Documentation 6.4

ZABBIX

02.01.2026

Contents

Zabbix Manual **6**

Copyright notice . . . 6

1 Introduction . . . 6

1 Manual structure . . . 6

2 What is Zabbix . . . 7

3 Zabbix features . . . 7

4 Zabbix overview . . . 8

5 What's new in Zabbix 6.4.0 . . . 9

6 What's new in Zabbix 6.4.1 . . . 17

7 What's new in Zabbix 6.4.2 . . . 18

8 What's new in Zabbix 6.4.3 . . . 18

9 What's new in Zabbix 6.4.4 . . . 19

10 What's new in Zabbix 6.4.5 . . . 19

11 What's new in Zabbix 6.4.6 . . . 20

12 What's new in Zabbix 6.4.7 . . . 21

13 What's new in Zabbix 6.4.8 . . . 21

14 What's new in Zabbix 6.4.9 . . . 22

15 What's new in Zabbix 6.4.10 . . . 23

16 What's new in Zabbix 6.4.11 . . . 23

17 What's new in Zabbix 6.4.12 . . . 24

18 What's new in Zabbix 6.4.13 . . . 24

19 What's new in Zabbix 6.4.14 . . . 25

20 What's new in Zabbix 6.4.15 . . . 25

21 What's new in Zabbix 6.4.16 . . . 26

22 What's new in Zabbix 6.4.17 . . . 26

23 What's new in Zabbix 6.4.18 . . . 26

24 What's new in Zabbix 6.4.19 . . . 27

25 What's new in Zabbix 6.4.20 . . . 27

26 What's new in Zabbix 6.4.21 . . . 27

2 Definitions . . . 27

3 Zabbix processes . . . 29

1 Server . . . 29

2 Agent . . . 37

3 Agent 2 . . . 40

4 Proxy . . . 42

5 Java gateway . . . 45

6 Sender . . . 49

7 Get . . . 50

8 JS . . . 50

9 Web service . . . 51

4 Installation . . . 51

1 Getting Zabbix . . . 51

2 Requirements . . . 52

3 Installation from sources . . . 66

4 Installation from packages . . . 76

5 Installation from containers . . . 92

6 Web interface installation . . . 101

7 Upgrade procedure . . . 106

8 Known issues . . . 117

9 Template changes . . . 125

10 Upgrade notes for 6.4.0 . . . 127
11 Upgrade notes for 6.4.1 . . . 129
12 Upgrade notes for 6.4.2 . . . 129
13 Upgrade notes for 6.4.3 . . . 129
14 Upgrade notes for 6.4.4 . . . 129
15 Upgrade notes for 6.4.5 . . . 130
16 Upgrade notes for 6.4.6 . . . 130
17 Upgrade notes for 6.4.7 . . . 130
18 Upgrade notes for 6.4.8 . . . 130
19 Upgrade notes for 6.4.9 . . . 131
20 Upgrade notes for 6.4.10 . . . 131
21 Upgrade notes for 6.4.11 . . . 131
22 Upgrade notes for 6.4.12 . . . 131
23 Upgrade notes for 6.4.13 . . . 131
24 Upgrade notes for 6.4.14 . . . 132
25 Upgrade notes for 6.4.15 . . . 132
26 Upgrade notes for 6.4.16 . . . 132
27 Upgrade notes for 6.4.17 . . . 132
28 Upgrade notes for 6.4.18 . . . 132
29 Upgrade notes for 6.4.19 . . . 132
30 Upgrade notes for 6.4.20 . . . 132
31 Upgrade notes for 6.4.21 . . . 132
5 Quickstart . . . 133
1 Login and configuring user . . . 133
2 New host . . . 136
3 New item . . . 138
4 New trigger . . . 139
5 Receiving problem notification . . . 141
6 New template . . . 145
6 Zabbix appliance . . . 147
7 Configuration . . . 150
1 Hosts and host groups . . . 159
2 Items . . . 170
3 Triggers . . . 345
4 Events . . . 363
5 Event correlation . . . 367
6 Tagging . . . 373
7 Visualization . . . 376
8 Templates and template groups . . . 403
9 Templates out of the box . . . 404
10 Notifications upon events . . . 414
11 Macros . . . 459
12 Users and user groups . . . 470
13 Storage of secrets . . . 479
14 Scheduled reports . . . 485
15 Data export . . . 489
8 Service monitoring . . . 494
1 Service tree . . . 494
2 SLA . . . 498
3 Setup example . . . 499
9 Web monitoring . . . 504
1 Web monitoring items . . . 513
2 Real-life scenario . . . 514
10 Virtual machine monitoring . . . 522
1 VMware monitoring item keys . . . 524
2 Virtual machine discovery key fields . . . 542
3 JSON examples for VMware items . . . 545
4 VMware monitoring setup example . . . 549
11 Maintenance . . . 553
12 Regular expressions . . . 557
13 Problem acknowledgment . . . 562
1 Problem suppression . . . 564
14 Configuration export/import . . . 565

1 Template groups . . . 566
2 Host groups . . . 567
3 Templates . . . 567
4 Hosts . . . 587
5 Network maps . . . 605
6 Media types . . . 611
15 Discovery . . . 618
1 Network discovery . . . 618
2 Active agent autoregistration . . . 626
3 Low-level discovery . . . 629
16 Distributed monitoring . . . 681
1 Proxies . . . 681
17 Encryption . . . 686
1 Using certificates . . . 693
2 Using pre-shared keys . . . 699
3 Troubleshooting . . . 702
18 Web interface . . . 704
1 Menu . . . 705
2 Frontend sections . . . 713
3 User settings . . . 870
4 Global search . . . 874
5 Frontend maintenance mode . . . 876
6 Page parameters . . . 877
7 Definitions . . . 878
8 Creating your own theme . . . 878
9 Debug mode . . . 879
10 Cookies used by Zabbix . . . 880
11 Time zones . . . 881
12 Resetting password . . . 881
19 API . . . 882
Method reference . . . 887
Appendix 1. Reference commentary . . . 1547
Appendix 2. Changes from 6.2 to 6.4 . . . 1552
Zabbix API changes in 6.4 . . . 1555
20 Extensions . . . 1556
1 Loadable modules . . . 1558
2 Plugins . . . 1567
3 Frontend modules . . . 1574
21 Appendixes . . . 1575
1 Installation and setup . . . 1575
2 Process configuration . . . 1630
3 Protocols . . . 1696
4 Items . . . 1722
5 Supported functions . . . 1751
6 Macros . . . 1779
7 Unit symbols . . . 1817
8 Time period syntax . . . 1819
9 Command execution . . . 1819
10 Version compatibility . . . 1820
11 Database error handling . . . 1821
12 Zabbix sender dynamic link library for Windows . . . 1821
13 Python library for Zabbix API . . . 1822
14 Service monitoring upgrade . . . 1822
15 Other issues . . . 1823
16 Agent vs agent 2 comparison . . . 1824
17 Escaping examples . . . 1825
22 Quick reference guides . . . 1827
Overview . . . 1827
1 Monitor Linux with Zabbix agent . . . 1827
2 Monitor Windows with Zabbix agent . . . 1830
3 Monitor Apache via HTTP . . . 1834
4 Monitor MySQL with Zabbix agent 2 . . . 1839
5 Monitor VMware with Zabbix . . . 1846

Developer Center **1849**
Copyright notice . . . 1849
Modules . . . 1849
Module file structure . . . 1850
Widgets . . . 1857
Tutorials . . . 1867
Examples . . . 1891
Plugins . . . 1891
Examples . . . 1894
Create a plugin (tutorial) . . . 1894
Plugin interfaces . . . 1898

Zabbix manpages **1899**
zabbix_agent2 . . . 1899
NAME . . . 1899
SYNOPSIS . . . 1899
DESCRIPTION . . . 1899
OPTIONS . . . 1899
FILES . . . 1900
SEE ALSO . . . 1900
AUTHOR . . . 1900
Index . . . 1900
zabbix_agentd . . . 1901
NAME . . . 1901
SYNOPSIS . . . 1901
DESCRIPTION . . . 1901
OPTIONS . . . 1901
FILES . . . 1902
SEE ALSO . . . 1902
AUTHOR . . . 1902
Index . . . 1902
zabbix_get . . . 1903
NAME . . . 1903
SYNOPSIS . . . 1903
DESCRIPTION . . . 1903
OPTIONS . . . 1903
EXAMPLES . . . 1904
SEE ALSO . . . 1904
AUTHOR . . . 1904
Index . . . 1905
zabbix_js . . . 1905
NAME . . . 1905
SYNOPSIS . . . 1905
DESCRIPTION . . . 1905
OPTIONS . . . 1905
EXAMPLES . . . 1906
SEE ALSO . . . 1906
Index . . . 1906
zabbix_proxy . . . 1906
NAME . . . 1906
SYNOPSIS . . . 1906
DESCRIPTION . . . 1906
OPTIONS . . . 1906
FILES . . . 1907
SEE ALSO . . . 1907
AUTHOR . . . 1907
Index . . . 1907
zabbix_sender . . . 1908
NAME . . . 1908
SYNOPSIS . . . 1908
DESCRIPTION . . . 1908
OPTIONS . . . 1908
EXIT STATUS . . . 1910

EXAMPLES . . . 1911
SEE ALSO . . . 1911
AUTHOR . . . 1911
Index . . . 1911
zabbix_server . . . 1912
NAME . . . 1912
SYNOPSIS . . . 1912
DESCRIPTION . . . 1912
OPTIONS . . . 1912
FILES . . . 1913
SEE ALSO . . . 1913
AUTHOR . . . 1914
Index . . . 1914
zabbix_web_service . . . 1914
NAME . . . 1914
SYNOPSIS . . . 1914
DESCRIPTION . . . 1914
OPTIONS . . . 1914
FILES . . . 1914
SEE ALSO . . . 1915
AUTHOR . . . 1915
Index . . . 1915

Zabbix Manual

Welcome to the user manual for Zabbix software. These pages are created to help users successfully manage their monitoring tasks with Zabbix, from the simple to the more complex ones.

Copyright notice

Zabbix documentation is NOT distributed under a GPL license. Use of Zabbix documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Zabbix disseminates it (that is, electronically for download on a Zabbix web site) or on a USB or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Zabbix. Zabbix reserves any and all rights to this documentation not expressly granted above.

1 Introduction

Please use the sidebar to access content in the Introduction section.

1 Manual structure

Structure

The content of this manual is divided into sections and subsections to provide easy access to particular subjects of interest.

When you navigate to respective sections, make sure that you expand section folders to reveal full content of what is included in subsections and individual pages.

Cross-linking between pages of related content is provided as much as possible to make sure that relevant information is not missed by the users.

Sections

Introduction provides general information about current Zabbix software. Reading this section should equip you with some good reasons to choose Zabbix.

Zabbix concepts explain the terminology used in Zabbix and provides details on Zabbix components.

Installation and Quickstart sections should help you to get started with Zabbix. Zabbix appliance is an alternative for getting a quick taster of what it is like to use Zabbix.

Configuration is one of the largest and more important sections in this manual. It contains loads of essential advice about how to set up Zabbix to monitor your environment, from setting up hosts to getting essential data to viewing data to configuring notifications and remote commands to be executed in case of problems.

Service monitoring section details how to use Zabbix for a high-level overview of your monitoring environment.

Web monitoring should help you learn how to monitor the availability of web sites.

Virtual machine monitoring presents a how-to for configuring VMware environment monitoring.

Maintenance, Regular expressions, Event acknowledgment and XML export/import are further sections that reveal how to use these various aspects of Zabbix software.

Discovery contains instructions for setting up automatic discovery of network devices, active agents, file systems, network interfaces, etc.

Distributed monitoring deals with the possibilities of using Zabbix in larger and more complex environments.

Encryption helps explaining the possibilities of encrypting communications between Zabbix components.

Web interface contains information specific for using the web interface of Zabbix.

API section presents details of working with Zabbix API.

Detailed lists of technical information are included in Appendixes. This is where you will also find a FAQ section.

2 What is Zabbix

Overview

Zabbix was created by Alexei Vladishev, and currently is actively developed and supported by Zabbix SIA.

Zabbix is an enterprise-class open source distributed monitoring solution.

Zabbix is a software that monitors numerous parameters of a network and the health and integrity of servers, virtual machines, applications, services, databases, websites, the cloud and more. Zabbix uses a flexible notification mechanism that allows users to configure email based alerts for virtually any event. This allows a fast reaction to server problems. Zabbix offers excellent reporting and data visualization features based on the stored data. This makes Zabbix ideal for capacity planning.

Zabbix supports both polling and trapping. All Zabbix reports and statistics, as well as configuration parameters, are accessed through a web-based frontend. A web-based frontend ensures that the status of your network and the health of your servers can be assessed from any location. Properly configured, Zabbix can play an important role in monitoring IT infrastructure. This is equally true for small organizations with a few servers and for large companies with a multitude of servers.

Zabbix is free of cost. Zabbix is written and distributed under the GPL General Public License version 2. It means that its source code is freely distributed and available for the general public.

Commercial support is available and provided by Zabbix Company and its partners around the world.

Learn more about Zabbix features.

Users of Zabbix

Many organizations of different size around the world rely on Zabbix as a primary monitoring platform.

3 Zabbix features

Overview

Zabbix is a highly integrated network monitoring solution, offering a multiplicity of features in a single package.

Data gathering

- availability and performance checks
- support for SNMP (both trapping and polling), IPMI, JMX, VMware monitoring
- custom checks
- gathering desired data at custom intervals
- performed by server/proxy and by agents

Flexible threshold definitions

- you can define very flexible problem thresholds, called triggers, referencing values from the backend database

Highly configurable alerting

- sending notifications can be customized for the escalation schedule, recipient, media type
- notifications can be made meaningful and helpful using macro variables
- automatic actions include remote commands

Real-time graphing

- monitored items are immediately graphed using the built-in graphing functionality

Web monitoring capabilities

- Zabbix can follow a path of simulated mouse clicks on a web site and check for functionality and response time

Extensive visualization options

- ability to create custom graphs that can combine multiple items into a single view
- network maps
- slideshows in a dashboard-style overview
- reports
- high-level (business) view of monitored resources

Historical data storage

- data stored in a database
- configurable history
- built-in housekeeping procedure

Easy configuration

- add monitored devices as hosts
- hosts are picked up for monitoring, once in the database
- apply templates to monitored devices

Use of templates

- grouping checks in templates
- templates can inherit other templates

Network discovery

- automatic discovery of network devices
- agent autoregistration
- discovery of file systems, network interfaces and SNMP OIDs

Fast web interface

- a web-based frontend in PHP
- accessible from anywhere
- you can click your way through
- audit log

Zabbix API

- Zabbix API provides programmable interface to Zabbix for mass manipulations, third-party software integration and other purposes.

Permissions system

- secure user authentication
- certain users can be limited to certain views

Full featured and easily extensible agent

- deployed on monitoring targets
- can be deployed on both Linux and Windows

Binary daemons

- written in C, for performance and small memory footprint
- easily portable

Ready for complex environments

- remote monitoring made easy by using a Zabbix proxy

4 Zabbix overview

Architecture

Zabbix consists of several major software components. Their responsibilities are outlined below.

Server

Zabbix server is the central component to which agents report availability and integrity information and statistics. The server is the central repository in which all configuration, statistical and operational data are stored.

Database storage

All configuration information as well as the data gathered by Zabbix is stored in a database.

Web interface

For an easy access to Zabbix from anywhere and from any platform, the web-based interface is provided. The interface is part of Zabbix server, and usually (but not necessarily) runs on the same physical machine as the one running the server.

Proxy

Zabbix proxy can collect performance and availability data on behalf of Zabbix server. A proxy is an optional part of Zabbix deployment; however, it may be very beneficial to distribute the load of a single Zabbix server.

Agent

Zabbix agents are deployed on monitoring targets to actively monitor local resources and applications and report the gathered data to Zabbix server. Since Zabbix 4.4, there are two types of agents available: the Zabbix agent (lightweight, supported on many platforms, written in C) and the Zabbix agent 2 (extra-flexible, easily extendable with plugins, written in Go).

Data flow

In addition it is important to take a step back and have a look at the overall data flow within Zabbix. In order to create an item that gathers data you must first create a host. Moving to the other end of the Zabbix spectrum you must first have an item to create a trigger. You must have a trigger to create an action. Thus if you want to receive an alert that your CPU load is too high on *Server X* you must first create a host entry for *Server X* followed by an item for monitoring its CPU, then a trigger which activates if the CPU is too high, followed by an action which sends you an email. While that may seem like a lot of steps, with the use of templating it really isn't. However, due to this design it is possible to create a very flexible setup.

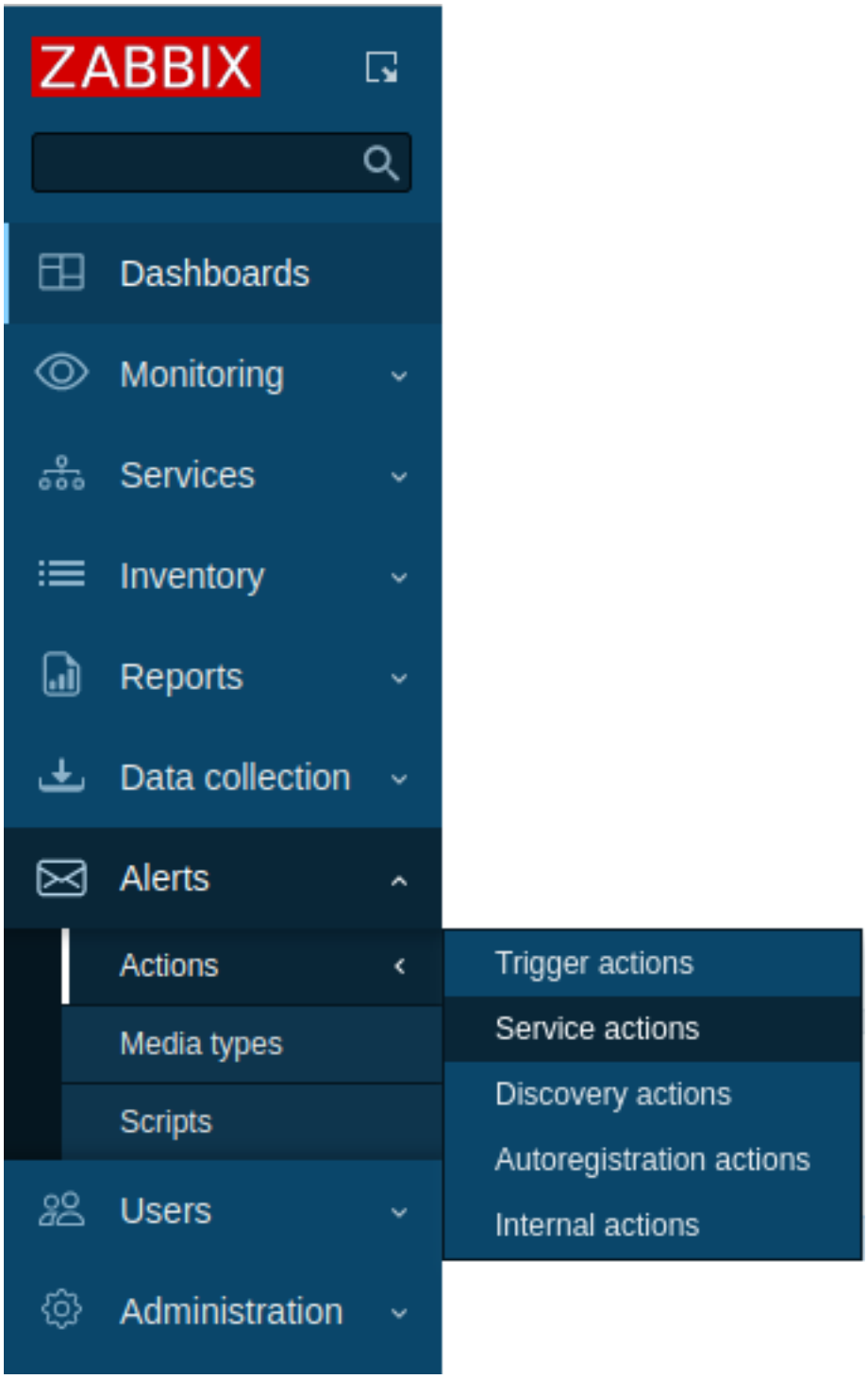
5 What's new in Zabbix 6.4.0

See breaking changes for this version.

Menu layout

The new version features an updated menu layout, which includes the following changes:

- **Dashboards** now are a top-level menu entry (previously *Dashboard* under the *Monitoring* menu).
- A new **Alerts** top-level menu has been added, which contains submenus related to alerting such as *Actions*, *Media types* and *Scripts*.
- *Actions* has a submenu with all types of actions (including *Service actions*, previously under *Services*).
- A new **Users** top-level menu has been added, which contains submenus related to user management such as *User groups*, *User roles*, *Users*, *API tokens* and *Authentication*.
- A new **Data collection** menu replaces the previous *Configuration* menu and contains subsections that are related to configuring data collection (with actions moved to *Alerts*).



In other changes:

- The *Audit* section has been renamed to *Audit log* under *Reports*.
- The **Administration** menu no longer contains submenus related to alerting and user management. Also:
- *Housekeeping*, *Macros* and *Audit log* that previously were submenus of *Administration -> General* have been moved one level up and now are submenus of *Administration*.

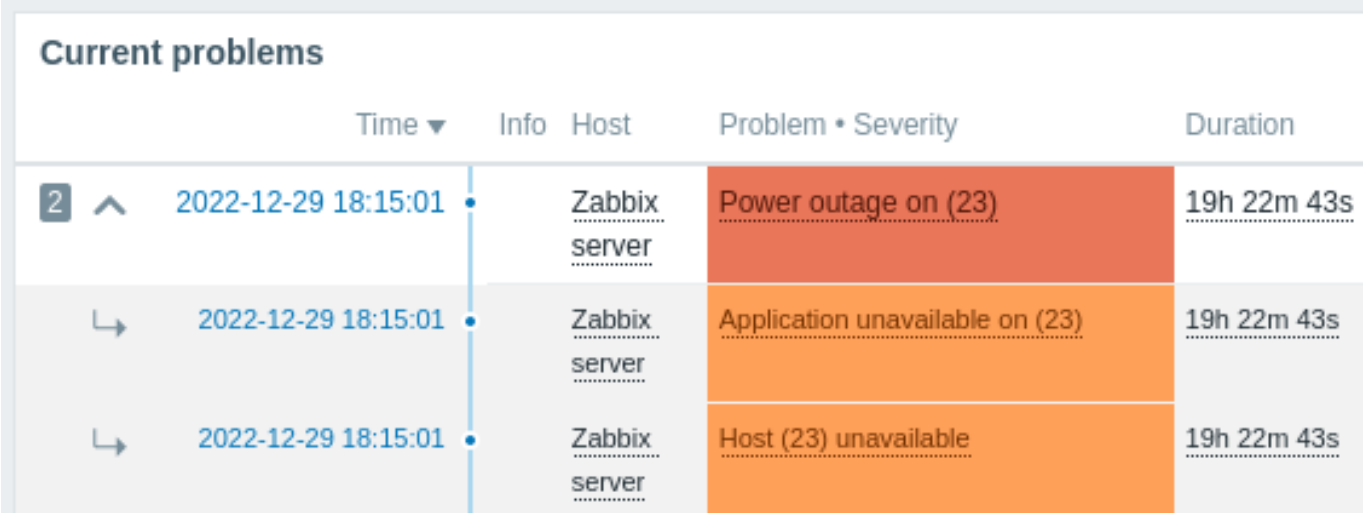
The new menu in full can be seen in frontend sections.

Note that access to the new menu depends on the user type and defined user role (see Permissions for more details).

Cause and symptom problems

The new version comes with an option to mark problems as cause or symptom problems.

For example, power outage may be the actual root cause why some host is unreachable or some service is down. In this case, the ”host is unreachable” and ”service is down” problems may be classified as the symptom problems of ”power outage” - the cause problem.



Symptom problems are grouped under the cause problem and marked accordingly, with an icon, smaller font and different background. The cause problem has a number showing how many symptoms are attributed to it.

By default all new problems are classified as cause problems. It is possible, in *Monitoring -> Problems*, to manually reclassify certain problems as symptom problems of the cause problem.

It is also possible to revert a symptom problem back to a cause problem.

New {EVENT.CAUSE.*} macros allow to reference the cause event in the symptom event notifications, while {EVENT.SYMPTOMS} allows to retrieve a list of symptoms in the cause event messages.

See also: Cause and symptom problems.

Streaming to external systems

It is now possible to stream item values and events from Zabbix to external systems over HTTP (see protocol details). The tag filter can be used to stream subsets of item values or events.

Warning:
This feature currently has experimental status.

The following steps are required to configure data streaming to an external system:

- Have a remote system set up for receiving data from Zabbix (see the documentation of a simple receiver).
- Set the required number of connector workers in Zabbix (see StartConnectors in zabbix_server.conf). Restart the Zabbix server.
- Configure a new connector in Zabbix (*Administration -> General -> Connectors*) with the receiver URL and other parameters.

When the connector has been saved, Zabbix will start sending data to the data receiver.

See also: Streaming to external systems

In this development, two new processes have been added to Zabbix server: `connector manager` and `connector worker`. A new Zabbix internal item `zabbix[connector_queue]` allows to monitor the count of values enqueued in the connector queue.

LDAP/SAML user provisioning

It is now possible to configure JIT (just-in-time) user provisioning for LDAP/SAML users. In this case, it is not required that a user already exists in Zabbix. The user account can be created when the user logs into Zabbix for the first time.

In addition to automatic user creation and updating the new functionality also allows to specify user group and user media matching between LDAP/SAML and Zabbix.

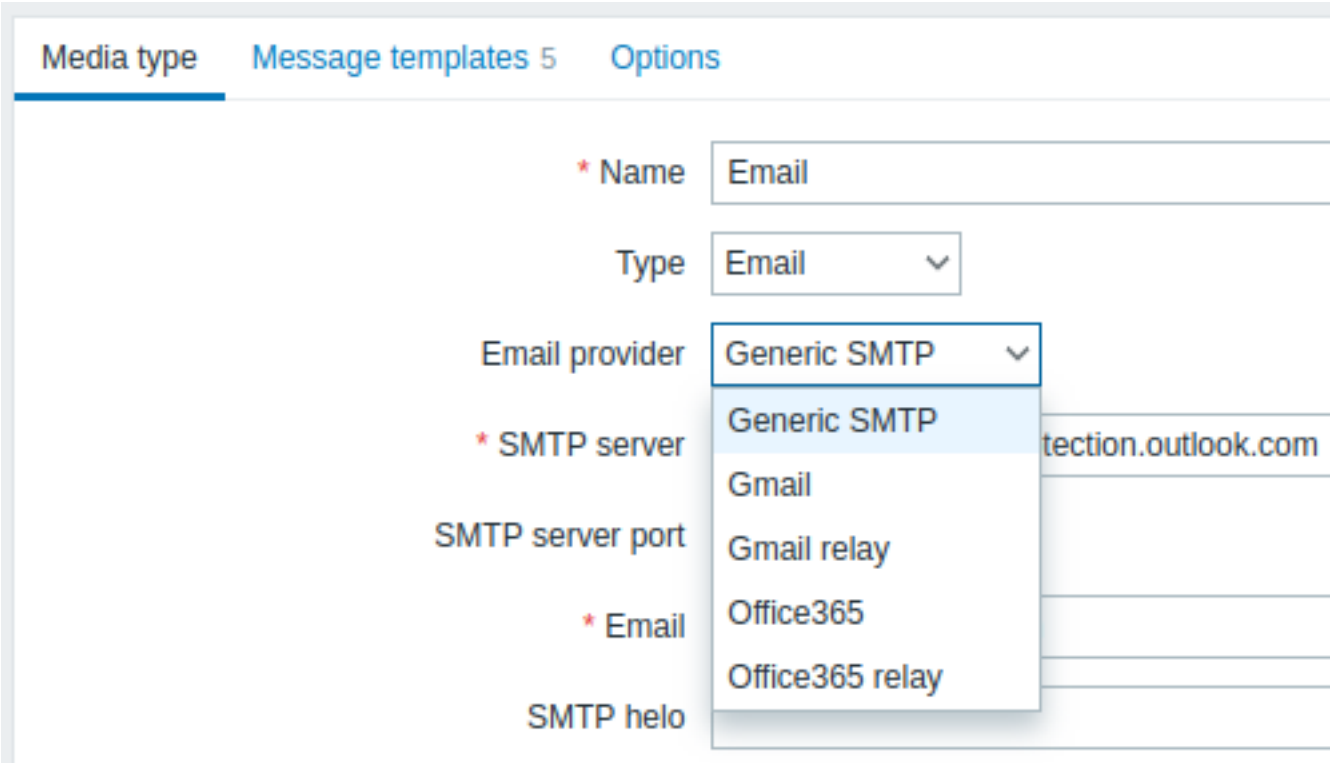
Provisioned users will be marked in the user list by a date entry in a new *Provisioned* column. Also, a *Provision now* option has been added to the user list to update users created from LDAP.

See also:

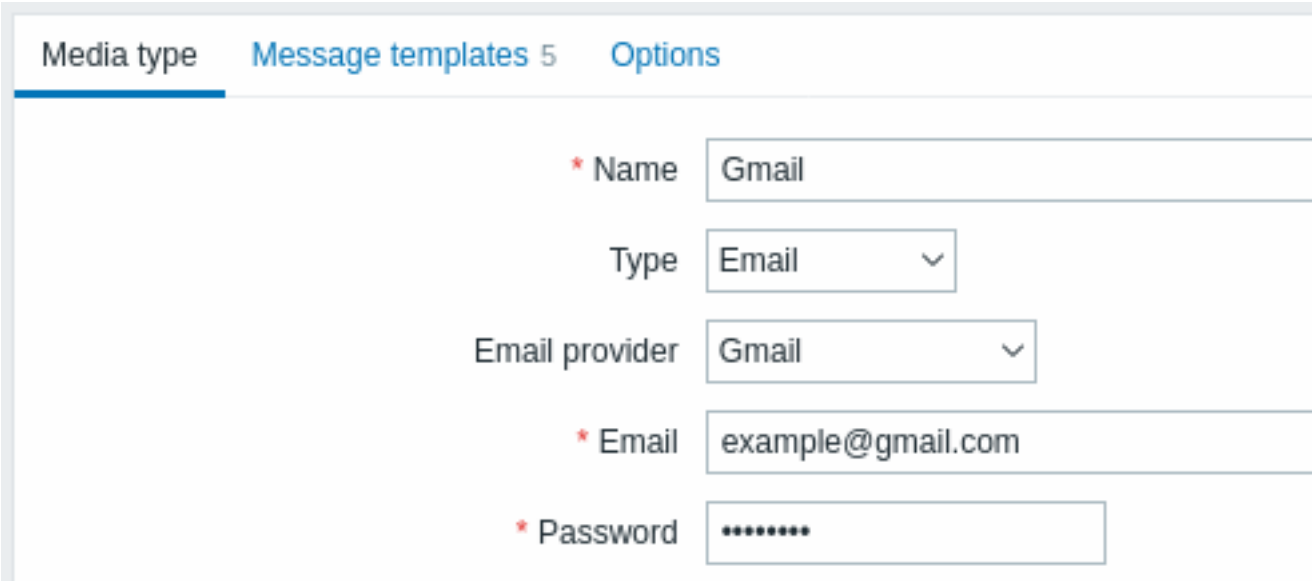
- LDAP authentication
- SAML authentication
- SAML setup with Okta
- Users

Automation for Gmail/Office365 media types

Gmail or Office365 users may now benefit from easier media type configuration. The new *Email provider* field in the mail media type configuration allows to select pre-configured options for Gmail and Office 365 (alongside the "Generic SMTP" option, which works as before).



When selecting the Gmail/Office365 related options, it is **only** required to supply the sender email address/password to create a working media type.



If the email address/password is supplied, Zabbix is able to automatically fill all required settings for Gmail/Office365 media types with the actual/recommended values, i.e. *SMTP server*, *SMTP server port*, *SMTP helo*, and *Connection security*. Because of this automation, these fields are not even shown, however, it is possible to see the SMTP server and email details in the media type list (see the *Details* column).

Note also that:

- The password is not required for the relay options.
- For Office365 relay, the domain name of the provided email address will be used to dynamically fill the SMTP server (i.e. replace "example.com" in `example-com.mail.protection.outlook.com` with the real value).
- The SMTP helo value is the domain name extracted from the provided email address.

In other changes:

- The *SMTP email* field is now called simply *Email*.
- The *SMTP helo* field is now optional (if empty, it will send the domain part of the sender email).
- New default media types have been added: *Gmail*, *Gmail relay*, *Office365*, *Office365 relay*.
- In new installations, all media types are disabled by default.
- In user media, where specific recipient addresses are specified, only enabled media types can be selected.
- In user media, there is now a yellow icon and status column shown indicating if the media type has been disabled.

See also:

- Email media type
- Media types
- User media
- User profile media

Optimized SNMP discovery and collection

SNMP discovery and data collection has been updated to use native SNMP bulk requests (GetBulkRequest-PDUs), available in SNMP versions 2/3.

A GetBulk request in SNMP executes multiple GetNext requests and returns the result in a single response. Previously, only the SNMP `discovery[]` item in Zabbix would use GetBulk requests; it was not available for regular SNMP items and also discovered SNMP items would have to query the devices independently.

Using the new SNMP **walk[OID1,OID2,...]** item, it becomes possible to collect data in one request and parse the response as needed, without additional requests to devices. For example,

`walk[1.3.6.1.2.1.2.2.1.2,1.3.6.1.2.1.2.2.1.3]`

returns a multi-line list of interface names and types. Using the Zabbix preprocessing options, the response of this item may be used to discover interfaces, create discovered items and populate their values. Note that the discovery rule in this case must be a dependent discovery rule to the `walk[]` master item and item prototypes must be dependent item prototypes of the `walk[]` master item.

To make this functionality possible, two new preprocessing steps have been added:

- *SNMP walk value* - extract and format SNMP walk value by the specified OID/MIB name;
- *SNMP walk to JSON* - convert SNMP walk values to JSON. This step may be used in low-level discovery of SNMP OIDs.

The walk[] item returns the output of the snmpwalk utility with -Oe -Ot -On parameters. MIB names are supported as parameters; thus `walk[1.3.6.1.2.1.2.2.1.2]` and `walk[ifDescr]` will return the same output. If several OIDs/MIBs are specified, i.e. `walk[ifDescr,ifType,ifPhysAddress]`, then the output is a concatenated list.

This item uses GetBulk requests with SNMPv2 and v3 interfaces and GetNext for SNMPv1 interfaces; the max repetition value for GetBulk requests is configured on the SNMP interface level.

The previous ”use bulk requests” option in Zabbix has been retained (under a new name ”use combined requests”). This is not related to native SNMP bulk requests in any way; it is Zabbix own way of combining multiple SNMP requests.

See also:

- SNMP agent checks
- Item preprocessing
- SNMP OID discovery

Notification macros in alert scripts

Alert script parameters now support user macros and all built-in macros that are supported in Zabbix notifications (including in trigger-based, autoregistration, discovery, internal, and service notifications) in addition to the {ALERT.*} macros supported previously.

Alert script testing form has been updated to allow specifying custom parameter values for the test procedure.

Interface not required for some checks

It is no longer required to define an interface when creating items of the following type:

- Simple check
- External check
- SSH agent
- Telnet agent

The interface field for these items is no longer mandatory. Also, it is now possible to select a ”None” option in the interface field.

Runtime commands for profiling

Runtime commands for profiling have been added to Zabbix server and Zabbix proxy.

- `prof_enable` - enable profiling
- `prof_disable` - disable profiling

Profiling can be enabled per server/proxy process. Enabled profiling provides details of all rwlocks/mutexes by function name.

See also:

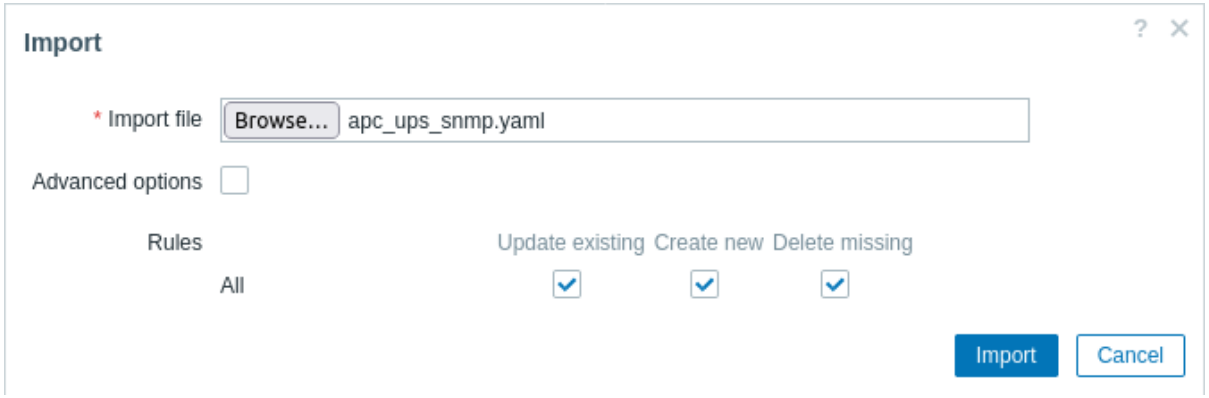
- Zabbix server runtime commands
- Zabbix proxy runtime commands

Date removed from export

When exporting objects (hosts, templates, etc.), note that the element `date` is removed from the export format.

Host and template import

The configuration import form for templates/hosts has a new row for *All* entities in the import rules section.



If you mark the checkbox in the *All* row, all importable entities become marked/unmarked.

To see the full list of entities in the import rules, it is now required to mark the *Advanced options* checkbox. In another change, the *Delete missing* option is now checked by default.

Action log export to CSV

It is now possible to export action log records to a CSV file. For more information, see Action log.

Higher limit for host metadata

The HostMetadataItem parameter in agent configuration, used in host autoregistration, now can return up to 65535 UTF-8 code points (instead of 255 in previous versions). A longer value will be truncated.

Note that on MySQL, the effective maximum length in characters will be less if the returned value contains multibyte characters. For example, a value containing 3-byte characters only will be limited to 21844 characters in total, while a value containing 4-byte characters only will be limited to 16383 symbols.

Also, the maximum length of the HostMetadata option in agent configuration has been increased to 2034 bytes.

Security Secure password change

When changing your user password in *User profile* or *User configuration* (for users with the *Super admin role*), Zabbix now asks and verifies the current (old) password to allow changing it. On a successful password change, the user will be logged out of all active sessions. Note that the password can only be changed for users using Zabbix internal authentication.

CSRF tokens

For enhanced security against CSRF (Cross Site Request Forgery) attacks, Zabbix frontend now uses randomly generated CSRF tokens instead of static session-based tokens.

Items New and updated agent items

New items have been added to Zabbix agent/agent 2:

- **system.sw.packages.get** - return information about installed packages in JSON format. In comparison to the existing `system.sw.packages` item, the new item returns more details.
- **system.sw.os.get** - return information about the operating system, such as version, type, distribution name, build number, minor and major version, etc., in JSON format.

The following Zabbix agent/agent 2 items have been updated:

- **system.sw.os** that returns OS information as a string is now also supported on Windows.
- **vfs.fs.get** and **vfs.fs.discovery**, used for discovery of mounted filesystems, now additionally return filesystem mount options in the "options" and "{#FSOPTIONS}" property respectively. These options allow to detect filesystems that remounted as read-only (e.g. on VMs) or to filter bind mounts or .dmg volumes on macOS.

See also:

- Zabbix agent items for detailed item description.
- Template changes for details how item-related changes are reflected in the existing templates.

Additional ssh.run options

The item **ssh.run[]** has been updated and now allows to pass additional SSH options as part of the item key. See SSH checks for details.

Templates Template versioning

To improve management and upgrade of templates, template versioning has been introduced.

In *Data collection → Templates* you can now see the template vendor and version, as well as filter templates by vendor and/or version.

For more information, see Upgrade notes for 6.4.0.

Updated templates

The following templates have been updated:

- Templates that discover filesystems (updated to make use of the `vfs.fs.get` item instead of `vfs.fs.discovery` item with some additional changes);
- *Linux by Zabbix agent* (`system.sw.packages` item replaced by `system.sw.packages.get` item; new trigger);
- *Windows by Zabbix agent*, *Windows by Zabbix agent active* (included `system.sw.os` item; new trigger).

For more information about the updates, see Template changes.

You can get these templates:

- In *Data collection → Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from Zabbix Git repository or find them in the *zabbix/templates* directory of the downloaded latest Zabbix version. Then, while in *Data collection → Templates*, you can import them manually into Zabbix.

Performance Instant refresh of active checks

Previously Zabbix agent (in active mode) received from Zabbix server or Zabbix proxy a full copy of the configuration once every two minutes (default). By introducing incremental configuration sync, full configuration is no longer sent if there are no changes to host or global regular expressions, thus default sync interval has been reduced to 5 seconds.

'RefreshActiveChecks' parameter supported in a Zabbix agent configuration file default value is changed to 5 seconds (previously 120).

JSON protocol for active agent checks has been updated to include `config_revision` and `session` ID. For more information, see Passive and active agent checks.

Thread-based preprocessing workers

The item value preprocessing has been rewritten to use thread-based preprocessing workers for enhanced parallelism and reduced overhead. This development should help avoid situations when the prolonged preprocessing of one item holds up others.

This change adds a new required library for Zabbix server/proxy - `libevent_pthreads`.

See also: Preprocessing details

Value cache optimization

Previously the value cache was optimized for working with frequently changing data. Values for items that were updated less than daily were removed from the value cache.

Now the value cache is optimized for a wider variety of monitoring patterns. Item values remain in the value cache until the item is deleted or until the item value is outside the time or count range specified in the trigger/calculated item expression.

For more information, see Value cache.

Optimized proxy configuration update

In previous Zabbix versions the server would send a full copy of the configuration to the proxy every time the configuration is synced.

Now it has been replaced with an incremental update of the proxy configuration. During a configuration sync only the modified entities are updated (thus, if no entities have been modified, nothing will be sent). This approach allows to save resources and set a smaller interval (down to almost instant) for the proxy configuration update.

For more details, see Synchronization of monitoring configuration.

Configuration parameters

The **ProxyConfigFrequency** parameter determines how often the proxy configuration is synced with the server (now 10 seconds by default).

Note that ProxyConfigFrequency is:

- the server parameter for passive proxies;
- the proxy parameter for active proxies.

On active proxies ProxyConfigFrequency is a **new** parameter and must be used instead of the now-deprecated ConfigFrequency.

> **Attention:**
> If both ProxyConfigFrequency and ConfigFrequency are used, the proxy will log an error and terminate.

Also, the default value of CacheUpdateFrequency on the server has been lowered from 60 to **10** seconds.

Zabbix server support for older proxies

In a server-proxy setup, the upgrade procedure has become more flexible. Now, Zabbix server officially supports data collection, execution of remote commands, and immediate item value checks by outdated proxies that are no older than Zabbix server previous LTS release version. Older proxies are not supported, and all communication with Zabbix server will fail with a warning. For more information, see Upgrade procedure.

Automated database upgrade on proxies with SQLite

Upon the first launch after an upgrade, Zabbix proxy with SQLite3 now automatically dumps the existing older version of the database file and creates a new one for the current version. History data that has been stored in the SQLite database file is not preserved. Previously, the SQLite database file had to be recreated manually.

Heartbeat dropped from Zabbix proxy

The *heartbeat sender* has been removed from the proxy. Therefore, Zabbix proxy item *zabbix [process,heartbeat sender]* is no longer supported and has been removed from templates. The *HeartbeatFrequency* parameter has been deprecated.

Frontend Context menu

Context menus for items, hosts, and events have become more functional.

New options

- Host context menu now offers options to navigate to the lists of host items, triggers, discovery rules, or web scenarios to quickly access configuration of the required entity.
- Item context menu now offers options to:
  - see all triggers based on an item and navigate to the configuration of any of them;
  - create a new trigger, a dependent item, or a discovery rule based on the item;
  - switch to the *Latest data* section filtered by the current host and item.
- Event context menu now offers options to:
  - acknowledge or update a problem;
  - see all items used by a problem trigger and navigate to the configuration of any of them;
  - mark problems as symptom or cause.

Note that links to the configuration section are only visible to Admin and Super admin level users with sufficient permissions.

Custom links

Host context menu and event context menu can be customized further by adding custom links. In the *Alerts* → *Scripts* menu section, it is now possible to add a global script of the new type URL. If configured, such links will be visible in the context menu of matching hosts/problem events.

Configurable trigger URL label

When configuring a trigger, it is now possible to add a custom label to a trigger URL. If configured, the custom label will be displayed instead of the default label (*Trigger URL*) in the event context menu.

A new {TRIGGER.URL.NAME} macro for trigger URL labels has been introduced. This macro is supported in all places currently supported by the existing macro {TRIGGER.URL}.

Additional item menu location

Item context menu is now also available in:

- *Data collection* → *Hosts* → *Items*
- *Data collection* → *Hosts* → *Discovery rules* → *Item prototypes*

Dashboard widgets

Dynamic background color for Item value widget

The *Item value* widget now allows to configure a dynamic background color based on the thresholds set and the latest received value.

Usability updates for Graph widget

When configuring a *Graph* widget, it is now possible to rename data sets by customizing the *Data set label*. This makes it easier to identify data sets in widget configuration, as well as identify aggregated data sets in graph *Legend*.

Autocomplete functionality of the widget has also been improved. The *Item pattern* field suggestions now only display the items that belong to the hosts selected in the *Host pattern* field.

Decimal places for item values in Top hosts widget

When configuring columns of data type ”Item value” in the *Top hosts* widget, it is now possible to specify how many decimal places will be displayed with the value.

Dynamic item field renamed

The field that enables dynamic widgets has been renamed from *Dynamic item* to ***Enable host selection*** for more clarity.

Ack link renamed to update

The link from the problem list to the problem update screen has been renamed from *Ack* (Acknowledge) to *Update* to correctly reflect multiple options that are available in the problem update screen in recent versions (not only acknowledgment).

The respective *Ack* (Acknowledgment) column in the problem list has also been renamed to *Update*. When a problem has been acknowledged it is no longer displayed by the link color in the acknowledgment column; it is now displayed by a green checkbox  icon in the Actions column.

Host filtering by status

Hosts displayed in the *Data collection* → *Hosts* section can now be filtered by status (enabled/disabled).

New action log filtering options

New filtering options have been added to *Reports* → *Action log* section. In addition to filtering records by notification recipients, now you can also filter records by actions, media types, status, or by the message/remote command content. These filtering options can also be configured for the *Action log* widget.

Frontend languages

Catalan language is now enabled in the frontend.

Modal forms

The forms for action configuration, maintenance period configuration, as well as the form for copying items, triggers and graphs between hosts or templates are now opened in a modal (pop-up) window.

6 What’s new in Zabbix 6.4.1

MariaDB 10.11 support The maximum supported version for MariaDB is now 10.11.X.

TimescaleDB 2.10 support The maximum supported version for TimescaleDB is now 2.10.

Connection options for Oracle plugin The Oracle plugin, supported for Zabbix agent 2, now allows to specify `as sysdba`, `as sysoper`, or `as sysasm` login option. The option can be appended either to the user item key parameter or to the plugin configuration parameter Plugins.Oracle.Sessions.<SessionName>.User in the format `user as sysdba` (login option is case-insensitive; must not contain a trailing space).

Signing data using RS256 A new `sign(hash,key,data)` JavaScript function has been implemented allowing to use the RS256 encryption algorithm to calculate the signature.

For more details see: Additional JavaScript objects.

7 What's new in Zabbix 6.4.2

Configuration sync optimization for Oracle For Zabbix installations with Oracle, it is now possible to manually change item and item preprocessing database field types from *nclob* to *nvarchar2* by applying a database patch.

Patch application may increase the speed of configuration sync in environments with large number of items and item preprocessing steps, but will reduce the maximum field size limit from 65535 bytes to 4000 bytes for some item parameters. See Known issues for details.

Webhook integrations New webhook media type for pushing Zabbix notifications to Event-Driven Ansible has been added.

Mixing item key and session parameters in Zabbix agent 2 plugins Zabbix agent 2 now allows to override named session parameters by specifying new values in the item key parameters. Previously, users had to select if they prefer to provide connection string values in a named session or in an item key. If a named session has been used, related item key parameters had to be empty. Now, if using named sessions, only the first parameter (usually, a URI) has to be specified in the named session, whereas other parameters can be defined either in the named session or in the item key.

HTML support in Geomap attribution dropped The attribution text for the Geomap dashboard widget can now only contain plain text; HTML support has been dropped.

In Geographical maps settings in the Administration → General section, the field *Attribution* is now only visible when *Tile provider* is set to *Other*.

8 What's new in Zabbix 6.4.3

Items docker.container_stats

The docker.container_stats item on Zabbix agent 2 now also returns a `pids_stats` property with the current number of processes/threads on the container.

Default values for Zabbix agent 2 Zabbix agent 2 plugins now allow to define default values for connecting to monitoring targets in the configuration file. If no value is specified in an item key or a named session, the plugin will use the value defined in the corresponding default parameter. New parameters have the structure `Plugins.<PluginName>.Default.<Parameter>` - for example, `Plugins.MongoDB.Default.Uri=tcp://localhost:27017`. See for more info:

- Configuring plugins
- Plugin configuration file parameters

Cleaner configuration export YAML files generated during Zabbix entity configuration export no longer contain empty lines between entities in an array, which makes such files shorter and more convenient to work with. See Configuration export/import section for updated export examples.

UTF-8 BOM in configuration import Configuration import now supports files with a UTF-8 byte-order mark (BOM).

Cosmos DB monitoring The template *Azure by HTTP* now also works with Azure Cosmos DB for MongoDB.

You can get this template:

- In *Data collection* → *Templates* in new installations.
- If you are upgrading from previous versions, you can download this template from Zabbix Git repository or find it in the *zabbix/templates* directory of the downloaded latest Zabbix version. Then, while in *Data collection* → *Templates* you can import it manually into Zabbix.

Proxy history housekeeping The limitation on the amount of outdated information deleted from the proxy database per proxy history housekeeping cycle has been removed.

Previously the housekeeper deleted only no more than 4 times the `HousekeepingFrequency` hours of outdated information. For example, if `HousekeepingFrequency` was set to "1", no more than 4 hours of outdated information (starting from the oldest entry) was deleted. In cases when a proxy would constantly receive data older than set in `ProxyOfflineBuffer`, this could result in excessive data accumulation.

Now this limitation has been removed, providing a more effective proxy history housekeeping solution.

Templates A new template *Google Cloud Platform by HTTP (GCP by HTTP)* is available.

9 What's new in Zabbix 6.4.4

User creation A new user cannot be created without assigning a user role to them anymore (user role setting can be found under *Permissions* tab). When trying to do so, you will face an error stating: *"Cannot add user: field "roleid" is mandatory"*.

TimescaleDB 2.11 support Support for TimescaleDB version 2.11 is now available.

Aggregate functions The **count_foreach** function now returns '0' for a matching item in the array, if no data are present for the item or the data do not match the filter. Previously such items would be ignored (no data added to the aggregation).

JavaScript preprocessing The heap limit for scripts has been upped from 64 to 512 megabytes.

Configurable TLS and connection parameters in MQTT plugin The MQTT plugin for Zabbix agent 2 now provides additional configuration options, which can be defined in the plugin configuration file as named session or default parameters:

- Connection-related parameters: broker URL, topic, username, and password;
- TLS encryption parameters: location of the top-level CA(s) certificate, MQTT certificate or certificate chain, private key.

All of the new parameters are optional.

Supported platforms Support for Debian 12 (Bookworm) has been added, and official packages are available for download on Zabbix website.

10 What's new in Zabbix 6.4.5

Templates New templates are available:

- AWS ECS Cluster by HTTP (along with its Serverless Cluster version)
- Cisco SD-WAN by HTTP
- OpenStack by HTTP, which includes *OpenStack Nova by HTTP* template for monitoring OpenStack Nova service
- PostgreSQL by ODBC

You can get these templates:

- In *Data collection* → *Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from Zabbix Git repository or find them in the *zabbix/templates* directory of the downloaded latest Zabbix version. Then, while in *Data collection* → *Templates* you can import them manually into Zabbix.

Frontend Spellcheck disabled in non-descriptive text areas

Spellcheck has been disabled for the text areas in which non-descriptive text is entered, such as scripts, expressions, macro values, etc.

Miscellaneous Database TLS connection for MySQL on SLES 12

The packages for server/proxy installation on SUSE Linux Enterprise Server version 12 are now built using MariaDB Connector/C library, thus enabling the encryption of connection to MySQL using the DBTLSConnect parameter. The supported encryption values are ”required” and ”verify_full”.

11 What’s new in Zabbix 6.4.6

MySQL 8.1 support

The maximum supported version for MySQL is now 8.1.X.

MariaDB 11.0 support

The maximum supported version for MariaDB is now 11.0.X.

Log file monitoring

For `log[]`, `logrt[]`, `log.count[]`, `logrt.count[]` items, regular expression runtime errors are now logged in the Zabbix agent log file. See more details.

Items New item for Zabbix agent 2

A new item has been added to MySQL plugin for Zabbix agent 2. This new item, mysql.custom.query, can be used for executing custom MySQL queries.

Templates New template is available:

- AWS Cost Explorer by HTTP

You can get this template:

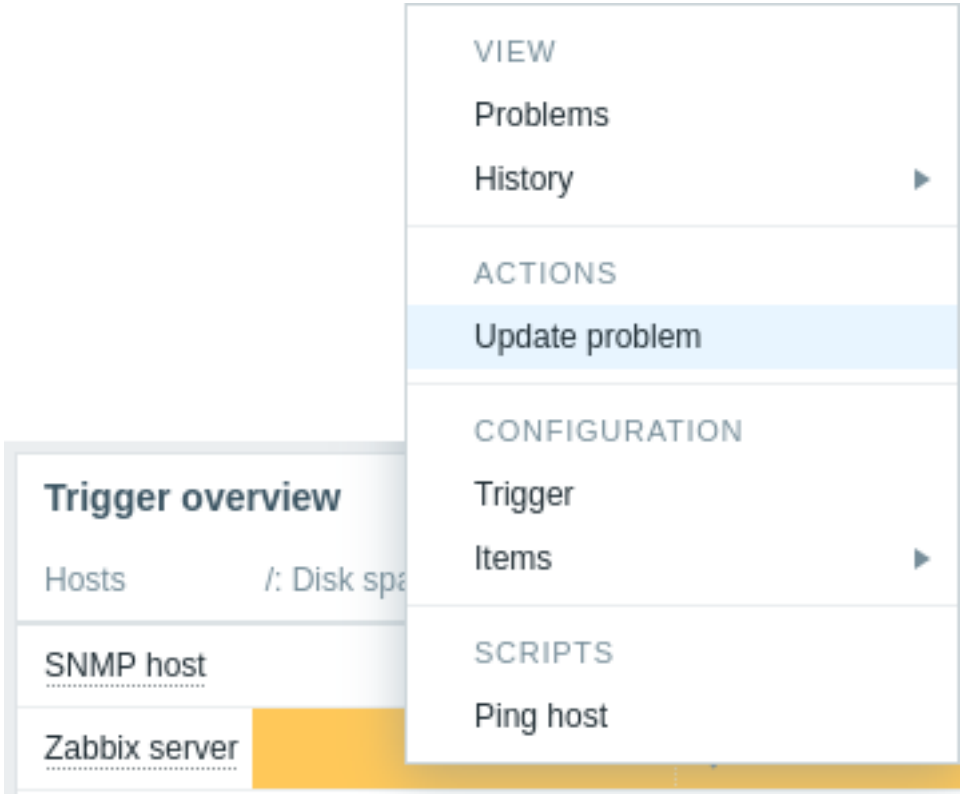
- In *Data collection* → *Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from Zabbix Git repository or find them in the *zabbix/templates* directory of the downloaded latest Zabbix version. Then, while in *Data collection* → *Templates* you can import them manually into Zabbix.

Notifications Webhook integrations

New webhook media type for pushing Zabbix notifications to Mantis Bug Tracker has been added.

Frontend Event menu layout

The layout of the event menu has been changed. The *Update problem* option has been moved to the Actions section. The Actions section is new and is placed between the View and Configuration sections.



The *Update problem* option is available in the event menu in *Trigger overview* widgets.

12 What's new in Zabbix 6.4.7

Aggregate functions

The **last_foreach** function is now also supported in the following aggregate functions: kurtosis, mad, skewness, stddevpop, stddevsamp, sumofsquares, varpop, and varsamp.

Return value limit

The return value limit for receiving data from external sources (such as scripts or other programs) has been raised to 16MB. This affects:

- Agent items system.run[] and vfs.file.contents[]
- Custom agent checks defined in user parameters
- SSH agent, External check, and Script items
- Remote commands

Templates New templates are available:

- Acronis Cyber Protect Cloud by HTTP
- HashiCorp Nomad by HTTP
- MantisBT by HTTP

You can get these templates:

- In *Data collection → Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from Zabbix Git repository or find them in the *zabbix/templates* directory of the downloaded latest Zabbix version. Then, while in *Data collection → Templates* you can import them manually into Zabbix.

13 What's new in Zabbix 6.4.8

See breaking changes for this version.

Databases Supported versions

PostgreSQL **16** and MariaDB **11.1** are now supported. See also: Requirements.

Plugins New item for PostgreSQL Zabbix agent 2 plugin

New item, **pgsql.version**, has been added to PostgreSQL Zabbix agent 2 plugin. This item is used for returning the PostgreSQL version.

Templates New templates

New templates are available:

- FortiGate by HTTP
- FortiGate by SNMP
- Nextcloud by HTTP

You can get these templates:

- In *Data collection* → *Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from Zabbix Git repository or find them in the *zabbix/templates* directory of the downloaded latest Zabbix version. Then, while in *Data collection* → *Templates* you can import them manually into Zabbix.

Updated templates

PostgreSQL by ODBC and PostgreSQL by Zabbix agent 2 templates now include the item and trigger for monitoring PostgreSQL version.

Cisco Meraki organization by HTTP template has been supplemented with items, item prototypes, LLD rules, and macros related to authentication, licenses, networks, SAML roles, and VPN statuses.

Frontend Miscellaneous

The *Clear history* button located in *Data collection* → *Hosts* → *Items* has been renamed *Clear history and trends* to more accurately describe its function, which is the same as the *Clear history and trends* button in the item configuration form.

In trigger action configuration, the condition type *Trigger name* has been renamed *Event name* to better describe its function. Note that by default, the event name matches the trigger name unless a custom event name is specified in trigger configuration.

14 What's new in Zabbix 6.4.9

Databases TimescaleDB 2.12 support

Support for TimescaleDB version 2.12 is now available.

Plugins New items in Zabbix agent 2 plugins

The items for returning the database server version are now available in MongoDB plugin (`mongodb.version`) and Oracle Database plugin (`oracle.version`).

Items Content conversion to UTF-8

HTTP agent items, web scenarios, web checks and JavaScript items have been improved to convert to UTF-8 from the character set specified in the HTTP header or HTTP meta tag.

Templates New templates

New template is available:

- HPE iLO by HTTP

You can get this template:

- In *Data collection* → *Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from Zabbix Git repository or find them in the *zabbix/templates* directory of the downloaded latest Zabbix version. Then, while in *Data collection* → *Templates*, you can import them manually into Zabbix.

Updated templates

Integration with OpenShift has been added to Kubernetes cluster state by HTTP template.

15 What's new in Zabbix 6.4.10

TimescaleDB 2.13 support

Support for TimescaleDB version 2.13 is now available.

User macro support in preprocessing custom error-handling parameters

User macros are now supported in item value preprocessing custom error-handling parameters (*Set value to* and *Set error to* fields).

Although undocumented, this functionality worked before Zabbix 6.4.0, but it stopped working in previous 6.4.x versions. Now the support has been restored and documented.

Plugins Cache mode parameter for PostgreSQL plugin

New parameters for controlling the cache mode by default or on session name level have been added to the PostgreSQL plugin configuration:

- `Plugins.PostgreSQL.Default.CacheMode`
- `Plugins.PostgreSQL.Sessions.<SessionName>.CacheMode`

The cache mode parameter may have one of two allowed values: *prepare* (default) or *describe*. Note that "describe" is primarily useful when the environment does not allow prepared statements such as when running a connection pooler like PgBouncer.

16 What's new in Zabbix 6.4.11

Templates New templates

The set of Azure by HTTP templates has been supplemented with the Azure Cost Management by HTTP template.

You can get this template:

- In *Data collection* → *Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from Zabbix Git repository or find them in the *zabbix/templates* directory of the downloaded latest Zabbix version. Then, while in *Data collection* → *Templates*, you can import them manually into Zabbix.

Updated templates

MSSQL by ODBC template has been updated for working with AlwaysOn features such as Failover Cluster Instances (FCI) and Availability Groups (AG). It is now possible to use the template for monitoring a host in cluster, standalone host and host by cluster name. A macro for instance name is no longer used; when the master is switched, it is not required to change any macros:

- new LLD rules and metrics for quorum and quorum members have been added;
- the type of the LLD rules has been changed from "Database monitor" to "Dependent item";
- items with `db.odbc.discovery` key have been turned into items dependent on the `db.odbc.get` item
- new item has been added - MSSQL DB '{#DBNAME}': Recovery model, which returns the database recovery model under the database discovery;
- new macros, namely, {$MSSQL.BACKUP_FULL.USED}, {$MSSQL.BACKUP_DIFF.USED}, {$MSSQL.BACKUP_LOG.USED}, have been added - those can be used for disabling backup age triggers for a certain database.

Frontend PHP support

The maximum supported version for PHP is now 8.3.

X-Frame-Options HTTP header

The *X-Frame-Options header* parameter has been renamed to *Use X-Frame-Options header*, now consists of a checkbox and an input field (allowing you to disable the header by unmarking a checkbox instead of specifying "null" in the input field), and supports additional values.

Other security parameters now also follow the same structure. For more information, see the security parameters in *Administration* → *General*.

Databases MySQL 8.2 support

The maximum supported version for MySQL is now 8.2.X.

17 What's new in Zabbix 6.4.12

See breaking changes for this version.

Agent interface availability behavior

The host availability behavior for Zabbix agent interfaces has been updated:

- if active checks are available, but at least one Agent interface (passive) is unknown while the host also has at least one item using this interface, the total Agent interface availability is displayed as gray (unknown);
- if active checks are available and all Agent interfaces (passive) are unknown (and no items are using this interface), the total Agent interface availability is displayed as green (available).

Databases MySQL 8.3 support

The maximum supported version for MySQL is now 8.3.X.

MariaDB 11.2 support

The maximum supported version for MariaDB is now 11.2.X.

Plugins MSSQL

A new plugin for direct monitoring of MSSQL by Zabbix agent 2 has been added.

For more information, see:

- MSSQL plugin readme
- Agent 2 items
- MSSQL plugin parameters
- Agent 2 installation

Templates New templates

A new template is available:

- YugabyteDB by HTTP, which includes the *YugabyteDB Cluster by HTTP* template for monitoring each YugabyteDB cluster.

You can get this template:

- In *Data collection → Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from the Zabbix Git repository or find them in the *zabbix/templates* directory of the latest Zabbix version you have downloaded. Then, while in *Data collection → Templates*, you can import them manually into Zabbix.

Platforms Zabbix agent 2 support on Windows

To prevent critical security vulnerabilities, the minimum Windows version for Zabbix agent 2 has been raised to Windows 10/Windows Server 2016. See note under Supported platforms for more information.

18 What's new in Zabbix 6.4.13

Databases TimescaleDB 2.14 support

The maximum supported version for TimescaleDB is now 2.14.X.

Templates New templates

New templates are available:

- AWS ELB Application Load Balancer by HTTP
- Check Point Next Generation Firewall by SNMP
- MSSQL by Zabbix agent 2

You can get these templates:

- In *Data collection → Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from the Zabbix Git repository or find them in the *zabbix/templates* directory of the latest Zabbix version you have downloaded. Then, while in *Data collection → Templates*, you can import them manually into Zabbix.

19 What's new in Zabbix 6.4.14

Databases MariaDB 11.3 support

The maximum supported version for MariaDB is now 11.3.X.

Templates New templates

A new template is available:

- Oracle Cloud by HTTP, a master template that discovers various Oracle Cloud Infrastructure (OCI) services and resources.

You can get this template:

- In *Data collection → Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from the Zabbix Git repository or find them in the *zabbix/templates* directory of the latest Zabbix version you have downloaded. Then, while in *Data collection → Templates*, you can import them manually into Zabbix.

Updated templates

- FortiGate by SNMP template has been supplemented with metrics regarding VPN, high availability (HA), wireless termination points (WTPs), SD-WAN health checks, and HW sensors.
- MySQL by ODBC template has been supplemented with the items "MySQL: Get database" and "MySQL: Get replication". The LLD rules "Database discovery" and "Replication discovery" have been changed to the "Dependent item" type.
- Oracle by ODBC template has been supplemented with the items "Oracle: Get archive log", "Oracle: Get ASM disk groups", "Oracle: Get database", "Oracle: Get PDB", and "Oracle: Get tablespace". The LLD rules "Archive log discovery", "ASM disk groups discovery", "Database discovery", "PDB discovery", and "Tablespace discovery" have been changed to the "Dependent item" type.
- The VMware Hypervisor template within the VMware and VMware FQDN template sets has been supplemented with a new LLD rule, "Sensor discovery".

20 What's new in Zabbix 6.4.15

Frontend Frontend languages

Danish, Georgian, and Spanish languages are now enabled in the frontend.

Templates New templates

The AWS ELB template set has been supplemented with the template AWS ELB Network Load Balancer by HTTP.

You can get this template:

- In *Data collection → Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from the Zabbix Git repository or find them in the *zabbix/templates* directory of the latest Zabbix version you have downloaded. Then, while in *Data collection → Templates*, you can import them manually into Zabbix.

Updated templates

The OS templates (agent, SNMP, and Prometheus-based) have been given a mounted filesystem overhaul. Changes include item, trigger, graph, and dashboard updates.

Macros Macro support for email media types

When configuring an email media type, it is now possible to use macros in the username and password fields.

21 What's new in Zabbix 6.4.16

Items More secure JavaScript preprocessing

The JavaScript function atob now returns an array of 8-bit unsigned integers instead of a decoded string.

Frontend Preprocessing test result truncation

When testing preprocessing steps, test results are now truncated to a maximum size of 512KB when sent to the frontend. Note that data larger than 512KB is still processed fully by Zabbix server.

Templates New templates

A new template is available:

- Jira Data Center by JMX, a template for monitoring Jira Data Center health.

You can get this template:

- In *Data collection* → *Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from the Zabbix Git repository or find them in the *zabbix/templates* directory of the latest Zabbix version you have downloaded. Then, while in *Data collection* → *Templates*, you can import them manually into Zabbix.

22 What's new in Zabbix 6.4.17

Databases MySQL 8.4 support

The maximum supported version for MySQL is now 8.4.X.

MariaDB 11.4 support

The maximum supported version for MariaDB is now 11.4.X.

TimescaleDB 2.15 support

The maximum supported version for TimescaleDB is now 2.15.X.

Templates New templates

The set of Azure by HTTP templates has been supplemented with the Azure VM Scale Set by HTTP template.

You can get this template:

- In *Data collection* → *Templates* in new installations;
- If you are upgrading from previous versions, you can download new templates from the Zabbix Git repository or find them in the *zabbix/templates* directory of the latest Zabbix version you have downloaded. Then, while in *Data collection* → *Templates*, you can import them manually into Zabbix.

23 What's new in Zabbix 6.4.18

Databases MySQL 9.0 support

The maximum supported version for MySQL is now 9.0.X.

24 What's new in Zabbix 6.4.19

Databases TimescaleDB 2.16 support

The maximum supported version for TimescaleDB is now 2.16.X.

MariaDB 11.5 support

The maximum supported version for MariaDB is now 11.5.X.

Items Special characters supported in ODBC check user password

Special characters are now supported when specifying the ODBC check user password in the frontend.

25 What's new in Zabbix 6.4.20

TimescaleDB 2.17 support

The maximum supported version for TimescaleDB is now 2.17.X.

PostgreSQL 17 support

PostgreSQL 17 is now supported.

26 What's new in Zabbix 6.4.21

See breaking changes for this version.

2 Definitions

Overview In this section you can learn the meaning of some terms commonly used in Zabbix.

Definitions ***host***

- any physical or virtual device, application, service, or any other logically-related collection of monitored parameters.

host group

- a logical grouping of hosts. Host groups are used when assigning access rights to hosts for different user groups.

item

- a particular piece of data that you want to receive from a host, a metric of data.

value preprocessing

- a transformation of received metric value before saving it to the database.

trigger

- a logical expression that defines a problem threshold and is used to "evaluate" data received in items.

When received data are above the threshold, triggers go from 'Ok' into a 'Problem' state. When received data are below the threshold, triggers stay in/return to an 'Ok' state.

template

- a set of entities (items, triggers, graphs, low-level discovery rules, web scenarios) ready to be applied to one or several hosts.

The job of templates is to speed up the deployment of monitoring tasks on a host; also to make it easier to apply mass changes to monitoring tasks. Templates are linked directly to individual hosts.

template group

- a logical grouping of templates. Template groups are used when assigning access rights to templates for different user groups.

event

- a single occurrence of something that deserves attention such as a trigger changing state or a discovery/agent autoregistration taking place.

event tag

- a pre-defined marker for the event. It may be used in event correlation, permission granulation, etc.

event correlation

- a method of correlating problems to their resolution flexibly and precisely.

For example, you may define that a problem reported by one trigger may be resolved by another trigger, which may even use a different data collection method.

problem

- a trigger that is in "Problem" state.

problem update

- problem management options provided by Zabbix, such as adding comment, acknowledging, changing severity or closing manually.

action

- a predefined means of reacting to an event.

An action consists of operations (e.g. sending a notification) and conditions (*when* the operation is carried out)

escalation

- a custom scenario for executing operations within an action; a sequence of sending notifications/executing remote commands.

media

- a means of delivering notifications; delivery channel.

notification

- a message about some event sent to a user via the chosen media channel.

remote command

- a pre-defined command that is automatically executed on a monitored host upon some condition.

web scenario

- one or several HTTP requests to check the availability of a web site.

frontend

- the web interface provided with Zabbix.

dashboard

- customizable section of the web interface displaying summaries and visualizations of important information in visual units called widgets.

widget

- visual unit displaying information of a certain kind and source (a summary, a map, a graph, the clock, etc.), used in the dashboard.

Zabbix API

- Zabbix API allows you to use the JSON RPC protocol to create, update and fetch Zabbix objects (like hosts, items, graphs and others) or perform any other custom tasks.

Zabbix server

- a central process of Zabbix software that performs monitoring, interacts with Zabbix proxies and agents, calculates triggers, sends notifications; a central repository of data.

Zabbix proxy

- a process that may collect data on behalf of Zabbix server, taking some processing load from the server.

Zabbix agent

- *a process deployed on monitoring targets to actively monitor local resources and applications.*

Zabbix agent 2

- *a new generation of Zabbix agent to actively monitor local resources and applications, allowing to use custom plugins for monitoring.*

> **Attention:**
> Because Zabbix agent 2 shares much functionality with Zabbix agent, the term "Zabbix agent" in documentation stands for both - Zabbix agent and Zabbix agent 2, if the functional behavior is the same. Zabbix agent 2 is only specifically named where its functionality differs.

encryption

- *support of encrypted communications between Zabbix components (server, proxy, agent, zabbix_sender and zabbix_get utilities)* using Transport Layer Security (TLS) protocol.

agent autoregistration

- *automated process whereby a Zabbix agent itself is registered* as a host and started to monitor.

network discovery

- *automated discovery of network devices*.

low-level discovery

- *automated discovery of low-level entities on a particular device* (e.g. file systems, network interfaces, etc).

low-level discovery rule

- *set of definitions for automated discovery of low-level entities* on a device.

item prototype

- *a metric with certain parameters as variables, ready for low-level discovery*. After low-level discovery the variables are automatically substituted with the real discovered parameters and the metric automatically starts gathering data.

trigger prototype

- *a trigger with certain parameters as variables, ready for low-level discovery*. After low-level discovery the variables are automatically substituted with the real discovered parameters and the trigger automatically starts evaluating data.

Prototypes of some other Zabbix entities are also in use in low-level discovery - graph prototypes, host prototypes, host group prototypes.

3 Zabbix processes

Please use the sidebar to access content in the Zabbix process section.

1 Server

Overview

Zabbix server is the central process of Zabbix software.

The server performs the polling and trapping of data, it calculates triggers, sends notifications to users. It is the central component to which Zabbix agents and proxies report data on availability and integrity of systems. The server can itself remotely check networked services (such as web servers and mail servers) using simple service checks.

The server is the central repository in which all configuration, statistical and operational data is stored, and it is the entity in Zabbix that will actively alert administrators when problems arise in any of the monitored systems.

The functioning of a basic Zabbix server is broken into three distinct components; they are: Zabbix server, web frontend and database storage.

All of the configuration information for Zabbix is stored in the database, which both the server and the web frontend interact with. For example, when you create a new item using the web frontend (or API) it is added to the items table in the database. Then, about once a minute Zabbix server will query the items table for a list of the items which are active that is then stored in a cache

within the Zabbix server. This is why it can take up to two minutes for any changes made in Zabbix frontend to show up in the latest data section.

Running server

If installed as package

Zabbix server runs as a daemon process. The server can be started by executing:

```
shell> systemctl start zabbix-server
```

This will work on most of GNU/Linux systems. On other systems you may need to run:

```
shell> /etc/init.d/zabbix-server start
```

Similarly, for stopping/restarting/viewing status, use the following commands:

```
shell> systemctl stop zabbix-server
shell> systemctl restart zabbix-server
shell> systemctl status zabbix-server
```

Start up manually

If the above does not work you have to start it manually. Find the path to the zabbix_server binary and execute:

```
shell> zabbix_server
```

You can use the following command line parameters with Zabbix server:

```
-c --config <file>              path to the configuration file (default is /usr/local/etc/zabbix_server.co
-f --foreground                 run Zabbix server in foreground
-R --runtime-control <option>   perform administrative functions
-h --help                       give this help
-V --version                    display version number
```

Examples of running Zabbix server with command line parameters:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf
shell> zabbix_server --help
shell> zabbix_server -V
```

Runtime control

Runtime control options:

Option	Description	Target
config_cache_reload	Reload configuration cache. Ignored if cache is being currently loaded.	
diaginfo[=**section**]	Gather diagnostic information in the server log file.	**historycache** - history cache statistics **valuecache** - value cache statistics **preprocessing** - preprocessing manager statistics **alerting** - alert manager statistics **lld** - LLD manager statistics **locks** - list of mutexes (is empty on *BSD* systems) **connector** - statistics for connectors with the largest queue
ha_status	Log high availability (HA) cluster status.	
ha_remove_node=**target**	Remove the high availability (HA) node specified by its name or ID. Note that active/standby nodes cannot be removed.	**target** - name or ID of the node (can be obtained by running ha_status)
ha_set_failover_delay=**delay**	Set high availability (HA) failover delay. Time suffixes are supported, e.g. 10s, 1m.	
proxy_config_cache_reload[=**target**]	Reload proxy configuration cache.	**target** - comma-delimited list of proxy names If no target is specified, reload configuration for all proxies
secrets_reload	Reload secrets from Vault.	
service_cache_reload	Reload the service manager cache.	

Option	Description	Target
snmp_cache_reload	Reload SNMP cache, clear the SNMP properties (engine time, engine boots, engine id, credentials) for all hosts.	
housekeeper_execute	Start the housekeeping procedure. Ignored if the housekeeping procedure is currently in progress.	
trigger_housekeeper_execute	Start the trigger housekeeping procedure for services to remove problems caused by triggers that have since been deleted, including service problems generated by such problems (considered as resolved at the time of housekeeping). Note that, until the housekeeping procedure is started, problems caused by now-deleted triggers might still generate service problems and assign them to services. If your setup involves many service status calculation rules based on frequently discovered/undiscovered triggers, consider increasing the frequency of the trigger housekeeping procedure by adjusting the ProblemHousekeepingFrequency server configuration parameter. Ignored if the trigger housekeeping procedure is currently in progress.	
log_level_increase[=<**target**>]	Increase log level, affects all processes if target is not specified. Not supported on *BSD* systems.	**process type** - All processes of specified type (e.g., poller) See all server process types. **process type,N** - Process type and number (e.g., poller,3) **pid** - Process identifier (1 to 65535). For larger values specify target as 'process type,N'.
log_level_decrease[=<**target**>]	Decrease log level, affects all processes if target is not specified. Not supported on *BSD* systems.	
prof_enable[=<**target**>]	Enable profiling. Affects all processes if target is not specified. Enabled profiling provides details of all rwlocks/mutexes by function name.	**process type** - All processes of specified type (e.g. history syncer) Supported process types as profiling targets: alerter, alert manager, availability manager, configuration syncer, discoverer, escalator, history poller, history syncer, housekeeper, http poller, icmp pinger, ipmi manager, ipmi poller, java poller, lld manager, lld worker, odbc poller, poller, preprocessing manager, preprocessing worker, proxy poller, self-monitoring, service manager, snmp trapper, task manager, timer, trapper, unreachable poller, vmware collector **process type,N** - Process type and number (e.g., history syncer,1) **pid** - Process identifier (1 to 65535). For larger values specify target as 'process type,N'. **scope** - `rwlock`, `mutex`, `processing` can be used with the process type and number (e.g., history syncer,1,processing) or all processes of type (e.g., history syncer,rwlock)

Option	Description	Target
prof_disable[=<**target**>]	Disable profiling. Affects all processes if target is not specified.	**process type** - All processes of specified type (e.g. history syncer) Supported process types as profiling targets: see `prof_enable` **process type,N** - Process type and number (e.g., history syncer,1) **pid** - Process identifier (1 to 65535). For larger values specify target as 'process type,N'.

Example of using runtime control to reload the server configuration cache:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R config_cache_reload
```

Examples of using runtime control to reload the proxy configuration:

```
Reload configuration of all proxies:
shell> zabbix_server -R proxy_config_cache_reload

Reload configuration of Proxy1 and Proxy2:
shell> zabbix_server -R proxy_config_cache_reload=Proxy1,Proxy2
```

Examples of using runtime control to gather diagnostic information:

```
Gather all available diagnostic information in the server log file:
shell> zabbix_server -R diaginfo

Gather history cache statistics in the server log file:
shell> zabbix_server -R diaginfo=historycache
```

Example of using runtime control to reload the SNMP cache:

```
shell> zabbix_server -R snmp_cache_reload
```

Example of using runtime control to trigger execution of housekeeper:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R housekeeper_execute
```

Examples of using runtime control to change log level:

```
Increase log level of all processes:
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase

Increase log level of second poller process:
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase=poller,2

Increase log level of process with PID 1234:
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_increase=1234

Decrease log level of all http poller processes:
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R log_level_decrease="http poller"
```

Example of setting the HA failover delay to the minimum of 10 seconds:

```
shell> zabbix_server -R ha_set_failover_delay=10s
```

Process user

Zabbix server is designed to run as a non-root user. It will run as whatever non-root user it is started as. So you can run server as any non-root user without any issues.

If you will try to run it as 'root', it will switch to a hardcoded 'zabbix' user, which must be present on your system. You can only run server as 'root' if you modify the 'AllowRoot' parameter in the server configuration file accordingly.

If Zabbix server and agent are run on the same machine it is recommended to use a different user for running the server than for running the agent. Otherwise, if both are run as the same user, the agent can access the server configuration file and any Admin level user in Zabbix can quite easily retrieve, for example, the database password.

Configuration file

See the configuration file options for details on configuring zabbix_server.

Start-up scripts

The scripts are used to automatically start/stop Zabbix processes during system's start-up/shutdown. The scripts are located under directory misc/init.d.

Server process types

- `alert manager` - alert queue manager
- `alert syncer` - alert DB writer
- `alerter` - process for sending notifications
- `availability manager` - process for host availability updates
- `configuration syncer` - process for managing in-memory cache of configuration data
- `connector manager` - manager process for connectors
- `connector worker` - process for handling requests from the connector manager
- `discoverer` - process for discovery of devices
- `escalator` - process for escalation of actions
- `ha manager` - process for managing high availability
- `history poller` - process for handling calculated checks requiring a database connection
- `history syncer` - history DB writer
- `housekeeper` - process for removal of old historical data
- `http poller` - web monitoring poller
- `icmp pinger` - poller for icmpping checks
- `ipmi manager` - IPMI poller manager
- `ipmi poller` - poller for IPMI checks
- `java poller` - poller for Java checks
- `lld manager` - manager process of low-level discovery tasks
- `lld worker` - worker process of low-level discovery tasks
- `odbc poller` - poller for ODBC checks
- `poller` - normal poller for passive checks
- `preprocessing manager` - manager of preprocessing tasks
- `preprocessing worker` - process for data preprocessing
- `proxy poller` - poller for passive proxies
- `report manager`- manager of scheduled report generation tasks
- `report writer` - process for generating scheduled reports
- `self-monitoring` - process for collecting internal server statistics
- `service manager` - process for managing services by receiving information about problems, problem tags, and problem recovery from history syncer, task manager, and alert manager
- `snmp trapper` - trapper for SNMP traps
- `task manager` - process for remote execution of tasks requested by other components (e.g., close problem, acknowledge problem, check item value now, remote command functionality)
- `timer` - timer for processing maintenances
- `trapper` - trapper for active checks, traps, proxy communication
- `trigger housekeeper` - process for removing problems generated by triggers that have been deleted
- `unreachable poller` - poller for unreachable devices
- `vmware collector` - VMware data collector responsible for data gathering from VMware services

The server log file can be used to observe these process types.

Various types of Zabbix server processes can be monitored using the **zabbix[process,<type>,<mode>,<state>]** internal item.

Supported platforms

Due to the security requirements and mission-critical nature of server operation, UNIX is the only operating system that can consistently deliver the necessary performance, fault tolerance and resilience. Zabbix operates on market leading versions.

Zabbix server is tested on the following platforms:

- Linux
- Solaris
- AIX
- HP-UX
- Mac OS X
- FreeBSD
- OpenBSD
- NetBSD
- SCO Open Server
- Tru64/OSF1

Note:
Zabbix may work on other Unix-like operating systems as well.

Locale

Note that the server requires a UTF-8 locale so that some textual items can be interpreted correctly. Most modern Unix-like systems have a UTF-8 locale as default, however, there are some systems where that may need to be set specifically.

1 High availability

Overview

High availability (HA) is typically required in critical infrastructures that can afford virtually no downtime. So for any service that may fail there must be a failover option in place to take over should the current service fail.

Zabbix offers a **native** high-availability solution that is easy to set up and does not require any previous HA expertise. Native Zabbix HA may be useful for an extra layer of protection against software/hardware failures of Zabbix server or to have less downtime due to maintenance.

In the Zabbix high availability mode multiple Zabbix servers are run as nodes in a cluster. While one Zabbix server in the cluster is active, others are on standby, ready to take over if necessary.



Switching to Zabbix HA is non-committal. You may switch back to standalone operation at any point.

See also: Implementation details

Enabling high availability

Starting Zabbix server as cluster node

Two parameters are required in the server configuration to start a Zabbix server as cluster node:

- **HANodeName** parameter must be specified for each Zabbix server that will be an HA cluster node.

This is a unique node identifier (e.g. `zabbix-node-01`) that the server will be referred to in agent and proxy configurations. If you do not specify HANodeName, then the server will be started in standalone mode.

- **NodeAddress** parameter must be specified for each node.

The NodeAddress parameter (address:port) will be used by Zabbix frontend to connect to the active server node. NodeAddress must match the IP or FQDN name of the respective Zabbix server.

Restart all Zabbix servers after making changes to the configuration files. They will now be started as cluster nodes. The new status of the servers can be seen in *Reports* → *System information* and also by running:

```
zabbix_server -R ha_status
```

This runtime command will log the current HA cluster status into the Zabbix server log (and to stdout):

```
Failover delay: 60 seconds
Cluster status:
   #  ID                        Name                    Address                  Status      Last Access
   1. ckzxxqg7u0001lsropenyzh3m zabbix-node-01          64.227.66.193:10051      standby     0s
   2. ckzxyqo1k00013frpq539e1jp zabbix-node-02          64.227.74.25:10051       active      3s
```

Preparing frontend

Make sure that Zabbix server address:port is **not defined** in the frontend configuration (found in `conf/zabbix.conf.php` of the frontend files directory).

```
// Uncomment and set to desired values to override Zabbix hostname/IP and port.
// $ZBX_SERVER                  = '';
// $ZBX_SERVER_PORT             = '';
```

Zabbix frontend will autodetect the active node by reading settings from the nodes table in Zabbix database. Node address of the active node will be used as the Zabbix server address.

Proxy configuration

HA cluster nodes (servers) must be listed in the configuration of either passive or active Zabbix proxy.

For a passive proxy, the node names must be listed in the Server parameter of the proxy, separated by a **comma**.

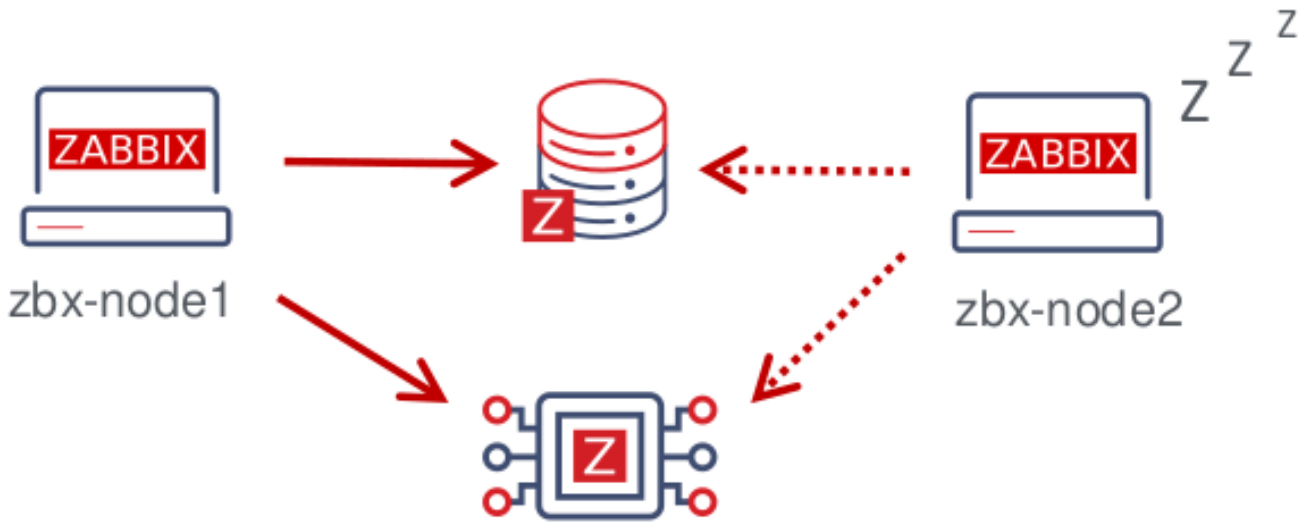
```
Server=zabbix-node-01,zabbix-node-02
```

For an active proxy, the node names must be listed in the Server parameter of the proxy, separated by a **semicolon**.

```
Server=zabbix-node-01;zabbix-node-02
```

Agent configuration

HA cluster nodes (servers) must be listed in the configuration of Zabbix agent or Zabbix agent 2.



To enable passive checks, the node names must be listed in the Server parameter, separated by a **comma**.

```
Server=zabbix-node-01,zabbix-node-02
```

To enable active checks, the node names must be listed in the ServerActive parameter. Note that for active checks the nodes must be separated by a comma from any other servers, while the nodes themselves must be separated by a **semicolon**, e.g.:

```
ServerActive=zabbix-node-01;zabbix-node-02
```

Failover to standby node

Zabbix will fail over to another node automatically if the active node stops. There must be at least one node in standby status for the failover to happen.

How fast will the failover be? All nodes update their last access time (and status, if it is changed) every 5 seconds. So:

- If the active node shuts down and manages to report its status as "stopped", another node will take over within **5 seconds**.
- If the active node shuts down/becomes unavailable without being able to update its status, standby nodes will wait for the **failover delay** + 5 seconds to take over

The failover delay is configurable, with the supported range between 10 seconds and 15 minutes (one minute by default). To change the failover delay, you may run:

```
zabbix_server -R ha_set_failover_delay=5m
```

Managing HA cluster

The current status of the HA cluster can be managed using the dedicated runtime control options:

- `ha_status` - log HA cluster status in the Zabbix server log (and to stdout)
- `ha_remove_node=target` - remove an HA node identified by its <target> - name or ID of the node (name/ID can be obtained from the output of running ha_status), e.g.:

```
zabbix_server -R ha_remove_node=zabbix-node-02
```

Note that active/standby nodes cannot be removed.

- `ha_set_failover_delay=delay` - set HA failover delay (between 10 seconds and 15 minutes; time suffixes are supported, e.g. 10s, 1m)

Node status can be monitored:

- in *Reports* → *System information*
- in the *System information* dashboard widget
- using the `ha_status` runtime control option of the server (see above).

The `zabbix[cluster,discovery,nodes]` internal item can be used for node discovery, as it returns a JSON with the high-availability node information.

Disabling HA cluster

To disable a high availability cluster:

- make backup copies of configuration files
- stop standby nodes
- remove the HANodeName parameter from the active primary server
- restart the primary server (it will start in standalone mode)

Upgrading HA cluster

To perform a major version upgrade for the HA nodes:

- stop all nodes;
- create a full database backup;
- if the database uses replication make sure that all nodes are in sync and have no issues. Do not upgrade if replication is broken.
- select a single node that will perform database upgrade, change its configuration to standalone mode by commenting out HANodeName and upgrade it;
- make sure that database upgrade is fully completed (*System information* should display that Zabbix server is running);
- restart the node in HA mode;
- upgrade and start the rest of nodes (it is not required to change them to standalone mode as the database is already upgraded at this point).

In a minor version upgrade it is sufficient to upgrade the first node, make sure it has upgraded and running, and then start upgrade on the next node.

Implementation details

The high availability (HA) cluster is an opt-in solution and it is supported for Zabbix server. The native HA solution is designed to be simple in use, it will work across sites and does not have specific requirements for the databases that Zabbix recognizes. Users are free to use the native Zabbix HA solution, or a third-party HA solution, depending on what best suits the high availability requirements in their environment.

The solution consists of multiple zabbix_server instances or nodes. Every node:

- is configured separately
- uses the same database
- may have several modes: active, standby, unavailable, stopped

Only one node can be active (working) at a time. A standby node runs only one process - the HA manager. A standby node does no data collection, processing or other regular server activities; they do not listen on ports; they have minimum database connections.

Both active and standby nodes update their last access time every 5 seconds. Each standby node monitors the last access time of the active node. If the last access time of the active node is over 'failover delay' seconds, the standby node switches itself to be the active node and assigns 'unavailable' status to the previously active node.

The active node monitors its own database connectivity - if it is lost for more than `failover delay`−5 seconds, it must stop all processing and switch to standby mode. The active node also monitors the status of the standby nodes - if the last access time of a standby node is over 'failover delay' seconds, the standby node is assigned the 'unavailable' status.

The nodes are designed to be compatible across minor Zabbix versions.

2 Agent

Overview

Zabbix agent is deployed on a monitoring target to actively monitor local resources and applications (hard drives, memory, processor statistics, etc.).

The agent gathers operational information locally and reports data to Zabbix server for further processing. In case of failures (such as a hard disk running full or a crashed service process), Zabbix server can actively alert the administrators of the particular machine that reported the failure.

Zabbix agents are highly efficient because of the use of native system calls for gathering statistical information.

Passive and active checks

Zabbix agents can perform passive and active checks.

In a passive check the agent responds to a data request. Zabbix server (or proxy) asks for data, for example, CPU load, and Zabbix agent sends back the result.

Active checks require more complex processing. The agent must first retrieve a list of items from Zabbix server for independent processing. Then it will periodically send new values to the server.

Whether to perform passive or active checks is configured by selecting the respective monitoring item type. Zabbix agent processes items of type 'Zabbix agent' or 'Zabbix agent (active)'.

Supported platforms

Pre-compiled Zabbix agent binaries are available for the supported platforms:

- Windows (all desktop and server versions since XP)
- Linux (also available in distribution packages)
- macOS
- IBM AIX
- FreeBSD
- OpenBSD
- Solaris

It is also possible to download legacy Zabbix agent binaries for NetBSD and HP-UX, and those are compatible with current Zabbix server/proxy version.

Agent on UNIX-like systems

Zabbix agent on UNIX-like systems is run on the host being monitored.

Installation

See the package installation section for instructions on how to install Zabbix agent as package.

Alternatively see instructions for manual installation if you do not want to use packages.

> **Attention:**
> In general, 32bit Zabbix agents will work on 64bit systems, but may fail in some cases.

If installed as package

Zabbix agent runs as a daemon process. The agent can be started by executing:

```
shell> systemctl start zabbix-agent
```

This will work on most of GNU/Linux systems. On other systems you may need to run:

```
shell> /etc/init.d/zabbix-agent start
```

Similarly, for stopping/restarting/viewing status of Zabbix agent, use the following commands:

```
shell> systemctl stop zabbix-agent
shell> systemctl restart zabbix-agent
shell> systemctl status zabbix-agent
```

Start up manually

If the above does not work you have to start it manually. Find the path to the zabbix_agentd binary and execute:

```
shell> zabbix_agentd
```

Agent on Windows systems

Zabbix agent on Windows runs as a Windows service.

Preparation

Zabbix agent is distributed as a zip archive. After you download the archive you need to unpack it. Choose any folder to store Zabbix agent and the configuration file, e. g.

```
C:\zabbix
```

Copy bin\zabbix_agentd.exe and conf\zabbix_agentd.conf files to c:\zabbix.

Edit the c:\zabbix\zabbix_agentd.conf file to your needs, making sure to specify a correct "Hostname" parameter.

Installation

After this is done use the following command to install Zabbix agent as Windows service:

```
C:\> c:\zabbix\zabbix_agentd.exe -c c:\zabbix\zabbix_agentd.conf -i
```

Now you should be able to configure "Zabbix agent" service normally as any other Windows service.

See more details on installing and running Zabbix agent on Windows.

Other agent options

It is possible to run multiple instances of the agent on a host. A single instance can use the default configuration file or a configuration file specified in the command line. In case of multiple instances each agent instance must have its own configuration file (one of the instances can use the default configuration file).

The following command line parameters can be used with Zabbix agent:

Parameter	Description
UNIX and Windows agent	
-c --config <config-file>	Path to the configuration file. You may use this option to specify a configuration file that is not the default one. On UNIX, default is /usr/local/etc/zabbix_agentd.conf or as set by compile-time variables *--sysconfdir* or *--prefix* On Windows, default is c:\zabbix_agentd.conf
-p --print	Print known items and exit. *Note*: To return user parameter results as well, you must specify the configuration file (if it is not in the default location).
-t --test <item key>	Test specified item and exit. *Note*: To return user parameter results as well, you must specify the configuration file (if it is not in the default location).
-h --help	Display help information
-V --version	Display version number
UNIX agent only	
-R --runtime-control <option>	Perform administrative functions. See runtime control.
Windows agent only	
-m --multiple-agents	Use multiple agent instances (with -i,-d,-s,-x functions). To distinguish service names of instances, each service name will include the Hostname value from the specified configuration file.
Windows agent only (functions)	
-i --install	Install Zabbix Windows agent as service
-d --uninstall	Uninstall Zabbix Windows agent service
-s --start	Start Zabbix Windows agent service
-x --stop	Stop Zabbix Windows agent service

Specific **examples** of using command line parameters:

- printing all built-in agent items with values
- testing a user parameter with "mysql.ping" key defined in the specified configuration file
- installing a "Zabbix Agent" service for Windows using the default path to configuration file c:\zabbix_agentd.conf
- installing a "Zabbix Agent [Hostname]" service for Windows using the configuration file zabbix_agentd.conf located in the same folder as agent executable and make the service name unique by extending it by Hostname value from the config file

```
shell> zabbix_agentd --print
shell> zabbix_agentd -t "mysql.ping" -c /etc/zabbix/zabbix_agentd.conf
shell> zabbix_agentd.exe -i
shell> zabbix_agentd.exe -i -m -c zabbix_agentd.conf
```

Runtime control

With runtime control options you may change the log level of agent processes.

Option	Description	Target
log_level_increase[=<target>]	Increase log level. If target is not specified, all processes are affected.	Target can be specified as: **process type** - all processes of specified type (e.g., listener) See all agent process types. **process type,N** - process type and number (e.g., listener,3) **pid** - process identifier (1 to 65535). For larger values specify target as 'process-type,N'.
log_level_decrease[=<target>]	Decrease log level. If target is not specified, all processes are affected.	
userparameter_reload	Reload values of the *UserParameter* and *Include* options from the current configuration file.	

Examples:

- increasing log level of all processes
- increasing log level of the third listener process
- increasing log level of process with PID 1234
- decreasing log level of all active check processes

```
shell> zabbix_agentd -R log_level_increase
shell> zabbix_agentd -R log_level_increase=listener,3
shell> zabbix_agentd -R log_level_increase=1234
shell> zabbix_agentd -R log_level_decrease="active checks"
```

Note:
Runtime control is not supported on OpenBSD, NetBSD and Windows.

Agent process types

- `active checks` - process for performing active checks
- `collector` - process for data collection
- `listener` - process for listening to passive checks

The agent log file can be used to observe these process types.

Process user

Zabbix agent on UNIX is designed to run as a non-root user. It will run as whatever non-root user it is started as. So you can run agent as any non-root user without any issues.

If you will try to run it as 'root', it will switch to a hardcoded 'zabbix' user, which must be present on your system. You can only run agent as 'root' if you modify the 'AllowRoot' parameter in the agent configuration file accordingly.

Configuration file

For details on configuring Zabbix agent see the configuration file options for zabbix_agentd or Windows agent.

Locale

Note that the agent requires a UTF-8 locale so that some textual agent items can return the expected content. Most modern Unix-like systems have a UTF-8 locale as default, however, there are some systems where that may need to be set specifically.

Exit code

Before version 2.2 Zabbix agent returned 0 in case of successful exit and 255 in case of failure. Starting from version 2.2 and higher Zabbix agent returns 0 in case of successful exit and 1 in case of failure.

3 Agent 2

Overview

Zabbix agent 2 is a new generation of Zabbix agent and may be used in place of Zabbix agent. Zabbix agent 2 has been developed to:

- reduce the number of TCP connections
- provide improved concurrency of checks
- be easily extendible with plugins. A plugin should be able to:
  - provide trivial checks consisting of only a few simple lines of code
  - provide complex checks consisting of long-running scripts and standalone data gathering with periodic sending back of the data
- be a drop-in replacement for Zabbix agent (in that it supports all the previous functionality)

Agent 2 is written in Go programming language (with some C code of Zabbix agent reused). A configured Go environment with a currently supported Go version is required for building Zabbix agent 2.

Agent 2 does not have built-in daemonization support on Linux; it can be run as a Windows service.

Passive and active checks

Passive checks work similarly to Zabbix agent. Active checks support scheduled/flexible intervals and check concurrency within one active server.

> **Note:**
> By default, after a restart, Zabbix agent 2 will schedule the first data collection for active checks at a conditionally random time within the item's update interval to prevent spikes in resource usage. To perform active checks that do not have *Scheduling* update interval immediately after the agent restart, set `ForceActiveChecksOnStart` parameter (global-level) or `Plugins.<Plugin name>.System.ForceActiveChecksOnStart` (affects only specific plugin checks) in the configuration file. Plugin-level parameter, if set, will override the global parameter. Forcing active checks on start is supported since Zabbix 6.0.2.

Check concurrency

Checks from different plugins can be executed concurrently. The number of concurrent checks within one plugin is limited by the plugin capacity setting. Each plugin may have a hardcoded capacity setting (100 being default) that can be lowered using the `Plugins.<PluginName>.System.Capacity=N` setting in the *Plugins* configuration parameter. Former name of this parameter `Plugins.<PluginName>.Capacity` is still supported, but has been deprecated in Zabbix 6.0.

Supported platforms

Zabbix agent 2 is supported on the following platforms:

- Windows (all desktop and server versions since Windows 10/Server 2016) - available as a pre-compiled binary or in Zabbix sources
- Linux - available in distribution packages or Zabbix sources

Installation

To install Zabbix agent 2, the following options are available:

Windows:

- from a pre-compiled binary - download the binary and follow the instructions on the Windows agent installation from MSI page
- from sources - see Building Zabbix agent 2 on Windows

Linux:

- from distribution packages - follow the instructions on the Zabbix packages page, available by choosing your distribution and the Agent 2 component
- from sources - see Installation from sources; note that you must configure the sources by specifying the `--enable-agent2` configuration option

Note:
Zabbix agent 2 monitoring capabilities can be extended with plugins. While built-in plugins are available out-of-the-box, loadable plugins must be installed separately. For more information, see Plugins.

Options

The following command line parameters can be used with Zabbix agent 2:

Parameter	Description
-c --config <config-file>	Path to the configuration file. You may use this option to specify a configuration file that is not the default one. On UNIX, default is /usr/local/etc/zabbix_agent2.conf or as set by compile-time variables *--sysconfdir* or *--prefix*
-f --foreground	Run Zabbix agent in foreground (default: true).
-p --print	Print known items and exit. *Note*: To return user parameter results as well, you must specify the configuration file (if it is not in the default location).
-t --test <item key>	Test specified item and exit. *Note*: To return user parameter results as well, you must specify the configuration file (if it is not in the default location).
-h --help	Print help information and exit.
-v --verbose	Print debugging information. Use this option with -p and -t options.
-V --version	Print agent version and license information.
-R --runtime-control <option>	Perform administrative functions. See runtime control.

Specific **examples** of using command line parameters:

- print all built-in agent items with values
- test a user parameter with ”mysql.ping” key defined in the specified configuration file

  shell> zabbix_agent2 --print shell> zabbix_agent2 -t ”mysql.ping” -c /etc/zabbix/zabbix_agentd.conf

Runtime control

Runtime control provides some options for remote control.

Option	Description
log_level_increase	Increase log level.
log_level_decrease	Decrease log level.
metrics	List available metrics.
version	Display agent version.
userparameter_reload	Reload values of the *UserParameter* and *Include* options from the current configuration file.
help	Display help information on runtime control.

Examples:

- increasing log level for agent 2
- print runtime control options

```
shell> zabbix_agent2 -R log_level_increase
shell> zabbix_agent2 -R help
```

Configuration file

The configuration parameters of agent 2 are mostly compatible with Zabbix agent with some exceptions.

New parameters	Description
ControlSocket	The runtime control socket path. Agent 2 uses a control socket for runtime commands.
EnablePersistentBuffer, *PersistentBufferFile*, *PersistentBufferPeriod*	These parameters are used to configure persistent storage on agent 2 for active items.

New parameters	Description
ForceActiveChecksOnStart	Determines whether the agent should perform active checks immediately after restart or spread evenly over time.
Plugins	Plugins may have their own parameters, in the format `Plugins.<Plugin name>.<Parameter>=<value>`. A common plugin parameter is *System.Capacity*, setting the limit of checks that can be executed at the same time.
StatusPort	The port agent 2 will be listening on for HTTP status request and display of a list of configured plugins and some internal parameters
Dropped parameters	**Description**
AllowRoot, *User*	Not supported because daemonization is not supported.
LoadModule, *LoadModulePath*	Loadable modules are not supported.
StartAgents	This parameter was used in Zabbix agent to increase passive check concurrency or disable them. In Agent 2, the concurrency is configured at a plugin level and can be limited by a capacity setting. Whereas disabling passive checks is not currently supported.
HostInterface, *HostInterfaceItem*	Not yet supported.

For more details see the configuration file options for zabbix_agent2.

Exit codes

Starting from version 4.4.8 Zabbix agent 2 can also be compiled with older OpenSSL versions (1.0.1, 1.0.2).

In this case Zabbix provides mutexes for locking in OpenSSL. If a mutex lock or unlock fails then an error message is printed to the standard error stream (STDERR) and Agent 2 exits with return code 2 or 3, respectively.

4 Proxy

Overview

Zabbix proxy is a process that may collect monitoring data from one or more monitored devices and send the information to the Zabbix server, essentially working on behalf of the server. All collected data is buffered locally and then transferred to the Zabbix server the proxy belongs to.

Deploying a proxy is optional, but may be very beneficial to distribute the load of a single Zabbix server. If only proxies collect data, processing on the server becomes less CPU and disk I/O hungry.

A Zabbix proxy is the ideal solution for centralized monitoring of remote locations, branches and networks with no local administrators.

Zabbix proxy requires a separate database.

Attention:
Note that databases supported with Zabbix proxy are SQLite, MySQL and PostgreSQL. Using Oracle is at your own risk and may contain some limitations as, for example, in return values of low-level discovery rules.

See also: Using proxies in a distributed environment

Running proxy

If installed as package

Zabbix proxy runs as a daemon process. The proxy can be started by executing:

```
shell> systemctl start zabbix-proxy
```

This will work on most of GNU/Linux systems. On other systems you may need to run:

```
shell> /etc/init.d/zabbix-proxy start
```

Similarly, for stopping/restarting/viewing status of Zabbix proxy, use the following commands:

```
shell> systemctl stop zabbix-proxy
shell> systemctl restart zabbix-proxy
shell> systemctl status zabbix-proxy
```

Start up manually

If the above does not work you have to start it manually. Find the path to the zabbix_proxy binary and execute:

```
shell> zabbix_proxy
```

You can use the following command line parameters with Zabbix proxy:

```
-c --config <file>              path to the configuration file
-f --foreground                 run Zabbix proxy in foreground
-R --runtime-control <option>   perform administrative functions
-h --help                       give this help
-V --version                    display version number
```

Examples of running Zabbix proxy with command line parameters:

```
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf
shell> zabbix_proxy --help
shell> zabbix_proxy -V
```

Runtime control

Runtime control options:

Option	Description	Target
config_cache_reload	Reload configuration cache. Ignored if cache is being currently loaded. Active Zabbix proxy will connect to the Zabbix server and request configuration data. Passive Zabbix proxy will request configuration data from Zabbix server the next time when the server connects to the proxy.	
diaginfo[=<**section**>]	Gather diagnostic information in the proxy log file.	**historycache** - history cache statistics **preprocessing** - preprocessing manager statistics **locks** - list of mutexes (is empty on *BSD* systems)
snmp_cache_reload	Reload SNMP cache, clear the SNMP properties (engine time, engine boots, engine id, credentials) for all hosts.	
housekeeper_execute	Start the housekeeping procedure. Ignored if the housekeeping procedure is currently in progress.	
log_level_increase[=<**target**>]	Increase log level, affects all processes if target is not specified. Not supported on *BSD* systems.	**process type** - All processes of specified type (e.g., poller) See all proxy process types. **process type,N** - Process type and number (e.g., poller,3) **pid** - Process identifier (1 to 65535). For larger values specify target as 'process type,N'.
log_level_decrease[=<**target**>]	Decrease log level, affects all processes if target is not specified. Not supported on *BSD* systems.	
prof_enable[=<**target**>]	Enable profiling. Affects all processes if target is not specified. Enabled profiling provides details of all rwlocks/mutexes by function name.	**process type** - All processes of specified type (e.g., history syncer) See all proxy process types. **process type,N** - Process type and number (e.g., history syncer,1) **pid** - Process identifier (1 to 65535). For larger values specify target as 'process type,N'. **scope** - `rwlock`, `mutex`, `processing` can be used with the process type and number (e.g., history syncer,1,processing) or all processes of type (e.g., history syncer,rwlock)

Option	Description	Target
prof_disable[=<**target**>]	Disable profiling. Affects all processes if target is not specified.	**process type** - All processes of specified type (e.g., history syncer) See all proxy process types. **process type,N** - Process type and number (e.g., history syncer,1) **pid** - Process identifier (1 to 65535). For larger values specify target as 'process type,N'.

Example of using runtime control to reload the proxy configuration cache:

```
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R config_cache_reload
```

Examples of using runtime control to gather diagnostic information:

```
Gather all available diagnostic information in the proxy log file:
shell> zabbix_proxy -R diaginfo

Gather history cache statistics in the proxy log file:
shell> zabbix_proxy -R diaginfo=historycache
```

Example of using runtime control to reload the SNMP cache:

```
shell> zabbix_proxy -R snmp_cache_reload
```

Example of using runtime control to trigger execution of housekeeper

```
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R housekeeper_execute
```

Examples of using runtime control to change log level:

```
Increase log level of all processes:
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase

Increase log level of second poller process:
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase=poller,2

Increase log level of process with PID 1234:
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_increase=1234

Decrease log level of all http poller processes:
shell> zabbix_proxy -c /usr/local/etc/zabbix_proxy.conf -R log_level_decrease="http poller"
```

Process user

Zabbix proxy is designed to run as a non-root user. It will run as whatever non-root user it is started as. So you can run proxy as any non-root user without any issues.

If you will try to run it as 'root', it will switch to a hardcoded 'zabbix' user, which must be present on your system. You can only run proxy as 'root' if you modify the 'AllowRoot' parameter in the proxy configuration file accordingly.

Configuration file

See the configuration file options for details on configuring zabbix_proxy.

Proxy process types

- `availability manager` - process for host availability updates
- `configuration syncer` - process for managing in-memory cache of configuration data
- `data sender` - proxy data sender
- `discoverer` - process for discovery of devices
- `history syncer` - history DB writer
- `housekeeper` - process for removal of old historical data
- `http poller` - web monitoring poller
- `icmp pinger` - poller for icmpping checks
- `ipmi manager` - IPMI poller manager
- `ipmi poller` - poller for IPMI checks
- `java poller` - poller for Java checks
- `odbc poller` - poller for ODBC checks
- `poller` - normal poller for passive checks

- `preprocessing manager` - manager of preprocessing tasks
- `preprocessing worker` - process for data preprocessing
- `self-monitoring` - process for collecting internal server statistics
- `snmp trapper` - trapper for SNMP traps
- `task manager` - process for remote execution of tasks requested by other components (e.g. close problem, acknowledge problem, check item value now, remote command functionality)
- `trapper` - trapper for active checks, traps, proxy communication
- `unreachable poller` - poller for unreachable devices
- `vmware collector` - VMware data collector responsible for data gathering from VMware services

The proxy log file can be used to observe these process types.

Various types of Zabbix proxy processes can be monitored using the **zabbix[process,<type>,<mode>,<state>]** internal item.

Supported platforms

Zabbix proxy runs on the same list of supported platforms as Zabbix server.

Locale

Note that the proxy requires a UTF-8 locale so that some textual items can be interpreted correctly. Most modern Unix-like systems have a UTF-8 locale as default, however, there are some systems where that may need to be set specifically.

5 Java gateway

Overview

Native support for monitoring JMX applications exists in the form of a Zabbix daemon called "Zabbix Java gateway", available since Zabbix 2.0. Zabbix Java gateway is a daemon written in Java. To find out the value of a particular JMX counter on a host, Zabbix server queries Zabbix Java gateway, which uses the JMX management API to query the application of interest remotely. The application does not need any additional software installed, it just has to be started with `-Dcom.sun.management.jmxremote` option on the command line.

Java gateway accepts incoming connection from Zabbix server or proxy and can only be used as a "passive proxy". As opposed to Zabbix proxy, it may also be used from Zabbix proxy (Zabbix proxies cannot be chained). Access to each Java gateway is configured directly in Zabbix server or proxy configuration file, thus only one Java gateway may be configured per Zabbix server or Zabbix proxy. If a host will have items of type **JMX agent** and items of other type, only the **JMX agent** items will be passed to Java gateway for retrieval.

When an item has to be updated over Java gateway, Zabbix server or proxy will connect to the Java gateway and request the value, which Java gateway in turn retrieves and passes back to the server or proxy. As such, Java gateway does not cache any values.

Zabbix server or proxy has a specific type of processes that connect to Java gateway, controlled by the option **StartJavaPollers**. Internally, Java gateway starts multiple threads, controlled by the **START_POLLERS** option. On the server side, if a connection takes more than **Timeout** seconds, it will be terminated, but Java gateway might still be busy retrieving value from the JMX counter. To solve this, there is the **TIMEOUT** option in Java gateway that allows to set timeout for JMX network operations.

Zabbix server or proxy will try to pool requests to a single JMX target together as much as possible (affected by item intervals) and send them to the Java gateway in a single connection for better performance.

It is suggested to have **StartJavaPollers** less than or equal to **START_POLLERS**, otherwise there might be situations when no threads are available in the Java gateway to service incoming requests; in such a case Java gateway uses ThreadPoolExecutor.CallerRunsPolicy, meaning that the main thread will service the incoming request and will not accept any new requests temporarily.

If you are trying to monitor Wildfly-based Java applications with Zabbix Java gateway, please install the latest jboss-client.jar available on the Wildfly download page.

Getting Java gateway

You can install Java gateway either from the sources or packages downloaded from Zabbix website.

Using the links below you can access information how to get and run Zabbix Java gateway, how to configure Zabbix server (or Zabbix proxy) to use Zabbix Java gateway for JMX monitoring, and how to configure Zabbix items in Zabbix frontend that correspond to particular JMX counters.

Installation from	Instructions	Instructions
Sources	Installation	Setup

Installation from	Instructions	Instructions
RHEL packages	Installation	Setup
Debian/Ubuntu packages	Installation	Setup

1 Setup from sources

Overview

If installed from sources, the following information will help you in setting up Zabbix Java gateway.

Overview of files

If you obtained Java gateway from sources, you should have ended up with a collection of shell scripts, JAR and configuration files under $PREFIX/sbin/zabbix_java. The role of these files is summarized below.

```
bin/zabbix-java-gateway-$VERSION.jar
```

Java gateway JAR file itself.

```
lib/logback-core-1.5.16.jar
lib/logback-classic-1.5.16.jar
lib/slf4j-api-2.0.16.jar
lib/android-json-4.3_r3.1.jar
```

Dependencies of Java gateway: Logback, SLF4J, and Android JSON library.

```
lib/logback.xml
lib/logback-console.xml
```

Configuration files for Logback.

```
shutdown.sh
startup.sh
```

Convenience scripts for starting and stopping Java gateway.

```
settings.sh
```

Configuration file that is sourced by startup and shutdown scripts above.

Configuring and running Java gateway

By default, Java gateway listens on port 10052. If you plan on running Java gateway on a different port, you can specify that in settings.sh script. See the description of Java gateway configuration file for how to specify this and other options.

> **Warning:**
> Port 10052 is not IANA registered.

Once you are comfortable with the settings, you can start Java gateway by running the startup script:

```
$ ./startup.sh
```

Likewise, once you no longer need Java gateway, run the shutdown script to stop it:

```
$ ./shutdown.sh
```

Note that unlike server or proxy, Java gateway is lightweight and does not need a database.

Configuring server for use with Java gateway

With Java gateway up and running, you have to tell Zabbix server where to find Zabbix Java gateway. This is done by specifying JavaGateway and JavaGatewayPort parameters in the server configuration file. If the host on which JMX application is running is monitored by Zabbix proxy, then you specify the connection parameters in the proxy configuration file instead.

```
JavaGateway=192.168.3.14
JavaGatewayPort=10052
```

By default, server does not start any processes related to JMX monitoring. If you wish to use it, however, you have to specify the number of pre-forked instances of Java pollers. You do this in the same way you specify regular pollers and trappers.

```
StartJavaPollers=5
```

Do not forget to restart server or proxy, once you are done with configuring them.

Debugging Java gateway

In case there are any problems with Java gateway or an error message that you see about an item in the frontend is not descriptive enough, you might wish to take a look at Java gateway log file.

By default, Java gateway logs its activities into /tmp/zabbix_java.log file with log level "info". Sometimes that information is not enough and there is a need for information at log level "debug". In order to increase logging level, modify file lib/logback.xml and change the level attribute of <root> tag to "debug":

```
<root level="debug">
  <appender-ref ref="FILE" />
</root>
```

Note that unlike Zabbix server or Zabbix proxy, there is no need to restart Zabbix Java gateway after changing logback.xml file - changes in logback.xml will be picked up automatically. When you are done with debugging, you can return the logging level to "info".

If you wish to log to a different file or a completely different medium like database, adjust logback.xml file to meet your needs. See Logback Manual for more details.

Sometimes for debugging purposes it is useful to start Java gateway as a console application rather than a daemon. To do that, comment out PID_FILE variable in settings.sh. If PID_FILE is omitted, startup.sh script starts Java gateway as a console application and makes Logback use lib/logback-console.xml file instead, which not only logs to console, but has logging level "debug" enabled as well.

Finally, note that since Java gateway uses SLF4J for logging, you can replace Logback with the framework of your choice by placing an appropriate JAR file in lib directory. See SLF4J Manual for more details.

JMX monitoring

See JMX monitoring page for more details.

2 Setup from RHEL packages

Overview

If installed from RHEL packages, the following information will help you in setting up Zabbix Java gateway.

Configuring and running Java gateway

Configuration parameters of Zabbix Java gateway may be tuned in the file:

```
/etc/zabbix/zabbix_java_gateway.conf
```

For more details, see Zabbix Java gateway configuration parameters.

To start Zabbix Java gateway:

```
# systemctl restart zabbix-java-gateway
```

To automatically start Zabbix Java gateway on boot:

RHEL 7 and later:

```
# systemctl enable zabbix-java-gateway
```

RHEL prior to 7:

```
# chkconfig --level 12345 zabbix-java-gateway on
```

Configuring server for use with Java gateway

With Java gateway up and running, you have to tell Zabbix server where to find Zabbix Java gateway. This is done by specifying JavaGateway and JavaGatewayPort parameters in the server configuration file. If the host on which JMX application is running is monitored by Zabbix proxy, then you specify the connection parameters in the proxy configuration file instead.

```
JavaGateway=192.168.3.14
JavaGatewayPort=10052
```

By default, server does not start any processes related to JMX monitoring. If you wish to use it, however, you have to specify the number of pre-forked instances of Java pollers. You do this in the same way you specify regular pollers and trappers.

```
StartJavaPollers=5
```

Do not forget to restart server or proxy, once you are done with configuring them.

Debugging Java gateway

Zabbix Java gateway log file is:

```
/var/log/zabbix/zabbix_java_gateway.log
```

If you like to increase the logging, edit the file:

```
/etc/zabbix/zabbix_java_gateway_logback.xml
```

and change `level="info"` to ”debug” or even ”trace” (for deep troubleshooting):

```
<configuration scan="true" scanPeriod="15 seconds">
[...]
      <root level="info">
              <appender-ref ref="FILE" />
      </root>

</configuration>
```

JMX monitoring

See JMX monitoring page for more details.

3 Setup from Debian/Ubuntu packages

Overview

If installed from Debian/Ubuntu packages, the following information will help you in setting up Zabbix Java gateway.

Configuring and running Java gateway

Java gateway configuration may be tuned in the file:

```
/etc/zabbix/zabbix_java_gateway.conf
```

For more details, see Zabbix Java gateway configuration parameters.

To start Zabbix Java gateway:

```
# systemctl restart zabbix-java-gateway
```

To automatically start Zabbix Java gateway on boot:

```
# systemctl enable zabbix-java-gateway
```

Configuring server for use with Java gateway

With Java gateway up and running, you have to tell Zabbix server where to find Zabbix Java gateway. This is done by specifying JavaGateway and JavaGatewayPort parameters in the server configuration file. If the host on which JMX application is running is monitored by Zabbix proxy, then you specify the connection parameters in the proxy configuration file instead.

```
JavaGateway=192.168.3.14
JavaGatewayPort=10052
```

By default, server does not start any processes related to JMX monitoring. If you wish to use it, however, you have to specify the number of pre-forked instances of Java pollers. You do this in the same way you specify regular pollers and trappers.

```
StartJavaPollers=5
```

Do not forget to restart server or proxy, once you are done with configuring them.

Debugging Java gateway

Zabbix Java gateway log file is:

```
/var/log/zabbix/zabbix_java_gateway.log
```

If you like to increase the logging, edit the file:

```
/etc/zabbix/zabbix_java_gateway_logback.xml
```

and change `level="info"` to ”debug” or even ”trace” (for deep troubleshooting):

```
<configuration scan="true" scanPeriod="15 seconds">
[...]
      <root level="info">
              <appender-ref ref="FILE" />
      </root>

</configuration>
```

JMX monitoring

See JMX monitoring page for more details.

6 Sender

Overview

Zabbix sender is a command line utility that may be used to send performance data to Zabbix server for processing.

The utility is usually used in long running user scripts for periodical sending of availability and performance data.

For sending results directly to Zabbix server or proxy, a trapper item type must be configured.

See also zabbix_utils - a Python library that has built-in functionality to act like Zabbix sender.

Running Zabbix sender

An example of running Zabbix UNIX sender:

```
shell> cd bin
shell> ./zabbix_sender -z zabbix -s "Linux DB3" -k db.connections -o 43
```

where:

- z - Zabbix server host (IP address can be used as well)
- s - technical name of monitored host (as registered in Zabbix frontend)
- k - item key
- o - value to send

Attention:
Options that contain whitespaces, must be quoted using double quotes.

Zabbix sender can be used to send multiple values from an input file. See the Zabbix sender manpage for more information.

If a configuration file is specified, Zabbix sender uses all addresses defined in the agent ServerActive configuration parameter for sending data. If sending to one address fails, the sender tries sending to the other addresses. If sending of batch data fails to one address, the following batches are not sent to this address.

Zabbix sender accepts strings in UTF-8 encoding (for both UNIX-like systems and Windows) without byte order mark (BOM) first in the file.

Zabbix sender on Windows can be run similarly:

```
zabbix_sender.exe [options]
```

Since Zabbix 1.8.4, zabbix_sender realtime sending scenarios have been improved to gather multiple values passed to it in close succession and send them to the server in a single connection. A value that is not further apart from the previous value than 0.2 seconds can be put in the same stack, but maximum polling time still is 1 second.

Note:
Zabbix sender will terminate if invalid (not following *parameter=value* notation) parameter entry is present in the specified configuration file.

Running Zabbix sender with low-level discovery

An example of running Zabbix sender for sending a JSON-formatted value for low-level discovery:

```
./zabbix_sender -z 192.168.1.113 -s "Zabbix server" -k trapper.discovery.item -o '[{"{#FSNAME}":"/","{#FST
```

For this to work, the low-level discovery rule must have a Zabbix trapper item type (in this example, with `trapper.discovery.item` key).

7 Get

Overview

Zabbix get is a command line utility which can be used to communicate with Zabbix agent and retrieve required information from the agent.

The utility is usually used for the troubleshooting of Zabbix agents.

See also zabbix_utils - a Python library that has built-in functionality to act like Zabbix get.

Running Zabbix get

An example of running Zabbix get under UNIX to get the processor load value from the agent:

```
shell> cd bin
shell> ./zabbix_get -s 127.0.0.1 -p 10050 -k system.cpu.load[all,avg1]
```

Another example of running Zabbix get for capturing a string from a website:

```
shell> cd bin
shell> ./zabbix_get -s 192.168.1.1 -p 10050 -k "web.page.regexp[www.example.com,,,\"USA: ([a-zA-Z0-9.-]+)\
```

Note that the item key here contains a space so quotes are used to mark the item key to the shell. The quotes are not part of the item key; they will be trimmed by the shell and will not be passed to Zabbix agent.

Zabbix get accepts the following command line parameters:

```
-s --host <host name or IP>      Specify host name or IP address of a host
-p --port <port number>          Specify port number of agent running on the host (default: 10050)
-I --source-address <IP address> Specify source IP address
-t --timeout <seconds>           Specify timeout. Valid range: 1-30 seconds (default: 30 seconds)
-k --key <item key>              Specify key of item to retrieve value for
-h --help                        Display this help message
-V --version                     Display version number
```

See also Zabbix get manpage for more information.

Zabbix get on Windows can be run similarly:

```
zabbix_get.exe [options]
```

8 JS

Overview

zabbix_js is a command line utility that can be used for embedded script testing.

This utility will execute a user script with a string parameter and print the result. Scripts are executed using the embedded Zabbix scripting engine.

In case of compilation or execution errors zabbix_js will print the error in stderr and exit with code 1.

Usage

```
zabbix_js -s script-file -p input-param [-l log-level] [-t timeout]
zabbix_js -s script-file -i input-file [-l log-level] [-t timeout]
zabbix_js -h
zabbix_js -V
```

zabbix_js accepts the following command line parameters:

```
  -s, --script script-file          Specify the file name of the script to execute. If '-' is specified as
  -i, --input input-file            Specify the file name of the input parameter. If '-' is specified as f
  -p, --param input-param           Specify the input parameter.
  -l, --loglevel log-level          Specify the log level.
  -t, --timeout timeout             Specify the timeout in seconds. Valid range: 1-60 seconds (default: 10
  -h, --help                        Display help information.
  -V, --version                     Display the version number.
```

Example:

```
zabbix_js -s script-file.js -p example
```

9 Web service

Overview

Zabbix web service is a process that is used for communication with external web services. Currently, Zabbix web service is used for generating and sending scheduled reports with plans to add additional functionality in the future.

Zabbix server connects to the web service via HTTP(S). Zabbix web service requires Google Chrome to be installed on the same host; on some distributions the service may also work with Chromium (see known issues).

Installation

The official zabbix-web-service package is available in the Zabbix repository.

To compile Zabbix web service from sources, specify the `--enable-webservice` configure option.

To configure Zabbix web service, update the `zabbix_web_service.conf` configuration file parameters.

> **Attention:**
> It is strongly recommended to set up encryption between Zabbix server and Zabbix web service using certificates. By default, data transmitted between Zabbix server and Zabbix web service is not encrypted, which can lead to unauthorized access.

4 Installation

Please use the sidebar to access content in the Installation section.

1 Getting Zabbix

Overview

There are four ways of getting Zabbix:

- Install it from the distribution packages
- Download the latest source archive and compile it yourself
- Install it from the containers
- Download the virtual appliance

To download the latest distribution packages, pre-compiled sources or the virtual appliance, go to the Zabbix download page, where direct links to latest versions are provided.

Getting Zabbix source code

There are several ways of getting Zabbix source code:

- You can download the released stable versions from the official Zabbix website
- You can download nightly builds from the official Zabbix website developer page
- You can get the latest development version from the Git source code repository system:
  - The primary location of the full repository is at https://git.zabbix.com/scm/zbx/zabbix.git
  - Master and supported releases are also mirrored to Github at https://github.com/zabbix/zabbix

A Git client must be installed to clone the repository. The official commandline Git client package is commonly called **git** in distributions. To install, for example, on Debian/Ubuntu, run:

```
sudo apt-get update
sudo apt-get install git
```

To grab all Zabbix source, change to the directory you want to place the code in and execute:

```
git clone https://git.zabbix.com/scm/zbx/zabbix.git
```

Hardware

Memory

Zabbix requires both physical and disk memory. The amount of required disk memory obviously depends on the number of hosts and parameters that are being monitored. If you're planning to keep a long history of monitored parameters, you should be thinking of at least a couple of gigabytes to have enough space to store the history in the database. Each Zabbix daemon process requires several connections to a database server. The amount of memory allocated for the connection depends on the configuration of the database engine.

> **Note:**
> The more physical memory you have, the faster the database (and therefore Zabbix) works.

CPU

Zabbix and especially Zabbix database may require significant CPU resources depending on number of monitored parameters and chosen database engine.

Other hardware

A serial communication port and a serial GSM modem are required for using SMS notification support in Zabbix. USB-to-serial converter will also work.

Examples of hardware configuration

The table provides examples of hardware configuration, assuming a **Linux/BSD/Unix** platform.

These are size and hardware configuration examples to start with. Each Zabbix installation is unique. Make sure to benchmark the performance of your Zabbix system in a staging or development environment, so that you can fully understand your requirements before deploying the Zabbix installation to its production environment.

Installation size	Monitored metrics[1]	CPU/vCPU cores	Memory (GiB)	Database	Amazon EC2[2]
Small	1 000	2	8	MySQL Server, Percona Server, MariaDB Server, PostgreSQL	m6i.large/m6g.large
Medium	10 000	4	16	MySQL Server, Percona Server, MariaDB Server, PostgreSQL	m6i.xlarge/m6g.xlarge
Large	100 000	16	64	MySQL Server, Percona Server, MariaDB Server, PostgreSQL, Oracle	m6i.4xlarge/m6g.4xlarge
Very large	1 000 000	32	96	MySQL Server, Percona Server, MariaDB Server, PostgreSQL, Oracle	m6i.8xlarge/m6g.8xlarge

[1] 1 metric = 1 item + 1 trigger + 1 graph
 [2] Example with Amazon general purpose EC2 instances, using ARM64 or x86_64 architecture, a proper instance type like Compute/Memory/Storage optimised should be selected during Zabbix installation evaluation and testing before installing in its production environment.

> **Note:**
> Actual configuration depends on the number of active items and refresh rates very much (see database size section of this page for details). It is highly recommended to run the database on a separate server for large installations.

Supported platforms

Due to security requirements and the mission-critical nature of the monitoring server, UNIX is the only operating system that can consistently deliver the necessary performance, fault tolerance, and resilience. Zabbix operates on market-leading versions.

Zabbix components are available and tested for the following platforms:

Platform	Server	Agent	Agent2
Linux	x	x	x
IBM AIX	x	x	-
FreeBSD	x	x	-
NetBSD	x	x	-
OpenBSD	x	x	-
HP-UX	x	x	-
Mac OS X	x	x	-
Solaris	x	x	-
Windows	-	x	x

Note:
Zabbix server/agent may work on other Unix-like operating systems as well. Zabbix agent is supported on all Windows desktop and server versions since XP (XP 64-bit version, since Zabbix 6.2).

To prevent critical security vulnerabilities in Zabbix agent 2, it is compiled only with supported Go releases. As of Go 1.21, the minimum required Windows versions are raised; therefore, since Zabbix 6.4.12, the minimum Windows version for Zabbix agent 2 is Windows 10/Server 2016.

Attention:
Zabbix disables core dumps if compiled with encryption and does not start if the system does not allow disabling of core dumps.

Required software

Zabbix is built around modern web servers, leading database engines, and PHP scripting language.

Third-party external surrounding software

If stated as mandatory, the required software/library is strictly necessary. Optional ones are needed for supporting some specific function.

Software	Mandatory status	Supported versions	Comments
MySQL/Percona	One of	8.0.30-9.0.X	Required if MySQL (or Percona) is used as Zabbix backend database. InnoDB engine is required. Added support for MySQL versions: - 8.1 since Zabbix 6.4.6; - 8.2 since Zabbix 6.4.11; - 8.3 since Zabbix 6.4.12; - 8.4 since Zabbix 6.4.17; - 9.0 since Zabbix 6.4.18. We recommend using the C API (libmysqlclient) library for building server/proxy.

Software	Mandatory status	Supported versions	Comments
MariaDB		10.5.00-11.5.X	InnoDB engine is required. The recommended version is 10.5. We recommend using the MariaDB Connector/C library for building server/proxy. Added support for MariaDB versions: - 10.11.X since Zabbix 6.4.1; - 11.0.X since Zabbix 6.4.6; - 11.1.X since Zabbix 6.4.8; - 11.2.X since Zabbix 6.4.12; - 11.3.X since Zabbix 6.4.14; - 11.4.X since Zabbix 6.4.17; - 11.5.X since Zabbix 6.4.19. See also: Possible deadlocks with MariaDB.
Oracle		19c - 21c	Required if Oracle is used as Zabbix backend database.
PostgreSQL		13.0-17.X	Required if PostgreSQL is used as Zabbix backend database. Depending on the installation size, it might be required to increase PostgreSQL *work_mem* configuration property (4MB being the default value), so that the amount of memory used by the database for particular operation is sufficient and query execution does not take too much time. Added support for PostgreSQL versions: - 16.X since Zabbix 6.4.8; - 17.X since Zabbix 6.4.20.
TimescaleDB for PostgreSQL		2.1.0-2.17.X	Required if TimescaleDB is used as a PostgreSQL database extension. Make sure to install TimescaleDB Community Edition, which supports compression. Note that PostgreSQL 15 is supported since TimescaleDB 2.10. You may also refer to the Timescale documentation for details regarding PostgreSQL and TimescaleDB version compatibility. Added support for TimescaleDB versions: - 2.10 since Zabbix 6.4.1; - 2.11 since Zabbix 6.4.4; - 2.12 since Zabbix 6.4.9; - 2.13 since Zabbix 6.4.10; - 2.14 since Zabbix 6.4.13; - 2.15 since Zabbix 6.4.17; - 2.16 since Zabbix 6.4.19; - 2.17 since Zabbix 6.4.20.
SQLite	Optional	3.3.5-3.34.X	SQLite is only supported with Zabbix proxies. Required if SQLite is used as Zabbix proxy database.
smartmontools		7.1 or later	Required for Zabbix agent 2.
who			Required for the user count plugin.
dpkg			Required for the system.sw.packages plugin.
pkgtool			Required for the system.sw.packages plugin.
rpm			Required for the system.sw.packages plugin.
pacman			Required for the system.sw.packages plugin.

Note:
Although Zabbix can work with databases available in the operating systems, for the best experience, we recommend using databases installed from the official database developer repositories.

Frontend

The minimum supported screen width for Zabbix frontend is 1200px.

If stated as mandatory, the required software/library is strictly necessary. Optional ones are needed for supporting some specific function.

Software	Mandatory status	Version	Comments
Apache *Nginx*	One of	2.4 or later 1.20 or later	
PHP	Yes	7.4.0 - 8.3.X	It is recommended to use PHP 8.0 or newer as PHP 7.4 is out of support. Added support for PHP versions: - 8.3.X since Zabbix 6.4.11.
PHP extensions:			
gd	Yes	2.0.28 or later	PHP GD extension must support PNG images (*--with-png-dir*), JPEG (*--with-jpeg-dir*) images and FreeType 2 (*--with-freetype-dir*). Version 2.3.0 or later might be required to avoid possible text overlapping in graphs for some frontend languages.
bcmath			php-bcmath (*--enable-bcmath*)
ctype			php-ctype (*--enable-ctype*)
libXML		2.6.15 or later	php-xml, if provided as a separate package by the distributor.
xmlreader			php-xmlreader, if provided as a separate package by the distributor.
xmlwriter			php-xmlwriter, if provided as a separate package by the distributor.
session			php-session, if provided as a separate package by the distributor.
sockets			php-net-socket (*--enable-sockets*). Required for user script support.
mbstring			php-mbstring (*--enable-mbstring*)
gettext			php-gettext (*--with-gettext*). Required for translations to work.
ldap	No		php-ldap. Required only if LDAP authentication is used in the frontend.
openssl			php-openssl. Required only if SAML authentication is used in the frontend.
mysqli			Required if MySQL is used as Zabbix backend database.
oci8			Required if Oracle is used as Zabbix backend database.
pgsql			Required if PostgreSQL is used as Zabbix backend database.

Third-party frontend libraries that are supplied with Zabbix:

Library	Mandatory status	Minimum version	Comments
jQuery JavaScript Library	Yes	3.6.0	JavaScript library that simplifies the process of cross-browser development.
jQuery UI		1.12.1	A set of user interface interactions, effects, widgets, and themes built on top of jQuery.
SAML PHP Toolkit		4.0.0	A PHP toolkit that adds SAML 2.0 authentication support to be able to sign in to Zabbix.
Symfony Yaml Component		5.1.0	Adds support to export and import Zabbix configuration elements in the YAML format.

Note:
Zabbix may work on previous versions of Apache, MySQL, Oracle, and PostgreSQL as well.

Attention:
For other fonts than the default DejaVu, PHP function imagerotate might be required. If it is missing, these fonts might be rendered incorrectly when a graph is displayed. This function is only available if PHP is compiled with bundled GD, which is not the case in Debian and other distributions.

Third-party libraries used for writing and debugging Zabbix frontend code:

Library	Mandatory status	Minimum version	Description
Composer	No	2.4.1	An application-level package manager for PHP that provides a standard format for managing dependencies of PHP software and required libraries.
PHPUnit		8.5.29	A PHP unit testing framework for testing Zabbix frontend.
SASS		3.4.22	A preprocessor scripting language that is interpreted and compiled into Cascading Style Sheets (CSS).

Web browser on client side

Cookies and JavaScript must be enabled.

The latest stable versions of Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari, and Opera are supported.

Warning:
The same-origin policy for IFrames is implemented, which means that Zabbix cannot be placed in frames on a different domain.

Still, pages placed into a Zabbix frame will have access to Zabbix frontend (through JavaScript) if the page that is placed in the frame and Zabbix frontend are on the same domain. A page like `http://secure-zabbix.com/cms/page.html`, if placed into dashboards on `http://secure-zabbix.com/zabbix/`, will have full JS access to Zabbix.

Server/proxy

If stated as mandatory, the required software/library is strictly necessary. Optional ones are needed for supporting some specific function.

Requirement	Mandatory status	Description
libpcre/libpcre2	One of	PCRE/PCRE2 library is required for Perl Compatible Regular Expression (PCRE) support. The naming may differ depending on the GNU/Linux distribution, for example 'libpcre3' or 'libpcre1'. PCRE v8.x and PCRE2 v10.x (from Zabbix 6.0.0) are supported.
libevent	Yes	Required for inter-process communication. Version 1.4 or higher.
libevent-pthreads		Required for inter-process communication.
libpthread		Required for mutex and read-write lock support (could be part of libc).
libresolv		Required for DNS resolution (could be part of libc).
libiconv		Required for text encoding/format conversion (could be part of libc). Mandatory for Zabbix server on Linux.
libz		Required for compression support.
libm		Math library. Required by Zabbix server only.
libmysqlclient	One of	Required if MySQL is used.
libmariadb		Required if MariaDB is used.
libclntsh		Required if Oracle is used; *libclntsh* version must match or be higher than the version of the Oracle database used.
libpq5		Required if PostgreSQL is used; *libpq5* version must match or be higher than the version of the PostgreSQL database used.
libsqlite3		Required if Sqlite is used. Required for Zabbix proxy only.
libOpenIPMI	No	Required for IPMI support. Required for Zabbix server only.
libssh2 or *libssh*		Required for SSH checks. Version 1.0 or higher (libssh2); 0.9.0 or higher (libssh). libssh is supported since Zabbix 4.4.6.
libcurl		Required for web monitoring, VMware monitoring, SMTP authentication, `web.page.*` Zabbix agent items, HTTP agent items and Elasticsearch (if used). Version 7.28.0 or higher is recommended. Libcurl version requirements: - SMTP authentication: version 7.20.0 or higher - Elasticsearch: version 7.28.0 or higher
libxml2		Required for VMware monitoring and XML XPath preprocessing.

Requirement	Mandatory status	Description
net-snmp		Required for SNMP support. Version 5.3.0 or higher. Support of strong encryption protocols (AES192/AES192C, AES256/AES256C) is available starting with net-snmp library 5.8; on RHEL 8+ based systems it is recommended to use net-snmp 5.8.15 or later.
libunixodbc		Required for database monitoring.
libgnutls or *libopenssl*		Required when using encryption. Minimum versions: *libgnutls* - 3.1.18, *libopenssl* - 1.0.1
libldap		Required for LDAP support.
fping		Required for ICMP ping items.

Agent

Requirement	Mandatory status	Description
libpcre/libpcre2	One of	PCRE/PCRE2 library is required for Perl Compatible Regular Expression (PCRE) support. The naming may differ depending on the GNU/Linux distribution, for example 'libpcre3' or 'libpcre1'. PCRE v8.x and PCRE2 v10.x (from Zabbix 6.0.0) are supported. Required for log monitoring. Also required on Windows.
libpthread	Yes	Required for mutex and read-write lock support (could be part of libc). Not required on Windows.
libresolv		Required for DNS resolution (could be part of libc). Not required on Windows.
libiconv		Required for text encoding/format conversion to UTF-8 in log items, file content, file regex and regmatch items (could be part of libc). Not required on Windows.
libgnutls or *libopenssl*	No	Required if using encryption. Minimum versions: *libgnutls* - 3.1.18, *libopenssl* - 1.0.1 On Microsoft Windows OpenSSL 1.1.1 or later is required.
libldap		Required if LDAP is used. Not supported on Windows.
libcurl		Required for `web.page.*` Zabbix agent items. Not supported on Windows. Version 7.28.0 or higher is recommended.
libmodbus		Only required if Modbus monitoring is used. Version 3.0 or higher.

Note:
Starting from version 5.0.3, Zabbix agent will not work on AIX platforms below versions 6.1 TL07 / AIX 7.1 TL01.

Agent 2

Requirement	Mandatory status	Description
libpcre/libpcre2	One of	PCRE/PCRE2 library is required for Perl Compatible Regular Expression (PCRE) support. The naming may differ depending on the GNU/Linux distribution, for example 'libpcre3' or 'libpcre1'. PCRE v8.x and PCRE2 v10.x (from Zabbix 6.0.0) are supported. Required for log monitoring. Also required on Windows.
libopenssl	No	Required when using encryption. OpenSSL 1.0.1 or later is required on UNIX platforms. The OpenSSL library must have PSK support enabled. LibreSSL is not supported. On Microsoft Windows systems OpenSSL 1.1.1 or later is required.

Go libraries

Requirement	Mandatory status	Minimum version	Description
git.zabbix.com/ap/plugin support	Yes	1.X.X	Zabbix own support library. Mostly for plugins.
github.com/BurntSushi/locker		0.0.0	Named read/write locks, access sync.
github.com/chromedp/cdproto		0.0.0	Generated commands, types, and events for the Chrome DevTools Protocol domains.
github.com/chromedp/chromedp		0.6.0	Chrome DevTools Protocol support (report generation).
github.com/dustin/gomemcached		0.0.0	A memcached binary protocol toolkit for go.
github.com/eclipse/paho.mqtt.golang		1.2.0	A library to handle MQTT connections.
github.com/fsnotify/fsnotify		1.4.9	Cross-platform file system notifications for Go.
github.com/go-ldap/ldap		3.0.3	Basic LDAP v3 functionality for the Go programming language.
github.com/go-ole/go-ole		1.2.4	Win32 ole implementation for Go.
github.com/godbus/dbus		4.1.0	Native Go bindings for D-Bus.
github.com/go-sql-driver/mysql		1.5.0	MySQL driver.
github.com/godror/godror		0.20.1	Oracle DB driver.
github.com/mattn/go-sqlite3		2.0.3	Sqlite3 driver.
github.com/mediocregopher/radix/v3		3.5.0	Redis client.
github.com/memcachier/mc/v3		3.0.1	Binary Memcached client.
github.com/miekg/dns		1.1.43	DNS library.
github.com/omeid/go-yarn		0.0.1	Embeddable filesystem mapped key-string store.
github.com/goburrow/modbus		0.1.0	Fault-tolerant implementation of Modbus.
golang.org/x/sys		0.0.0	Go packages for low-level interactions with the operating system. Also used in plugin support lib. Used in MongoDB and PostgreSQL plugins.
github.com/Microsoft/go-winio	On Windows. Yes, indirect[1]	0.6.0	Windows named pipe implementation. Also used in plugin support lib. Used in MongoDB and PostgreSQL plugins.
github.com/goburrow/serial	Yes, indirect[1]	0.1.0	Serial library for Modbus.
golang.org/x/xerrors		0.0.0	Functions to manipulate errors.
gopkg.in/asn1-ber.v1		1.0.0	Encoding/decoding library for ASN1 BER.
github.com/go-stack/stack	No, indirect[1]	1.8.0	
github.com/golang/snappy		0.0.1	
github.com/klauspost/compress		1.13.6	
github.com/xdg-go/pbkdf2		1.0.0	
github.com/xdg-go/scram		1.0.2	
github.com/xdg-go/stringprep		1.0.2	
github.com/youmark/pkcs8		0.0.0	

[1] "Indirect" means that it is used in one of the libraries that the agent uses. It's required since Zabbix uses the library that uses the package.

See also dependencies for loadable plugins:

- PostgreSQL
- MongoDB

Java gateway

If you obtained Zabbix from the source repository or an archive, then the necessary dependencies are already included in the source tree.

If you obtained Zabbix from your distribution's package, then the necessary dependencies are already provided by the packaging system.

In both cases above, the software is ready to be used and no additional downloads are necessary.

If, however, you wish to provide your versions of these dependencies (for instance, if you are preparing a package for some Linux distribution), below is the list of library versions that Java gateway is known to work with. Zabbix may work with other versions of these libraries, too.

The following table lists JAR files that are currently bundled with Java gateway in the original code:

Library	Mandatory status	Minimum version	Comments
android-json	Yes	4.3r1	JSON (JavaScript Object Notation) is a lightweight data-interchange format. This is the org.json compatible Android implementation extracted from the Android SDK.
logback-classic		1.5.16	
logback-core		1.5.16	
slf4j-api		2.0.16	

Java gateway can be built using either Oracle Java or open source OpenJDK (version 1.6 or newer). Packages provided by Zabbix are compiled using OpenJDK. The following table lists OpenJDK packages used for building Zabbix packages by distribution:

Distribution	OpenJDK package
Debian 12	`default-jdk-headless` (amd64, arm64: 2:1.17-74)
Debian 11	`default-jdk-headless` (amd64: 2:1.11-72)
Debian 10	`default-jdk-headless` (amd64, i386: 2:1.11-71)
OpenSUSE Leap 15	`java-17-openjdk-devel` (amd64: 17.0.5.0-150400.3.9.3; arm64: 17.0.8.0-150400.3.27.1)
Oracle Linux 9	`java-11-openjdk-devel` (amd64: 11.0.19.0.7-4.0.1; arm64: 11.0.20.0.8-2.0.1)
Oracle Linux 8	`java-1.8.0-openjdk-devel` (amd64: 1.8.0.372.b07-4.0.1); `java-11-openjdk-devel` (arm64: 11.0.20.0.8-3.0.1)
Oracle Linux 7	`java-1.8.0-openjdk-devel` (amd64: 1.8.0.282.b08-1)
Raspberry Pi OS 11	`default-jdk-headless` (arm64: 2:1.11-72; armhf: 2:1.11-72+b4)
Raspberry Pi OS 10	`default-jdk` (armhf: 2:1.11-71+b2)
RHEL 9	`java-11-openjdk-devel` (amd64: 11.0.19.0.7-4; arm64: 11.0.20.0.8-3)
RHEL 8	`java-1.8.0-openjdk-devel` (amd64: 1.8.0.372.b07-4; arm64: 1.8.0.382.b05-2)
RHEL 7	`java-1.8.0-openjdk-devel` (amd64: 1.8.0.282.b08-1)
SLES 15	`java-17-openjdk-devel` (amd64: 17.0.5.0-150400.3.9.3; arm64: 17.0.8.0-150400.3.27.1)
SLES 12	`java-1_8_0-openjdk-devel` (amd64: 1.8.0.252-27.45.6)
Ubuntu 24.04	`default-jdk-headless` (amd64, arm64: 2:1.21-75+exp1)
Ubuntu 22.04	`default-jdk-headless` (amd64, arm64: 2:1.11-72build2)
Ubuntu 20.04	`default-jdk-headless` (amd64, arm64: 2:1.11-72)
Ubuntu 18.04	`default-jdk` (amd64: 2:1.11-68ubuntu1~18.04.1; i386: 2:1.10-63ubuntu1~02)

Default port numbers

The following list of open ports per component is applicable for default configuration:

Zabbix component	Port number	Protocol	Type of connection
Zabbix agent	10050	TCP	on demand
Zabbix agent 2	10050	TCP	on demand
Zabbix server	10051	TCP	on demand
Zabbix proxy	10051	TCP	on demand
Zabbix Java gateway	10052	TCP	on demand
Zabbix web service	10053	TCP	on demand
Zabbix frontend	80	HTTP	on demand
	443	HTTPS	on demand
Zabbix trapper	10051	TCP	on demand

Note:
The port numbers should be open in firewall to enable Zabbix communications. Outgoing TCP connections usually do not require explicit firewall settings.

Database size

Zabbix configuration data require a fixed amount of disk space and do not grow much.

Zabbix database size mainly depends on these variables, which define the amount of stored historical data:

- Number of processed values per second

This is the average number of new values Zabbix server receives every second. For example, if we have 3000 items for monitoring with a refresh rate of 60 seconds, the number of values per second is calculated as 3000/60 = **50**.

It means that 50 new values are added to Zabbix database every second.

- Housekeeper settings for history

Zabbix keeps values for a fixed period of time, normally several weeks or months. Each new value requires a certain amount of disk space for data and index.

So, if we would like to keep 30 days of history and we receive 50 values per second, the total number of values will be around (**30***24*3600)* **50** = 129.600.000, or about 130M of values.

Depending on the database engine used, type of received values (floats, integers, strings, log files, etc), the disk space for keeping a single value may vary from 40 bytes to hundreds of bytes. Normally it is around 90 bytes per value for numeric items[2]. In our case, it means that 130M of values will require 130M * 90 bytes = **10.9GB** of disk space.

Note:
The size of text/log item values is impossible to predict exactly, but you may expect around 500 bytes per value.

- Housekeeper setting for trends

Zabbix keeps a 1-hour max/min/avg/count set of values for each item in the table **trends**. The data is used for trending and long period graphs. The one hour period can not be customized.

Zabbix database, depending on the database type, requires about 90 bytes per each total. Suppose we would like to keep trend data for 5 years. Values for 3000 items will require 3000*24*365* **90** = **2.2GB** per year, or **11GB** for 5 years.

- Housekeeper settings for events

Each Zabbix event requires approximately 250 bytes of disk space[1]. It is hard to estimate the number of events generated by Zabbix daily. In the worst-case scenario, we may assume that Zabbix generates one event per second.

For each recovered event, an event_recovery record is created. Normally most of the events will be recovered so we can assume one event_recovery record per event. That means additional 80 bytes per event.

Optionally events can have tags, each tag record requiring approximately 100 bytes of disk space[1]. The number of tags per event (#tags) depends on configuration. So each will need an additional #tags * 100 bytes of disk space.

It means that if we want to keep 3 years of events, this would require 3*365*24*3600* (250+80+#tags*100) = **~30GB**+#tags*100B disk space[2].

Note:
[1] More when having non-ASCII event names, tags and values.
[2] The size approximations are based on MySQL and might be different for other databases.

The table contains formulas that can be used to calculate the disk space required for Zabbix system:

Parameter	Formula for required disk space (in bytes)
Zabbix configuration	Fixed size. Normally 10MB or less.
History	days*(items/refresh rate)*24*3600*bytes items : number of items days : number of days to keep history refresh rate : average refresh rate of items bytes : number of bytes required to keep single value, depends on database engine, normally ~90 bytes.

Parameter	Formula for required disk space (in bytes)
Trends	days*(items/3600)*24*3600*bytes items : number of items days : number of days to keep history bytes : number of bytes required to keep single trend, depends on the database engine, normally ~90 bytes.
Events	days*events*24*3600*bytes events : number of event per second. One (1) event per second in worst-case scenario. days : number of days to keep history bytes : number of bytes required to keep single trend, depends on the database engine, normally ~330 + average number of tags per event * 100 bytes.

So, the total required disk space can be calculated as:
Configuration + History + Trends + Events
The disk space will NOT be used immediately after Zabbix installation. Database size will grow then it will stop growing at some point, which depends on housekeeper settings.

Time synchronization

It is very important to have precise system time on the server with Zabbix running. ntpd is the most popular daemon that synchronizes the host's time with the time of other machines. It's strongly recommended to maintain synchronized system time on all systems Zabbix components are running on.

Network requirements

A following list of open ports per component is applicable for default configuration.

Port	Components
Frontend	http on 80, https on 443
Server	10051 (for use with active proxy/agents)
Active Proxy	10051
Passive Proxy	10051
Agent2	10050
Trapper	
JavaGateway	10052
WebService	10053

Note:
The port numbers should be opened in the firewall to enable external communications with Zabbix. Outgoing TCP connections usually do not require explicit firewall settings.

1 Plugins

Please use the sidebar to access content in the Plugins section.

1 PostgreSQL plugin dependencies

Overview

The required libraries for the PostgreSQL loadable plugin are listed in this page.

Go libraries

Requirement	Mandatory status	Minimum version	Description
git.zabbix.com/ap/plugin support	Yes	1.X.X	Zabbix own support library. Mostly for plugins.
github.com/jackc/pgx/v4		4.17.2	PostgreSQL driver.
github.com/omeid/go-yarn		0.0.1	Embeddable filesystem mapped key-string store.

Requirement	Mandatory status	Minimum version	Description
github.com/jackc/chunkreader	Indirect[1]	2.0.1	
github.com/jackc/pgconn		1.13.0	
github.com/jackc/pgio		1.0.0	
github.com/jackc/pgpassfile		1.0.0	
github.com/jackc/pgproto3		2.3.1	
github.com/jackc/pgservicefile		0.0.0	
github.com/jackc/pgtype		1.12.0	
github.com/jackc/puddle		1.3.0	
github.com/Microsoft/go-winio		0.6.0	Required package for PostgreSQL plugin on Windows.
golang.org/x/crypto		0.0.0	
golang.org/x/sys		0.0.0	
golang.org/x/text		0.3.7	

[1] "Indirect" means that it is used in one of the libraries that the agent uses. It's required since Zabbix uses the library that uses the package.

2 MongoDB plugin dependencies

Overview

The required libraries for the MongoDB loadable plugin are listed in this page.

Go libraries

Requirement	Mandatory status	Minimum version	Description
git.zabbix.com/ap/plugin support	Yes	1.X.X	Zabbix own support library. Mostly for plugins.
go.mongodb.org/mongo-driver		1.7.6	Named read/write locks, access sync.
github.com/go-stack/stack	Indirect[1]	1.8.0	Required package for MongoDB plugin mongo-driver lib.
github.com/golang/snappy		0.0.1	Required package for MongoDB plugin mongo-driver lib.
github.com/klauspost/compress		1.13.6	Required package for MongoDB plugin mongo-driver lib.
github.com/Microsoft/go-winio		0.6.0	Required package for MongoDB plugin mongo-driver lib on Windows.
github.com/pkg/errors		0.9.1	Required package for MongoDB plugin mongo-driver lib.
github.com/xdg-go/pbkdf2		1.0.0	Required package for MongoDB plugin mongo-driver lib.
github.com/xdg-go/scram		1.0.2	Required package for MongoDB plugin mongo-driver lib.
github.com/xdg-go/stringprep		1.0.2	Required package for MongoDB plugin mongo-driver lib.
github.com/youmark/pkcs8		0.0.0	Required package for MongoDB plugin mongo-driver lib.
golang.org/x/crypto		0.0.0	Required package for MongoDB plugin mongo-driver lib.
golang.org/x/sync		0.0.0	Required package for MongoDB plugin mongo-driver lib.
golang.org/x/sys		0.0.0	Required package for MongoDB plugin mongo-driver lib.
golang.org/x/text		0.3.7	Required package for MongoDB plugin mongo-driver lib.

[1] "Indirect" means that it is used in one of the libraries that the agent uses. It's required since Zabbix uses the library that uses the package.

2 Best practices for secure Zabbix setup

Overview

This section contains best practices that should be observed in order to set up Zabbix in a secure way.

The practices contained here are not required for the functioning of Zabbix. They are recommended for better security of the system.

Access control

Principle of least privilege

The principle of least privilege should be used at all times for Zabbix. This principle means that user accounts (in Zabbix frontend) or process user (for Zabbix server/proxy or agent) have only those privileges that are essential to perform intended functions. In other words, user accounts at all times should run with as few privileges as possible.

Attention:
Giving extra permissions to 'zabbix' user will allow it to access configuration files and execute operations that can compromise the overall security of the infrastructure.

When implementing the least privilege principle for user accounts, Zabbix frontend user types should be taken into account. It is important to understand that while a "Admin" user type has less privileges than "Super Admin" user type, it has administrative permissions that allow managing configuration and execute custom scripts.

Note:
Some information is available even for non-privileged users. For example, while *Alerts* → *Scripts* is not available for non-Super Admins, scripts themselves are available for retrieval by using Zabbix API. Limiting script permissions and not adding sensitive information (like access credentials, etc) should be used to avoid exposure of sensitive information available in global scripts.

Secure user for Zabbix agent

In the default configuration, Zabbix server and Zabbix agent processes share one 'zabbix' user. If you wish to make sure that the agent cannot access sensitive details in server configuration (e.g. database login information), the agent should be run as a different user:

1. Create a secure user
2. Specify this user in the agent configuration file ('User' parameter)
3. Restart the agent with administrator privileges. Privileges will be dropped to the specified user.

Revoke write access to SSL configuration (Windows)

If you have compiled Zabbix agent on Windows, with OpenSSL located in an unprotected directory (e.g., `c:\openssl-64bit`, `C:\OpenSSL-Win64-111-static`, or `C:\dev\openssl`), make sure to revoke write access from non-administrator users to this directory. Otherwise, the agent loads SSL settings from a path that can be modified by unprivileged users, resulting in a potential security vulnerability.

Cryptography

Setting up SSL for Zabbix frontend

On RHEL-based systems, install the `mod_ssl` package:

```
dnf install mod_ssl
```

Create a directory for SSL keys:

```
mkdir -p /etc/httpd/ssl/private
chmod 700 /etc/httpd/ssl/private
```

Create the SSL certificate:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/httpd/ssl/private/apache-selfsigned.key -
```

Fill out the prompts appropriately. The most important line is the one that requests the `Common Name`. You need to enter the domain name that you want to be associated with your server. You can enter the public IP address instead if you do not have a domain name.

```
Country Name (2 letter code) [XX]:
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:example.com
Email Address []:
```

Edit the Apache SSL configuration file (`/etc/httpd/conf.d/ssl.conf`):

```
DocumentRoot "/usr/share/zabbix"
ServerName example.com:443
SSLCertificateFile /etc/httpd/ssl/apache-selfsigned.crt
SSLCertificateKeyFile /etc/httpd/ssl/private/apache-selfsigned.key
```

Restart the Apache service to apply the changes:

```
systemctl restart httpd.service
```

Web server hardening

Enabling Zabbix on root directory of URL

On RHEL-based systems, add a virtual host to Apache configuration (`/etc/httpd/conf/httpd.conf`) and set permanent redirect for document root to Zabbix SSL URL. Note that *example.com* should be replaced with the actual name of the server.

```
#### Add lines:

<VirtualHost *:*>
    ServerName example.com
    Redirect permanent / https://example.com
</VirtualHost>
```

Restart the Apache service to apply the changes:

```
systemctl restart httpd.service
```

Enabling HTTP Strict Transport Security (HSTS) on the web server

To protect Zabbix frontend against protocol downgrade attacks, we recommend enabling HSTS policy on the web server.

To enable HSTS policy for your Zabbix frontend in Apache configuration, follow these steps:

1. Locate your virtual host's configuration file:
   - `/etc/httpd/conf/httpd.conf` on RHEL-based systems
   - `/etc/apache2/sites-available/000-default.conf` on Debian/Ubuntu
2. Add the following directive to your virtual host's configuration file:

```
<VirtualHost *:*>
    Header set Strict-Transport-Security "max-age=31536000"
</VirtualHost>
```

3. Restart the Apache service to apply the changes:

```
#### On RHEL-based systems:
systemctl restart httpd.service

#### On Debian/Ubuntu
systemctl restart apache2.service
```

Enabling Content Security Policy (CSP) on the web server

To protect Zabbix frontend against Cross Site Scripting (XSS), data injection, and similar attacks, we recommend enabling Content Security Policy on the web server. To do so, configure the web server to return the HTTP header.

> **Attention:**
> The following CSP header configuration is only for the default Zabbix frontend installation and for cases when all content originates from the site's domain (excluding subdomains). A different CSP header configuration may be required if you are, for example, configuring the *URL* widget to display content from the site's subdomains or external domains, switching from *OpenStreetMap* to another map engine, or adding external CSS or widgets.

To enable CSP for your Zabbix frontend in Apache configuration, follow these steps:

1. Locate your virtual host's configuration file:
   - `/etc/httpd/conf/httpd.conf` on RHEL-based systems
   - `/etc/apache2/sites-available/000-default.conf` on Debian/Ubuntu
2. Add the following directive to your virtual host's configuration file:

```
<VirtualHost *:*>
    Header set Content-Security-Policy: "default-src 'self' *.openstreetmap.org; script-src 'self' 'unsafe
</VirtualHost>
```

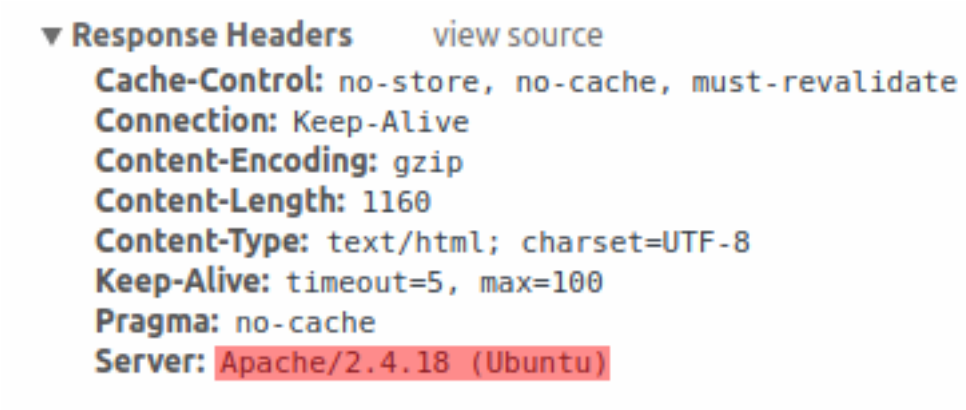
3. Restart the Apache service to apply the changes:

```
#### On RHEL-based systems:
systemctl restart httpd.service

#### On Debian/Ubuntu
systemctl restart apache2.service
```

Disabling web server information exposure

It is recommended to disable all web server signatures as part of the web server hardening process. The web server is exposing software signature by default:



The signature can be disabled by adding two lines to the Apache (used as an example) configuration file:

```
ServerSignature Off
ServerTokens Prod
```

PHP signature (X-Powered-By HTTP header) can be disabled by changing the php.ini configuration file (signature is disabled by default):

```
expose_php = Off
```

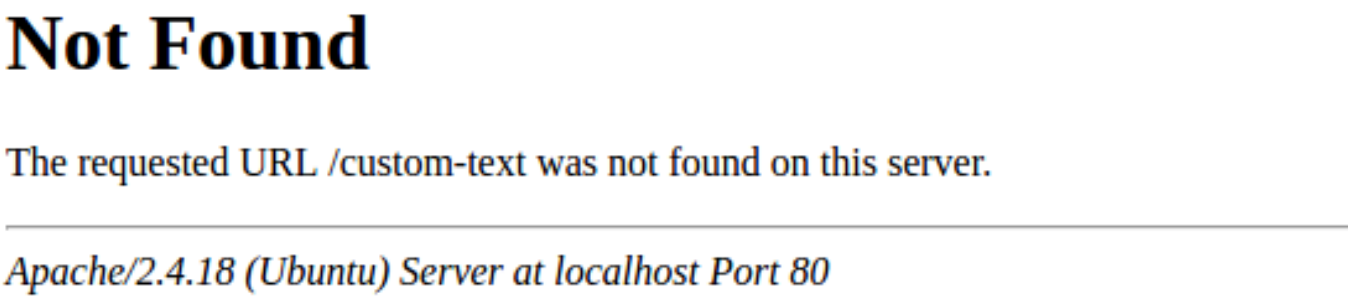
Web server restart is required for configuration file changes to be applied.

Additional security level can be achieved by using the mod_security (package libapache2-mod-security2) with Apache. mod_security allows to remove server signature instead of only removing version from server signature. Signature can be altered to any value by changing ”SecServerSignature” to any desired value after installing mod_security.

Please refer to documentation of your web server to find help on how to remove/change software signatures.

Disabling default web server error pages

It is recommended to disable default error pages to avoid information exposure. Web server is using built-in error pages by default:

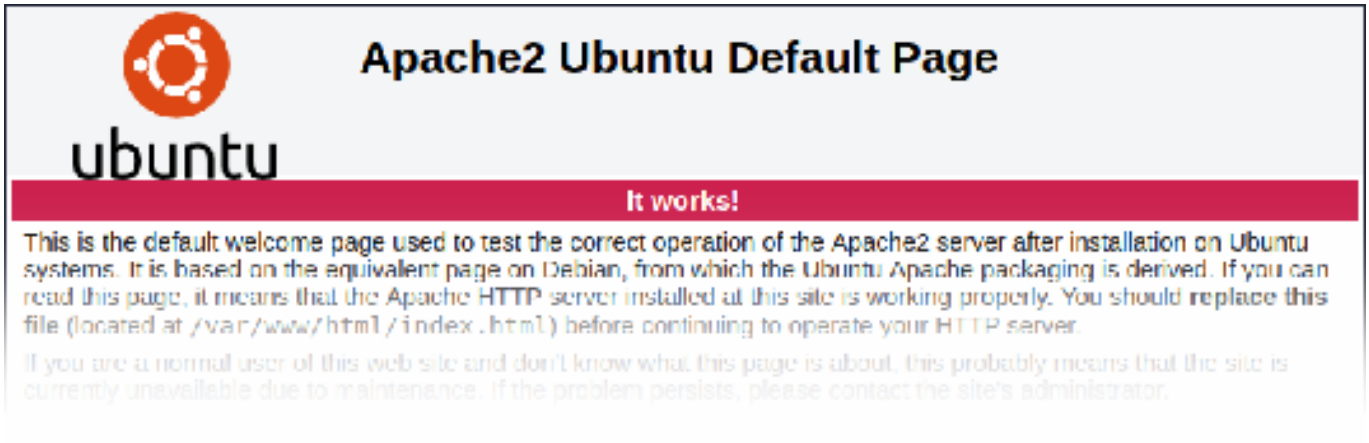


Default error pages should be replaced/removed as part of the web server hardening process. The ”ErrorDocument” directive can be used to define a custom error page/text for Apache web server (used as an example).

Please refer to documentation of your web server to find help on how to replace/remove default error pages.

Removing web server test page

It is recommended to remove the web server test page to avoid information exposure. By default, web server webroot contains a test page called index.html (Apache2 on Ubuntu is used as an example):



The test page should be removed or should be made unavailable as part of the web server hardening process.

Set X-Frame-Options HTTP response header

By default, Zabbix is configured with X-Frame-Options HTTP header* set to `SAMEORIGIN`. This means that content can only be loaded in a frame that has the same origin as the page itself.

Zabbix frontend elements that pull content from external URLs (namely, the URL dashboard widget) display retrieved content in a sandbox with all sandboxing restrictions enabled.

These settings enhance the security of the Zabbix frontend and provide protection against XSS and clickjacking attacks. *Super admin* users can modify the *Use iframe sandboxing* and *Use X-Frame-Options HTTP header* parameters as needed. Please carefully weigh the risks and benefits before changing default settings. Turning iframe sandboxing or X-Frame-Options HTTP header off completely is not recommended.

Hiding the file with list of common passwords

To increase the complexity of password brute force attacks, it is suggested to limit access to the file `ui/data/top_passwords.txt` by modifying web server configuration. This file contains a list of the most common and context-specific passwords, and is used to prevent users from setting such passwords if *Avoid easy-to-guess passwords* parameter is enabled in the password policy.

For example, on NGINX file access can be limited by using the `location` directive:

```
location = /data/top_passwords.txt {
    deny all;
    return 404;
}
```

On Apache - by using `.htaccess` file:

```
<Files "top_passwords.txt">
    Order Allow,Deny
    Deny from all
</Files>
```

UTF-8 encoding

UTF-8 is the only encoding supported by Zabbix. It is known to work without any security flaws. Users should be aware that there are known security issues if using some of the other encodings.

Windows installer paths

When using Windows installers, it is recommended to use default paths provided by the installer as using custom paths without proper permissions could compromise the security of the installation.

Zabbix Security Advisories and CVE database

See Zabbix Security Advisories and CVE database.

3 Installation from sources

You can get the very latest version of Zabbix by compiling it from the sources.

A step-by-step tutorial for installing Zabbix from the sources is provided here.

1 Installing Zabbix daemons

1 Download the source archive

Go to the Zabbix download page and download the source archive. Once downloaded, extract the sources, by running:

```
$ tar -zxvf zabbix-6.4.0.tar.gz
```

Note:
Enter the correct Zabbix version in the command. It must match the name of the downloaded archive.

2 Create user account

For all of the Zabbix daemon processes, an unprivileged user is required. If a Zabbix daemon is started from an unprivileged user account, it will run as that user.

However, if a daemon is started from a 'root' account, it will switch to a 'zabbix' user account, which must be present. To create such a user account (in its own group, "zabbix"),

on a RedHat-based system, run:

```
groupadd --system zabbix
useradd --system -g zabbix -d /usr/lib/zabbix -s /sbin/nologin -c "Zabbix Monitoring System" zabbix
```

on a Debian-based system, run:

```
addgroup --system --quiet zabbix
adduser --quiet --system --disabled-login --ingroup zabbix --home /var/lib/zabbix --no-create-home zabbix
```

Attention:
Zabbix processes do not need a home directory, which is why we do not recommend creating it. However, if you are using some functionality that requires it (e. g. store MySQL credentials in `$HOME/.my.cnf`) you are free to create it using the following commands.

On RedHat-based systems, run:

```
mkdir -m u=rwx,g=rwx,o= -p /usr/lib/zabbix
chown zabbix:zabbix /usr/lib/zabbix
```

On Debian-based systems, run:

```
mkdir -m u=rwx,g=rwx,o= -p /var/lib/zabbix
chown zabbix:zabbix /var/lib/zabbix
```

A separate user account is not required for Zabbix frontend installation.

If Zabbix server and agent are run on the same machine it is recommended to use a different user for running the server than for running the agent. Otherwise, if both are run as the same user, the agent can access the server configuration file and any Admin level user in Zabbix can quite easily retrieve, for example, the database password.

Attention:
Running Zabbix as `root`, `bin`, or any other account with special rights is a security risk.

3 Create Zabbix database

For Zabbix server and proxy daemons, as well as Zabbix frontend, a database is required. It is not needed to run Zabbix agent.

SQL scripts are provided for creating database schema and inserting the dataset. Zabbix proxy database needs only the schema while Zabbix server database requires also the dataset on top of the schema.

Having created a Zabbix database, proceed to the following steps of compiling Zabbix.

4 Configure the sources

C99 with GNU extensions is required for building Zabbix server, Zabbix proxy or Zabbix agent. This version can be explicitly specified by setting CFLAGS="-std=gnu99":

```
export CFLAGS="-std=gnu99"
```

Note:
If installing from Zabbix Git repository, it is required to run first:

```
./bootstrap.sh
```

When configuring the sources for a Zabbix server or proxy, you must specify the database type to be used. Only one database type can be compiled with a server or proxy process at a time.

To see all of the supported configuration options, inside the extracted Zabbix source directory run:

```
./configure --help
```

To configure the sources for a Zabbix server and agent, you may run something like:

```
./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --wit
```

To configure the sources for a Zabbix server (with PostgreSQL etc.), you may run:

```
./configure --enable-server --with-postgresql --with-net-snmp
```

To configure the sources for a Zabbix proxy (with SQLite etc.), you may run:

```
./configure --prefix=/usr --enable-proxy --with-net-snmp --with-sqlite3 --with-ssh2
```

To configure the sources for a Zabbix agent, you may run:

```
./configure --enable-agent
```

or, for Zabbix agent 2:

```
./configure --enable-agent2
```

Note:
A configured Go environment with a currently supported Go version is required for building Zabbix agent 2. See go.dev for installation instructions.

Notes on compilation options:

- Command-line utilities zabbix_get and zabbix_sender are compiled if --enable-agent option is used.
- --with-libcurl and --with-libxml2 configuration options are required for virtual machine monitoring; --with-libcurl is also required for SMTP authentication and `web.page.*` Zabbix agent items. Note that cURL 7.20.0 or higher is required with the --with-libcurl configuration option.
- Zabbix always compiles with the PCRE library (since version 3.4.0); installing it is not optional. --with-libpcre=[DIR] only allows pointing to a specific base install directory, instead of searching through a number of common places for the libpcre files.
- You may use the --enable-static flag to statically link libraries. If you plan to distribute compiled binaries among different servers, you must use this flag to make these binaries work without required libraries. Note that --enable-static does not work in Solaris.
- Using --enable-static option is not recommended when building server. In order to build the server statically, you must have a static version of every external library needed. There is no strict check for that in configure script.
- Add optional path to the MySQL configuration file --with-mysql=/<path_to_the_file>/mysql_config to select the desired MySQL client library when there is a need to use one that is not located in the default location. It is useful when there are several versions of MySQL installed or MariaDB installed alongside MySQL on the same system.
- Use --with-oracle flag to specify location of the OCI API.

Attention:
If ./configure fails due to missing libraries or some other circumstance, please see the `config.log` file for more details on the error. For example, if `libssl` is missing, the immediate error message may be misleading:

```
checking for main in -lmysqlclient... no
configure: error: Not found mysqlclient library
```

While `config.log` has a more detailed description:

```
/usr/bin/ld: cannot find -lssl
/usr/bin/ld: cannot find -lcrypto
```

See also:

- Compiling Zabbix with encryption support for encryption support
- Known issues with compiling Zabbix agent on HP-UX

5 Make and install everything

> **Note:**
> If installing from Zabbix Git repository, it is required to run first:
>
> ```
> $ make dbschema
> ```

```
make install
```

This step should be run as a user with sufficient permissions (commonly 'root', or by using `sudo`).

Running `make install` will by default install the daemon binaries (zabbix_server, zabbix_agentd, zabbix_proxy) in /usr/local/sbin and the client binaries (zabbix_get, zabbix_sender) in /usr/local/bin.

> **Note:**
> To specify a different location than /usr/local, use a --prefix key in the previous step of configuring sources, for example --prefix=/home/zabbix. In this case daemon binaries will be installed under <prefix>/sbin, while utilities under <prefix>/bin. Man pages will be installed under <prefix>/share.

6 Review and edit configuration files

- edit the Zabbix agent configuration file **/usr/local/etc/zabbix_agentd.conf**

You need to configure this file for every host with zabbix_agentd installed.

You must specify the Zabbix server **IP address** in the file. Connections from other hosts will be denied.

- edit the Zabbix server configuration file **/usr/local/etc/zabbix_server.conf**

You must specify the database name, user and password (if using any).

The rest of the parameters will suit you with their defaults if you have a small installation (up to ten monitored hosts). You should change the default parameters if you want to maximize the performance of Zabbix server (or proxy) though.

- if you have installed a Zabbix proxy, edit the proxy configuration file **/usr/local/etc/zabbix_proxy.conf**

You must specify the server IP address and proxy hostname (must be known to the server), as well as the database name, user and password (if using any).

> **Note:**
> With SQLite the full path to database file must be specified; DB user and password are not required.

7 Start up the daemons

Run zabbix_server on the server side.

```
shell> zabbix_server
```

> **Note:**
> Make sure that your system allows allocation of 36MB (or a bit more) of shared memory, otherwise the server may not start and you will see "Cannot allocate shared memory for <type of cache>." in the server log file. This may happen on FreeBSD, Solaris 8.

Run zabbix_agentd on all the monitored machines.

```
shell> zabbix_agentd
```

> **Note:**
> Make sure that your system allows allocation of 2MB of shared memory, otherwise the agent may not start and you will see "Cannot allocate shared memory for collector." in the agent log file. This may happen on Solaris 8.

If you have installed Zabbix proxy, run zabbix_proxy.

```
shell> zabbix_proxy
```

2 Installing Zabbix web interface

Copying PHP files

Zabbix frontend is written in PHP, so to run it a PHP supported webserver is needed. Installation is done by simply copying the PHP files from the ui directory to the webserver HTML documents directory.

Common locations of HTML documents directories for Apache web servers include:

- /usr/local/apache2/htdocs (default directory when installing Apache from source)
- /srv/www/htdocs (OpenSUSE, SLES)
- /var/www/html (Debian, Ubuntu, Fedora, RHEL)

It is suggested to use a subdirectory instead of the HTML root. To create a subdirectory and copy Zabbix frontend files into it, execute the following commands, replacing the actual directory:

```
mkdir <htdocs>/zabbix
cd ui
cp -a . <htdocs>/zabbix
```

If planning to use any other language than English, see Installation of additional frontend languages for instructions.

Installing frontend

Please see Web interface installation page for information about Zabbix frontend installation wizard.

3 Installing Java gateway

It is required to install Java gateway only if you want to monitor JMX applications. Java gateway is lightweight and does not require a database.

To install from sources, first download and extract the source archive.

To compile Java gateway, run the `./configure` script with `--enable-java` option. It is advisable that you specify the `--prefix` option to request installation path other than the default /usr/local, because installing Java gateway will create a whole directory tree, not just a single executable.

```
$ ./configure --enable-java --prefix=$PREFIX
```

To compile and package Java gateway into a JAR file, run `make`. Note that for this step you will need `javac` and `jar` executables in your path.

```
$ make
```

Now you have a zabbix-java-gateway-$VERSION.jar file in src/zabbix_java/bin. If you are comfortable with running Java gateway from src/zabbix_java in the distribution directory, then you can proceed to instructions for configuring and running Java gateway. Otherwise, make sure you have enough privileges and run `make install`.

```
$ make install
```

Proceed to setup for more details on configuring and running Java gateway.

4 Installing Zabbix web service

Installing Zabbix web service is only required if you want to use scheduled reports.

To install from sources, first download and extract the source archive.

To compile Zabbix web service, run the `./configure` script with `--enable-webservice` option.

> **Note:**
> A configured Go version 1.13+ environment is required for building Zabbix web service.

Run zabbix_web_service on the machine, where the web service is installed:

```
shell> zabbix_web_service
```

Proceed to setup for more details on configuring Scheduled reports generation.

1 Building Zabbix agent on Windows

Overview

This section demonstrates how to build Zabbix Windows agent binaries from sources with or without TLS.

Compiling OpenSSL

The following steps will help you to compile OpenSSL from sources on MS Windows 10 (64-bit).

1. For compiling OpenSSL you will need on Windows machine:
    1. C compiler (e.g. VS 2017 RC),
    2. NASM (https://www.nasm.us/),
    3. Perl (e.g. Strawberry Perl from http://strawberryperl.com/),
    4. Perl module Text::Template (cpan Text::Template).
2. Get OpenSSL sources from https://www.openssl.org/. OpenSSL 1.1.1 is used here.
3. Unpack OpenSSL sources, for example, in E:\openssl-1.1.1.
4. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 RC.
5. Go to the OpenSSL source directory, e.g. E:\openssl-1.1.1.
    1. Verify that NASM can be found:`e:\openssl-1.1.1> nasm --version NASM version 2.13.01 compiled on May 1 2017`
6. Configure OpenSSL, for example:`e:\openssl-1.1.1> perl E:\openssl-1.1.1\Configure VC-WIN64A no-shared no-capieng no-srp no-gost no-dgram no-dtls1-method no-dtls1_2-method --api=1.1.0 --prefix=C:\OpenSSL --openssldir=C:\OpenSSL-Win64-111-static`
    - Make sure to revoke write access from non-administrator users to the OpenSSL install directory (`C:\OpenSSL-Win64-111-stati`). Otherwise, Zabbix agent will load SSL settings from a path that can be modified by unprivileged users, resulting in a potential security vulnerability.
    - Note the option 'no-shared': if 'no-shared' is used then the OpenSSL static libraries libcrypto.lib and libssl.lib will be 'self-sufficient' and resulting Zabbix binaries will include OpenSSL in themselves, no need for external OpenSSL DLLs. Advantage: Zabbix binaries can be copied to other Windows machines without OpenSSL libraries. Disadvantage: when a new OpenSSL bugfix version is released, Zabbix agent needs to recompiled and reinstalled.
    - If 'no-shared' is not used, then the static libraries libcrypto.lib and libssl.lib will be using OpenSSL DLLs at runtime. Advantage: when a new OpenSSL bugfix version is released, probably you can upgrade only OpenSSL DLLs, without recompiling Zabbix agent. Disadvantage: copying Zabbix agent to another machine requires copying OpenSSL DLLs, too.
7. Compile OpenSSL, run tests, install:`e:\openssl-1.1.1> nmake e:\openssl-1.1.1> nmake test ... All tests successful. Files=152, Tests=1152, 501 wallclock secs ( 0.67 usr + 0.61 sys = 1.28 CPU) Result: PASS e:\openssl-1.1.1> nmake install_sw`'install_sw' installs only software components (i.e. libraries, header files, but no documentation). If you want everything, use "nmake install".

Compiling PCRE

1. Download the PCRE or PCRE2 (supported since Zabbix 6.0) library (https://pcre.org/).
2. Extract to directory *E:\pcre2-10.39*.
3. Install CMake from https://cmake.org/download/, during install select: and ensure that cmake\bin is on your path (tested version 3.9.4).
4. Create a new, empty build directory, preferably a subdirectory of the source dir. For example, *E:\pcre2-10.39\build*.
5. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 and from that shell environment run cmake-gui. Do not try to start Cmake from the Windows Start menu, as this can lead to errors.
6. Enter *E:\pcre2-10.39* and *E:\pcre2-10.39\build* for the source and build directories, respectively.
7. Hit the "Configure" button.
8. When specifying the generator for this project select "NMake Makefiles".
9. Create a new, empty install directory. For example, *E:\pcre2-10.39-install*.
10. The GUI will then list several configuration options. Make sure the following options are selected:
    - **PCRE_SUPPORT_UNICODE_PROPERTIES** ON
    - **PCRE_SUPPORT_UTF** ON
    - **CMAKE_INSTALL_PREFIX** *E:\pcre2-10.39-install*
11. Hit "Configure" again. The adjacent "Generate" button should now be active.
12. Hit "Generate".
13. In the event that errors occur, it is recommended that you delete the CMake cache before attempting to repeat the CMake build process. In the CMake GUI, the cache can be deleted by selecting "File > Delete Cache".
14. The build directory should now contain a usable build system - *Makefile*.
15. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 and navigate to the *Makefile* mentioned above.
16. Run NMake command: `E:\pcre2-10.39\build> nmake install`

Compiling Zabbix

The following steps will help you to compile Zabbix from sources on MS Windows 10 (64-bit). When compiling Zabbix with/without TLS support the only significant difference is in step 4.

1. On a Linux machine check out the source from git:`$ git clone https://git.zabbix.com/scm/zbx/zabbix.git $ cd zabbix $ ./bootstrap.sh $ ./configure --enable-agent --enable-ipv6 --prefix=`pwd` $ make dbschema $ make dist`
2. Copy and unpack the archive, e.g. zabbix-4.4.0.tar.gz, on a Windows machine.

3. Let's assume that sources are in e:\zabbix-4.4.0. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 RC. Go to E:\zabbix-4.4.0\build\win32\project.
4. Compile zabbix_get, zabbix_sender and zabbix_agent.
    - without TLS: `E:\zabbix-4.4.0\build\win32\project> nmake /K PCRE2INCDIR=E:\pcre2-10.39-install\include PCRE2LIBDIR=E:\pcre2-10.39-install\lib`
    - with TLS: `E:\zabbix-4.4.0\build\win32\project> nmake /K -f Makefile_get TLS=openssl TLSINCDIR=C:\OpenSSL-Win64-111-static\include TLSLIBDIR=C:\OpenSSL-Win64-111-static\lib PCRE2INCDIR=E:\pcre2-10.39-install\include PCRE2LIBDIR=E:\pcre2-10.39-install\lib E:\zabbix-4.4.0\build\win32\project> nmake /K -f Makefile_sender TLS=openssl TLSINCDIR="C:\OpenSSL-Win64-111-static\include TLSLIBDIR="C:\OpenSSL-Win64-111-static\lib PCRE2INCDIR=E:\pcre2-10.39-install\include PCRE2LIBDIR=E:\pcre2-10.39-install\lib E:\zabbix-4.4.0\build\win32\project> nmake /K -f Makefile_agent TLS=openssl TLSINCDIR=C:\OpenSSL-Win64-111-static\include TLSLIBDIR=C:\OpenSSL-Win64-111-static\lib PCRE2INCDIR=E:\pcre2-10.39-install\include PCRE2LIBDIR=E:\pcre2-10.39-install\lib`
5. New binaries are located in e:\zabbix-4.4.0\bin\win64. Since OpenSSL was compiled with 'no-shared' option, Zabbix binaries contain OpenSSL within themselves and can be copied to other machines that do not have OpenSSL.

Compiling Zabbix with LibreSSL

The process is similar to compiling with OpenSSL, but you need to make small changes in files located in the `build\win32\project` directory:

- In Makefile_tls delete `/DHAVE_OPENSSL_WITH_PSK`, i.e. find:

```
CFLAGS = $(CFLAGS) /DHAVE_OPENSSL /DHAVE_OPENSSL_WITH_PSK
```

and replace it with

```
CFLAGS = $(CFLAGS) /DHAVE_OPENSSL
```

- In Makefile_common.inc add `/NODEFAULTLIB:LIBCMT` i.e. find:

```
/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO /PDB:$(TARGETDIR)\$(TARGETNAME).pdb
```

and replace it with

```
/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO /PDB:$(TARGETDIR)\$(TARGETNAME).pdb /NOD
```

2 Building Zabbix agent 2 on Windows

Overview

This section demonstrates how to build Zabbix agent 2 (Windows) from sources.

Installing MinGW Compiler

1. Download MinGW-w64 with SJLJ (set jump/long jump) Exception Handling and Windows threads (for example *x86_64-8.1.0-release-win32-sjlj-rt_v6-rev0.7z*)
2. Extract and move to *c:\mingw*
3. Setup environmental variable

```
@echo off
set PATH=%PATH%;c:\mingw\bin
cmd
```

When compiling use Windows prompt instead of MSYS terminal provided by MinGW

Compiling PCRE development libraries

The following instructions will compile and install 64-bit PCRE libraries in *c:\dev\pcre* and 32-bit libraries in *c:\dev\pcre32*:

1. Download the PCRE or PCRE2 (supported since Zabbix 6.0) library (https://pcre.org/) and extract
2. Open *cmd* and navigate to the extracted sources

Build 64bit PCRE

1. Delete old configuration/cache if exists:

```
del CMakeCache.txt
rmdir /q /s CMakeFiles
```

2. Run cmake (CMake can be installed from https://cmake.org/download/):

```
cmake -G "MinGW Makefiles" -DCMAKE_C_COMPILER=gcc -DCMAKE_C_FLAGS="-O2 -g" -DCMAKE_CXX_FLAGS="-O2 -g" -DCM
```

3. Next, run:

```
mingw32-make clean
mingw32-make install
```

Build 32bit PCRE

1. Run:

```
mingw32-make clean
```

2. Delete *CMakeCache.txt*:

```
del CMakeCache.txt
rmdir /q /s CMakeFiles
```

3. Run cmake:

```
cmake -G "MinGW Makefiles" -DCMAKE_C_COMPILER=gcc -DCMAKE_C_FLAGS="-m32 -O2 -g" -DCMAKE_CXX_FLAGS="-m32 -O
```

4. Next, run:

```
mingw32-make install
```

Building OpenSSL libraries using MinGW

1. If you don't have it installed already, install *Cygwin* and *Perl for Cygwin* and configure them according to instructions in the official Perl documentation.
2. Run:

```
perl Configure mingw64 no-shared no-ui-console no-tests no-capieng --libdir=lib --api=1.1.0 --prefix=c:\de

make build_sw

make install_dev
```

Make sure to revoke write access from non-administrator users to the OpenSSL install directory (`C:\dev\openssl32` or `C:\dev\openssl`). Otherwise, Zabbix agent 2 will load SSL settings from a path that can be modified by unprivileged users, resulting in a potential security vulnerability.

Compiling Zabbix agent 2

32 bit

Open MinGW environment (Windows command prompt) and navigate to *build/mingw* directory in the Zabbix source tree.

Run:

```
mingw32-make clean
mingw32-make ARCH=x86 PCRE=c:\dev\pcre32 OPENSSL=c:\dev\openssl32
```

64 bit

Open MinGW environment (Windows command prompt) and navigate to *build/mingw* directory in the Zabbix source tree.

Run:

```
mingw32-make clean
mingw32-make PCRE=c:\dev\pcre OPENSSL=c:\dev\openssl
```

> **Note:**
> Both 32- and 64- bit versions can be built on a 64-bit platform, but only a 32-bit version can be built on a 32-bit platform. When working on the 32-bit platform, follow the same steps as for 64-bit version on 64-bit platform.

3 Building Zabbix agent on macOS

Overview

This section demonstrates how to build Zabbix macOS agent binaries from sources with or without TLS.

Prerequisites

You will need command line developer tools (Xcode is not required), Automake, pkg-config and PCRE (v8.x) or PCRE2 (v10.x). If you want to build agent binaries with TLS, you will also need OpenSSL or GnuTLS.

To install Automake and pkg-config, you will need a Homebrew package manager from https://brew.sh/. To install it, open terminal and run the following command:

```
$ /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Then install Automake and pkg-config:

```
$ brew install automake
$ brew install pkg-config
```

Preparing PCRE, OpenSSL and GnuTLS libraries depends on the way how they are going to be linked to the agent.

If you intend to run agent binaries on a macOS machine that already has these libraries, you can use precompiled libraries that are provided by Homebrew. These are typically macOS machines that use Homebrew for building Zabbix agent binaries or for other purposes.

If agent binaries will be used on macOS machines that don't have the shared version of libraries, you should compile static libraries from sources and link Zabbix agent with them.

Building agent binaries with shared libraries

Install PCRE2 (replace *pcre2* with *pcre* in the commands below, if needed):

```
$ brew install pcre2
```

When building with TLS, install OpenSSL and/or GnuTLS:

```
$ brew install openssl
$ brew install gnutls
```

Download Zabbix source:

```
$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
```

Build agent without TLS:

```
$ cd zabbix
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6
$ make
$ make install
```

Build agent with OpenSSL:

```
$ cd zabbix
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-openssl=/usr/local/op
$ make
$ make install
```

Build agent with GnuTLS:

```
$ cd zabbix-source/
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-gnutls=/usr/local/opt
$ make
$ make install
```

Building agent binaries with static libraries without TLS

Let's assume that PCRE static libraries will be installed in `$HOME/static-libs`. We will use PCRE2 10.39.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre2-10.39"
```

Download and build PCRE with Unicode properties support:

```
$ mkdir static-libs-source
$ cd static-libs-source
$ curl --remote-name https://github.com/PhilipHazel/pcre2/releases/download/pcre2-10.39/pcre2-10.39.tar.gz
$ tar xf pcre2-10.39.tar.gz
$ cd pcre2-10.39
$ ./configure --prefix="$PCRE_PREFIX" --disable-shared --enable-static --enable-unicode-properties
$ make
$ make check
$ make install
```

Download Zabbix source and build agent:

```
$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
$ cd zabbix
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre2="$PCRE_PREFI
$ make
$ make install
```

Building agent binaries with static libraries with OpenSSL

When building OpenSSL, it's recommended to run `make test` after successful building. Even if building was successful, tests sometimes fail. If this is the case, problems should be researched and resolved before continuing.

Let's assume that PCRE and OpenSSL static libraries will be installed in `$HOME/static-libs`. We will use PCRE2 10.39 and OpenSSL 1.1.1a.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre2-10.39"
$ OPENSSL_PREFIX="$HOME/static-libs/openssl-1.1.1a"
```

Let's build static libraries in `static-libs-source`:

```
$ mkdir static-libs-source
$ cd static-libs-source
```

Download and build PCRE with Unicode properties support:

```
$ curl --remote-name https://github.com/PhilipHazel/pcre2/releases/download/pcre2-10.39/pcre2-10.39.tar.gz
$ tar xf pcre2-10.39.tar.gz
$ cd pcre2-10.39
$ ./configure --prefix="$PCRE_PREFIX" --disable-shared --enable-static --enable-unicode-properties
$ make
$ make check
$ make install
$ cd ..
```

Download and build OpenSSL:

```
$ curl --remote-name https://www.openssl.org/source/openssl-1.1.1a.tar.gz
$ tar xf openssl-1.1.1a.tar.gz
$ cd openssl-1.1.1a
$ ./Configure --prefix="$OPENSSL_PREFIX" --openssldir="$OPENSSL_PREFIX" --api=1.1.0 no-shared no-capieng n
$ make
$ make test
$ make install_sw
$ cd ..
```

Download Zabbix source and build agent:

```
$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
$ cd zabbix
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre2="$PCRE_PREFI
$ make
$ make install
```

Building agent binaries with static libraries with GnuTLS

GnuTLS depends on the Nettle crypto backend and GMP arithmetic library. Instead of using full GMP library, this guide will use mini-gmp which is included in Nettle.

When building GnuTLS and Nettle, it's recommended to run `make check` after successful building. Even if building was successful, tests sometimes fail. If this is the case, problems should be researched and resolved before continuing.

Let's assume that PCRE, Nettle and GnuTLS static libraries will be installed in `$HOME/static-libs`. We will use PCRE2 10.39, Nettle 3.4.1 and GnuTLS 3.6.5.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre2-10.39"
$ NETTLE_PREFIX="$HOME/static-libs/nettle-3.4.1"
$ GNUTLS_PREFIX="$HOME/static-libs/gnutls-3.6.5"
```

Let's build static libraries in `static-libs-source`:

```
$ mkdir static-libs-source
$ cd static-libs-source
```

Download and build Nettle:

```
$ curl --remote-name https://ftp.gnu.org/gnu/nettle/nettle-3.4.1.tar.gz
$ tar xf nettle-3.4.1.tar.gz
$ cd nettle-3.4.1
$ ./configure --prefix="$NETTLE_PREFIX" --enable-static --disable-shared --disable-documentation --disable
$ make
$ make check
$ make install
$ cd ..
```

Download and build GnuTLS:

```
$ curl --remote-name https://www.gnupg.org/ftp/gcrypt/gnutls/v3.6/gnutls-3.6.5.tar.xz
$ tar xf gnutls-3.6.5.tar.xz
$ cd gnutls-3.6.5
$ PKG_CONFIG_PATH="$NETTLE_PREFIX/lib/pkgconfig" ./configure --prefix="$GNUTLS_PREFIX" --enable-static --d
$ make
$ make check
$ make install
$ cd ..
```

Download Zabbix source and build agent:

```
$ git clone https://git.zabbix.com/scm/zbx/zabbix.git
$ cd zabbix
$ ./bootstrap.sh
$ CFLAGS="-Wno-unused-command-line-argument -framework Foundation -framework Security" \
> LIBS="-lgnutls -lhogweed -lnettle" \
> LDFLAGS="-L$GNUTLS_PREFIX/lib -L$NETTLE_PREFIX/lib" \
> ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre2="$PCRE_PREFI
$ make
$ make install
```

4 Installation from packages

From Zabbix official repository

Zabbix SIA provides official RPM and DEB packages for:

- Red Hat Enterprise Linux
- Debian/Ubuntu/Raspbian
- SUSE Linux Enterprise Server

Package files for yum/dnf, apt and zypper repositories for various OS distributions are available at repo.zabbix.com.

Some OS distributions (in particular, Debian-based distributions) provide their own Zabbix packages. Note that these packages are **not** supported by Zabbix. Third-party Zabbix packages can be out of date and may lack the latest features and bug fixes. It is recommended to use only the official packages from repo.zabbix.com. If you have previously used unofficial Zabbix packages, see notes about upgrading the Zabbix packages from OS repositories.

1 Red Hat Enterprise Linux

Overview

Official Zabbix 6.4 packages for Red Hat Enterprise Linux versions 6, 7, 8, and 9 and Oracle Linux versions 8 and 9 are available for download.

Attention:
Zabbix packages for Red Hat Enterprise Linux systems are intended only for RHEL systems. Alternative environments, such as Red Hat Universal Base Image, may lack the necessary dependencies and repository access requirements for successful installation. To address such issues, verify compatibility with the target environment and ensure access to required repositories and dependencies before proceeding with Zabbix installation from packages. For more information, see Known issues.

Packages are available with:

- MySQL or PostgreSQL database
- Apache or Nginx web server support

Note that separate packages for Zabbix agent, the Zabbix get and Zabbix sender utilities are available for RHEL 6, 7, 8, and 9. The same packages can be used to install Zabbix components on Alma Linux and Rocky Linux.

The official Zabbix repository provides `fping`, `iksemel` and `libssh2` packages as well. These packages are located in the non-supported directory.

Attention:
The EPEL repository for EL9 also provides Zabbix packages. If both the official Zabbix repository and EPEL repositories are installed, then the Zabbix packages in EPEL **must be** excluded by adding the following clause to the EPEL repo configuration file under `/etc/yum.repos.d/`:

```
[epel]
...
excludepkgs=zabbix*
```

See also: Accidental installation of EPEL Zabbix packages

Notes on installation

See installation instructions per platform in the download page for:

- installing the repository
- installing server/agent/frontend
- creating initial database, importing initial data
- configuring database for Zabbix server
- configuring PHP for Zabbix frontend
- starting server/agent processes
- configuring Zabbix frontend

If you want to run Zabbix agent as root, see Running agent as root.

Zabbix web service process, which is used for scheduled report generation, requires Google Chrome browser. The browser is not included into packages and has to be installed manually.

Importing data with Timescale DB

With TimescaleDB, in addition to the import command for PostgreSQL, also run:

```
cat /usr/share/zabbix-sql-scripts/postgresql/timescaledb.sql | sudo -u zabbix psql zabbix
```

Warning:
TimescaleDB is supported with Zabbix server only.

SELinux configuration

Zabbix uses socket-based inter-process communication. On systems where SELinux is enabled, it may be required to add SELinux rules to allow Zabbix create/use UNIX domain sockets in the SocketDir directory. Currently, socket files are used by server (alerter, preprocessing, IPMI) and proxy (IPMI). Socket files are persistent, meaning they are present while the process is running.

Having SELinux status enabled in enforcing mode, you need to execute the following commands to enable communication between Zabbix frontend and server:

RHEL 7 and later:

```
setsebool -P httpd_can_connect_zabbix on
```

If the database is accessible over network (including 'localhost' in case of PostgreSQL), you need to allow Zabbix frontend to connect to the database too:

```
setsebool -P httpd_can_network_connect_db on
```

RHEL prior to 7:

```
setsebool -P httpd_can_network_connect on
setsebool -P zabbix_can_network on
```

After the frontend and SELinux configuration is done, restart the Apache web server:

```
systemctl restart httpd
```

In addition, Zabbix provides the zabbix-selinux-policy package as part of source RPM packages for RHEL 7, RHEL 8, and RHEL 9. This package provides a basic default policy for SELinux and makes zabbix components work out-of-the-box by allowing Zabbix to create and use sockets and enabling httpd connection to PostgreSQL (used by frontend).

The source *zabbix_policy.te* file contains the following rules:

```
module zabbix_policy 1.2;

require {
  type zabbix_t;
  type zabbix_port_t;
  type zabbix_var_run_t;
  type postgresql_port_t;
  type httpd_t;
  class tcp_socket name_connect;
  class sock_file { create unlink };
  class unix_stream_socket connectto;
}

#============= zabbix_t ==============
allow zabbix_t self:unix_stream_socket connectto;
allow zabbix_t zabbix_port_t:tcp_socket name_connect;
allow zabbix_t zabbix_var_run_t:sock_file create;
allow zabbix_t zabbix_var_run_t:sock_file unlink;
allow httpd_t zabbix_port_t:tcp_socket name_connect;

#============= httpd_t ==============
allow httpd_t postgresql_port_t:tcp_socket name_connect;
```

This package has been created to prevent users from turning off SELinux because of the configuration complexity. It contains the default policy that is sufficient to speed up Zabbix deployment and configuration. For maximum security level, it is recommended to set custom SELinux settings.

Proxy installation

Once the required repository is added, you can install Zabbix proxy by running:

```
dnf install zabbix-proxy-mysql zabbix-sql-scripts
```

Substitute 'mysql' in the commands with 'pgsql' to use PostgreSQL, or with 'sqlite3' to use SQLite3 (proxy only).

The package 'zabbix-sql-scripts' contains database schemas for all supported database management systems for both Zabbix server and Zabbix proxy and will be used for data import.

Creating database

Create a separate database for Zabbix proxy.

Zabbix server and Zabbix proxy cannot use the same database. If they are installed on the same host, the proxy database must have a different name.

Importing data

Import initial schema:

```
cat /usr/share/zabbix-sql-scripts/mysql/proxy.sql | mysql -uzabbix -p zabbix
```

For proxy with PostgreSQL (or SQLite):

```
cat /usr/share/zabbix-sql-scripts/postgresql/proxy.sql | sudo -u zabbix psql zabbix
cat /usr/share/zabbix-sql-scripts/sqlite3/proxy.sql | sqlite3 zabbix.db
```

Configure database for Zabbix proxy

Edit Zabbix proxy configuration file (`/etc/zabbix/zabbix_proxy.conf`):

```
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=<password>
```

In DBName for Zabbix proxy use a separate database from Zabbix server.

In DBPassword use Zabbix database password for MySQL; PostgreSQL user password for PostgreSQL.

Use `DBHost=` with PostgreSQL. You might want to keep the default setting `DBHost=localhost` (or an IP address), but this would make PostgreSQL use a network socket for connecting to Zabbix. See SELinux configuration for instructions.

Starting Zabbix proxy process

To start a Zabbix proxy process and make it start at system boot:

```
systemctl start zabbix-proxy
systemctl enable zabbix-proxy
```

Frontend configuration

A Zabbix proxy does not have a frontend; it communicates with Zabbix server only.

Java gateway installation

It is required to install Java gateway only if you want to monitor JMX applications. Java gateway is lightweight and does not require a database.

Once the required repository is added, you can install Zabbix Java gateway by running:

```
dnf install zabbix-java-gateway
```

Proceed to setup for more details on configuring and running Java gateway.

Installing debuginfo packages

> **Note:**
> Debuginfo packages are currently available for RHEL versions 7 and 6.

To enable debuginfo repository, edit */etc/yum.repos.d/zabbix.repo* file. Change `enabled=0` to `enabled=1` for zabbix-debuginfo repository.

```
[zabbix-debuginfo]
name=Zabbix Official Repository debuginfo - $basearch
baseurl=http://repo.zabbix.com/zabbix/6.4/rhel/7/$basearch/debuginfo/
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-A14FE591
gpgcheck=1
```

This will allow you to install the zabbix-debuginfo package.

```
dnf install zabbix-debuginfo
```

This single package contains debug information for all binary Zabbix components.

2 Debian/Ubuntu/Raspbian

Overview

Official Zabbix 6.4 packages for Debian, Ubuntu, and Raspberry Pi OS (Raspbian) are available on Zabbix website.

Packages are available with either MySQL/PostgreSQL database and Apache/Nginx web server support.

Notes on installation

See the installation instructions per platform in the download page for:

- installing the repository
- installing server/agent/frontend
- creating initial database, importing initial data
- configuring database for Zabbix server
- configuring PHP for Zabbix frontend
- starting server/agent processes
- configuring Zabbix frontend

If you want to run Zabbix agent as root, see running agent as root.

Zabbix web service process, which is used for scheduled report generation, requires Google Chrome browser. The browser is not included into packages and has to be installed manually.

Importing data with Timescale DB

With TimescaleDB, in addition to the import command for PostgreSQL, also run:

```
cat /usr/share/zabbix-sql-scripts/postgresql/timescaledb.sql | sudo -u zabbix psql zabbix
```

Warning:
TimescaleDB is supported with Zabbix server only.

SELinux configuration

See SELinux configuration for RHEL.

After the frontend and SELinux configuration is done, restart the Apache web server:

```
systemctl restart apache2
```

Proxy installation

Once the required repository is added, you can install Zabbix proxy by running:

```
apt install zabbix-proxy-mysql zabbix-sql-scripts
```

Substitute 'mysql' in the command with 'pgsql' to use PostgreSQL, or with 'sqlite3' to use SQLite3.

The package 'zabbix-sql-scripts' contains database schemas for all supported database management systems for both Zabbix server and Zabbix proxy and will be used for data import.

Creating database

Create a separate database for Zabbix proxy.

Zabbix server and Zabbix proxy cannot use the same database. If they are installed on the same host, the proxy database must have a different name.

Importing data

Import initial schema:

```
cat /usr/share/zabbix-sql-scripts/mysql/proxy.sql | mysql -uzabbix -p zabbix
```

For proxy with PostgreSQL (or SQLite):

```
cat /usr/share/zabbix-sql-scripts/postgresql/proxy.sql | sudo -u zabbix psql zabbix
cat /usr/share/zabbix-sql-scripts/sqlite3/proxy.sql | sqlite3 zabbix.db
```

Configure database for Zabbix proxy

Edit Zabbix proxy configuration file (`/etc/zabbix/zabbix_proxy.conf`):

```
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=<password>
```

In DBName for Zabbix proxy use a separate database from Zabbix server.

In DBPassword use Zabbix database password for MySQL; PostgreSQL user password for PostgreSQL.

Use `DBHost=` with PostgreSQL. You might want to keep the default setting `DBHost=localhost` (or an IP address), but this would make PostgreSQL use a network socket for connecting to Zabbix. Refer to the respective section for RHEL for instructions.

Starting Zabbix proxy process

To start a Zabbix proxy process and make it start at system boot:

```
systemctl restart zabbix-proxy
systemctl enable zabbix-proxy
```

Frontend configuration

A Zabbix proxy does not have a frontend; it communicates with Zabbix server only.

Java gateway installation

It is required to install Java gateway only if you want to monitor JMX applications. Java gateway is lightweight and does not require a database.

Once the required repository is added, you can install Zabbix Java gateway by running:

```
apt install zabbix-java-gateway
```

Proceed to setup for more details on configuring and running Java gateway.

3 SUSE Linux Enterprise Server

Overview

Official Zabbix 6.4 packages for SUSE Linux Enterprise Server are available on Zabbix website.

Zabbix agent packages and utilities *Zabbix get* and *Zabbix sender* are available in Zabbix Official Repository for SLES 15 (SP4 and newer) and SLES 12 (SP4 and newer).

Please note that on SLES 12 the following features are not available:

- *Verify CA* encryption mode with MySQL does not work due to older MySQL libraries.
- Since Zabbix 6.4, *SSH checks* are not supported for both proxy and server because of the older libssh version.

Adding Zabbix repository

Install the repository configuration package. This package contains yum (software package manager) configuration files.

SLES 15:

```
rpm -Uvh --nosignature https://repo.zabbix.com/zabbix/6.4/sles/15/x86_64/zabbix-release-latest.sles15.noar
zypper --gpg-auto-import-keys refresh 'Zabbix Official Repository'
```

SLES 12:

```
rpm -Uvh --nosignature https://repo.zabbix.com/zabbix/6.4/sles/12/x86_64/zabbix-release-latest.sles12.noar
zypper --gpg-auto-import-keys refresh 'Zabbix Official Repository'
```

Please note that Zabbix web service process, which is used for scheduled report generation, requires Google Chrome browser. The browser is not included into packages and has to be installed manually.

Server/frontend/agent installation

To install Zabbix server/frontend/agent with PHP 7, Apache and MySQL support, run:

```
zypper install zabbix-server-mysql zabbix-web-mysql zabbix-apache-conf zabbix-agent
```

Substitute component names in this command as needed:

- **For Nginx**: use `zabbix-nginx-conf` instead of `zabbix-apache-conf`. See also: Nginx setup for Zabbix on SLES 12/15.
- **For PHP 8**: use `zabbix-apache-conf-php8` instead of `zabbix-apache-conf` for Apache; use `zabbix-nginx-conf-php8` instead of `zabbix-nginx-conf` for Nginx.
- **For PostgreSQL**: use `zabbix-server-pgsql` instead of `zabbix-server-mysql`; use `zabbix-web-pgsql` instead of `zabbix-web-mysql`.
- **For Zabbix agent 2** (only SLES 15): use `zabbix-agent2` instead of or in addition to `zabbix-agent`.

To install Zabbix proxy with MySQL support:

```
zypper install zabbix-proxy-mysql zabbix-sql-scripts
```

For PostgreSQL, use `zabbix-proxy-pgsql` instead of `zabbix-proxy-mysql`.

For SQLite3, use `zabbix-proxy-sqlite3` instead of `zabbix-proxy-mysql`.

The package 'zabbix-sql-scripts' contains database schemas for all supported database management systems for both Zabbix server and Zabbix proxy and will be used for data import.

Creating database

Zabbix server and proxy daemons require a database. Zabbix agent does not need a database.

To create a database, follow the instructions for MySQL or PostgreSQL. An SQLite3 database (supported for Zabbix proxy only) will be created automatically and does not require additional installation steps.

Warning:
Separate databases are required for Zabbix server and Zabbix proxy; they cannot share the same database. If a server and a proxy are installed on the same host, their databases must be created with different names!

Importing data

Now import initial schema and data for the **server** with MySQL:

```
zcat /usr/share/packages/zabbix-sql-scripts/mysql/create.sql.gz | mysql -uzabbix -p zabbix
```

You will be prompted to enter your newly created database password.

With PostgreSQL:

```
zcat /usr/share/packages/zabbix-sql-scripts/postgresql/create.sql.gz | sudo -u zabbix psql zabbix
```

With TimescaleDB, in addition to the previous command, also run:

```
zcat /usr/share/packages/zabbix-sql-scripts/postgresql/timescaledb.sql.gz | sudo -u <username> psql zabbix
```

Warning:
TimescaleDB is supported with Zabbix server only.

For proxy, import initial schema:

```
zcat /usr/share/packages/zabbix-sql-scripts/mysql/schema.sql.gz | mysql -uzabbix -p zabbix
```

For proxy with PostgreSQL:

```
zcat /usr/share/packages/zabbix-sql-scripts/postgresql/schema.sql.gz | sudo -u zabbix psql zabbix
```

Configure database for Zabbix server/proxy

Edit Zabbix server configuration file (`/etc/zabbix/zabbix_server.conf`) and, if required, Zabbix proxy configuration file (`/etc/zabbix/zabbix_proxy.conf`) for their respective databases. For example:

```
DBHost=localhost
DBName=zabbix
DBUser=zabbix
DBPassword=<password>
```

In DBPassword use Zabbix database password for MySQL; PostgreSQL user password for PostgreSQL.

Use `DBHost=` with PostgreSQL. You might want to keep the default setting `DBHost=localhost` (or an IP address), but this would make PostgreSQL use a network socket for connecting to Zabbix.

Zabbix frontend configuration

Depending on the web server used (Apache/Nginx), edit the corresponding configuration file for Zabbix frontend. While some PHP settings may already be configured, it's essential that you uncomment the `date.timezone` setting and specify the appropriate timezone setting that suits your requirements.

- For Apache the configuration file is located in `/etc/apache2/conf.d/zabbix.conf`.

```
php_value max_execution_time 300
php_value memory_limit 128M
php_value post_max_size 16M
php_value upload_max_filesize 2M
php_value max_input_time 300
php_value max_input_vars 10000
php_value always_populate_raw_post_data -1
# php_value date.timezone Europe/Riga
```

- The zabbix-nginx-conf package installs a separate Nginx server for Zabbix frontend. Its configuration file is located in `/etc/nginx/conf.d/zabbix.conf`. For Zabbix frontend to work, it's necessary to uncomment and set `listen` and/or `server_name` directives.

```
# listen 80;
# server_name example.com;
```

- Zabbix uses its own dedicated php-fpm connection pool with Nginx:

Its configuration file is located in `/etc/php7/fpm/php-fpm.d/zabbix.conf` (the path may vary slightly depending on the service pack).

```
php_value[max_execution_time] = 300
php_value[memory_limit] = 128M
php_value[post_max_size] = 16M
php_value[upload_max_filesize] = 2M
php_value[max_input_time] = 300
php_value[max_input_vars] = 10000
; php_value[date.timezone] = Europe/Riga
```

Now you are ready to proceed with frontend installation steps that will allow you to access your newly installed Zabbix.

Note that a Zabbix proxy does not have a frontend; it communicates with Zabbix server only.

Starting Zabbix server/agent process

Start Zabbix server and agent processes and make it start at system boot.

With Apache web server:

```
systemctl restart zabbix-server zabbix-agent apache2 php-fpm
systemctl enable zabbix-server zabbix-agent apache2 php-fpm
```

For Nginx, substitute `apache2` with `nginx`.

Installing debuginfo packages

To enable debuginfo repository edit */etc/zypp/repos.d/zabbix.repo* file. Change `enabled=0` to `enabled=1` for zabbix-debuginfo repository.

```
[zabbix-debuginfo]
name=Zabbix Official Repository debuginfo
type=rpm-md
baseurl=http://repo.zabbix.com/zabbix/6.4/sles/15/x86_64/debuginfo/
gpgcheck=1
gpgkey=http://repo.zabbix.com/zabbix/6.4/sles/15/x86_64/debuginfo/repodata/repomd.xml.key
enabled=0
update=1
```

This will allow you to install zabbix-***<component>***-debuginfo packages.

4 Windows agent installation from MSI

Overview

Zabbix Windows agent can be installed from Windows MSI installer packages (32-bit or 64-bit) available for download.

A 32-bit package cannot be installed on a 64-bit Windows.

The minimum requirement for MSI installation is:

- Windows XP 64-bit and Server 2003 for Zabbix agent;
- Windows 10 32-bit and Server 2016 for Zabbix agent 2.

The Zabbix get and sender utilities can also be installed, either together with Zabbix agent/agent 2 or separately.

All packages come with TLS support, however, configuring TLS is optional.

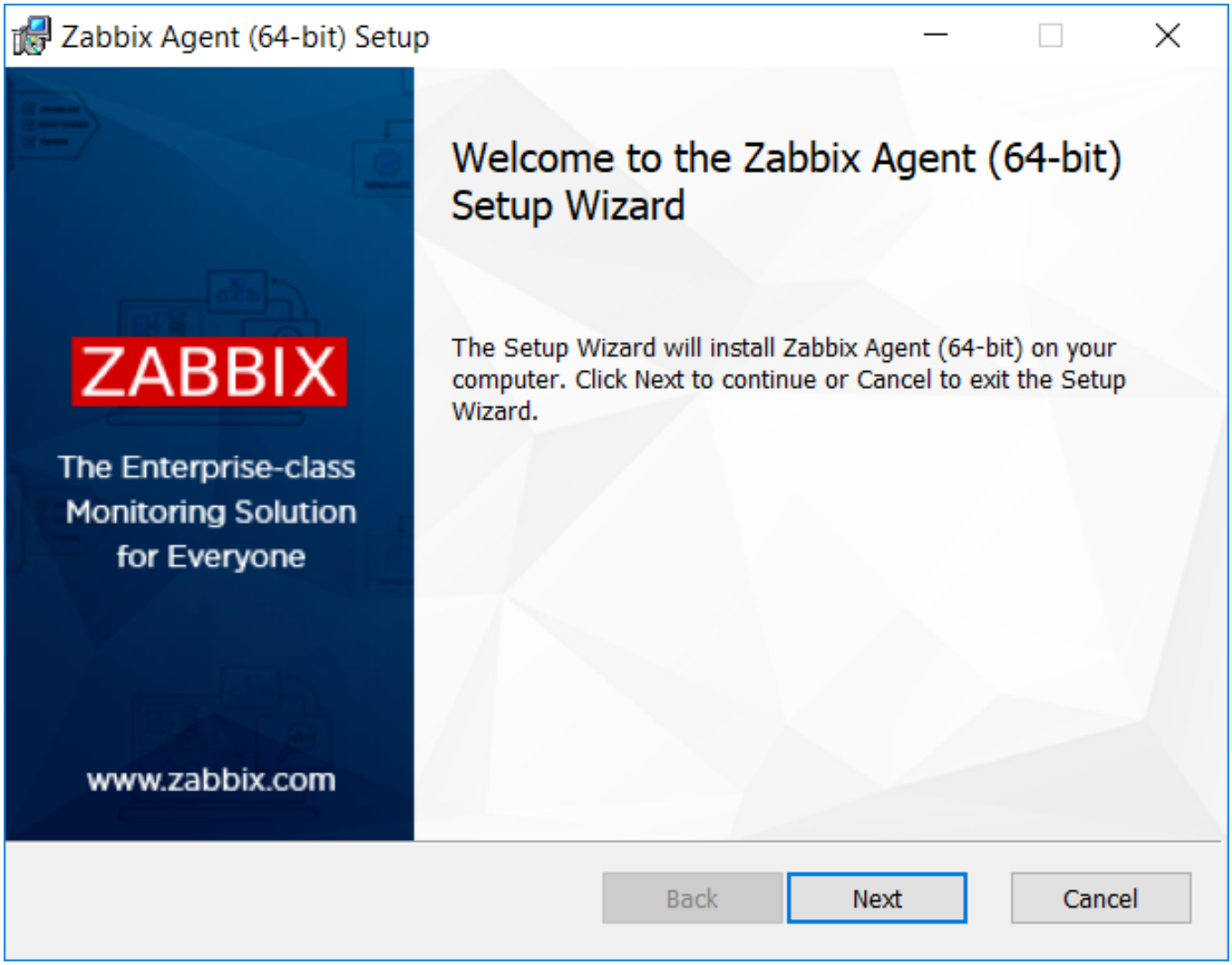
Both UI and command-line based installation is supported.

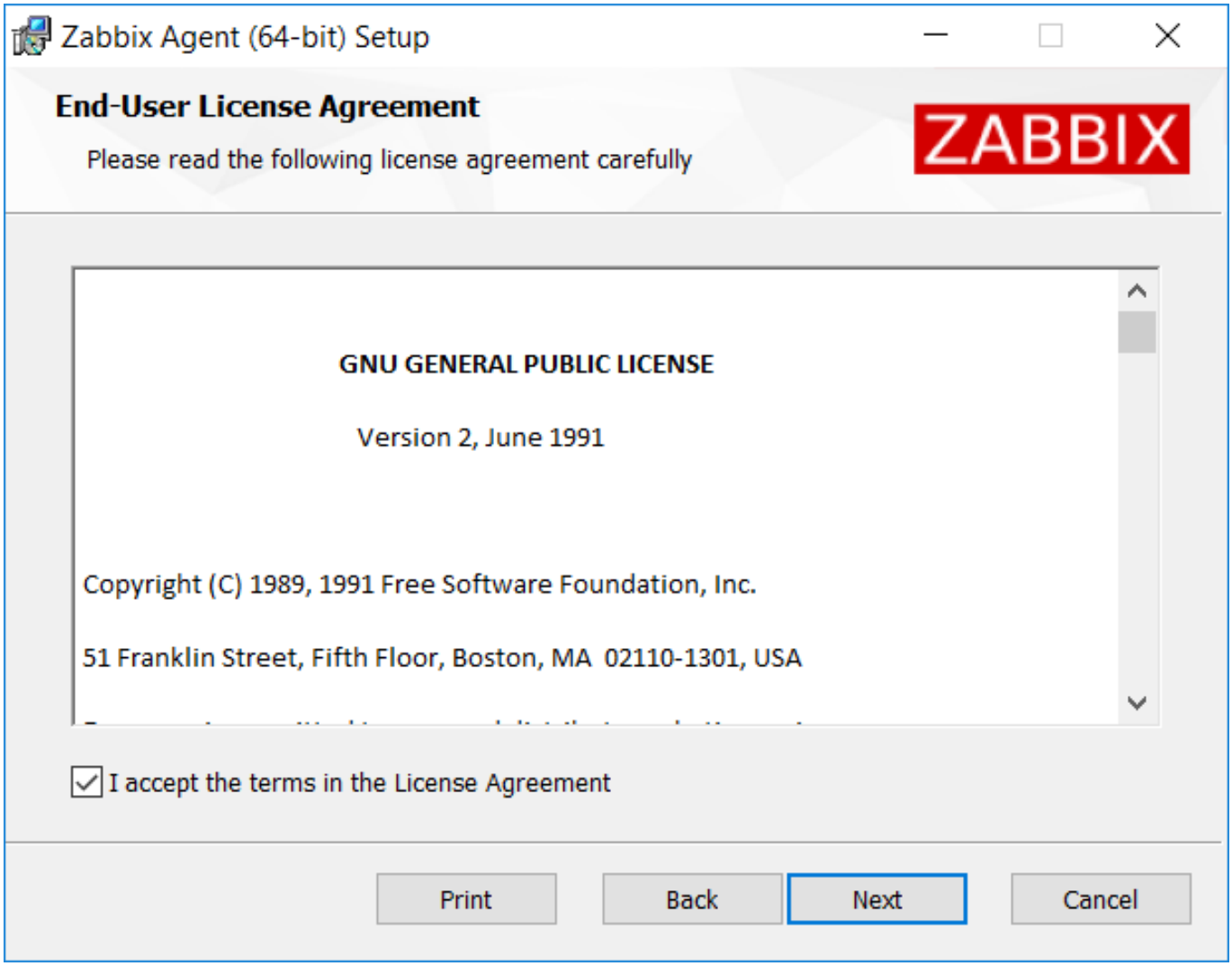
Note:
Although Zabbix installation from MSI installer packages is fully supported, it is recommended to install at least *Microsoft .NET Framework 2* for proper error handling. See Microsoft Download .NET Framework.

Attention:
It is recommended to use default paths provided by the installer as using custom paths without proper permissions could compromise the security of the installation.

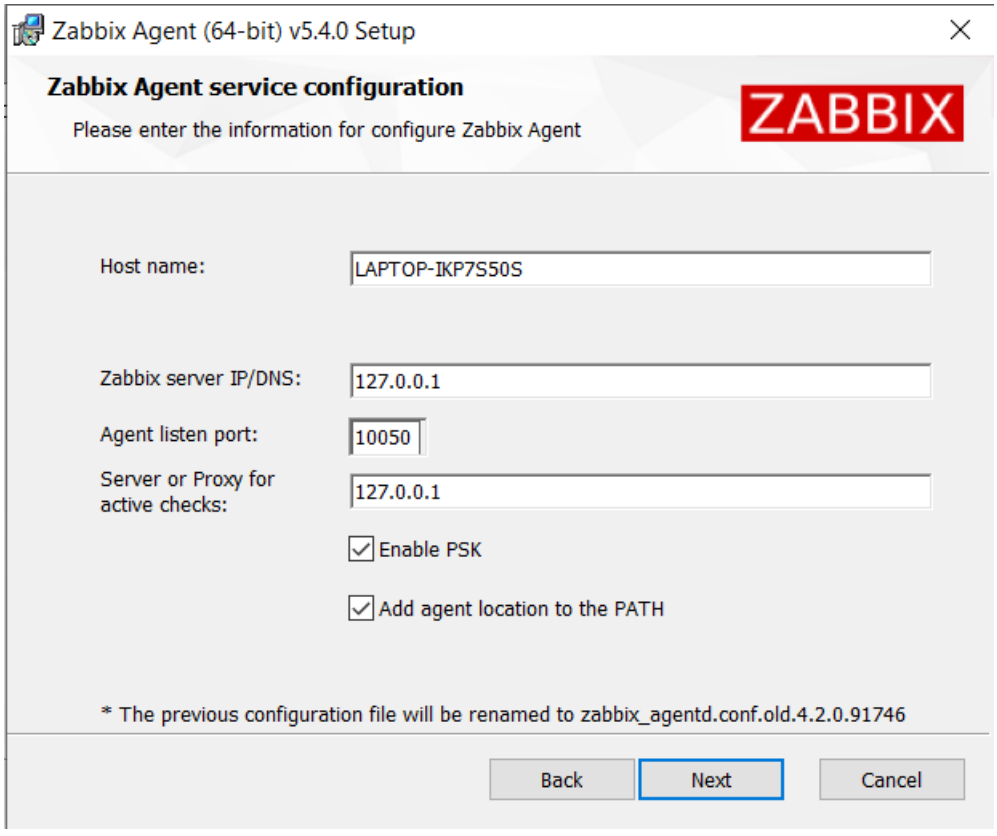
Installation steps

To install, double-click the downloaded MSI file.



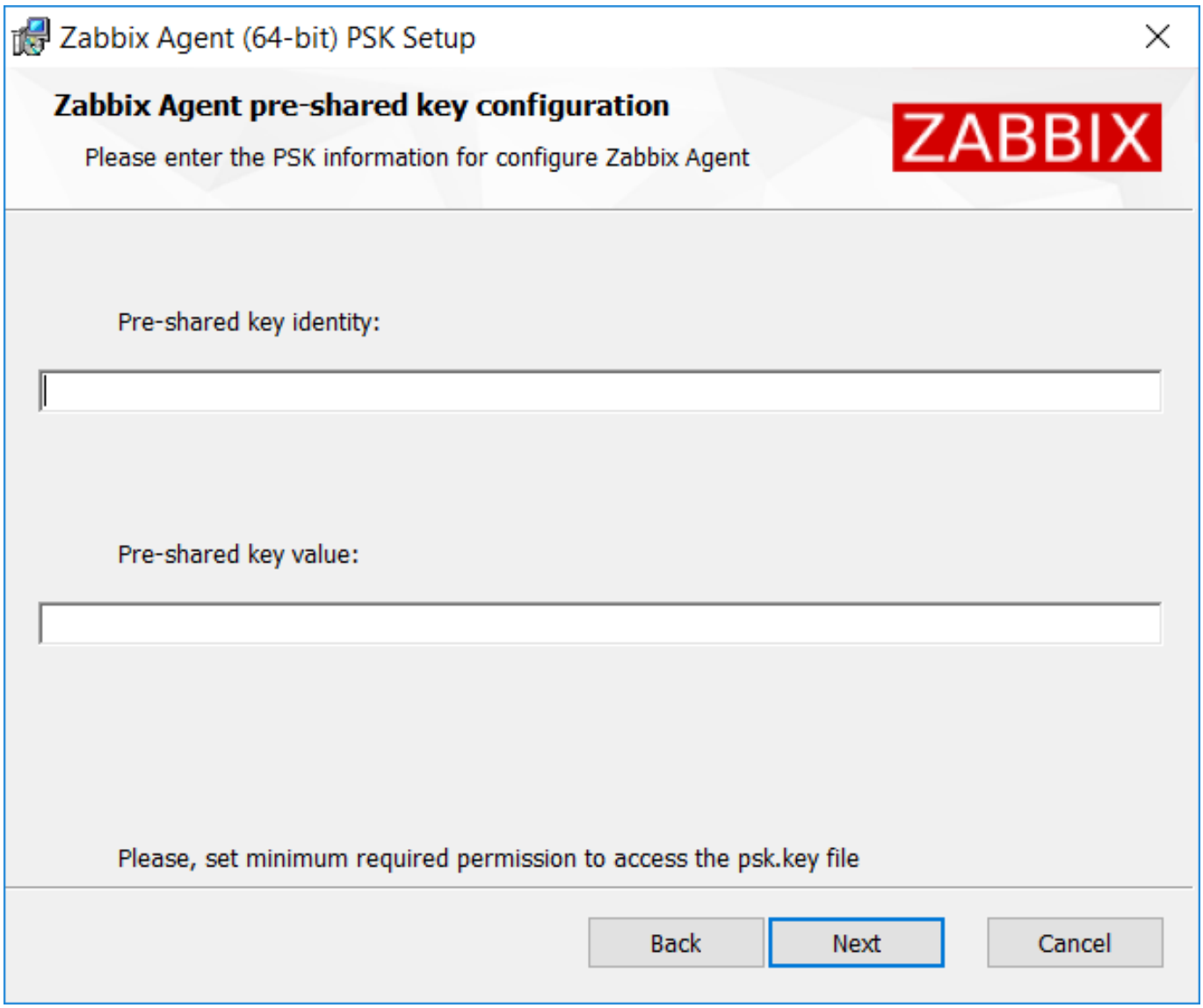


Accept the license to proceed to the next step.

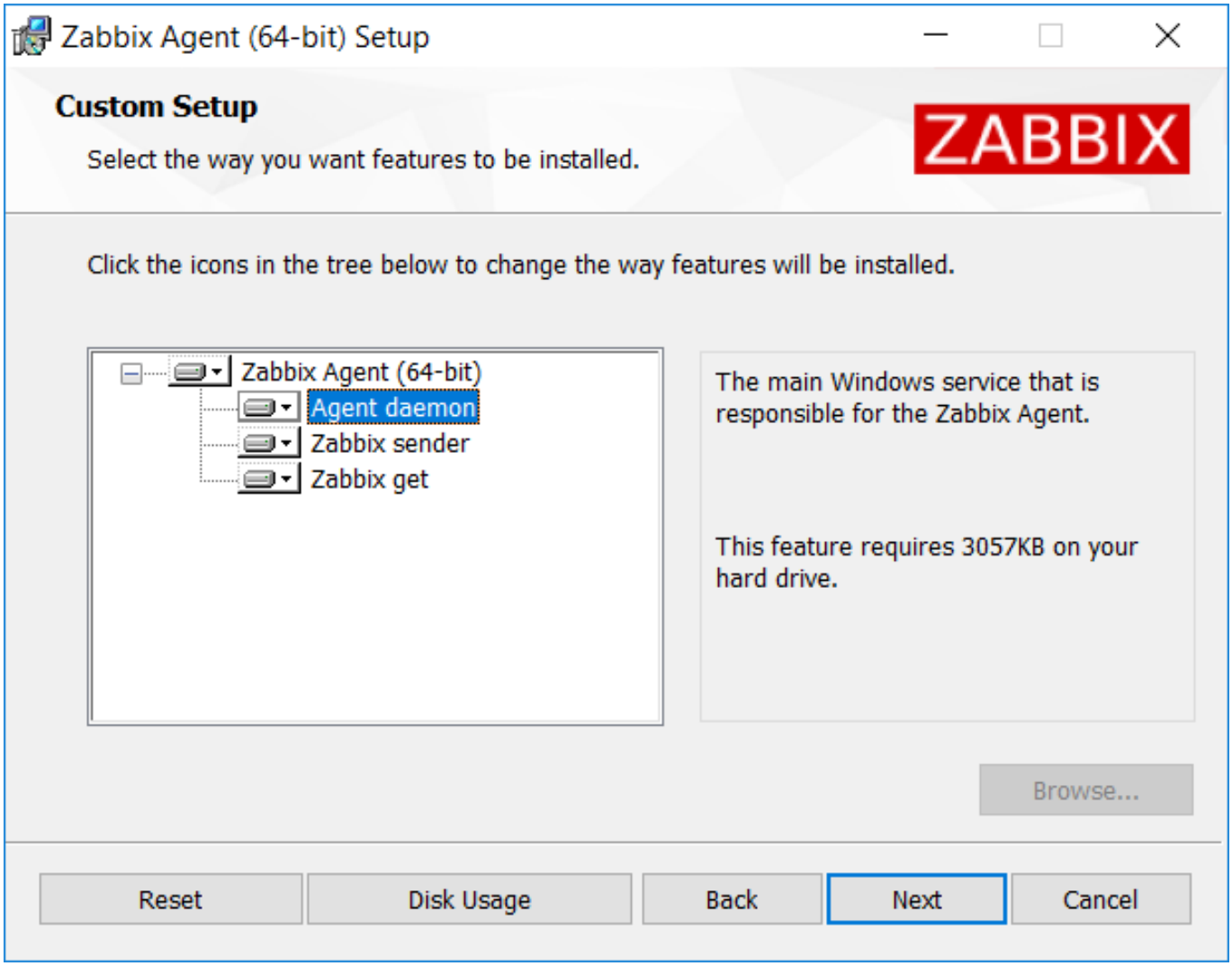


Specify the following parameters.

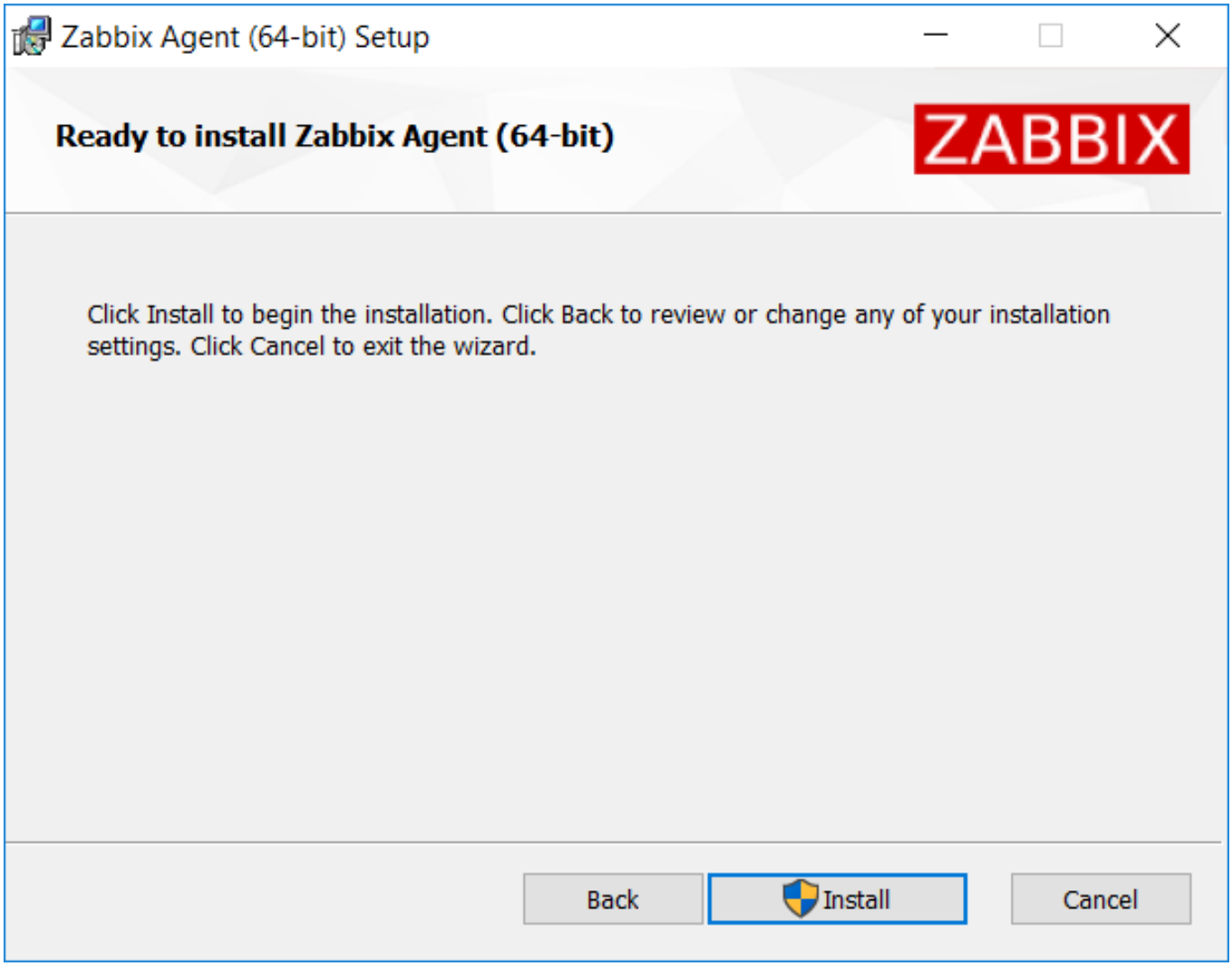
Parameter	Description
Host name	Specify host name.
Zabbix server IP/DNS	Specify IP/DNS of Zabbix server.
Agent listen port	Specify agent listen port (10050 by default).
Server or Proxy for active checks	Specify IP/DNS of Zabbix server/proxy for active agent checks.
Enable PSK	Mark the checkbox to enable TLS support via pre-shared keys.
Add agent location to the PATH	Add agent location to the PATH variable.



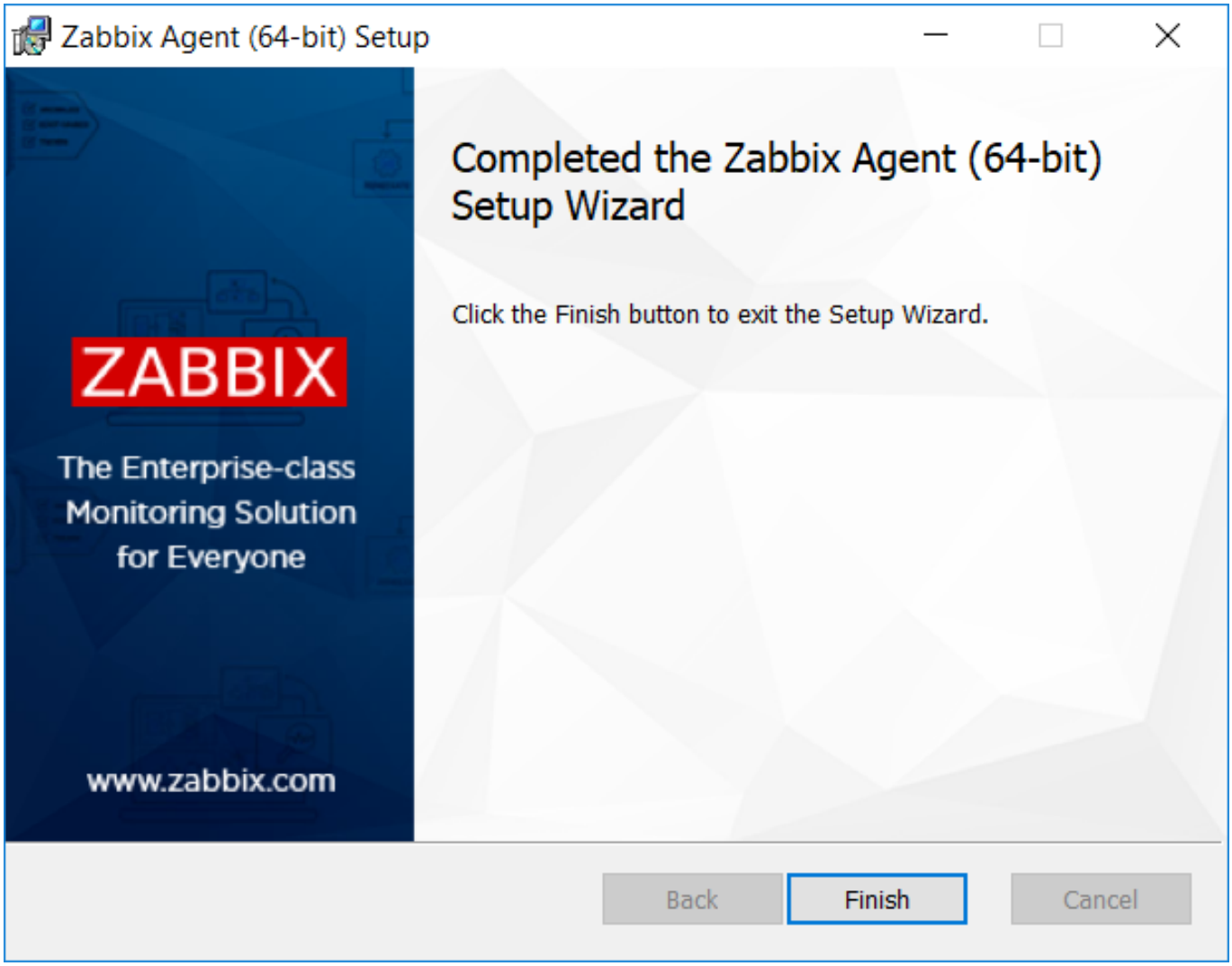
Enter pre-shared key identity and value. This step is only available if you checked *Enable PSK* in the previous step.



Select Zabbix components to install - Zabbix agent daemon, Zabbix sender, Zabbix get.



Zabbix components along with the configuration file will be installed in a *Zabbix Agent* folder in Program Files. zabbix_agentd.exe will be set up as Windows service with automatic startup.



Command-line based installation

Supported parameters

The following parameters are supported by created MSIs.

Parameter	Description
ADDDEFAULT	A comma-delimited list of programs to install in their default configuration. Possible values: `AgentProgram`, `GetProgram`, `SenderProgram`, `ALL`. Example: `ADDDEFAULT=AgentProgram,GetProgram`
ADDLOCAL	A comma-delimited list of programs to install locally. Possible values: `AgentProgram`, `GetProgram`, `SenderProgram`, `ALL`. Example: `ADDLOCAL=AgentProgram,SenderProgram`
ALLOWDENYKEY	Sequence of "AllowKey" and "DenyKey" parameters separated by `;` Use `\\;` to escape the delimiter. Example: `ALLOWDENYKEY="AllowKey=system.run[type c:\windows\system32\drivers\etc\hosts];DenyKey=system.run[*]"`
CONF	The full pathname to a custom configuration file. Example: `CONF=c:\full\path\to\user.conf`
ENABLEPATH	Add agent location to the PATH variable.
ENABLEPERSISTENTBUFFER	Zabbix agent 2 only. Enable the usage of local persistent storage for active items.
HOSTINTERFACE	An optional parameter that defines the host interface.
HOSTMETADATA	An optional parameter that defines the host metadata.
HOSTMETADATAITEM	An optional parameter that defines a Zabbix agent item used for getting the host metadata.
HOSTNAME	An optional parameter that defines the hostname.
INCLUDE	Sequence of includes separated by `;`
INSTALLFOLDER	The full pathname of the folder in which Zabbix components along with the configuration file will be installed.
LISTENIP	A list of comma-delimited IP addresses that the agent should listen on.

Parameter	Description
LISTENPORT	The agent will listen on this port for connections from the server.
LOGFILE	The name of the log file.
LOGTYPE	The type of the log output.
PERSISTENTBUFFERFILE	Zabbix agent 2 only. The file where Zabbix agent 2 should keep the SQLite database.
PERSISTENTBUFFERPERIOD	Zabbix agent 2 only. The time period for which data should be stored when there is no connection to the server or proxy.
SERVER	A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies.
SERVERACTIVE	The Zabbix server/proxy address or cluster configuration to get active checks from.
SKIP	`SKIP=fw` - do not install the firewall exception rule.
STATUSPORT	Zabbix agent 2 only. If set, the agent will listen on this port for HTTP status requests (http://localhost:<port>/status).
TIMEOUT	Spend no more than Timeout seconds on processing.
TLSACCEPT	What incoming connections to accept.
TLSCAFILE	The full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCERTFILE	The full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix components.
TLSCONNECT	How the agent should connect to Zabbix server or proxy.
TLSCRLFILE	The full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.
TLSKEYFILE	The full pathname of a file containing the agent private key, used for encrypted communications between Zabbix components.
TLSPSKFILE	The full pathname of a file containing the agent pre-shared key, used for encrypted communications with Zabbix server.
TLSPSKIDENTITY	The pre-shared key identity string, used for encrypted communications with Zabbix server.
TLSPSKVALUE	The pre-shared key string value, used for encrypted communications with Zabbix server.
TLSSERVERCERTISSUER	The allowed server (proxy) certificate issuer.
TLSSERVERCERTSUBJECT	The allowed server (proxy) certificate subject.

Examples

To install Zabbix Windows agent from the command-line, you may run, for example:

```
SET INSTALLFOLDER=C:\Program Files\Zabbix Agent

msiexec /l*v log.txt /i zabbix_agent-6.4.0-x86.msi /qn^
 LOGTYPE=file^
 LOGFILE="%INSTALLFOLDER%\zabbix_agentd.log"^
 SERVER=192.168.6.76^
 LISTENPORT=12345^
 SERVERACTIVE=::1^
 HOSTNAME=myHost^
 TLSCONNECT=psk^
 TLSACCEPT=psk^
 TLSPSKIDENTITY=MyPSKID^
 TLSPSKFILE="%INSTALLFOLDER%\mykey.psk"^
 TLSCAFILE="c:\temp\f.txt1"^
 TLSCRLFILE="c:\temp\f.txt2"^
 TLSSERVERCERTISSUER="My CA"^
 TLSSERVERCERTSUBJECT="My Cert"^
 TLSCERTFILE="c:\temp\f.txt5"^
 TLSKEYFILE="c:\temp\f.txt6"^
 ENABLEPATH=1^
 INSTALLFOLDER="%INSTALLFOLDER%"^
 SKIP=fw^
 ALLOWDENYKEY="DenyKey=vfs.file.contents[/etc/passwd]"
```

You may also run, for example:

```
msiexec /l*v log.txt /i zabbix_agent-6.4.0-x86.msi /qn^
 SERVER=192.168.6.76^
 TLSCONNECT=psk^
```

```
TLSACCEPT=psk^
TLSPSKIDENTITY=MyPSKID^
TLSPSKVALUE=1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

Note:
If both TLSPSKFILE and TLSPSKVALUE are passed, then TLSPSKVALUE will be written to TLSPSKFILE.

5 Mac OS agent installation from PKG

Overview

Zabbix Mac OS agent can be installed from PKG installer packages available for download. Versions with or without encryption are available.

Installing agent

The agent can be installed using the graphical user interface or from the command line, for example:

```
sudo installer -pkg zabbix_agent-6.4.0-macos-amd64-openssl.pkg -target /
```

Make sure to use the correct Zabbix package version in the command. It must match the name of the downloaded package.

Running agent

The agent will start automatically after installation or restart.

You may edit the configuration file at `/usr/local/etc/zabbix/zabbix_agentd.conf` if necessary.

To start the agent manually, you may run:

```
sudo launchctl start com.zabbix.zabbix_agentd
```

To stop the agent manually:

```
sudo launchctl stop com.zabbix.zabbix_agentd
```

During upgrade, the existing configuration file is not overwritten. Instead a new `zabbix_agentd.conf.NEW` file is created to be used for reviewing and updating the existing configuration file, if necessary. Remember to restart the agent after manual changes to the configuration file.

Troubleshooting and removing agent

This section lists some useful commands that can be used for troubleshooting and removing Zabbix agent installation.

See if Zabbix agent is running:

```
ps aux | grep zabbix_agentd
```

See if Zabbix agent has been installed from packages:

```
$ pkgutil --pkgs | grep zabbix
com.zabbix.pkg.ZabbixAgent
```

See the files that were installed from the installer package (note that the initial / is not displayed in this view):

```
$ pkgutil --only-files --files com.zabbix.pkg.ZabbixAgent
Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
usr/local/bin/zabbix_get
usr/local/bin/zabbix_sender
usr/local/etc/zabbix/zabbix_agentd/userparameter_examples.conf.NEW
usr/local/etc/zabbix/zabbix_agentd/userparameter_mysql.conf.NEW
usr/local/etc/zabbix/zabbix_agentd.conf.NEW
usr/local/sbin/zabbix_agentd
```

Stop Zabbix agent if it was launched with `launchctl`:

```
sudo launchctl unload /Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
```

Remove files (including configuration and logs) that were installed with installer package:

```
sudo rm -f /Library/LaunchDaemons/com.zabbix.zabbix_agentd.plist
sudo rm -f /usr/local/sbin/zabbix_agentd
sudo rm -f /usr/local/bin/zabbix_get
sudo rm -f /usr/local/bin/zabbix_sender
```

```
sudo rm -rf /usr/local/etc/zabbix
sudo rm -rf /var/log/zabbix
```

Forget that Zabbix agent has been installed:

```
sudo pkgutil --forget com.zabbix.pkg.ZabbixAgent
```

6 Unstable releases

Overview

The instructions below are for enabling unstable Zabbix release repositories (disabled by default) used for minor Zabbix version release candidates.

First, install or update to the latest zabbix-release package. To enable rc packages on your system do the following:

Red Hat Enterprise Linux

Open the `/etc/yum.repos.d/zabbix.repo` file and set enabled=1 for the `zabbix-unstable` repo.

```
[zabbix-unstable]
name=Zabbix Official Repository (unstable) - $basearch
baseurl=https://repo.zabbix.com/zabbix/6.3/rhel/8/$basearch/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-A14FE591
```

Debian/Ubuntu

Open the `/etc/apt/sources.list.d/zabbix.list` and uncomment "Zabbix unstable repository".

```
#### Zabbix unstable repository
deb https://repo.zabbix.com/zabbix/6.3/debian bullseye main
deb-src https://repo.zabbix.com/zabbix/6.3/debian bullseye main
```

SUSE

Open the `/etc/zypp/repos.d/zabbix.repo` file and set enable=1 for the `zabbix-unstable` repo.

```
[zabbix-unstable]
name=Zabbix Official Repository
type=rpm-md
baseurl=https://repo.zabbix.com/zabbix/6.3/sles/15/x86_64/
gpgcheck=1
gpgkey=https://repo.zabbix.com/zabbix/6.3/sles/15/x86_64/repodata/repomd.xml.key
enabled=1
update=1
```

5 Installation from containers

Overview This section describes how to deploy Zabbix with Docker or Docker Compose.

Zabbix officially provides:

- Separate Docker images for each Zabbix component to run as portable and self-sufficient containers.
- Compose files for defining and running multi-container Zabbix components in Docker.

Attention:
Since Zabbix 6.0, deterministic triggers need to be created during the installation. If binary logging is enabled for MySQL/MariaDB, this requires superuser privileges or setting the variable/configuration parameter *log_bin_trust_function_creators = 1*. See Database creation scripts for instructions how to set the variable.
Note that if executing from a console, the variable will only be set temporarily and will be dropped when a Docker is restarted. In this case, keep your SQL service running, only stop zabbix-server service by running 'docker compose down zabbix-server' and then 'docker compose up -d zabbix-server'.
Alternatively, you can set this variable in the configuration file.

Source files

Docker file sources are stored in the Zabbix official repository on GitHub, where you can follow latest file changes or fork the project to make your own images.

Docker Zabbix provides images based on a variety of OS base images. To get the list of supported base operating system images for a specific Zabbix component, see the component's description in Docker Hub. All Zabbix images are configured to rebuild latest images if base images are updated.

Installation

To get Zabbix component image, run:

```
docker pull zabbix/zabbix-server-mysql
```

Replace `zabbix/zabbix-server-mysql` with the name of the required docker repository.

This command will pull the latest stable Zabbix component version based on the Alpine Linux OS. You can append tags to the repository name to get an image based on another operating system or of the specific Zabbix major or minor version.

The following repositories are available in Docker Hub:

Component		Docker repository
Zabbix agent		zabbix/zabbix-agent
Zabbix server		
	with MySQL support	zabbix/zabbix-server-mysql
	with PostgreSQL support	zabbix/zabbix-server-pgsql
Zabbix web interface		
	based on Apache2 web server with MySQL support	zabbix/zabbix-web-apache-mysql
	based on Apache2 web server with PostgreSQL support	zabbix/zabbix-web-apache-pgsql
	based on Nginx web server with MySQL support	zabbix/zabbix-web-nginx-mysql
	based on Nginx web server with PostgreSQL support	zabbix/zabbix-web-nginx-pgsql
Zabbix proxy		
	with SQLite3 support	zabbix/zabbix-proxy-sqlite3
	with MySQL support	zabbix/zabbix-proxy-mysql
Zabbix Java gateway		zabbix/zabbix-java-gateway

Note:
SNMP trap support is provided in a separate repository zabbix/zabbix-snmptraps. It can be linked with Zabbix server and Zabbix proxy.

Tags

Official Zabbix component images may contain the following tags:

Tag	Description	Example
latest	The latest stable version of a Zabbix component based on Alpine Linux image.	zabbix-agent:latest

Tag	Description	Example
<OS>-trunk	The latest nightly build of the Zabbix version that is currently being developed on a specific operating system. **<OS>** - the base operating system. Supported values: *alpine* - Alpine Linux; *ltsc2019* - Windows 10 LTSC 2019 (agent only); *ol* - Oracle Linux; *ltsc2022* - Windows 11 LTSC 2022 (agent only); *ubuntu* - Ubuntu	zabbix agent:ubuntu-trunk
<OS>-latest	The latest stable version of a Zabbix component on a specific operating system. **<OS>** - the base operating system. Supported values: *alpine* - Alpine Linux; *ltsc2019* - Windows 10 LTSC 2019 (agent only); *ol* - Oracle Linux; *ltsc2022* - Windows 11 LTSC 2022 (agent only); *ubuntu* - Ubuntu	zabbix-agent:ol-latest
<OS>-X.X-latest	The latest minor version of a Zabbix component of a specific major version and operating system. **<OS>** - the base operating system. Supported values: *alpine* - Alpine Linux; *ltsc2019* - Windows 10 LTSC 2019 (agent only); *ol* - Oracle Linux; *ltsc2022* - Windows 11 LTSC 2022 (agent only); *ubuntu* - Ubuntu **X.X** - the major Zabbix version (i.e. *5.0*, *6.0*, *6.4*).	zabbix-agent:alpine-6.4-latest
<OS>-X.X.*	The latest minor version of a Zabbix component of a specific major version and operating system. **<OS>** - the base operating system. Supported values: *alpine* - Alpine Linux; *ltsc2019* - Windows 10 LTSC 2019 (agent only); *ol* - Oracle Linux; *ltsc2022* - Windows 11 LTSC 2022 (agent only); *ubuntu* - Ubuntu **X.X** - the major Zabbix version (i.e. *5.0*, *6.0*, *6.4*). * - the Zabbix minor version	zabbix-agent:alpine-6.4.1

Initial configuration

After downloading the images, start the containers by executing `docker run` command followed by additional arguments to specify required environment variables and/or mount points. Some configuration examples are provided below.

Attention:
Zabbix must not be run as PID1/as an init process in containers.

Note:
To enable communication between Zabbix components, some ports, such as 10051/TCP for Zabbix server (trapper), 10050/TCP for Zabbix agent, 162/UDP for SNMP traps and 80/TCP for Zabbix web interface will be exposed to a host machine. Full list of default ports used by Zabbix components is available on the Requirements page. For Zabbix server and agent the default port can be changed by setting ZBX_LISTENPORT environment variable.

Environment variables

All Zabbix component images provide environment variables to control configuration. Supported environment variables are listed in the component repository.

These environment variables are options from Zabbix configuration files, but with different naming method. For example, ZBX_LOGSLOWQUERIES is equal to LogSlowQueries from Zabbix server or Zabbix proxy configuration files.

Attention:
Some configuration options (e.g., PIDFile and LogType) cannot be changed.

The following environment variables are specific to Docker components and do not exist in Zabbix configuration files:

Variable	Components	Default value	Description
DB_SERVER_HOST	Server Proxy Web interface	mysql-server for MYSQL postgres-server for PostgreSQL	IP or DNS name of MySQL or PostgreSQL server.
DB_SERVER_PORT	Server Proxy Web interface	3306 for MYSQL 5432 for PostgreSQL	Port of MySQL or PostgreSQL server.
MYSQL_USER	Server Proxy Web-interface	zabbix	MySQL database user.
MYSQL_PASSWORD	Server Proxy Web interface	zabbix	MySQL database password.
MYSQL_DATABASE	Server Proxy Web interface	zabbix for Zabbix server zabbix_proxy for Zabbix proxy	Zabbix database name.
POSTGRES_USER	Server Web interface	zabbix	PostgreSQL database user.
POSTGRES_PASSWORD	Server Web interface	zabbix	PostgreSQL database password.
POSTGRES_DB	Server Web interface	zabbix for Zabbix server zabbix_proxy for Zabbix proxy	Zabbix database name.
PHP_TZ	Web-interface	Europe/Riga	Timezone in PHP format. Full list of supported timezones is available on php.net.
ZBX_SERVER_NAME	Web interface	Zabbix Docker	Visible Zabbix installation name in right top corner of the web interface.
ZBX_JAVAGATEWAY_ENABLE	Server Proxy	false	Enables communication with Zabbix Java gateway to collect Java related checks.
ZBX_ENABLE_SNMP_TRAPS	Server Proxy	false	Enables SNMP trap feature. It requires **zabbix-snmptraps** instance and shared volume */var/lib/zabbix/snmptraps* to Zabbix server or Zabbix proxy.

Volumes

The images allow to mount volumes using the following mount points:

Volume	Description
Zabbix agent	
/etc/zabbix/zabbix_agentd.d	The volume allows to include **.conf* files and extend Zabbix agent using the UserParameter feature
/var/lib/zabbix/modules	The volume allows to load additional modules and extend Zabbix agent using the LoadModule feature
/var/lib/zabbix/enc	The volume is used to store TLS-related files. These file names are specified using ZBX_TLSCAFILE, ZBX_TLSCRLFILE, ZBX_TLSKEY_FILE and ZBX_TLSPSKFILE environment variables
Zabbix server	

Volume	Description
Zabbix server	
/usr/lib/zabbix/alertscripts	The volume is used for custom alert scripts. It is the `AlertScriptsPath` parameter in zabbix_server.conf
/usr/lib/zabbix/externalscripts	The volume is used by external checks. It is the `ExternalScripts` parameter in zabbix_server.conf
/var/lib/zabbix/modules	The volume allows to load additional modules and extend Zabbix server using the LoadModule feature
/var/lib/zabbix/enc	The volume is used to store TLS related files. These file names are specified using `ZBX_TLSCAFILE`, `ZBX_TLSCRLFILE`, `ZBX_TLSKEY_FILE` and `ZBX_TLSPSKFILE` environment variables
/var/lib/zabbix/ssl/certs	The volume is used as location of SSL client certificate files for client authentication. It is the `SSLCertLocation` parameter in zabbix_server.conf
/var/lib/zabbix/ssl/keys	The volume is used as location of SSL private key files for client authentication. It is the `SSLKeyLocation` parameter in zabbix_server.conf
/var/lib/zabbix/ssl/ssl_ca	The volume is used as location of certificate authority (CA) files for SSL server certificate verification. It is the `SSLCALocation` parameter in zabbix_server.conf
/var/lib/zabbix/snmptraps	The volume is used as location of snmptraps.log file. It could be shared by zabbix-snmptraps container and inherited using the volumes_from Docker option while creating a new instance of Zabbix server. SNMP trap processing feature could be enabled by using shared volume and switching the `ZBX_ENABLE_SNMP_TRAPS` environment variable to 'true'
/var/lib/zabbix/mibs	The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in `/var/lib/zabbix/mibs`
Zabbix proxy	
/usr/lib/zabbix/externalscripts	The volume is used by external checks. It is the `ExternalScripts` parameter in zabbix_proxy.conf
/var/lib/zabbix/db_data	The volume allows to store database files on external devices. Supported only for Zabbix proxy with SQLite3
/var/lib/zabbix/modules	The volume allows to load additional modules and extend Zabbix server using the LoadModule feature
/var/lib/zabbix/enc	The volume is used to store TLS related files. These file names are specified using `ZBX_TLSCAFILE`, `ZBX_TLSCRLFILE`, `ZBX_TLSKEY_FILE` and `ZBX_TLSPSKFILE` environment variables
/var/lib/zabbix/ssl/certs	The volume is used as location of SSL client certificate files for client authentication. It is the `SSLCertLocation` parameter in zabbix_proxy.conf
/var/lib/zabbix/ssl/keys	The volume is used as location of SSL private key files for client authentication. It is the `SSLKeyLocation` parameter in zabbix_proxy.conf
/var/lib/zabbix/ssl/ssl_ca	The volume is used as location of certificate authority (CA) files for SSL server certificate verification. It is the `SSLCALocation` parameter in zabbix_proxy.conf
/var/lib/zabbix/snmptraps	The volume is used as location of snmptraps.log file. It could be shared by the zabbix-snmptraps container and inherited using the volumes_from Docker option while creating a new instance of Zabbix server. SNMP trap processing feature could be enabled by using shared volume and switching the `ZBX_ENABLE_SNMP_TRAPS` environment variable to 'true'
/var/lib/zabbix/mibs	The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in `/var/lib/zabbix/mibs`
Zabbix web interface based on Apache2 web server	
/etc/ssl/apache2	The volume allows to enable HTTPS for Zabbix web interface. The volume must contain the two `ssl.crt` and `ssl.key` files prepared for Apache2 SSL connections

Volume	Description
Zabbix web interface based on Nginx web server	
/etc/ssl/nginx	The volume allows to enable HTTPS for Zabbix web interface. The volume must contain the two `ssl.crt`, `ssl.key` files and `dhparam.pem` prepared for Nginx SSL connections
Zabbix snmptraps	
/var/lib/zabbix/snmptraps	The volume contains the `snmptraps.log` log file named with received SNMP traps
/var/lib/zabbix/mibs	The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in `/var/lib/zabbix/mibs`

For additional information, see Zabbix official repositories in Docker Hub.

Examples

Example 1

The example demonstrates how to run Zabbix server with MySQL database support, Zabbix web interface based on the Nginx web server and Zabbix Java gateway.

1. Create network dedicated for Zabbix component containers:

```
# docker network create --subnet 172.20.0.0/16 --ip-range 172.20.240.0/20 zabbix-net
```

2. Start empty MySQL server instance

```
# docker run --name mysql-server -t \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      --network=zabbix-net \
      --restart unless-stopped \
      -d mysql:8.0-oracle \
      --character-set-server=utf8 --collation-server=utf8_bin \
      --default-authentication-plugin=mysql_native_password
```

3. Start Zabbix Java gateway instance

```
# docker run --name zabbix-java-gateway -t \
      --network=zabbix-net \
      --restart unless-stopped \
      -d zabbix/zabbix-java-gateway:alpine-6.4-latest
```

4. Start Zabbix server instance and link the instance with created MySQL server instance

```
# docker run --name zabbix-server-mysql -t \
      -e DB_SERVER_HOST="mysql-server" \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      -e ZBX_JAVAGATEWAY="zabbix-java-gateway" \
      --network=zabbix-net \
      -p 10051:10051 \
      --restart unless-stopped \
      -d zabbix/zabbix-server-mysql:alpine-6.4-latest
```

5. Start Zabbix web interface and link the instance with created MySQL server and Zabbix server instances

```
# docker run --name zabbix-web-nginx-mysql -t \
      -e ZBX_SERVER_HOST="zabbix-server-mysql" \
      -e DB_SERVER_HOST="mysql-server" \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      --network=zabbix-net \
      -p 80:8080 \
      --restart unless-stopped \
      -d zabbix/zabbix-web-nginx-mysql:alpine-6.4-latest
```

Example 2

The example demonstrates how to run Zabbix server with PostgreSQL database support, Zabbix web interface based on the Nginx web server and SNMP trap feature.

1. Create network dedicated for Zabbix component containers:

```
# docker network create --subnet 172.20.0.0/16 --ip-range 172.20.240.0/20 zabbix-net
```

2. Start empty PostgreSQL server instance

```
# docker run --name postgres-server -t \
      -e POSTGRES_USER="zabbix" \
      -e POSTGRES_PASSWORD="zabbix_pwd" \
      -e POSTGRES_DB="zabbix" \
      --network=zabbix-net \
      --restart unless-stopped \
      -d postgres:latest
```

3. Start Zabbix snmptraps instance

```
# docker run --name zabbix-snmptraps -t \
      -v /zbx_instance/snmptraps:/var/lib/zabbix/snmptraps:rw \
      -v /var/lib/zabbix/mibs:/usr/share/snmp/mibs:ro \
      --network=zabbix-net \
      -p 162:1162/udp \
      --restart unless-stopped \
      -d zabbix/zabbix-snmptraps:alpine-6.4-latest
```

4. Start Zabbix server instance and link the instance with created PostgreSQL server instance

```
# docker run --name zabbix-server-pgsql -t \
      -e DB_SERVER_HOST="postgres-server" \
      -e POSTGRES_USER="zabbix" \
      -e POSTGRES_PASSWORD="zabbix_pwd" \
      -e POSTGRES_DB="zabbix" \
      -e ZBX_ENABLE_SNMP_TRAPS="true" \
      --network=zabbix-net \
      -p 10051:10051 \
      --volumes-from zabbix-snmptraps \
      --restart unless-stopped \
      -d zabbix/zabbix-server-pgsql:alpine-6.4-latest
```

5. Start Zabbix web interface and link the instance with created PostgreSQL server and Zabbix server instances

```
# docker run --name zabbix-web-nginx-pgsql -t \
      -e ZBX_SERVER_HOST="zabbix-server-pgsql" \
      -e DB_SERVER_HOST="postgres-server" \
      -e POSTGRES_USER="zabbix" \
      -e POSTGRES_PASSWORD="zabbix_pwd" \
      -e POSTGRES_DB="zabbix" \
      --network=zabbix-net \
      -p 443:8443 \
      -p 80:8080 \
      -v /etc/ssl/nginx:/etc/ssl/nginx:ro \
```

```
      --restart unless-stopped \
      -d zabbix/zabbix-web-nginx-pgsql:alpine-6.4-latest
```

Example 3

The example demonstrates how to run Zabbix server with MySQL database support, Zabbix web interface based on the Nginx web server and Zabbix Java gateway using `podman` on Red Hat 8.

1. Create new pod with name `zabbix` and exposed ports (web-interface, Zabbix server trapper):

```
podman pod create --name zabbix -p 80:8080 -p 10051:10051
```

2. (optional) Start Zabbix agent container in `zabbix` pod location:

```
podman run --name zabbix-agent \
    -e ZBX_SERVER_HOST="127.0.0.1,localhost" \
    --restart=always \
    --pod=zabbix \
    -d registry.connect.redhat.com/zabbix/zabbix-agent-64:latest
```

3. Create `./mysql/` directory on host and start Oracle MySQL server 8.0:

```
podman run --name mysql-server -t \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      -v ./mysql/:/var/lib/mysql/:Z \
      --restart=always \
      --pod=zabbix \
      -d mysql:8.0 \
      --character-set-server=utf8 --collation-server=utf8_bin \
      --default-authentication-plugin=mysql_native_password
```

4. Start Zabbix server container:

```
podman run --name zabbix-server-mysql -t \
                  -e DB_SERVER_HOST="127.0.0.1" \
                  -e MYSQL_DATABASE="zabbix" \
                  -e MYSQL_USER="zabbix" \
                  -e MYSQL_PASSWORD="zabbix_pwd" \
                  -e MYSQL_ROOT_PASSWORD="root_pwd" \
                  -e ZBX_JAVAGATEWAY="127.0.0.1" \
                  --restart=always \
                  --pod=zabbix \
                  -d registry.connect.redhat.com/zabbix/zabbix-server-mysql-64
```

5. Start Zabbix Java Gateway container:

```
podman run --name zabbix-java-gateway -t \
      --restart=always \
      --pod=zabbix \
      -d registry.connect.redhat.com/zabbix/zabbix-java-gateway-64
```

6. Start Zabbix web-interface container:

```
podman run --name zabbix-web-mysql -t \
                  -e ZBX_SERVER_HOST="127.0.0.1" \
                  -e DB_SERVER_HOST="127.0.0.1" \
                  -e MYSQL_DATABASE="zabbix" \
                  -e MYSQL_USER="zabbix" \
                  -e MYSQL_PASSWORD="zabbix_pwd" \
                  -e MYSQL_ROOT_PASSWORD="root_pwd" \
                  --restart=always \
                  --pod=zabbix \
                  -d registry.connect.redhat.com/zabbix/zabbix-web-mysql-64
```

Note:
Pod `zabbix` exposes 80/TCP port (HTTP) to host machine from 8080/TCP of `zabbix-web-mysql` container.

Docker Compose Alternatively, Zabbix can be installed using Docker Compose plugin. Compose files for defining and running multi-container Zabbix components are available in the official Zabbix Docker repository on GitHub.

> **Attention:**
> Official Zabbix compose files support version 3 of Docker Compose.

These compose files are added as examples; they are overloaded. For example, they contain proxies with both MySQL and SQLite3 support.

To obtain Docker compose files provided by Zabbix, clone the repository:

```
git clone https://github.com/zabbix/zabbix-docker.git
```

Switch to the required version:

```
git checkout 6.4
```

Compose configuration files and create and start containers:

```
docker compose -f ./docker-compose_v3_alpine_mysql_latest.yaml up
```

Replace `docker-compose_v3_alpine_mysql_latest.yaml` in the command above with the required configuration file.

The following options are available:

File name	Description
`docker-compose_v3_alpine_mysql_latest.yaml`	The compose file runs the latest version of Zabbix 6.4 components on Alpine Linux with MySQL database support.
`docker-compose_v3_alpine_mysql_local.yaml`	The compose file locally builds the latest version of Zabbix 6.4 and runs Zabbix components on Alpine Linux with MySQL database support.
`docker-compose_v3_alpine_pgsql_latest.yaml`	The compose file runs the latest version of Zabbix 6.4 components on Alpine Linux with PostgreSQL database support.
`docker-compose_v3_alpine_pgsql_local.yaml`	The compose file locally builds the latest version of Zabbix 6.4 and runs Zabbix components on Alpine Linux with PostgreSQL database support.
`docker-compose_v3_ol_mysql_latest.yaml`	The compose file runs the latest version of Zabbix 6.4 components on Oracle Linux with MySQL database support.
`docker-compose_v3_ol_mysql_local.yaml`	The compose file locally builds the latest version of Zabbix 6.4 and runs Zabbix components on Oracle Linux with MySQL database support.
`docker-compose_v3_ol_pgsql_latest.yaml`	The compose file runs the latest version of Zabbix 6.4 components on Oracle Linux with PostgreSQL database support.
`docker-compose_v3_ol_pgsql_local.yaml`	The compose file locally builds the latest version of Zabbix 6.4 and runs Zabbix components on Oracle Linux with PostgreSQL database support.
`docker-compose_v3_ubuntu_mysql_latest.yaml`	The compose file runs the latest version of Zabbix 6.4 components on Ubuntu 20.04 with MySQL database support.
`docker-compose_v3_ubuntu_mysql_local.yaml`	The compose file locally builds the latest version of Zabbix 6.4 and runs Zabbix components on Ubuntu 20.04 with MySQL database support.
`docker-compose_v3_ubuntu_pgsql_latest.yaml`	The compose file runs the latest version of Zabbix 6.4 components on Ubuntu 20.04 with PostgreSQL database support.
`docker-compose_v3_ubuntu_pgsql_local.yaml`	The compose file locally builds the latest version of Zabbix 6.4 and runs Zabbix components on Ubuntu 20.04 with PostgreSQL database support.

Storage

Compose files are configured to support local storage on a host machine. Docker Compose will create a `zbx_env` directory in the folder with the compose file when you run Zabbix components using the compose file. The directory will contain the same structure as described in the Volumes section and directory for database storage.

There are also volumes in read-only mode for `/etc/localtime` and `/etc/timezone` files.

Environment variables

The variable files have the following naming structure: `.env_<type of component>` and are located in the *env_vars* directory. See environment variables for details about variable naming and available selection.

Examples

Example 1

```
# git checkout 6.4
# docker compose -f ./docker-compose_v3_alpine_mysql_latest.yaml up -d
```

The command will download the latest Zabbix 6.4 images for each Zabbix component and run them in detach mode.

> **Attention:**
> Do not forget to download `.env_<type of component>` files from github.com official Zabbix repository with compose files.

Example 2

```
# git checkout 6.4
# docker compose -f ./docker-compose_v3_ubuntu_mysql_local.yaml up -d
```

The command will download base image Ubuntu 22.04 (jammy), then build Zabbix 6.4 components locally and run them in detach mode.

6 Web interface installation

This section provides step-by-step instructions for installing Zabbix web interface. Zabbix frontend is written in PHP, so to run it a PHP supported webserver is needed.

> **Note:**
> You can find out more about setting up SSL for Zabbix frontend by referring to these best practices.

Welcome screen

Open Zabbix frontend URL in the browser. If you have installed Zabbix from packages, the URL is:

- for Apache: *http://<server_ip_or_name>/zabbix*
- for Nginx: *http://<server_ip_or_name>*

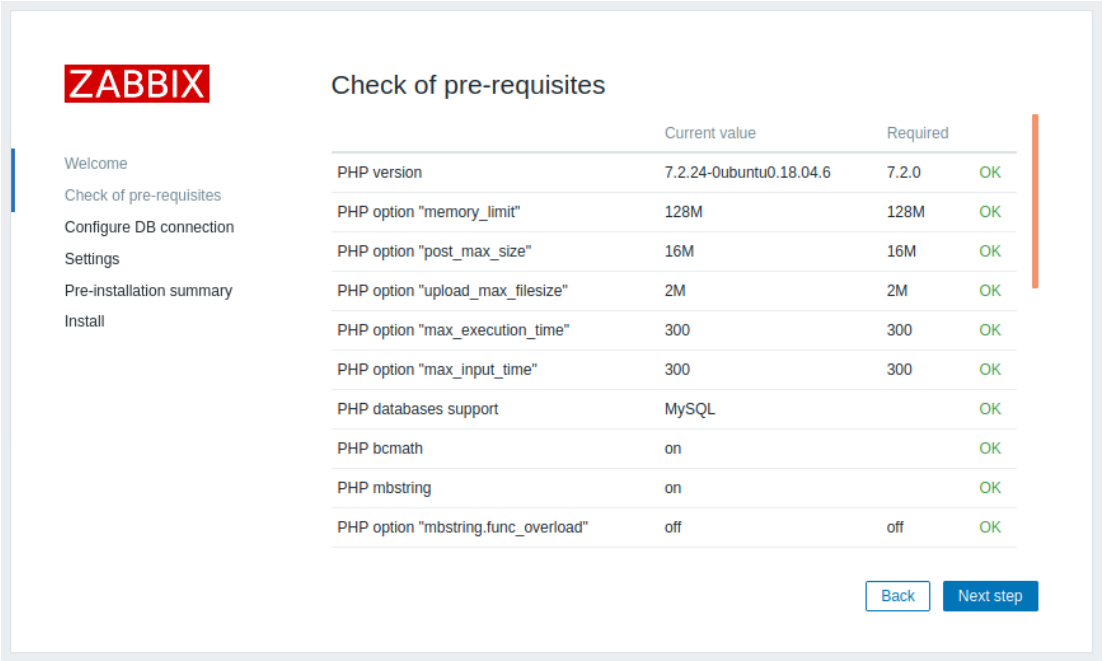
You should see the first screen of the frontend installation wizard.

Use the *Default language* drop-down menu to change system default language and continue the installation process in the selected language (optional). For more information, see Installation of additional frontend languages.



Check of pre-requisites

Make sure that all software prerequisites are met.



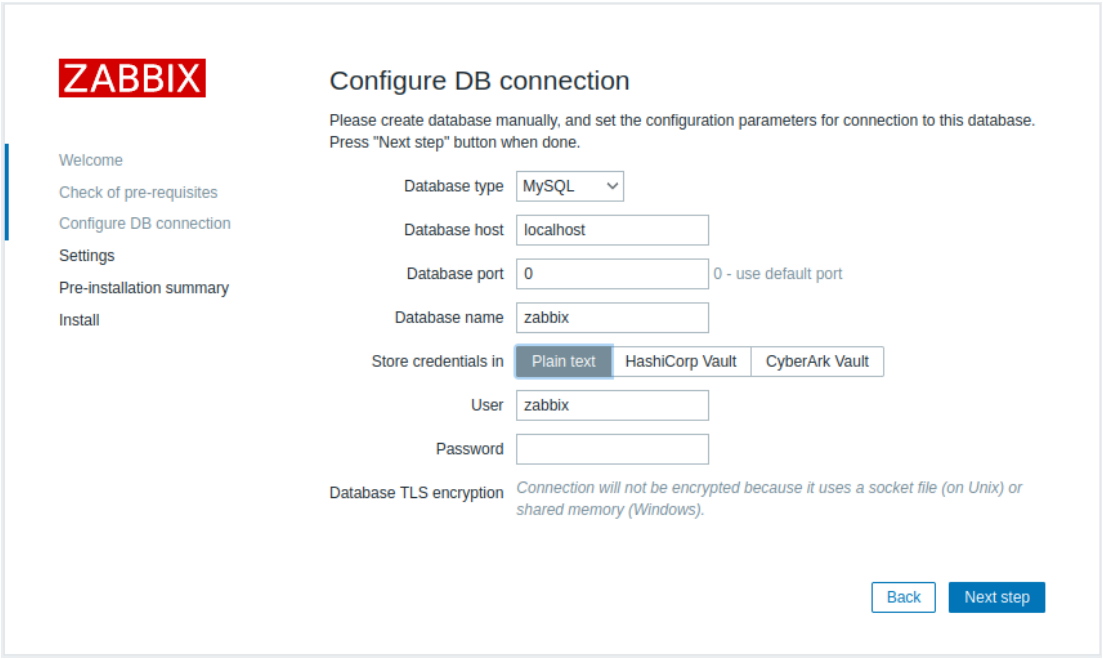
Pre-requisite	Minimum value	Description
PHP version	7.4.0	
PHP memory_limit option	128MB	In php.ini: memory_limit = 128M
PHP post_max_size option	16MB	In php.ini: post_max_size = 16M
PHP upload_max_filesize option	2MB	In php.ini: upload_max_filesize = 2M
PHP max_execution_time option	300 seconds (values 0 and -1 are allowed)	In php.ini: max_execution_time = 300
PHP max_input_time option	300 seconds (values 0 and -1 are allowed)	In php.ini: max_input_time = 300
PHP session.auto_start option	must be disabled	In php.ini: session.auto_start = 0
Database support	One of: MySQL, Oracle, PostgreSQL.	One of the following modules must be installed: mysql, oci8, pgsql
bcmath		php-bcmath
mbstring		php-mbstring
PHP mbstring.func_overload option	must be disabled	In php.ini: mbstring.func_overload = 0
sockets		php-net-socket. Required for user script support.
gd	2.0.28	php-gd. PHP GD extension must support PNG images (*--with-png-dir*), JPEG (*--with-jpeg-dir*) images and FreeType 2 (*--with-freetype-dir*).
libxml	2.6.15	php-xml
xmlwriter		php-xmlwriter
xmlreader		php-xmlreader
ctype		php-ctype
session		php-session
gettext		php-gettext Since Zabbix 2.2.1, the PHP gettext extension is not a mandatory requirement for installing Zabbix. If gettext is not installed, the frontend will work as usual, however, the translations will not be available.

Optional pre-requisites may also be present in the list. A failed optional prerequisite is displayed in orange and has a *Warning* status. With a failed optional pre-requisite, the setup may continue.

Attention:
If there is a need to change the Apache user or user group, permissions to the session folder must be verified. Otherwise Zabbix setup may be unable to continue.

Configure DB connection

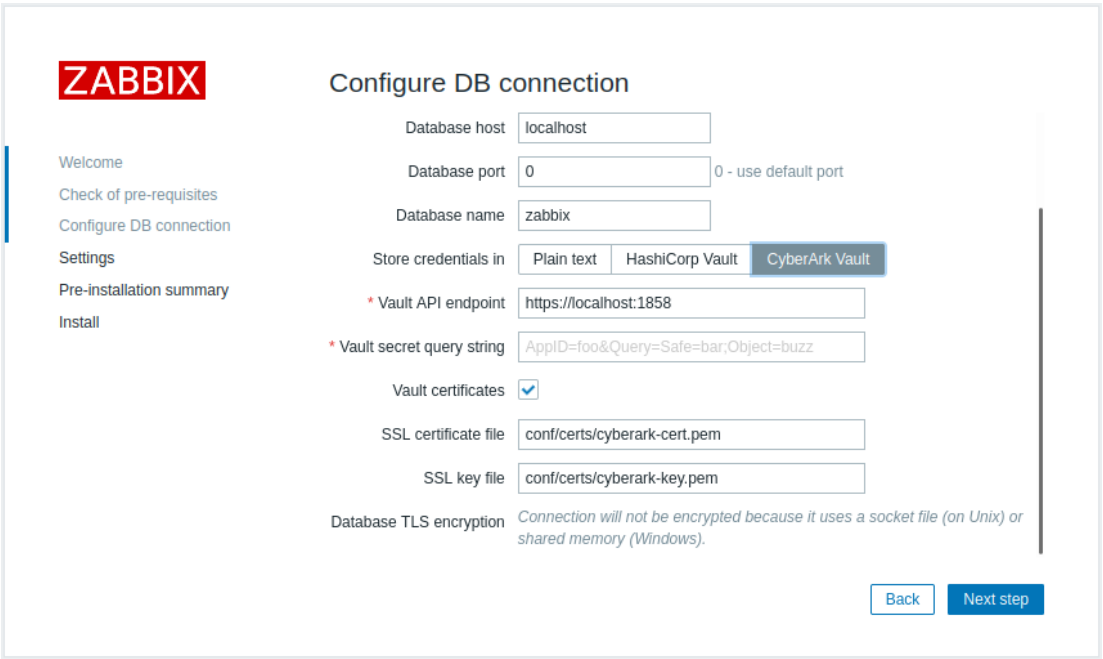
Enter details for connecting to the database. Zabbix database must already be created.



If the *Database TLS encryption* option is checked, then additional fields for configuring the TLS connection to the database appear in the form (MySQL or PostgreSQL only).

If *Store credentials in* is set to HashiCorp Vault or CyberArk Vault, additional parameters will become available:

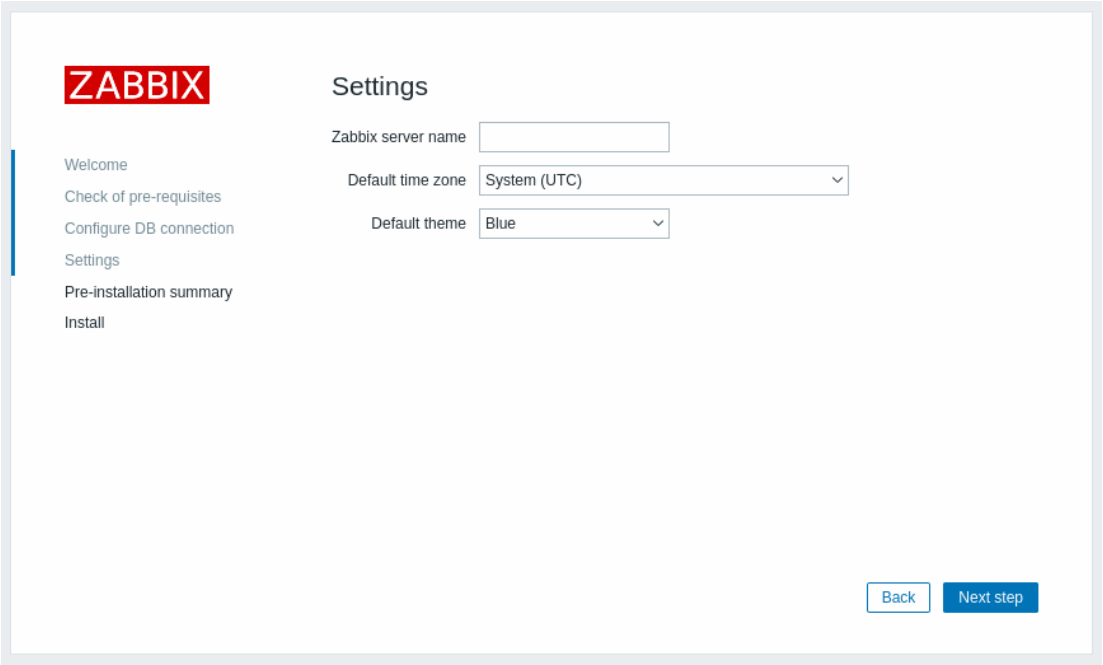
- for HashiCorp Vault: Vault API endpoint, secret path and authentication token;
- for CyberArk Vault: Vault API endpoint, secret query string and certificates. Upon marking *Vault certificates* checkbox, two new fields for specifying paths to SSL certificate file and SSL key file will appear.



Settings

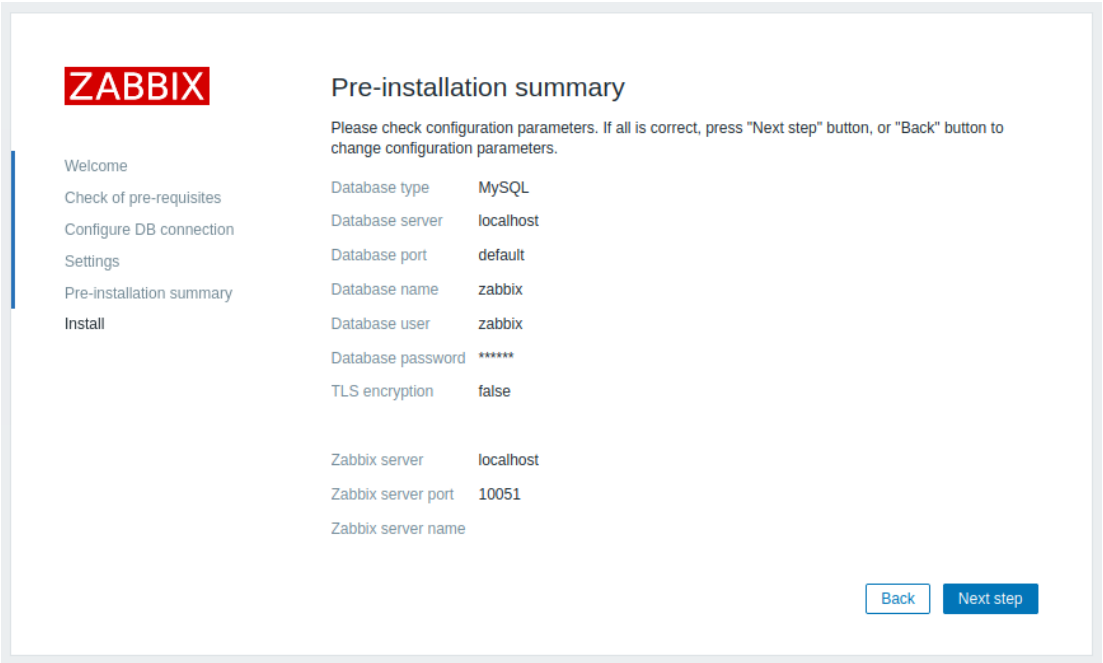
Entering a name for Zabbix server is optional, however, if submitted, it will be displayed in the menu bar and page titles.

Set the default time zone and theme for the frontend.



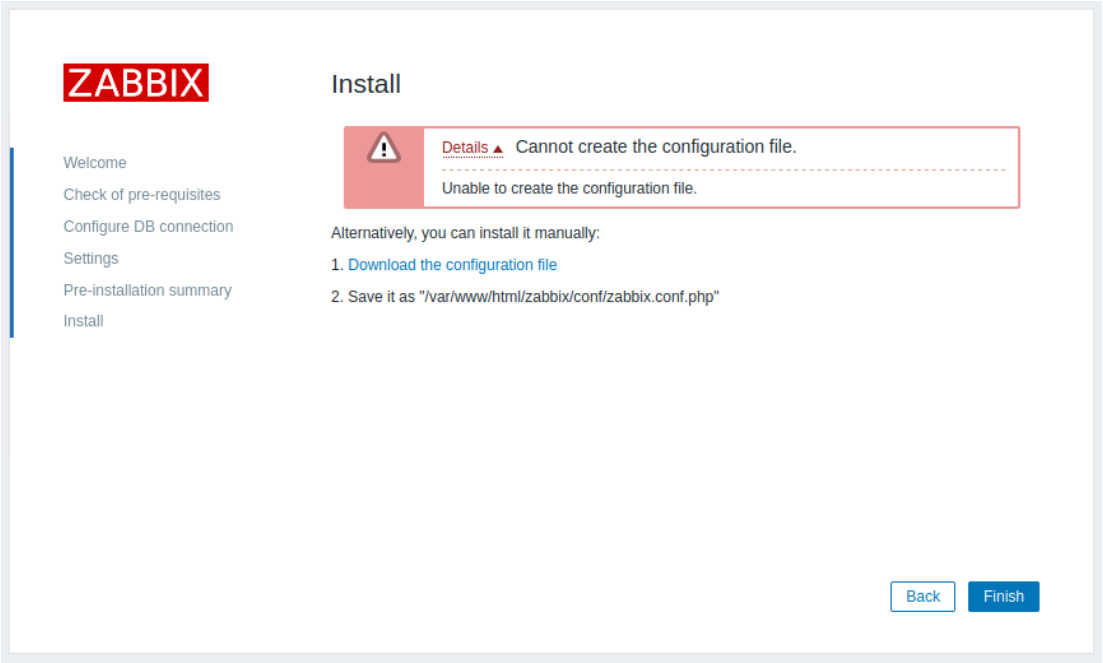
Pre-installation summary

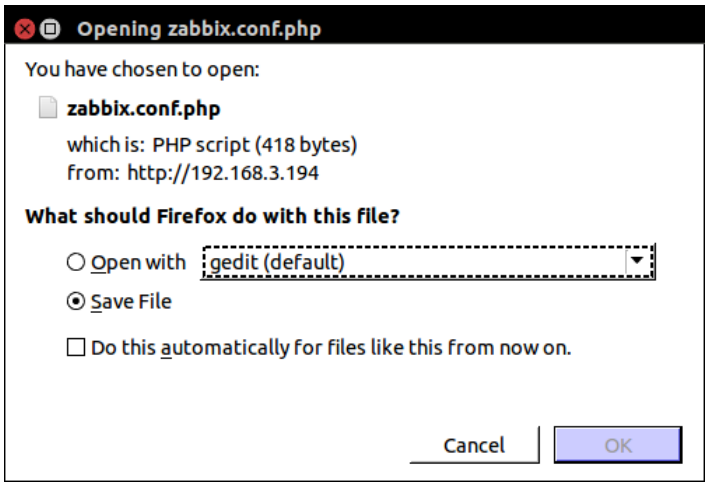
Review a summary of settings.



Install

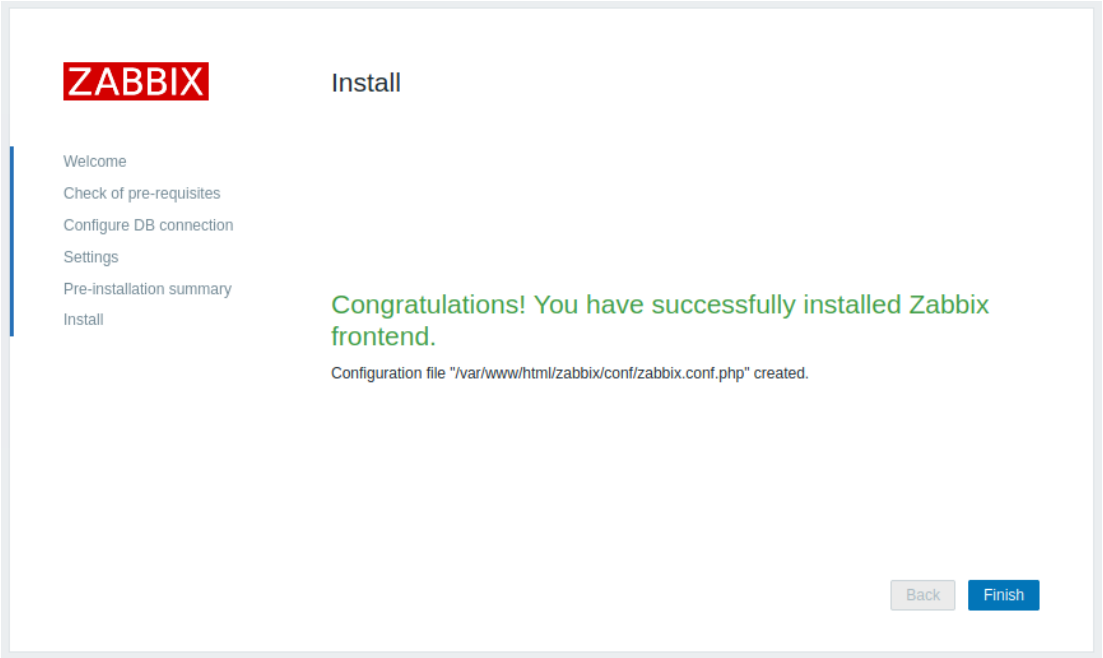
If installing Zabbix from sources, download the configuration file and place it under conf/ in the webserver HTML documents subdirectory where you copied Zabbix PHP files to.





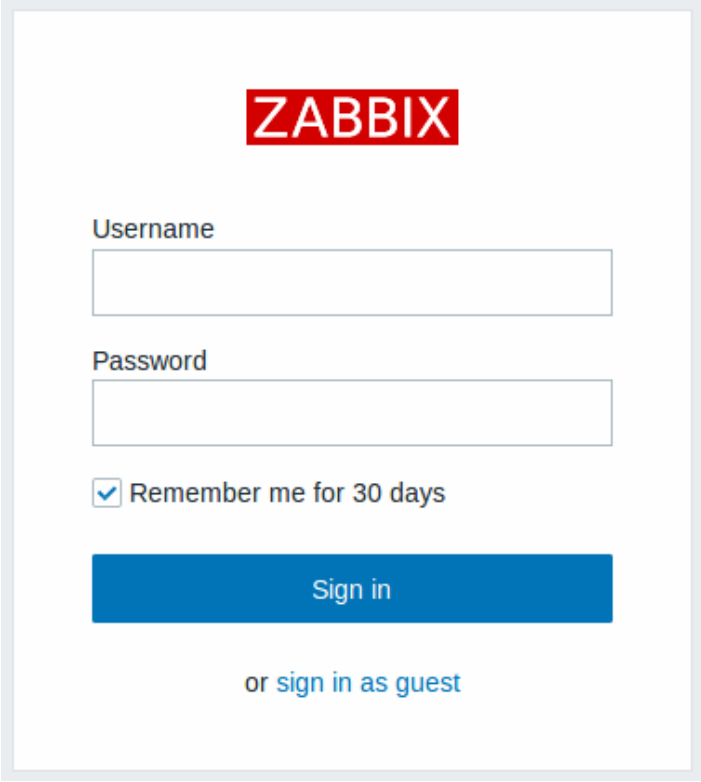
Note:
Providing the webserver user has write access to conf/ directory the configuration file would be saved automatically and it would be possible to proceed to the next step right away.

Finish the installation.



Log in

Zabbix frontend is ready! The default user name is **Admin**, password **zabbix**.



Proceed to getting started with Zabbix.

7 Upgrade procedure

Overview

This section provides upgrade information for Zabbix **6.4**:

- using packages:
  - for Red Hat Enterprise Linux
  - for Debian/Ubuntu
- using sources

See also upgrade instructions for servers in a **high-availability** (HA) cluster.

Upgrading Zabbix agents is recommended but not mandatory.

Upgrading Zabbix proxies is highly recommended. Zabbix server fully supports proxies that are of the same major version as the server. Zabbix server also supports proxies that are **no older** than Zabbix server previous LTS release version, but with limited functionality (data collection, execution of remote commands, immediate item value checks). Configuration update is also disabled, and outdated proxies will only work with old configuration.

Attention:
Proxies that are older than Zabbix server previous LTS release version or newer than Zabbix server major version are not supported. Zabbix server will ignore data from unsupported proxies and all communication with Zabbix server will fail with a warning. For more information, see Version compatibility.

To minimize downtime and data loss during the upgrade, it is recommended to stop, upgrade, and start Zabbix server and then stop, upgrade, and start Zabbix proxies one after another. During server downtime, running proxies will continue data collection. Once the server is up and running, outdated proxies will send the data to the newer server (proxy configuration will not be updated though) and will remain partly functional. Any notifications for problems during Zabbix server downtime will be generated only after the upgraded server is started.

If Zabbix proxy is started for the first time and the SQLite database file is missing, proxy creates it automatically. **Note that starting with Zabbix 6.4.0, if Zabbix proxy uses SQLite3 and on startup detects that existing database file version is older than required, it will delete the database file automatically and create a new one**. Therefore, history data stored in the SQLite database file will be lost. If Zabbix proxy's version is older than the database file version, Zabbix will log an error and exit.

Depending on the database size, the database upgrade to version 6.4 may take a long time.

Direct upgrade to Zabbix 6.4.x is possible from Zabbix **6.2**.x, **6.0**.x, **5.4**.x, **5.2**.x, **5.0**.x, **4.4**.x, **4.2**.x, **4.0**.x, **3.4**.x, **3.2**.x, **3.0**.x, **2.4**.x, **2.2**.x and **2.0**.x. For upgrading from earlier versions consult Zabbix documentation for 2.0 and earlier.

Note:
Please be aware that after upgrading some third-party software integrations in Zabbix might be affected, if the external software is not compatible with the upgraded Zabbix version.

The following upgrade notes are available:

Upgrade from	Read full upgrade notes	Most important changes between versions
6.2.x	For: Zabbix 6.4	Minimum required MySQL version raised from 8.0.0 to 8.0.30. 'libevent_pthreads' library is required for Zabbix server/proxy. Upon the first launch after an upgrade, Zabbix proxy with SQLite3 automatically drops the old version of the database (with all the history) and creates a new one.
6.0.x LTS	For: Zabbix 6.2 Zabbix 6.4	Minimum required PHP version upped from 7.2.5 to 7.4.0. Deterministic triggers need to be created during the upgrade. If binary logging is enabled for MySQL/MariaDB, this requires superuser privileges or setting the variable/configuration parameter *log_bin_trust_function_creators = 1*. See Database creation scripts for instructions how to set the variable.
5.4.x	For: Zabbix 6.0 Zabbix 6.2 Zabbix 6.4	Minimum required database versions upped. Server/proxy will not start if outdated database. Audit log records lost because of database structure change.
5.2.x	For: Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4	Minimum required database versions upped. Aggregate items removed as a separate type.
5.0.x LTS	For: Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4	Minimum required PHP version upped from 7.2.0 to 7.2.5. Password hashing algorithm changed from MD5 to bcrypt.

Upgrade from	Read full upgrade notes	Most important changes between versions
4.4.x	For:	Support of IBM DB2 dropped.
	Zabbix 5.0	Minimum required PHP version upped from 5.4.0 to 7.2.0.
	Zabbix 5.2	Minimum required database versions upped.
	Zabbix 5.4	Changed Zabbix PHP file directory.
	Zabbix 6.0	
	Zabbix 6.2	
	Zabbix 6.4	
4.2.x	For:	Jabber, Ez Texting media types removed.
	Zabbix 4.4	
	Zabbix 5.0	
	Zabbix 5.2	
	Zabbix 5.4	
	Zabbix 6.0	
	Zabbix 6.2	
	Zabbix 6.4	
4.0.x LTS	For:	Older proxies no longer can report data to an upgraded server.
	Zabbix 4.2	Newer agents no longer will be able to work with an older Zabbix
	Zabbix 4.4	server.
	Zabbix 5.0	
	Zabbix 5.2	
	Zabbix 5.4	
	Zabbix 6.0	
	Zabbix 6.2	
	Zabbix 6.4	
3.4.x	For:	'libpthread' and 'zlib' libraries now mandatory.
	Zabbix 4.0	Support for plain text protocol dropped and header is mandatory.
	Zabbix 4.2	Pre-1.4 version Zabbix agents are no longer supported.
	Zabbix 4.4	The Server parameter in passive proxy configuration now mandatory.
	Zabbix 5.0	
	Zabbix 5.2	
	Zabbix 5.4	
	Zabbix 6.0	
	Zabbix 6.2	
	Zabbix 6.4	
3.2.x	For:	SQLite support as backend database dropped for Zabbix
	Zabbix 3.4	server/frontend.
	Zabbix 4.0	Perl Compatible Regular Expressions (PCRE) supported instead of
	Zabbix 4.2	POSIX extended.
	Zabbix 4.4	'libpcre' and 'libevent' libraries mandatory for Zabbix server.
	Zabbix 5.0	Exit code checks added for user parameters, remote commands and
	Zabbix 5.2	system.run[] items without the 'nowait' flag as well as Zabbix server
	Zabbix 5.4	executed scripts.
	Zabbix 6.0	Zabbix Java gateway has to be upgraded to support new functionality.
	Zabbix 6.2	
	Zabbix 6.4	
3.0.x LTS	For:	Database upgrade may be slow, depending on the history table size.
	Zabbix 3.2	
	Zabbix 3.4	
	Zabbix 4.0	
	Zabbix 4.2	
	Zabbix 4.4	
	Zabbix 5.0	
	Zabbix 5.2	
	Zabbix 5.4	
	Zabbix 6.0	
	Zabbix 6.2	
	Zabbix 6.4	

Upgrade from	Read full upgrade notes	Most important changes between versions
2.4.x	For: Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4	Minimum required PHP version upped from 5.3.0 to 5.4.0. LogFile agent parameter must be specified.
2.2.x LTS	For: Zabbix 2.4 Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4	Node-based distributed monitoring removed.
2.0.x	For: Zabbix 2.2 Zabbix 2.4 Zabbix 3.0 Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2 Zabbix 6.4	Minimum required PHP version upped from 5.1.6 to 5.3.0. Case-sensitive MySQL database required for proper server work; character set utf8 and utf8_bin collation is required for Zabbix server to work properly with MySQL database. See database creation scripts. 'mysqli' PHP extension required instead of 'mysql'.

1 Upgrade from sources

Overview

This section provides the steps required for a successful upgrade from Zabbix **6.2**.x to Zabbix **6.4**.x using official Zabbix sources.

Warning:
Before the upgrade make sure to read the relevant **upgrade notes**!

You may also want to check the requirements for 6.4.

Note:
It may be handy to run two parallel SSH sessions during the upgrade, executing the upgrade steps in one and monitoring the server/proxy logs in another. For example, run `tail -f zabbix_server.log` or `tail -f zabbix_proxy.log` in the second SSH session showing you the latest log file entries and possible errors in real time. This can be critical for production instances.

Server upgrade process

1 Stop server

Stop Zabbix server to make sure that no new data is inserted into database.

2 Back up the existing Zabbix database

This is a very important step. Make sure that you have a backup of your database. It will help if the upgrade procedure fails (lack of disk space, power off, any unexpected problem).

3 Back up configuration files, PHP files and Zabbix binaries

Make a backup copy of Zabbix binaries, configuration files and the PHP file directory.

4 Install new server binaries

Use these instructions to compile Zabbix server from sources.

5 Review server configuration parameters

Make sure to review Upgrade notes to check if any changes in the configuration parameters are required.

For new optional parameters, see the What's new page.

6 Start new Zabbix binaries

Start new binaries. Check log files to see if the binaries have started successfully.

Zabbix server will automatically upgrade the database. When starting up, Zabbix server reports the current (mandatory and optional) and required database versions. If the current mandatory version is older than the required version, Zabbix server automatically executes the required database upgrade patches. The start and progress level (percentage) of the database upgrade is written to the Zabbix server log file. When the upgrade is completed, a "database upgrade fully completed" message is written to the log file. If any of the upgrade patches fail, Zabbix server will not start. Zabbix server will also not start if the current mandatory database version is newer than the required one. Zabbix server will only start if the current mandatory database version corresponds to the required mandatory version.

```
8673:20161117:104750.259 current database version (mandatory/optional): 03040000/03040000
8673:20161117:104750.259 required mandatory version: 03040000
```

Before you start the server:

- Make sure the database user has enough permissions (create table, drop table, create index, drop index)
- Make sure you have enough free disk space.

7 Install new Zabbix web interface

The minimum required PHP version is 7.4. Update if needed and follow installation instructions.

8 Clear web browser cookies and cache

After the upgrade you may need to clear web browser cookies and web browser cache for the Zabbix web interface to work properly.

Proxy upgrade process

1 Stop proxy

Stop Zabbix proxy.

2 Back up configuration files and Zabbix proxy binaries

Make a backup copy of the Zabbix proxy binary and configuration file.

3 Install new proxy binaries

Use these instructions to compile Zabbix proxy from sources.

4 Review proxy configuration parameters

There are no mandatory changes in this version to proxy parameters.

5 Start new Zabbix proxy

Start the new Zabbix proxy. Check log files to see if the proxy has started successfully.

Zabbix proxy will automatically upgrade the database. Database upgrade takes place similarly as when starting Zabbix server.

Agent upgrade process

Attention:
Upgrading agents is not mandatory. You only need to upgrade agents if it is required to access the new functionality.

The upgrade procedure described in this section may be used for upgrading both the Zabbix agent and the Zabbix agent 2.

1 Stop agent

Stop Zabbix agent.

2 Back up configuration files and Zabbix agent binaries

Make a backup copy of the Zabbix agent binary and configuration file.

3 Install new agent binaries

Use these instructions to compile Zabbix agent from sources.

Alternatively, you may download pre-compiled Zabbix agents from the Zabbix download page.

4 Review agent configuration parameters

There are no mandatory changes in this version neither to agent nor to agent 2 parameters.

5 Start new Zabbix agent

Start the new Zabbix agent. Check log files to see if the agent has started successfully.

Upgrade between minor versions

When upgrading between minor versions of 6.4.x (for example from 6.4.1 to 6.4.3) it is required to execute the same actions for server/proxy/agent as during the upgrade between major versions. The only difference is that when upgrading between minor versions no changes to the database are made.

2 Upgrade from packages

Overview

This section provides the steps required for a successful upgrade using official RPM and DEB packages provided by Zabbix for:

- Red Hat Enterprise Linux
- Debian/Ubuntu

Zabbix packages from OS repositories

Often, OS distributions (in particular, Debian-based distributions) provide their own Zabbix packages.
Note that these packages are not supported by Zabbix, they are typically out of date and lack the latest features and bug fixes. Only the packages from repo.zabbix.com are officially supported.

If you are upgrading from packages provided by OS distributions (or had them installed at some point), follow this procedure to switch to official Zabbix packages:

1. Always uninstall the old packages first.
2. Check for residual files that may have been left after deinstallation.
3. Install official packages following installation instructions provided by Zabbix.

Never do a direct update, as this may result in a broken installation.

1 Red Hat Enterprise Linux

Overview

This section provides the steps required for a successful upgrade from Zabbix **6.2**.x to the latest version of Zabbix **6.4**.x using official Zabbix packages for Red Hat Enterprise Linux.

Warning:
Before the upgrade make sure to read the relevant **upgrade notes**!

You may also want to check the requirements for 6.4.

Note:
It may be handy to run two parallel SSH sessions during the upgrade, executing the upgrade steps in one and monitoring the server/proxy logs in another. For example, run `tail -f zabbix_server.log` or `tail -f zabbix_proxy.log` in the second SSH session showing you the latest log file entries and possible errors in real time. This can be critical for production instances.

For instructions on upgrading between Zabbix 6.4.x minor versions (for example, from 6.4.1 to 6.4.3), see Upgrade between minor versions.

Upgrade procedure

1 Stop Zabbix processes

Stop Zabbix server to make sure that no new data is inserted into database.

```
# systemctl stop zabbix-server
```

If upgrading Zabbix proxy, agent, or agent 2, stop these components too:

```
# systemctl stop zabbix-proxy
# systemctl stop zabbix-agent
# systemctl stop zabbix-agent2
```

2 Back up the existing Zabbix database

This is a very important step. Make sure that you have a backup of your database. It will help if the upgrade procedure fails (lack of disk space, power off, any unexpected problem).

3 Back up configuration files, PHP files and Zabbix binaries

Make a backup copy of Zabbix binaries, configuration files and the PHP file directory.

Configuration files:

```
# mkdir /opt/zabbix-backup/
# cp /etc/zabbix/zabbix_server.conf /opt/zabbix-backup/
# cp /etc/httpd/conf.d/zabbix.conf  /opt/zabbix-backup/
```

PHP files and Zabbix binaries:

```
# cp -R /usr/share/zabbix/ /opt/zabbix-backup/
# cp -R /usr/share/zabbix-* /opt/zabbix-backup/
```

4 Update repository configuration package

Before proceeding with the upgrade, update your current repository package to the latest version to ensure compatibility with the newest packages and to include any recent security patches or bug fixes.

On **RHEL 9**, run:

```
# rpm -Uvh https://repo.zabbix.com/zabbix/6.4/rhel/9/x86_64/zabbix-release-latest.el9.noarch.rpm
```

On **RHEL 8**, run:

```
# rpm -Uvh https://repo.zabbix.com/zabbix/6.4/rhel/8/x86_64/zabbix-release-latest.el8.noarch.rpm
```

For older RHEL versions, replace the link above with the correct one from Zabbix repository. Note, however, that packages for those versions may not include all Zabbix components. For a list of components included, see Zabbix packages.

Then, clean up the `dnf` package manager's cache (including headers, metadata, and package files downloaded during previous installations or updates):

```
# dnf clean all
```

On the next `dnf` operation, `dnf` will download fresh metadata from the repositories since the old metadata is cleared.

See also: Known issues for updating the repository configuration package on RHEL.

5 Upgrade Zabbix components

To upgrade Zabbix components you may run something like:

```
# dnf install zabbix-server-mysql zabbix-web-mysql zabbix-agent
```

- If using PostgreSQL, substitute `mysql` with `pgsql` in the command.
- If upgrading the proxy, substitute `server` with `proxy` in the command.
- If upgrading the agent 2, substitute `zabbix-agent` with `zabbix-agent2 zabbix-agent2-plugin-*` in the command.

Attention:
Upgrading Zabbix agent 2 with the `dnf install zabbix-agent2` command could lead to an error. For more information, see *Known issues*.

Then, to upgrade the web frontend with Apache and restart Apache, run:

```
# dnf install zabbix-apache-conf
# systemctl restart httpd
```

6 Review component configuration parameters

Make sure to review Upgrade notes to check if any changes in the configuration parameters are required.

For new optional parameters, see the What's new page.

7 Start Zabbix processes

Start the updated Zabbix components.

```
# systemctl start zabbix-server
# systemctl start zabbix-proxy
# systemctl start zabbix-agent
# systemctl start zabbix-agent2
```

8 Clear web browser cookies and cache

After the upgrade you may need to clear web browser cookies and web browser cache for the Zabbix web interface to work properly.

Upgrade between minor versions

It is possible to upgrade between Zabbix 6.4.x minor versions (for example, from 6.4.1 to 6.4.3).

To upgrade Zabbix minor version, please run:

```
# dnf upgrade 'zabbix-*'
```

To upgrade Zabbix server minor version only, please run:

```
# dnf upgrade 'zabbix-server-*'
```

To upgrade Zabbix agent minor version only, please run:

```
# dnf upgrade 'zabbix-agent-*'
```

or, for Zabbix agent 2:

```
# dnf upgrade 'zabbix-agent2-*'
```

2 Debian/Ubuntu

Overview

This section provides the steps required for a successful upgrade from Zabbix **6.2**.x to the latest version of Zabbix **6.4**.x using official Zabbix packages for Debian/Ubuntu.

Warning:
Before the upgrade make sure to read the relevant **upgrade notes**!

You may also want to check the requirements for 6.4.

Note:
It may be handy to run two parallel SSH sessions during the upgrade, executing the upgrade steps in one and monitoring the server/proxy logs in another. For example, run `tail -f zabbix_server.log` or `tail -f zabbix_proxy.log` in the second SSH session showing you the latest log file entries and possible errors in real time. This can be critical for production instances.

For instructions on upgrading between Zabbix 6.4.x minor versions (for example, from 6.4.1 to 6.4.3), see Upgrade between minor versions.

Upgrade procedure

1 Stop Zabbix processes

Stop Zabbix server to make sure that no new data is inserted into database.

```
# systemctl stop zabbix-server
```

If upgrading Zabbix proxy, agent, or agent 2, stop these components too:

```
# systemctl stop zabbix-proxy
# systemctl stop zabbix-agent
# systemctl stop zabbix-agent2
```

2 Back up the existing Zabbix database

This is a very important step. Make sure that you have a backup of your database. It will help if the upgrade procedure fails (lack of disk space, power off, any unexpected problem).

3 Back up configuration files, PHP files and Zabbix binaries

Make a backup copy of Zabbix binaries, configuration files and the PHP file directory.

Configuration files:

```
# mkdir /opt/zabbix-backup/
# cp /etc/zabbix/zabbix_server.conf /opt/zabbix-backup/
# cp /etc/apache2/conf-enabled/zabbix.conf /opt/zabbix-backup/
```

PHP files and Zabbix binaries:

```
# cp -R /usr/share/zabbix/ /opt/zabbix-backup/
# cp -R /usr/share/zabbix-* /opt/zabbix-backup/
```

4 Update repository configuration package

Before proceeding with the upgrade, uninstall your current repository package:

```
# rm -Rf /etc/apt/sources.list.d/zabbix.list
```

Then, install the latest repository configuration package to ensure compatibility with the newest packages and to include any recent security patches or bug fixes.

On **Debian 12**, run:

```
# wget https://repo.zabbix.com/zabbix/6.4/debian/pool/main/z/zabbix-release/zabbix-release_latest+debian12
# dpkg -i zabbix-release_latest+debian12_all.deb
```

On **Debian 11**, run:

```
# wget https://repo.zabbix.com/zabbix/6.4/debian/pool/main/z/zabbix-release/zabbix-release_latest+debian11
# dpkg -i zabbix-release_latest+debian11_all.deb
```

For older Debian versions, replace the link above with the correct one from Zabbix repository. Note, however, that packages for those versions may not include all Zabbix components. For a list of components included, see Zabbix packages.

On **Ubuntu 24.04**, run:

```
# wget https://repo.zabbix.com/zabbix/6.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_latest+ubuntu24
# dpkg -i zabbix-release_latest+ubuntu24.04_all.deb
```

On **Ubuntu 22.04**, run:

```
# wget https://repo.zabbix.com/zabbix/6.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_latest+ubuntu22
# dpkg -i zabbix-release_latest+ubuntu22.04_all.deb
```

On **Ubuntu 20.04**, run:

```
# wget https://repo.zabbix.com/zabbix/6.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_latest+ubuntu20
# dpkg -i zabbix-release_latest+ubuntu20.04_all.deb
```

For older Ubuntu versions, replace the link above with the correct one from Zabbix repository. Note, however, that packages for those versions may not include all Zabbix components. For a list of components included, see Zabbix packages.

You may see a prompt about the Zabbix repository configuration:

```
# Configuration file '/etc/apt/sources.list.d/zabbix.list'
# ==> Deleted (by you or by a script) since installation.
# ==> Package distributor has shipped an updated version.
# What would you like to do about it ?  Your options are:
# Y or I  : install the package maintainer's version
```

```
# N or O  : keep your currently-installed version
# D       : show the differences between the versions
# Z       : start a shell to examine the situation
# The default action is to keep your current version.
# *** zabbix.list (Y/I/N/O/D/Z) [default=N] ?
```

Enter `Y` (or `I`) to install the package maintainer's version of the Zabbix repository configuration.

Then, update the repository information:

```
# apt update
```

5 Upgrade Zabbix components

To upgrade Zabbix components you may run something like:

```
# apt install --only-upgrade zabbix-server-mysql zabbix-frontend-php zabbix-agent
```

- If using PostgreSQL, substitute `mysql` with `pgsql` in the command.
- If upgrading the proxy, substitute `server` with `proxy` in the command.
- If upgrading the Zabbix agent 2, substitute `zabbix-agent` with `zabbix-agent2 zabbix-agent2-plugin-*` in the command.

Attention:
Upgrading Zabbix agent 2 with the `apt install zabbix-agent2` command could lead to an error. For more information, see *Known issues*.

You may see a prompt about the Zabbix server (or proxy) configuration:

```
# Configuration file '/etc/zabbix/zabbix_server.conf'
# ==> Modified (by you or by a script) since installation.
# ==> Package distributor has shipped an updated version.
# What would you like to do about it ?  Your options are:
# Y or I  : install the package maintainer's version
# N or O  : keep your currently-installed version
# D       : show the differences between the versions
# Z       : start a shell to examine the situation
# The default action is to keep your current version.
# *** zabbix_server.conf (Y/I/N/O/D/Z) [default=N] ?
```

Enter the option that best fits your situation. For example, enter `D` to compare the current and new configuration, then decide if you want to install the package maintainer's version (`Y` or `I`).

Then, to upgrade the web frontend with Apache and restart Apache, run:

```
# apt install zabbix-apache-conf
# systemctl restart apache2
```

Distributions **prior to Debian 10 (buster)** / **Ubuntu 18.04 (bionic)** / **Raspbian 10 (buster)** do not provide PHP 7.2 or newer, which is required for Zabbix frontend 5.0. See information about installing Zabbix frontend on older distributions.

6 Review component configuration parameters

Make sure to review Upgrade notes to check if any changes in the configuration parameters are required.

For new optional parameters, see the What's new page.

7 Start Zabbix processes

Start the updated Zabbix components.

```
# systemctl start zabbix-server
# systemctl start zabbix-proxy
# systemctl start zabbix-agent
# systemctl start zabbix-agent2
```

8 Clear web browser cookies and cache

After the upgrade you may need to clear web browser cookies and web browser cache for the Zabbix web interface to work properly.

Upgrade between minor versions

It is possible to upgrade between Zabbix 6.4.x minor versions (for example, from 6.4.1 to 6.4.3).

First, update the repository information:

```
# apt update
```

Then, to upgrade Zabbix minor version, please run:

```
# apt install --only-upgrade 'zabbix*'
```

To upgrade Zabbix server minor version only, please run:

```
# apt install --only-upgrade 'zabbix-server*'
```

To upgrade Zabbix agent minor version only, please run:

```
# apt install --only-upgrade 'zabbix-agent*'
```

or, for Zabbix agent 2:

```
# apt install --only-upgrade 'zabbix-agent2*'
```

3 Upgrade from containers

Overview

This section describes steps required for a successful upgrade to Zabbix **6.4**.x containers.

Separate sets of instructions are available for upgrading individual Zabbix component images and Docker compose files.

Warning:
Before the upgrade make sure to read the relevant **upgrade notes**!

Attention:
Before starting the upgrade, verify that users have the necessary permissions to the database for upgrading purposes.

For upgrades from Zabbix 6.0 or older, deterministic triggers will need to be created during the upgrade. If binary logging is enabled for MySQL/MariaDB, this requires superuser privileges or setting the variable/configuration parameter *log_bin_trust_function_creators = 1*. See Database creation scripts for instructions how to set the variable.
Note that if executing from a console, the variable will only be set temporarily and will be dropped when a Docker is restarted. In this case, keep your SQL service running, only stop zabbix-server service by running 'docker compose down zabbix-server' and then 'docker compose up -d zabbix-server'.
Alternatively, you can set this variable in the configuration file.

Depending on the size of a database upgrade to version 6.4 may take quite a long time.

Zabbix image upgrade

The steps listed below can be used to upgrade any Zabbix component. Replace `zabbix-server-mysql` with the required component image name.

1. Check current image version:

```
docker inspect -f '{{ .Config.Image }}' zabbix-server-mysql
```

2. Pull desired image version, for example:

```
docker pull zabbix/zabbix-server-mysql:alpine-6.4-latest
```

`zabbix/zabbix-server-mysql:alpine-6.4-latest` will pull the latest released minor version of Zabbix server 6.4 with MySQL support based on Alpine Linux. Replace it with the name of the Docker repository and tags combination you need. See Installation from containers for a list of available options.

3. Stop the container:

```
docker stop zabbix-server-mysql
```

4. Remove the container:

```
docker rm zabbix-server-mysql
```

5. Launch the updated container by executing `docker run` command followed by additional arguments to specify required environment variables and/or mount points.

Configuration examples

Zabbix server with MySQL:

```
docker run --name zabbix-server-mysql -t \
      -e DB_SERVER_HOST="mysql-server" \
      -e MYSQL_DATABASE="zabbix" \
      -e MYSQL_USER="zabbix" \
      -e MYSQL_PASSWORD="zabbix_pwd" \
      -e MYSQL_ROOT_PASSWORD="root_pwd" \
      -e ZBX_JAVAGATEWAY="zabbix-java-gateway" \
      --network=zabbix-net \
      -p 10051:10051 \
      --restart unless-stopped \
      -d zabbix/zabbix-server-mysql:alpine-6.4-latest
```

Zabbix server with PostgreSQL:

```
docker run --name zabbix-server-pgsql -t \
         -e DB_SERVER_HOST="postgres-server" \
         -e POSTGRES_USER="zabbix" \
         -e POSTGRES_PASSWORD="zabbix_pwd" \
         -e POSTGRES_DB="zabbix" \
         -e ZBX_ENABLE_SNMP_TRAPS="true" \
         --network=zabbix-net \
         -p 10051:10051 \
         --volumes-from zabbix-snmptraps \
         --restart unless-stopped \
         -d zabbix/zabbix-server-pgsql:alpine-6.4-latest
```

More configuration examples, including examples for other Zabbix components, are available on the Installation from containers page.

6. Verify the update:

```
docker logs -f zabbix-server-mysql
```

Compose files

Follow upgrade instructions in this section, if you installed Zabbix using compose file.

1. Check current image version:

```
docker inspect -f '{{ .Config.Image }}' zabbix-server-mysql
```

2. Pull the latest updates from the GitHub repository and switch to the required branch:

```
git pull
git checkout 6.4
```

3. Start Zabbix components using new compose file:

```
docker-compose -f ./docker-compose_v3_alpine_mysql_latest.yaml up -d
```

4. Verify the update:

```
docker logs -f zabbix-server-mysql
```

See Installation from containers for more details, including lists of supported environment variables and volume mount points.

8 Known issues

See also: Compilation issues.

Template nesting issue in Zabbix 6.4.0rc1

Zabbix 6.4.0rc1 (rc1 = Release Candidate 1) does not support template nesting (restored in 6.4.0rc2). If you have upgraded to Zabbix 6.4.0rc1, a DB patch will convert all nested templates into a flat template structure. This means that all entities (items, triggers, etc.) from nested templates will be transferred to the template that contained these nested templates. The support for template nesting has been fully restored in Zabbix 6.4.0rc2. However, if you have already upgraded to Zabbix 6.4.0rc1, the previously existing template structure will not be recovered.

Proxy startup with MySQL 8.0.0-8.0.17

zabbix_proxy on MySQL versions 8.0.0-8.0.17 fails with the following "access denied" error:

```
[Z3001] connection to database 'zabbix' failed: [1227] Access denied; you need (at least one of) the SUPER
```

That is due to MySQL 8.0.0 starting to enforce special permissions for setting session variables. However, in 8.0.18 this behavior was removed: As of MySQL 8.0.18, setting the session value of this system variable is no longer a restricted operation.

The workaround is based on granting additional privileges to the `zabbix` user:

For MySQL versions 8.0.14 - 8.0.17:

```
grant SESSION_VARIABLES_ADMIN on *.* to 'zabbix'@'localhost';
```

For MySQL versions 8.0.0 - 8.0.13:

```
grant SYSTEM_VARIABLES_ADMIN on *.* to 'zabbix'@'localhost';
```

Timescale DB: high memory usage with large number of partitions

PostgreSQL versions 9.6-12 use too much memory when updating tables with a large number of partitions (see problem report). This issue manifests itself when Zabbix updates trends on systems with TimescaleDB if trends are split into relatively small (e.g. 1 day) chunks. This leads to hundreds of chunks present in the trends tables with default housekeeping settings - the condition where PostgreSQL is likely to run out of memory.

The issue has been resolved since Zabbix 5.0.1 for new installations with TimescaleDB, but if TimescaleDB was set up with Zabbix before that, please see ZBX-16347 for the migration notes.

Timescale DB 2.5.0: compression policy can fail on tables that contain integers

This issue manifests when TimescaleDB 2.5.0/2.5.1 is used. It has been resolved since TimescaleDB 2.5.2.

For more information, please see TimescaleDB Issue #3773.

Upgrade

SQL mode setting for successful upgrade

The `sql_mode` setting in MySQL/MariaDB must have the "STRICT_TRANS_TABLES" mode set. If it is absent, the Zabbix database upgrade will fail (see also ZBX-19435).

Upgrade with MariaDB 10.2.1 and before

Upgrading Zabbix may fail if database tables were created with MariaDB 10.2.1 and before, because in those versions the default row format is compact. This can be fixed by changing the row format to dynamic (see also ZBX-17690).

Templates

Template compatibility in dual-stack (IPv4/IPv6) environments

In dual-stack environments (systems configured to support both IPv4 and IPv6), the hostname `localhost` typically resolves to both IPv4 and IPv6 addresses. Due to the common prioritization of IPv6 over IPv4 by many operating systems and DNS resolvers, Zabbix templates may fail to work correctly if the service being monitored is configured to listen only on IPv4.

Services that are not configured to listen on IPv6 addresses may become inaccessible, leading to monitoring failures. Users might configure access correctly for IPv4 but still face connectivity issues due to the default behavior of prioritizing IPv6.

A workaround for this is to ensure that the services (Nginx, Apache, PostgreSQL, etc.) are configured to listen on both IPv4 and IPv6 addresses, and Zabbix server/agent is allowed access via IPv6. Additionally, in Zabbix templates and configurations, use `localhost` explicitly instead of `127.0.0.1` to ensure compatibility with both IPv4 and IPv6.

For example, when monitoring PostgreSQL with the PostgreSQL by Zabbix agent 2 template, you may need to edit the `pg_hba.conf` file to allow connections for the `zbx_monitor` user. If the dual-stack environment prioritizes IPv6 (system resolves localhost to `::1`) and you configure `localhost` but only add an IPv4 entry (`127.0.0.1/32`), the connection will fail because there is no matching IPv6 entry.

The following `pg_hba.conf` file example ensures that the `zbx_monitor` user can connect to any database from the local machine using both IPv4 and IPv6 addresses with different authentication methods:

```
# TYPE      DATABASE      USER            ADDRESS           METHOD
  host      all           zbx_monitor     localhost         trust
  host      all           zbx_monitor     127.0.0.1/32      md5
  host      all           zbx_monitor     ::1/128           scram-sha-256
```

If necessary, you can also use the IPv4 address (`127.0.0.1`) directly when configuring the PostgreSQL by Zabbix agent 2 template macro for the connection string.

Accidental installation of EPEL Zabbix packages

With EPEL repository installed and enabled, installing Zabbix from packages will lead to EPEL Zabbix packages being installed rather than official Zabbix packages.

In this case uninstall Zabbix packages from EPEL, i.e.:

```
dnf remove zabbix-server-mysql
```

Block Zabbix packages from EPEL. Add the following line in the `/etc/yum.conf` file:

```
exclude=zabbix6.4*
```

Install Zabbix server again:

```
dnf install zabbix-server-mysql
```

Notice that official Zabbix packages have the word `release` in their version string:

```
6.4.10-release1.el8
```

Zabbix packages for RHEL on Red Hat UBI environments

When installing Zabbix from Red Hat Enterprise Linux packages on Red Hat Universal Base Image environments, ensure access to required repositories and dependencies. Zabbix packages depend on `libOpenIPMI.so` and `libOpenIPMIposix.so` libraries, which are not provided by any package in the default package manager repositories enabled on UBI systems and will result in installation failures.

The `libOpenIPMI.so` and `libOpenIPMIposix.so` libraries are available in the `OpenIPMI-libs` package, which is provided by the `redhat-#-for-<arch>-appstream-rpms` repository. Access to this repository is curated by subscriptions, which, in the case of UBI environments, get propagated by mounting repository configuration and secrets directories of the RHEL host into the container file-system namespace.

For more information, see ZBX-24291.

Expired signing key for RHEL packages

When upgrading Zabbix on Red Hat Enterprise Linux, you may encounter an expired signing key issue for packages on Zabbix repository. When a signing key expires, attempts to verify package signatures will result in an error indicating that the certificate or key is no longer valid. For example:

```
error: Verifying a signature using certificate D9AA84C2B617479C6E4FCF4D19F2475308EFA7DD (Zabbix LLC (Jul 2
  1. Certificate 19F2475308EFA7DD invalid: certificate is not alive
      because: The primary key is not live
      because: Expired on 2024-07-04T11:41:23Z
  2. Key 19F2475308EFA7DD invalid: key is not alive
      because: The primary key is not live
      because: Expired on 2024-07-04T11:41:23Z
```

To resolve such issues, manually reinstall the latest `zabbix-release` package for your specific variant of RHEL (replace the link below with the correct one from Zabbix repository).

For example, on **RHEL 9**, run:

```
rpm -Uvh https://repo.zabbix.com/zabbix/6.4/rhel/9/x86_64/zabbix-release-latest.el9.noarch.rpm
```

Then, update the repository information:

```
dnf update
```

For more information, see ZBX-24761.

Database TLS connection with MariaDB

Database TLS connection is not supported with the 'verify_ca' option for the DBTLSConnect parameter if MariaDB is used.

Possible deadlocks with MySQL/MariaDB

When running under high load, and with more than one LLD worker involved, it is possible to run into a deadlock caused by an InnoDB error related to the row-locking strategy (see upstream bug). The error has been fixed in MySQL since 8.0.29, but not in MariaDB. For more details, see ZBX-21506.

Global event correlation

Events may not get correlated correctly if the time interval between the first and second event is very small, i.e. half a second and less.

Numeric (float) data type range with PostgreSQL 11 and earlier

PostgreSQL 11 and earlier versions only support floating point value range of approximately -1.34E-154 to 1.34E+154.

NetBSD 8.0 and newer

Various Zabbix processes may randomly crash on startup on the NetBSD versions 8.X and 9.X. That is due to the too small default stack size (4MB), which must be increased by running:

```
ulimit -s 10240
```

For more information, please see the related problem report: ZBX-18275.

Regular expression limitations in Zabbix agent 2

Zabbix agent 2 does not support lookaheads and lookbehinds in regular expressions due to the standard Go regexp library limitations.

IPMI checks

IPMI checks will not work with the standard OpenIPMI library package on Debian prior to 9 (stretch) and Ubuntu prior to 16.04 (xenial). To fix that, recompile OpenIPMI library with OpenSSL enabled as discussed in ZBX-6139.

SSH checks

- Some Linux distributions like Debian, Ubuntu do not support encrypted private keys (with passphrase) if the libssh2 library is installed from packages. Please see ZBX-4850 for more details.
- When using libssh 0.9.x on some Linux distributions with OpenSSH 8, SSH checks may occasionally report "Cannot read data from SSH server". This is caused by a libssh issue (more detailed report). The error is expected to have been fixed by a stable libssh 0.9.5 release. See also ZBX-17756 for details.
- Using the pipe "|" in the SSH script may lead to a "Cannot read data from SSH server" error. In this case it is recommended to upgrade the libssh library version. See also ZBX-21337 for details.

ODBC checks

- MySQL unixODBC driver should not be used with Zabbix server or Zabbix proxy compiled against MariaDB connector library and vice versa, if possible it is also better to avoid using the same connector as the driver due to an upstream bug. Suggested setup:

  PostgreSQL, SQLite or Oracle connector → MariaDB or MySQL unixODBC driver MariaDB connector → MariaDB unixODBC driver MySQL connector → MySQL unixODBC driver

See ZBX-7665 for more information and available workarounds.

- XML data queried from Microsoft SQL Server may get truncated in various ways on Linux and UNIX systems.
- It has been observed that using ODBC checks for monitoring Oracle databases using various versions of Oracle Instant Client for Linux causes Zabbix server to crash. See also: ZBX-18402, ZBX-20803.
- If using FreeTDS UnixODBC driver, you need to prepend a 'SET NOCOUNT ON' statement to an SQL query (for example, `SET NOCOUNT ON DECLARE @strsql NVARCHAR(max) SET @strsql = ....`). Otherwise, database monitor item in Zabbix will fail to retrieve the information with an error "SQL query returned empty result".
  See ZBX-19917 for more information.

Incorrect request method parameter in items

The request method parameter, used only in HTTP checks, may be incorrectly set to '1', a non-default value for all items as a result of upgrade from a pre-4.0 Zabbix version. For details on how to fix this situation, see ZBX-19308.

Web monitoring and HTTP agent

Zabbix server leaks memory on some Linux distributions due to an upstream bug when "SSL verify peer" is enabled in web scenarios or HTTP agent. Please see ZBX-10486 for more information and available workarounds.

Simple checks

There is a bug in **fping** versions earlier than v3.10 that mishandles duplicate echo replay packets. This may cause unexpected results for `icmpping`, `icmppingloss`, `icmppingsec` items. It is recommended to use the latest version of **fping**. Please see ZBX-11726 for more details.

Errors with fping execution in rootless containers

When containers are running in rootless mode or in a specific-restrictions environment, you may face errors related to fping execution when performing ICMP checks, such as `fping: Operation not permitted` or all packets to all resources lost.

To fix this problem add `--cap-add=net_raw` to "docker run" or "podman run" commands.

Additionally fping execution in non-root environments may require sysctl modification, i.e.:

```
sudo sysctl -w "net.ipv4.ping_group_range=0 1995"
```

where ”1995” is the zabbix GID. For more details, see ZBX-22833.

SNMP checks

If the OpenBSD operating system is used, a use-after-free bug in the Net-SNMP library up to the 5.7.3 version can cause a crash of Zabbix server if the SourceIP parameter is set in the Zabbix server configuration file. As a workaround, please do not set the SourceIP parameter. The same problem applies also for Linux, but it does not cause Zabbix server to stop working. A local patch for the net-snmp package on OpenBSD was applied and will be released with OpenBSD 6.3.

SNMP data spikes

Spikes in SNMP data have been observed that may be related to certain physical factors like voltage spikes in the mains. See ZBX-14318 more details.

SNMP traps

The ”net-snmp-perl” package, needed for SNMP traps, has been removed in RHEL 8.0-8.2; re-added in RHEL 8.3.

So if you are using RHEL 8.0-8.2, the best solution is to upgrade to RHEL 8.3.

Please also see ZBX-17192 for more information.

Alerter process crash in RHEL 7

Instances of a Zabbix server alerter process crash have been encountered in RHEL 7. Please see ZBX-10461 for details.

Upgrading Zabbix agent 2 (6.0.5 or older)

When upgrading Zabbix agent 2 (version 6.0.5 or older) from packages, a plugin-related file conflict error may occur. To fix the error, back up your agent 2 configuration (if necessary), uninstall agent 2 and install it anew.

On RHEL-based systems, run:

```
dnf remove zabbix-agent2
dnf install zabbix-agent2
```

On Debian-based systems, run:

```
apt remove zabbix-agent2
apt install zabbix-agent2
```

For more information, see ZBX-23250.

Flipping frontend locales

It has been observed that frontend locales may flip without apparent logic, i. e. some pages (or parts of pages) are displayed in one language while other pages (or parts of pages) in a different language. Typically the problem may appear when there are several users, some of whom use one locale, while others use another.

A known workaround to this is to disable multithreading in PHP and Apache.

The problem is related to how setting the locale works in PHP: locale information is maintained per process, not per thread. So in a multi-thread environment, when there are several projects run by same Apache process, it is possible that the locale gets changed in another thread and that changes how data can be processed in the Zabbix thread.

For more information, please see related problem reports:

- ZBX-10911 (Problem with flipping frontend locales)
- ZBX-16297 (Problem with number processing in graphs using the `bcdiv` function of BC Math functions)

PHP 7.3 opcache configuration

If ”opcache” is enabled in the PHP 7.3 configuration, Zabbix frontend may show a blank screen when loaded for the first time. This is a registered PHP bug. To work around this, please set the ”opcache.optimization_level” parameter to `0x7FFFBFDF` in the PHP configuration (php.ini file).

Graphs

Daylight Saving Time

Changes to Daylight Saving Time (DST) result in irregularities when displaying X axis labels (date duplication, date missing, etc).

Sum aggregation

When using sum aggregation in a graph for period that is less than one hour, graphs display incorrect (multiplied) values when data come from trends.

Text overlapping

For some frontend languages (e.g., Japanese), local fonts can cause text overlapping in graph legend. To avoid this, use version 2.3.0 (or later) of PHP GD extension.

Log file monitoring

`log[]` and `logrt[]` items repeatedly reread log file from the beginning if file system is 100% full and the log file is being appended (see ZBX-10884 for more information).

Slow MySQL queries

Zabbix server generates slow `SELECT` queries in case of non-existing values for items. This issue is known to occur in MySQL 5.6/5.7 versions (for an extended discussion, see ZBX-10652), and, in specific cases, may also occur in later MySQL versions. A workaround to this is disabling the `index_condition_pushdown` or `prefer_ordering_index` optimizer in MySQL. Note, however, that this workaround may not fix all issues related to slow queries.

Slow configuration sync with Oracle

Configuration sync might be slow in Zabbix 6.0 installations with Oracle DB that have high number of items and item preprocessing steps. This is caused by the Oracle database engine speed processing *nclob* type fields.

To improve performance, you can convert the field types from *nclob* to *nvarchar2* by manually applying the database patch items_nvarchar_prepare.sql. Note that this conversion will reduce the maximum field size limit from 65535 bytes to 4000 bytes for item preprocessing parameters and item parameters such as *Description*, Script item's field *Script*, HTTP agent item's fields *Request body* and *Headers*, Database monitor item's field *SQL query*. Queries to determine template names that need to be deleted before applying the patch are provided in the patch as a comment. Alternatively, if MAX_STRING_SIZE is set you can change *nvarchar2(4000)* to *nvarchar2(32767)* in the patch queries to set the 32767 bytes field size limit.

For an extended discussion, see ZBX-22363.

API login

A large number of open user sessions can be created when using custom scripts with the `user.login` method without a following `user.logout`.

Persistent filter settings from links

When opening a link to Zabbix frontend page that contains filter settings, including the time selector, the filter is automatically saved in the database for the user, replacing the previously saved filter and/or time selector settings for that page. These settings remain active until the user manually updates or resets them.

IPv6 address issue in SNMPv3 traps

Due to a net-snmp bug, IPv6 address may not be correctly displayed when using SNMPv3 in SNMP traps. For more details and a possible workaround, see ZBX-14541.

Trimmed long IPv6 IP address in failed login information

A failed login attempt message will display only the first 39 characters of a stored IP address as that's the character limit in the database field. That means that IPv6 IP addresses longer than 39 characters will be shown incompletely.

Zabbix agent checks on Windows

Non-existing DNS entries in a `Server` parameter of Zabbix agent configuration file (zabbix_agentd.conf) may increase Zabbix agent response time on Windows. This happens because Windows DNS caching daemon doesn't cache negative responses for IPv4 addresses. However, for IPv6 addresses negative responses are cached, so a possible workaround to this is disabling IPv4 on the host.

YAML export/import

There are some known issues with YAML export/import:

- Error messages are not translatable;
- Valid JSON with a .yaml file extension sometimes cannot be imported;
- Unquoted human-readable dates are automatically converted to Unix timestamps.

Setup wizard on SUSE with NGINX and php-fpm

Frontend setup wizard cannot save configuration file on SUSE with NGINX + php-fpm. This is caused by a setting in /usr/lib/systemd/system/php-fpm.service unit, which prevents Zabbix from writing to /etc. (introduced in PHP 7.4).

There are two workaround options available:

- Set the ProtectSystem option to 'true' instead of 'full' in the php-fpm systemd unit.
- Manually save /etc/zabbix/web/zabbix.conf.php file.

Chromium for Zabbix web service on Ubuntu 20

Though in most cases, Zabbix web service can run with Chromium, on Ubuntu 20.04 using Chromium causes the following error:

```
Cannot fetch data: chrome failed to start:cmd_run.go:994:
WARNING: cannot create user data directory: cannot create
"/var/lib/zabbix/snap/chromium/1564": mkdir /var/lib/zabbix: permission denied
Sorry, home directories outside of /home are not currently supported. See https://forum.snapcraft.io/t/112
```

This error occurs because `/var/lib/zabbix` is used as a home directory of user 'zabbix'.

MySQL custom error codes

If Zabbix is used with MySQL installation on Azure, an unclear error message *[9002] Some errors occurred* may appear in Zabbix logs. This generic error text is sent to Zabbix server or proxy by the database. To get more information about the cause of the error, check Azure logs.

Invalid regular expressions after switching to PCRE2

In Zabbix 6.0 support for PCRE2 has been added. Even though PCRE is still supported, Zabbix installation packages for RHEL 7 and newer, SLES (all versions), Debian 9 and newer, Ubuntu 16.04 and newer have been updated to use PCRE2. While providing many benefits, switching to PCRE2 may cause certain existing PCRE regexp patterns becoming invalid or behaving differently. In particular, this affects the pattern *^[\w-\.]*. In order to make this regexp valid again without affecting semantics, change the expression to *^[-\w\.]* . This happens due to the fact that PCRE2 treats the dash sign as a delimiter, creating a range inside a character class.

Geomap widget error

The maps in the Geomap widget may not load correctly, if you have upgraded from an older Zabbix version with NGINX and didn't switch to the new NGINX configuration file during the upgrade.

To fix the issue, you can discard the old configuration file, use the configuration file from the current version package and reconfigure it as described in the download instructions in section *e. Configure PHP for Zabbix frontend*.

Alternatively, you can manually edit an existing NGINX configuration file (typically, */etc/zabbix/nginx.conf*). To do so, open the file and locate the following block:

```
location ~ /(api\/|conf[^\.]|include|locale|vendor) {
        deny            all;
        return          404;
}
```

Then, replace this block with:

```
location ~ /(api\/|conf[^\.]|include|locale) {
        deny            all;
        return          404;
}

location /vendor {
        deny            all;
        return          404;
}
```

Logrotate for Zabbix server and proxy

In Zabbix versions 6.4.3 and older, logrotate is only included into packages for zabbix-agent, zabbix-agent2 and zabbix-web-service, but needs to be installed separately for Zabbix server and proxy. The logrotate dependency has been added to the server and proxy packages for RHEL and SUSE starting from Zabbix 6.4.4rc1.

Missing files in Windows agent archive

The Windows Zabbix agent download ZIP file is missing zabbix_sender.h and zabbix_sender.lib files in versions 6.4.0-6.4.12, required for zabbix_sender.dll.

Server/proxy compatibility issue in 6.4.12

Zabbix server 6.4.12 and Zabbix proxy 6.4.12 are **not compatible** with other versions of proxy/server. If either server or proxy is 6.4.12, then both server and proxy must be 6.4.12.

This issue is fixed in 6.4.13 and later. However, while the following releases are compatible with 6.4.11 server/proxy (or earlier), they are still not compatible with 6.4.12 server/proxy.

Use case with global variables shared across webhook calls

As global variables are shared across different webhook calls, the following code will result in the tag value counter gradually increasing:

```
try
{
   aa = aa + 1;
}
catch(e)
{
   aa = 0;
}

result = {
        'tags': {
            'endpoint': aa
        }
    };
return JSON.stringify(result);
```

Using local variables instead of global ones is recommended to make sure that each script operates on its own data and that there are no collisions between simultaneous calls.

Limits of filtering with utf8mb4 collations

Filters (e.g., in *Data collection* → *Maintenance*) may not function correctly when applied to entities containing certain Unicode characters (e.g., ℰ, ∉). This issue arises due to how the default utf8mb4_bin collation for MySQL or MariaDB databases handles sorting and comparison of Unicode characters.

To address this limitation, users can change the collation of database columns to alternatives such as utf8mb4_0900_bin, utf8mb4_0900_ai_ci, or utf8mb4_unicode_520_ci. Note, however, that changing the collation may cause unexpected behavior in the handling of empty spaces, as well as sorting and filtering for other characters.

For more information on changing collations, see MySQL documentation or MariaDB documentation. For details on collation differences, see Unicode Character Sets in MySQL documentation.

1 Compilation issues

These are the known issues regarding Zabbix compilation from sources. For all other cases, see the Known issues page.

Compiling Zabbix agent on HP-UX

If you install the PCRE library from the popular HP-UX package site http://hpux.connect.org.uk (for example, from file `pcre-8.42-ia64_64-11.31.depot`), only the 64-bit version of the library will be installed in the `/usr/local/lib/hpux64` directory.

In this case, for successful agent compilation, a customized option is needed for the `configure` script, for example:

```
CFLAGS="+DD64" ./configure --enable-agent --with-libpcre-include=/usr/local/include --with-libpcre-lib=/us
```

Library in a non-standard location

Zabbix allows you to specify a library located in a non-standard location. In the example below, Zabbix will run `curl-config` from the specified non-standard location and use its output to determine the correct libcurl to use.

```
$ ./configure --enable-server --with-mysql --with-libcurl=/usr/local/bin/curl-config
```

This will work if it is the only libcurl installed in the system, but might not if there is another libcurl installed in a standard location (by the package manager, for example). Such is the case when you need a newer version of the library for Zabbix and the older one for other applications.

Therefore, specifying a component in a non-standard location will not always work when the same component also exists in a standard location.

For example, if you use a newer libcurl installed in `/usr/local` with the libcurl package still installed, Zabbix might pick up the wrong one and compilation will fail:

```
usr/bin/ld: ../../src/libs/zbxhttp/libzbxhttp.a(http.o): in function 'zbx_http_convert_to_utf8':
/tmp/zabbix-master/src/libs/zbxhttp/http.c:957: undefined reference to 'curl_easy_header'
collect2: error: ld returned 1 exit status
```

Here, the function `curl_easy_header()` is not available in the older `/usr/lib/x86_64-linux-gnu/libcurl.so`, but is available in the newer `/usr/local/lib/libcurl.so`.

The problem lies with the order of linker flags, and one solution is to specify the full path to the library in an LDFLAGS variable:

```
$ LDFLAGS="-Wl,--no-as-needed /usr/local/lib/libcurl.so" ./configure --enable-server --with-mysql --with-l
```

Note the `-Wl,--no-as-needed` option which might be needed on some systems (see also: default linking options on Debian-based systems).

9 Template changes

This page lists all changes to the stock templates that are shipped with Zabbix.

Note that upgrading to the latest Zabbix version will not automatically upgrade the templates used. It is suggested to modify the templates in existing installations by:

- Downloading the latest templates from the Zabbix Git repository;
- Then, while in *Data collection* → *Templates* you can import them manually into Zabbix. If templates with the same names already exist, the *Delete missing* option should be checked when importing to achieve a clean import. This way the old items that are no longer in the updated template will be removed (note that it will mean losing history of these old items).

Note:
Please be informed that since Zabbix 6.0, all templates follow an updated format, which may impact the import of pre-6.0 templates. For more information, see Template changes in 6.0.

CHANGES IN 6.4.0

Updated templates

Templates that discover filesystems

Zabbix templates that discover filesystems now make use of the `vfs.fs.get` item instead of `vfs.fs.discovery` item, and:

- The update interval for the discovery has been reduced to one minute.
- Item prototypes are now dependent on the `vfs.fs.get` master item.
- A trigger to detect volumes that remounted in read-only mode has been added to UNIX templates.
- Filters have been added to filter bind mounts on Linux and .dmg volumes on macOS in the respective templates.

Linux by Zabbix agent

The `system.sw.packages` item has been replaced by `system.sw.packages.get` in the generic *Linux by Zabbix agent* template. A new trigger has been added to fire when the number of packages changes.

Windows by Zabbix agent/Zabbix agent active

The templates *Windows by Zabbix agent* and *Windows by Zabbix agent active* have been updated and now include the `system.sw.os` item and a new trigger for monitoring system version changes.

CHANGES IN 6.4.3

A new template has been added: Google Cloud Platform (GCP) by HTTP.

The template *Azure by HTTP* now also works with Azure Cosmos DB for MongoDB.

CHANGES IN 6.4.5

New templates are available:

- AWS ECS Cluster by HTTP (along with its Serverless Cluster version)
- Cisco SD-WAN by HTTP
- OpenStack by HTTP, which includes *OpenStack Nova by HTTP* template for monitoring OpenStack Nova service
- PostgreSQL by ODBC

CHANGES IN 6.4.6

New template is available:

- AWS Cost Explorer by HTTP

CHANGES IN 6.4.7

New templates are available:

- Acronis Cyber Protect Cloud by HTTP
- HashiCorp Nomad by HTTP
- MantisBT by HTTP

CHANGES IN 6.4.8

New templates

New templates are available:

- FortiGate by HTTP
- FortiGate by SNMP
- Nextcloud by HTTP

Updated templates

- PostgreSQL by ODBC and PostgreSQL by Zabbix agent 2 templates now include the item and trigger for monitoring PostgreSQL version.
- Cisco Meraki organization by HTTP template has been supplemented with items, item prototypes, LLD rules, and macros related to authentication, licenses, networks, SAML roles, and VPN statuses.

CHANGES IN 6.4.9

New templates

New template is available:

- HPE iLO by HTTP

Updated templates

Integration with OpenShift has been added to Kubernetes cluster state by HTTP template.

CHANGES IN 6.4.11

New templates

The set of Azure by HTTP templates has been supplemented with the Azure Cost Management by HTTP template.

Updated templates

MSSQL by ODBC template has been updated for working with AlwaysOn features such as Failover Cluster Instances (FCI) and Availability Groups (AG). It is now possible to use the template for monitoring a host in cluster, standalone host and host by cluster name. A macro for instance name is no longer used; when the master is switched, it is not required to change any macros:

- new LLD rules and metrics for quorum and quorum members have been added;
- the type of the LLD rules has been changed from “Database monitor” to “Dependent item”;
- items with `db.odbc.discovery` key have been turned into items dependent on the `db.odbc.get` item
- new item has been added - MSSQL DB ’{#DBNAME}’: Recovery model, which returns the database recovery model under the database discovery;
- new macros, namely, {$MSSQL.BACKUP_FULL.USED}, {$MSSQL.BACKUP_DIFF.USED}, {$MSSQL.BACKUP_LOG.USED}, have been added - those can be used for disabling backup age triggers for a certain database.

CHANGES IN 6.4.12

New templates

A new template is available:

- YugabyteDB by HTTP, which includes the *YugabyteDB Cluster by HTTP* template for monitoring each YugabyteDB cluster.

CHANGES IN 6.4.13

New templates

New templates are available:

- AWS ELB Application Load Balancer by HTTP
- Check Point Next Generation Firewall by SNMP
- MSSQL by Zabbix agent 2

CHANGES IN 6.4.14

New templates

A new template is available:

- Oracle Cloud by HTTP, a master template that discovers various Oracle Cloud Infrastructure (OCI) services and resources.

Updated templates

- FortiGate by SNMP template has been supplemented with metrics regarding VPN, high availability (HA), wireless termination points (WTPs), SD-WAN health checks, and HW sensors.
- MySQL by ODBC template has been supplemented with the items "MySQL: Get database" and "MySQL: Get replication". The LLD rules "Database discovery" and "Replication discovery" have been changed to the "Dependent item" type.
- Oracle by ODBC template has been supplemented with the items "Oracle: Get archive log", "Oracle: Get ASM disk groups", "Oracle: Get database", "Oracle: Get PDB", and "Oracle: Get tablespace". The LLD rules "Archive log discovery", "ASM disk groups discovery", "Database discovery", "PDB discovery", and "Tablespace discovery" have been changed to the "Dependent item" type.
- The VMware Hypervisor template within the VMware and VMware FQDN template sets has been supplemented with a new LLD rule, "Sensor discovery".

CHANGES IN 6.4.15

New templates

The AWS ELB template set has been supplemented with the template AWS ELB Network Load Balancer by HTTP.

Updated templates

The OS templates (agent, SNMP, and Prometheus-based) have been given a mounted filesystem overhaul. Changes include item, trigger, graph, and dashboard updates.

CHANGES IN 6.4.16

New templates

A new template is available:

- Jira Data Center by JMX, a template for monitoring Jira Data Center health.

CHANGES IN 6.4.17

New templates

The set of Azure by HTTP templates has been supplemented with the Azure VM Scale Set by HTTP template.

10 Upgrade notes for 6.4.0

These notes are for upgrading from Zabbix 6.2.x to Zabbix 6.4.0.

All notes are grouped into:

- **Breaking changes** - changes that may break existing installations and other critical information related to the upgrade process
- **Other** - all remaining information describing the changes in Zabbix functionality

See also:

- Upgrade procedure for all relevant information about upgrading from versions before Zabbix 6.2.0;
- Upgrading HA cluster for instructions on upgrading servers in a **high-availability** (HA) cluster.

Upgrade process

To complete successful Zabbix server upgrade on MySQL/MariaDB, you may require to set `GLOBAL log_bin_trust_function_creators = 1` in MySQL if binary logging is enabled, there are no superuser privileges and `log_bin_trust_function_creators = 1` is not set in MySQL configuration file.

To set the variable using the MySQL console, run:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
```

Once the upgrade has been successfully completed, this option can be disabled:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 0;
```

Breaking changes MySQL version

MySQL versions below 8.0.30 are no longer supported. The minimum required MySQL version has been raised from 8.0.0 to 8.0.30.

libssh version

The minimum required libssh version has been raised from 0.6.0 to 0.9.0.

SLES versions

For SUSE Linux Enterprise Server (SLES) 15 the minimum supported service pack version is now SP4.

SLES 12 with SP4 and newer are also still supported, but due to the older libssh library, support for the SSH checks for this OS version has been discontinued.

Optimized proxy configuration update

Incremental update of the proxy configuration has been introduced in the new version.

As a result the **ConfigFrequency** parameter for active proxies is now deprecated; instead the ProxyConfigFrequency parameter must be used on both server (for passive proxies) and proxy (for active proxies). Note that if both ProxyConfigFrequency and ConfigFrequency are used, the proxy will log an error and terminate.

The default value of ProxyConfigFrequency has been lowered from 3600 to **10** seconds. Also, the default value of CacheUpdateFrequency on the server has been lowered from 60 to **10** seconds.

The server now uses the configuration cache to track proxy configuration changes using revision numbers, so the amount of cached data will increase and it might be necessary to increase the configuration cache size.

Note that the server configuration must be synced before any changes are synced to proxy. So if a user makes some changes and wants them to be synced to the proxy - the server configuration cache must be reloaded first.

Other Automated database upgrade on proxies with SQLite

Upon the first launch after an upgrade, Zabbix proxy with SQLite3 now automatically dumps the existing older version of the database file and creates a new one for the current version. History data that has been stored in the SQLite database file is not preserved. Previously, the SQLite database file had to be recreated manually.

If Zabbix proxy version is older than the database file version, the proxy will log an error and terminate.

CSRF tokens

For enhanced security against CSRF (Cross Site Request Forgery) attacks, Zabbix frontend now uses randomly generated CSRF tokens instead of static session-based tokens. If your installation uses an IDS/IPS that checks for specific token names, it may be necessary to update its configuration to recognize the new token name, `_csrf_token`, in order to avoid false positive detections.

Thread-based preprocessing workers

As the item value preprocessing has been rewritten to use thread-based preprocessing workers, a new library is now required for Zabbix server/proxy - `libevent_pthreads`.

Non-mandatory fields in user configuration

The user group and user role fields are no longer mandatory in user configuration. However, if an internal Zabbix user is created without a user role, the user will not be able to log into Zabbix, using internal authentication.

Instant refresh of active checks

Previously, Zabbix agent (in active mode) received from Zabbix server or Zabbix proxy a full copy of the configuration once every two minutes (default). By introducing incremental configuration sync, full configuration is no longer sent when there are no changes to host or global regular expressions, thus the default sync interval has been reduced to 5 seconds.

'RefreshActiveChecks' parameter supported in Zabbix agent configuration file default value is changed to 5 seconds (previously 120). This can make connections more frequent and increase network usage especially if encryption is used.

JSON protocol for active agent checks has been updated to include `config_revision` and `session` ID. For more information, see Passive and active agent checks.

User macros also affect the incremental configuration sync. It is advisable to use host macros instead of global macros because adding, updating or deleting global macros forces incremental configuration update for all hosts.

Heartbeat support is deprecated

Starting from Zabbix 6.4 *heartbeat sender* is removed from the proxy, and therefore, Zabbix proxy item *zabbix [process,heartbeat sender]* is not supported and is removed from templates. *HeartbeatFrequency* parameter has been deprecated.

Autoregistration heartbeat for Zabbix server

The active agent autoregistration heartbeat for Zabbix server has been set to 120 seconds (the same as for Zabbix proxy). So in case a discovered host is deleted, the autoregistration will be rerun in 120 seconds.

Old numeric (float) value type is deprecated

Since Zabbix 5.0, numeric (float) data type has been switched to the new format to support precision of approximately 15 digits and extended value range from approximately -1.79E+308 to 1.79E+308. Now the old numeric type, which has been disabled, but remained supported, is deprecated and will be removed in an upcoming version.

If your installation is not upgraded to the numeric values of extended range, System information in the frontend will display a warning: "Database history tables upgraded: No. Support for the old numeric type is deprecated. Please upgrade to numeric values of extended range". In this case, see instructions for enabling the extended range of numeric values.

Date removed from export

The `date` element has been removed from the export format when exporting objects (hosts, templates, etc.).

Template versioning

To improve management and upgrade of templates, template versioning has been introduced.

In *Data collection → Templates* you can now see the template vendor and version, as well as filter templates by vendor and/or version.

Importing updated templates

Please note that templates must be upgraded manually if you are upgrading from previous versions. This can be done by importing the updated templates. You can find the updated templates in the *zabbix/templates* directory of the downloaded latest Zabbix version. While in *Data collection → Templates* you can import them from this directory.

It is also possible to download templates from Zabbix git repository directly.

Updating custom template vendor and version

For existing custom templates, template vendor and version can also be modified through the Template API.

11 Upgrade notes for 6.4.1

This minor version doesn't have any upgrade notes.

12 Upgrade notes for 6.4.2

HTML support in Geomap attribution dropped The attribution text for the Geomap dashboard widget can now only contain plain text; HTML support has been dropped. If this field already contains HTML, it will be rendered as plain text after the upgrade.

In Geographical maps settings in the Administration → General section, the field *Attribution* is now only visible when *Tile provider* is set to *Other*.

13 Upgrade notes for 6.4.3

Proxy history housekeeping The limitation on the amount of outdated information deleted from the proxy database per proxy history housekeeping cycle has been removed.

Previously the housekeeper deleted only no more than 4 times the `HousekeepingFrequency` hours of outdated information. For example, if `HousekeepingFrequency` was set to "1", no more than 4 hours of outdated information (starting from the oldest entry) was deleted. In cases when a proxy would constantly receive data older than set in `ProxyOfflineBuffer`, this could result in excessive data accumulation.

Now this limitation has been removed, providing a more effective proxy history housekeeping solution.

14 Upgrade notes for 6.4.4

User creation A new user cannot be created without assigning a user role to them anymore (user role setting can be found under *Permissions* tab). When trying to do so, you will face an error stating: *"Cannot add user: field "roleid" is mandatory"*.

Aggregate functions The **count_foreach** function now returns '0' for a matching item in the array, if no data are present for the item or the data do not match the filter. Previously such items would be ignored (no data added to the aggregation).

15 Upgrade notes for 6.4.5

This minor version doesn't have any upgrade notes.

16 Upgrade notes for 6.4.6

Maximum JSON depth

The maximum allowed JSON depth has been set to 64 to improve security and performance during JSON parsing.

Macro functions

The range of the `fmtnum` macro function is now limited to 0-20.

17 Upgrade notes for 6.4.7

Autoregistration table cleared from orphaned records

Previously the `autoreg_host` table was never cleared. Now orphaned records that reference neither an autoregistration event nor an existing host will be removed by the Housekeeper.

PostgreSQL plugin parameters

The following PostgreSQL plugin parameters are no longer mandatory if Plugins.PostgreSQL.Sessions.<SessionName>.TLSConnect is set to *verify_ca* or *verify_full*:

- Plugins.PostgreSQL.Sessions.<SessionName>.TLSCertFile
- Plugins.PostgreSQL.Sessions.<SessionName>.TLSKeyFile

18 Upgrade notes for 6.4.8

Breaking changes Agent 2 plugins

MySQL

The correct master host is now returned if specified in the `mysql.replication.get_slave_status[]` item "masterHost" parameter. Previously the "masterHost" parameter was ignored and always the first master host was returned.

If this parameter is not specified, all hosts are returned.

MySQL plugin parameters

The following MySQL plugin parameters are no longer mandatory if Plugins.Mysql.Sessions.<SessionName>.TLSConnect is set to *verify_ca* or *verify_full*:

- Plugins.Mysql.Sessions.<SessionName>.TLSCertFile
- Plugins.Mysql.Sessions.<SessionName>.TLSKeyFile

PostgreSQL-TimescaleDB combination no longer checked

Zabbix no longer checks for the supported PostgreSQL-TimescaleDB combination. As before, Zabbix does check for the supported PostgreSQL version or TimescaleDB version separately.

19 Upgrade notes for 6.4.9

MongoDB plugin parameters

The following MongoDB plugin parameters are no longer mandatory if Plugins.MongoDB.Sessions.<SessionName>.TLSConnect is set to `verify_ca` or `verify_full`:

- Plugins.MongoDB.Sessions.<SessionName>.TLSCertFile
- Plugins.MongoDB.Sessions.<SessionName>.TLSKeyFile

See also: MongoDB plugin parameters

20 Upgrade notes for 6.4.10

This minor version doesn't have any upgrade notes.

21 Upgrade notes for 6.4.11

Package size handling for system.sw.packages.get item

When monitoring the `system.sw.packages.get` item using Zabbix agent 2, if the returned package data lacks information about the package size, the size value is now automatically set to 0.

Consistency introduced in sha256 checksums of vfs.file.cksum item

In earlier Zabbix versions, the Zabbix agent item `vfs.file.cksum` produced different sha256 sums for the same file depending on the platform (processor architecture) – sha256 sums on AIX, HP-UX (Itanium) and Solaris (SPARC) differed from those produced on Intel x86-64 platforms.

The issue has now been fixed; however, after upgrading, monitoring sha256 sums of files on AIX, HP-UX, or Solaris SPARC may result in false positives of files having been modified.

22 Upgrade notes for 6.4.12

Breaking changes Compatibility issue between server/proxy versions

Zabbix server 6.4.12 and Zabbix proxy 6.4.12 are **not compatible** with other versions of proxy/server. If either server or proxy is 6.4.12, then both server and proxy must be 6.4.12.

This issue will be fixed in the next release. However, while the next release will be compatible with 6.4.11 server/proxy (or earlier), it will still not be compatible with 6.4.12 server/proxy.

Zabbix agent 2 support on Windows

To prevent critical security vulnerabilities, the minimum Windows version for Zabbix agent 2 has been raised to Windows 10/Server 2016. See note under Supported platforms for more information.

23 Upgrade notes for 6.4.13

Guest user authorization

Automatic login for the guest user has been removed. After this change, the guest user will need to log in like any other user. Previously, a guest could immediately get to almost any monitoring or reporting page without going through authorization.

Invalid regular expression in proc.* items

proc.* agent items will now become 'not supported' if an invalid regular expression is supplied.

24 Upgrade notes for 6.4.14

This minor version doesn't have any upgrade notes.

25 Upgrade notes for 6.4.15

This minor version doesn't have any upgrade notes.

26 Upgrade notes for 6.4.16

This minor version doesn't have any upgrade notes.

27 Upgrade notes for 6.4.17

New index on auditlog table

A new index has been added to the `auditlog` table to improve database and frontend response times when filtering records by *Recordset ID* in the Audit log.

Note that users with large audit logs may experience extended upgrade times due to the database size.

Server to stop with read-only database

A standalone Zabbix server will now stop if the database becomes read-only.

28 Upgrade notes for 6.4.18

This minor version doesn't have any upgrade notes.

29 Upgrade notes for 6.4.19

Databases Database changes

A slow template cloning issue has been resolved by adding indexes for the uuid field. This change may cause long upgrade times on large datasets.

30 Upgrade notes for 6.4.20

This minor version doesn't have any upgrade notes.

31 Upgrade notes for 6.4.21

Breaking changes Java 11 required for Java gateway

Zabbix Java gateway now requires Java 11 for runtime (building from source is still possible with Java 8), due to updated minimum logback library versions:

Library	New minimum version	Old minimum version
logback-classic	**1.5.16**	1.2.9
logback-core	**1.5.16**	1.2.9
slf4j-api	**2.0.16**	1.7.32

5 Quickstart

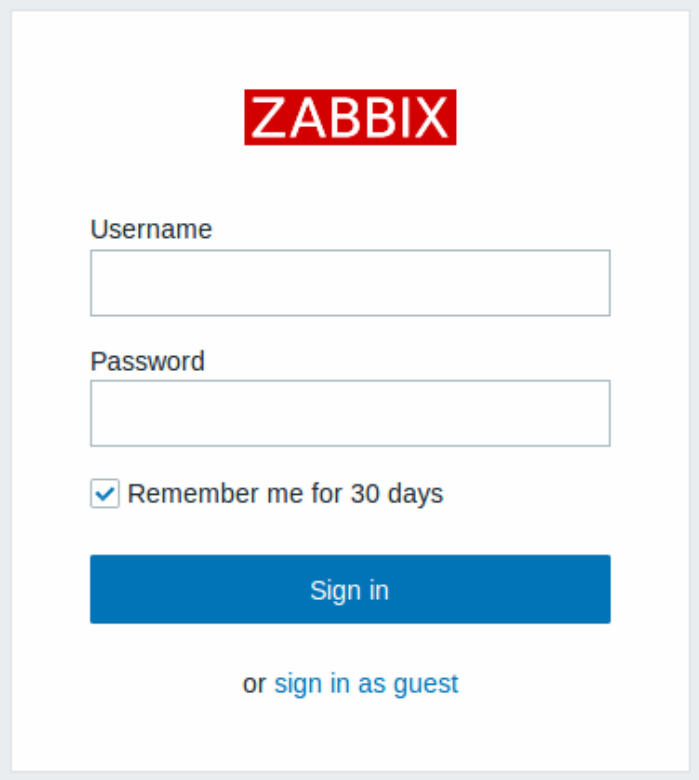
Please use the sidebar to access content in the Quickstart section.

1 Login and configuring user

Overview

In this section, you will learn how to log in and set up a system user in Zabbix.

Login



This is the Zabbix welcome screen. Enter the user name **Admin** with password **zabbix** to log in as a Zabbix superuser. Access to all menu sections will be granted.

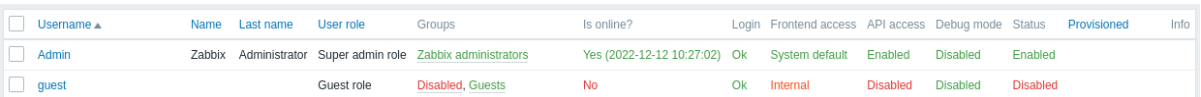
Protection against brute force attacks

In case of five consecutive failed login attempts, Zabbix interface will pause for 30 seconds in order to prevent brute force and dictionary attacks.

The IP address of a failed login attempt will be displayed after a successful login.

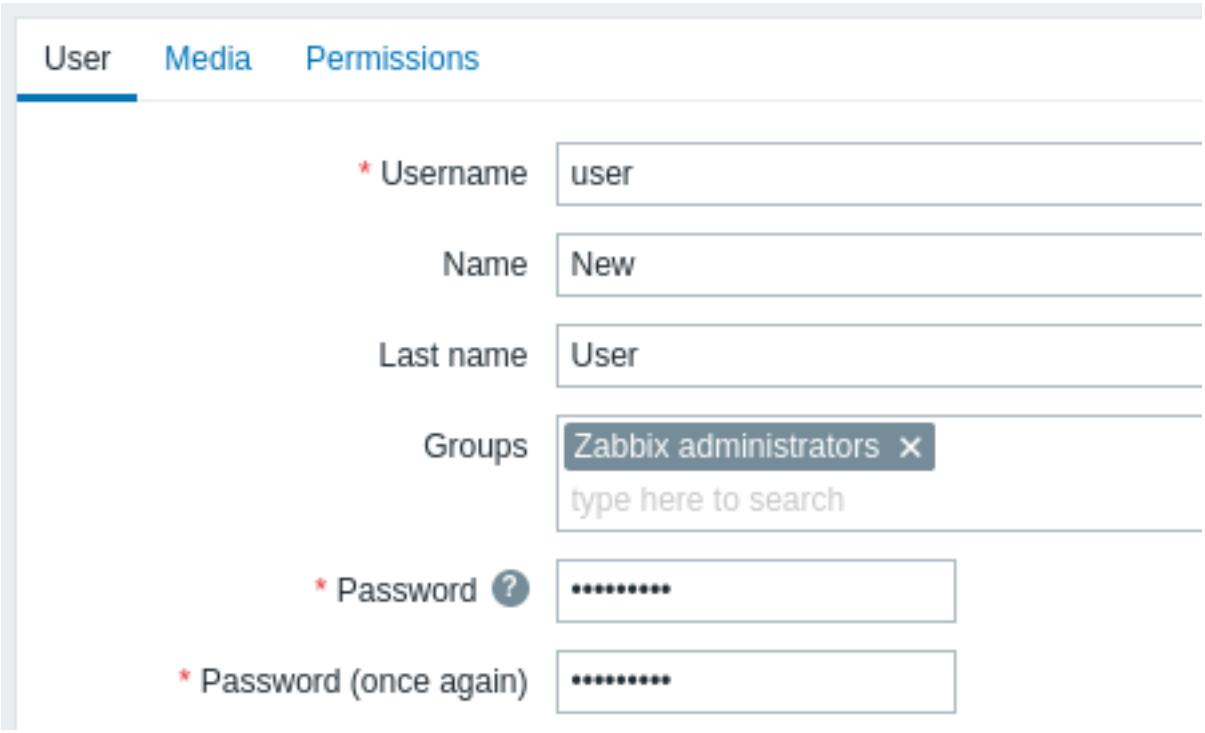
Adding user

To view information about users, go to *Users* → *Users*.



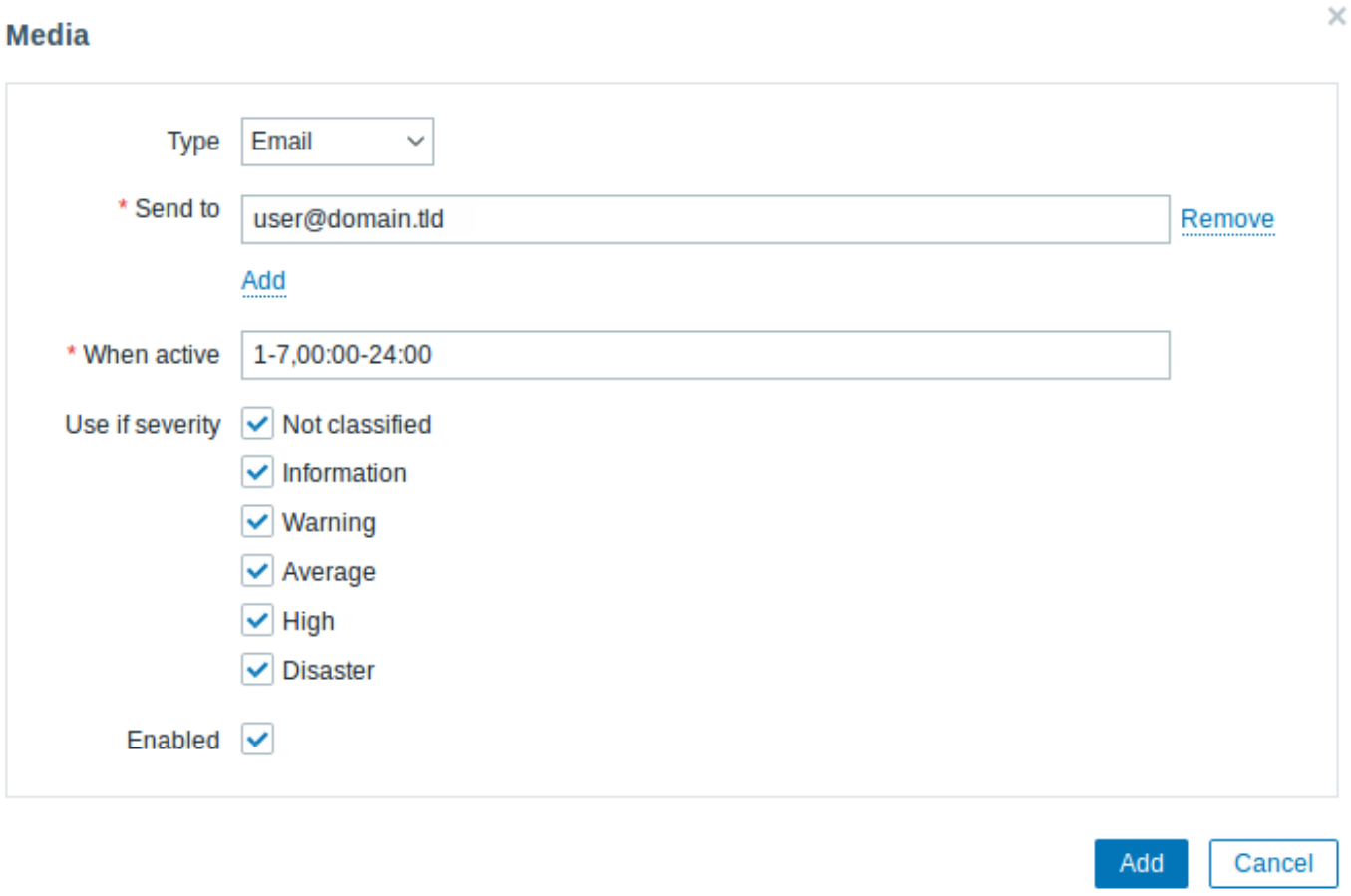
To add a new user, click on *Create user*.

In the new user form, make sure to add your user to one of the existing user groups, for example 'Zabbix administrators'.



All mandatory input fields are marked with a red asterisk.

By default, new users have no media (notification delivery methods) defined for them. To create one, go to the 'Media' tab and click on *Add*.

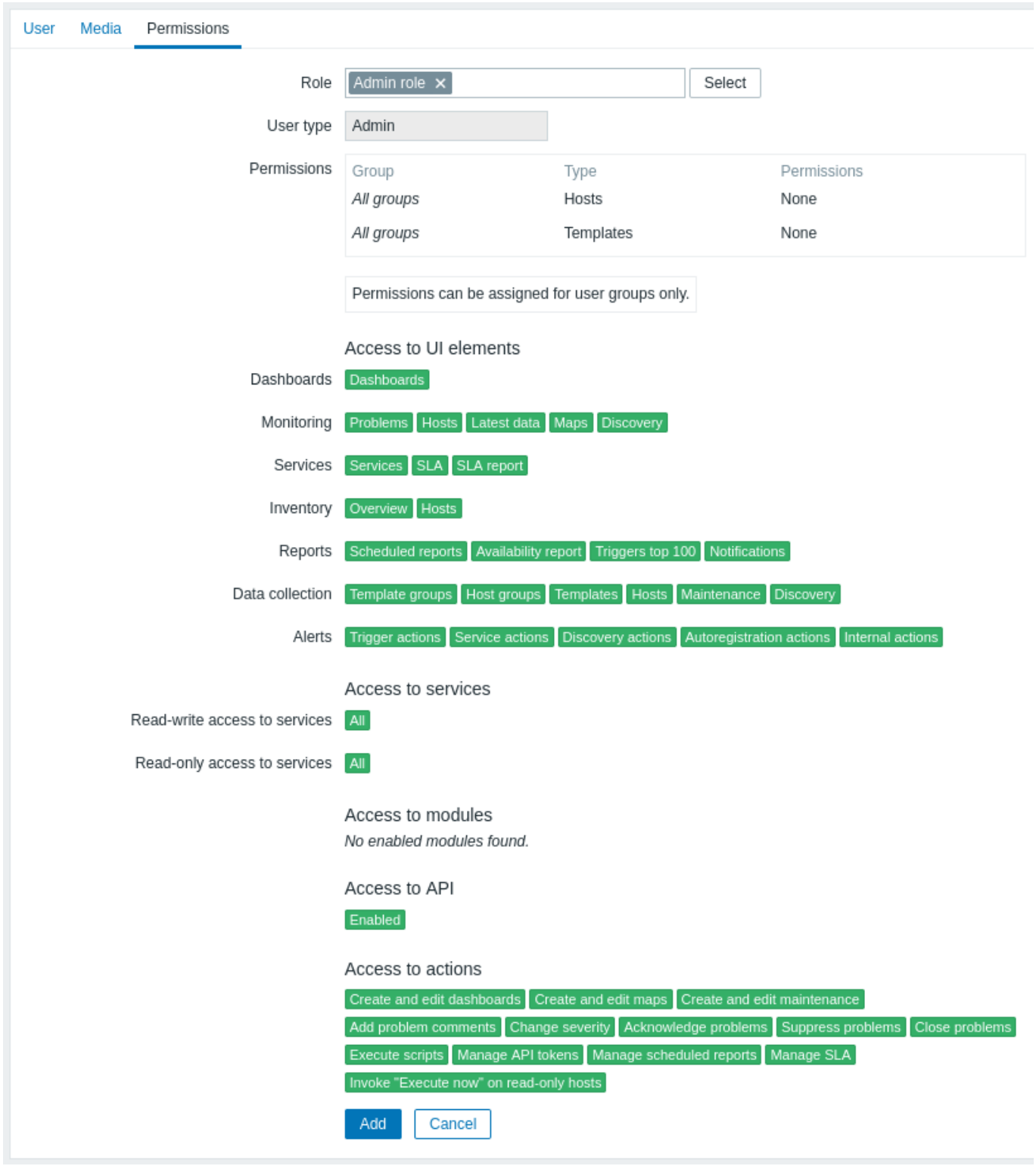


In this pop-up, enter an email address for the user.

You can specify a time period when the medium will be active (see Time period specification page for a description of the format), by default a medium is always active. You can also customize trigger severity levels for which the medium will be active, but leave all of them enabled for now.

Click on *Add* to save the medium, then go to the Permissions tab.

Permissions tab has a mandatory field *Role*. The role determines which frontend elements the user can view and which actions he is allowed to perform. Press Select and select one of the roles from the list. For example, select *Admin role* to allow access to all Zabbix frontend sections, except Administration. Later on, you can modify permissions or create more user roles. Upon selecting a role, permissions will appear in the same tab:

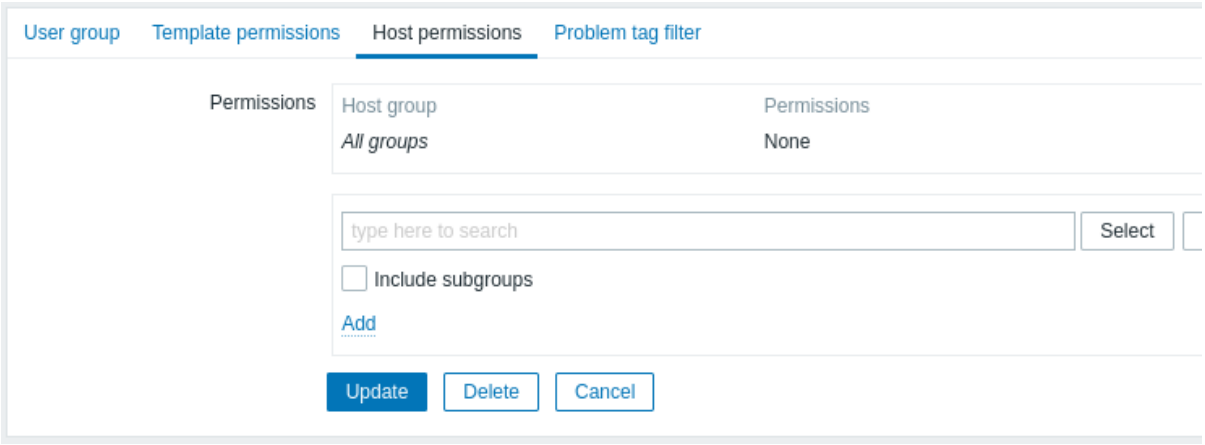


Click *Add* in the user properties form to save the user. The new user appears in the userlist.

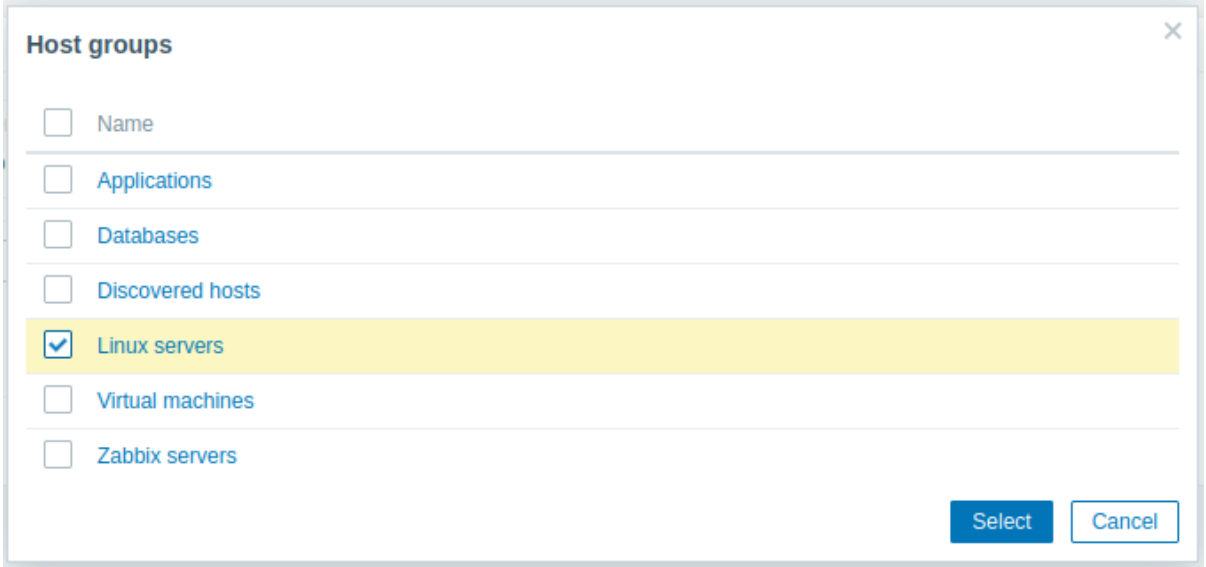


Adding permissions

By default, a new user has no permissions to access hosts and templates. To grant the user rights, click on the group of the user in the *Groups* column (in this case - 'Zabbix administrators'). In the group properties form, go to the *Host permissions* tab to assign permissions to host groups.



This user is to have read-only access to *Linux servers* group, so click on *Select* next to the host group selection field.



In this pop-up, mark the checkbox next to 'Linux servers', then click *Select*. *Linux servers* should be displayed in the selection field. Click the 'Read' button to set the permission level and then *Add* to add the group to the list of permissions. In the user group properties form, click *Update*.

To grant permissions to templates, you will need to switch to the *Template permissions* tab and specify template groups.

Attention:
In Zabbix, access rights to hosts and templates are assigned to user groups, not individual users.

Done! You may try to log in using the credentials of the new user.

2 New host

Overview

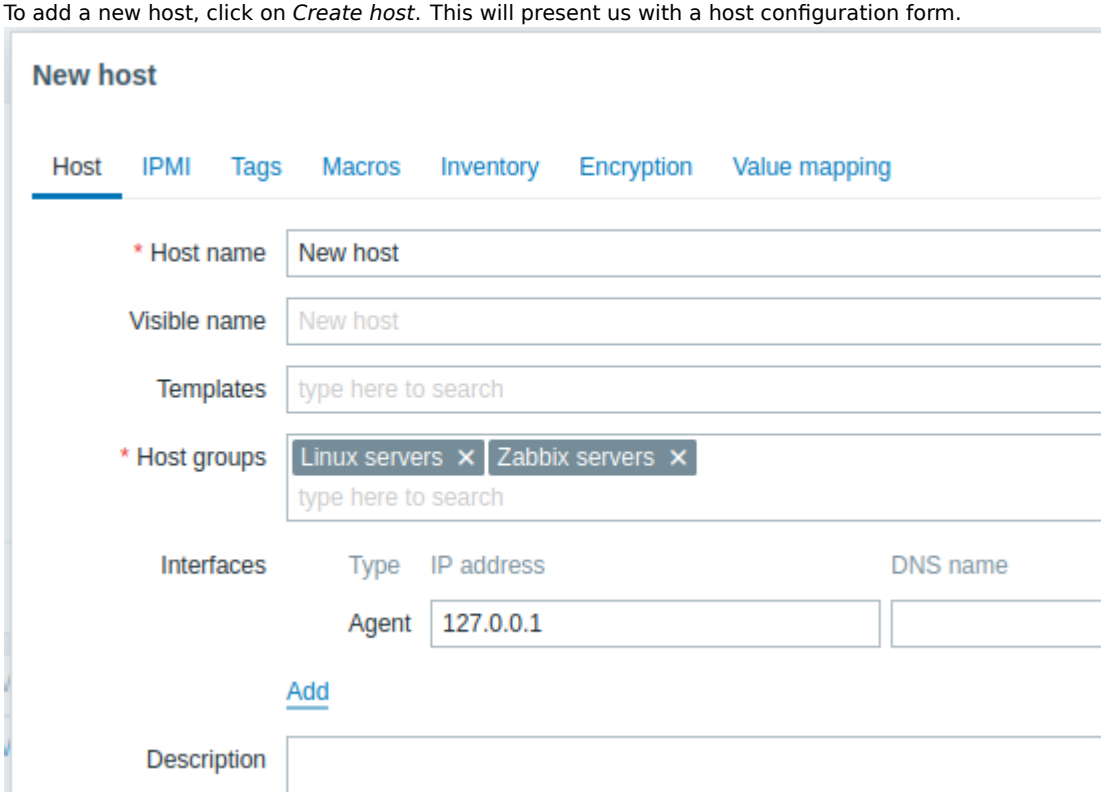
In this section you will learn how to set up a new host.

A host in Zabbix is a networked entity (physical, virtual) that you wish to monitor. The definition of what can be a "host" in Zabbix is quite flexible. It can be a physical server, a network switch, a virtual machine or some application.

Adding host

Information about configured hosts in Zabbix is available in *Data collection → Hosts* as well as *Monitoring → Hosts*. There is already one pre-defined host, called "Zabbix server", but we want to learn adding another.

To add a new host, click on *Create host*. This will present us with a host configuration form.



All mandatory input fields are marked with a red asterisk.

The bare minimum to enter here is:

Host name

- Enter a host name. Alphanumerics, spaces, dots, dashes and underscores are allowed.

Host groups

- Select one or several existing groups by clicking *Select* button or enter a non-existing group name to create a new group.

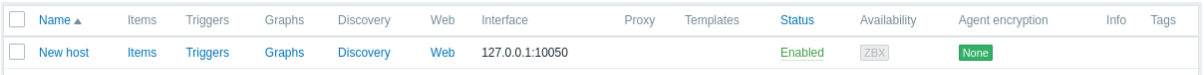
Note:
All access permissions are assigned to host groups, not individual hosts. That is why a host must belong to at least one group.

Interfaces: IP address

- Although not a required field technically, a host interface is necessary for collecting certain metrics. To use Zabbix agent passive checks, specify the agent's IP or DNS in this field. Note that you should also specify Zabbix server's IP or DNS in the Zabbix agent configuration file 'Server' directive. If Zabbix agent and Zabbix server are installed on the same machine, you need to specify the same IP/DNS in both places.

Other options will suit us with their defaults for now.

When done, click *Add*. Your new host should be visible in the host list.



The Availability column contains indicators of host availability per each interface. We have defined a Zabbix agent interface, so we can use the agent availability icon (with 'ZBX' on it) to understand host availability:

-  - host status has not been established; no metric check has happened yet
-  - host is available, a metric check has been successful
-  - host is unavailable, a metric check has failed (move your mouse cursor over the icon to see the error message). There might be some error with communication, possibly caused by incorrect interface credentials. Check that Zabbix server is running, and try refreshing the page later as well.

3 New item

Overview

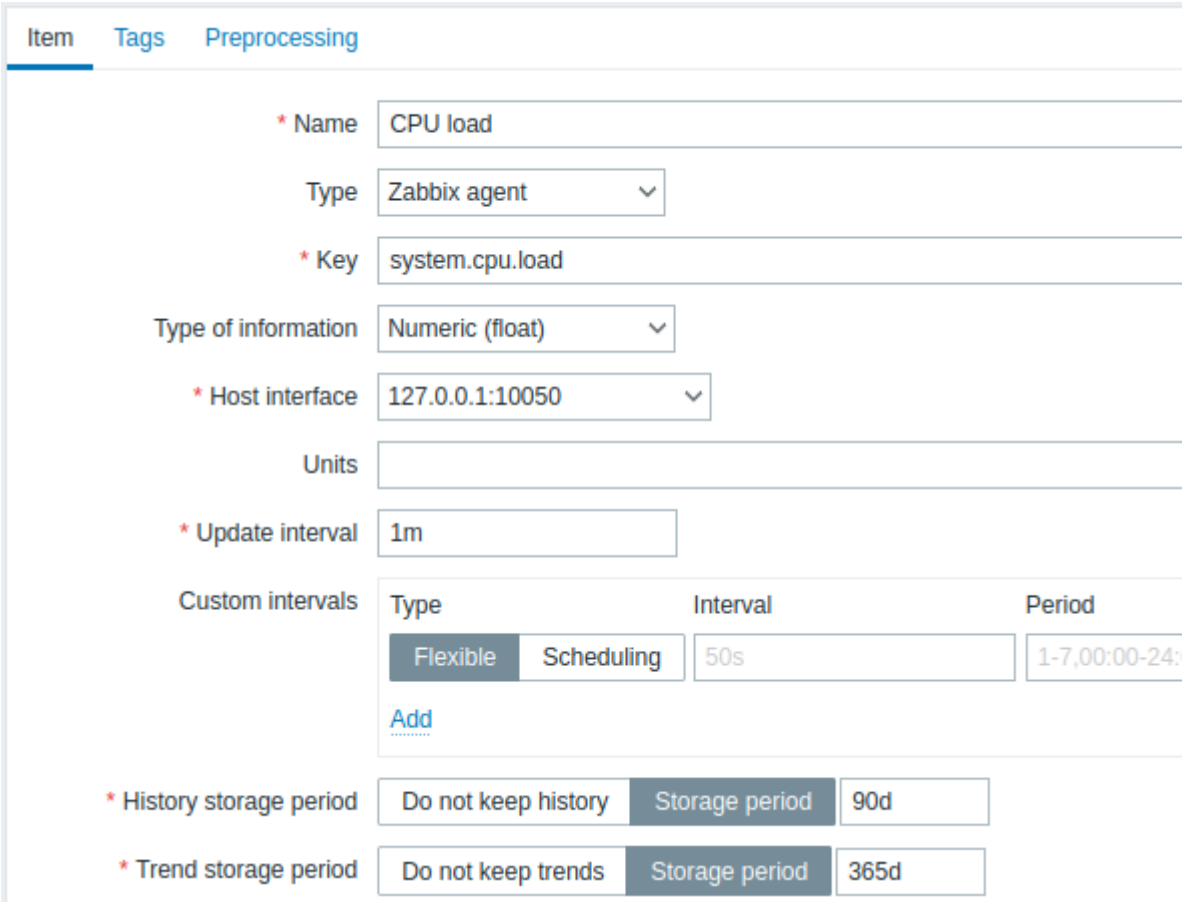
In this section, you will learn how to set up an item.

Items are the basis of gathering data in Zabbix. Without items, there is no data - because only an item defines a single metric or what kind of data to collect from a host.

Adding item

All items are grouped around hosts. That is why to configure a sample item we go to *Data collection* → *Hosts* and find the "New host" we have created.

Click on the *Items* link in the row of "New host", and then click on *Create item*. This will present us with an item definition form.



All mandatory input fields are marked with a red asterisk.

For our sample item, the essential information to enter is:

Name

- Enter *CPU load* as the value. This will be the item name displayed in lists and elsewhere.

Key

- Manually enter *system.cpu.load* as the value. This is the technical name of an item that identifies the type of information that will be gathered. The particular key is just one of pre-defined keys that come with Zabbix agent.

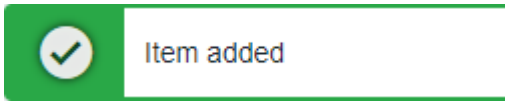
Type of information

- This attribute defines the format of the expected data. For the *system.cpu.load* key, this field will be automatically set to *Numeric (float)*.

Note:
You may also want to reduce the number of days item history will be kept, to 7 or 14. This is good practice to relieve the database from keeping lots of historical values.

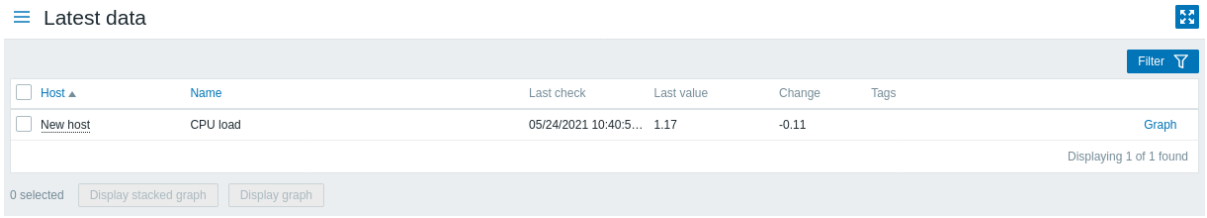
Other options will suit us with their defaults for now.

When done, click *Add*. The new item should appear in the item list, and you should see a success message.



Viewing data

With an item defined, you might be curious if it is actually gathering data. For that, go to *Monitoring* → *Latest data*, select 'New host' in the filter and click on *Apply*.



With that said, it may take up to 60 seconds for the first data to arrive. That, by default, is how often the server reads configuration changes and picks up new items to execute.

If you see no value in the 'Change' column, maybe only one value has been received so far. Wait 30 seconds for another value to arrive.

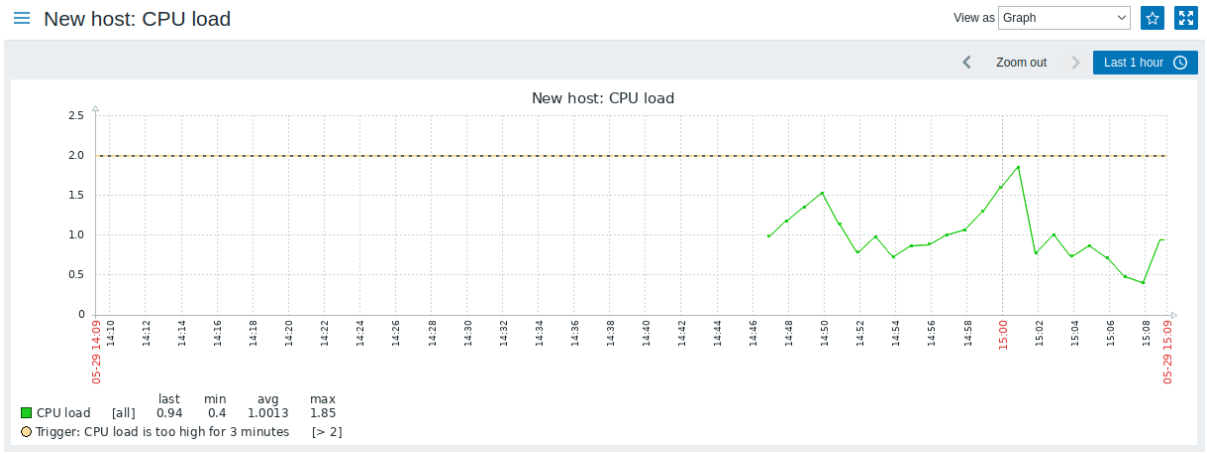
If you do not see information about the item as in the screenshot, make sure that:

- you have filled out the item 'Key' and 'Type of information' fields exactly as in the screenshot;
- both the agent and the server are running;
- host status is 'Monitored' and its availability icon is green;
- the host selected in the host filter is correct;
- the item is enabled.

Graphs

With the item working for a while, it might be time to see something visual. Simple graphs are available for any monitored numeric item without any additional configuration. These graphs are generated on runtime.

To view the graph, go to *Monitoring* → *Latest data* and click on the 'Graph' link next to the item.



4 New trigger

Overview

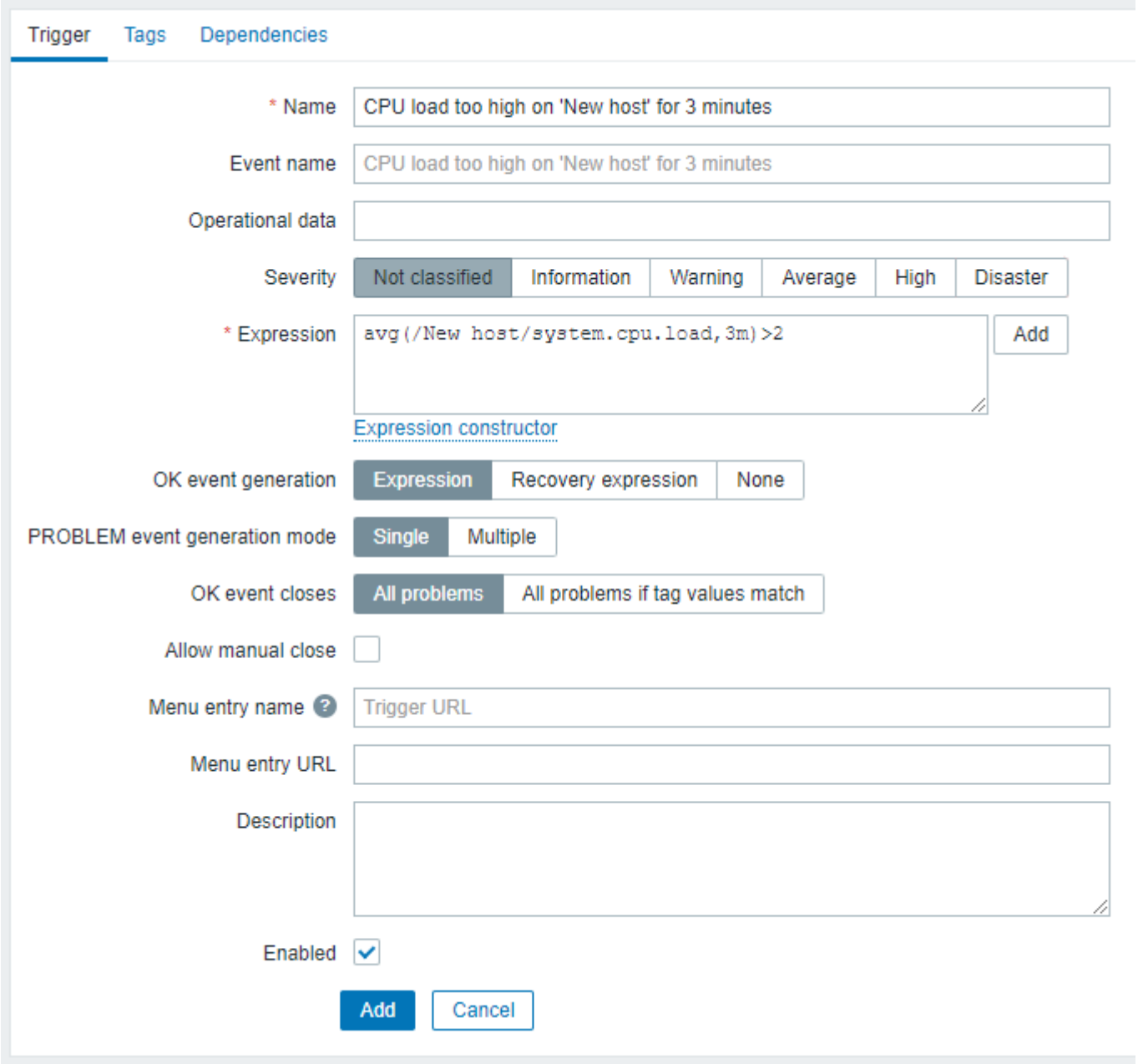
In this section you will learn how to set up a trigger.

Items only collect data. To automatically evaluate incoming data we need to define triggers. A trigger contains an expression that defines a threshold of what is an acceptable level for the data.

If that level is surpassed by the incoming data, a trigger will "fire" or go into a 'Problem' state - letting us know that something has happened that may require attention. If the level is acceptable again, trigger returns to an 'Ok' state.

Adding trigger

To configure a trigger for our item, go to *Data collection → Hosts*, find 'New host' and click on *Triggers* next to it and then on *Create trigger*. This presents us with a trigger definition form.



For our trigger, the essential information to enter here is:

Name

- Enter *CPU load too high on 'New host' for 3 minutes* as the value. This will be the trigger name displayed in lists and elsewhere.

Expression

- Enter: avg(/New host/system.cpu.load,3m)>2

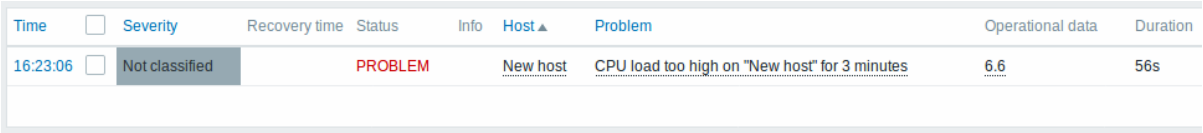
This is the trigger expression. Make sure that the expression is entered right, down to the last symbol. The item key here (system.cpu.load) is used to refer to the item. This particular expression basically says that the problem threshold is exceeded when the CPU load average value for 3 minutes is over 2. You can learn more about the syntax of trigger expressions.

When done, click *Add*. The new trigger should appear in the trigger list.

Displaying trigger status

With a trigger defined, you might be interested to see its status.

If the CPU load has exceeded the threshold level you defined in the trigger, the problem will be displayed in *Monitoring → Problems*.



The flashing in the status column indicates a recent change of trigger status, one that has taken place in the last 30 minutes.

5 Receiving problem notification

Overview

In this section you will learn how to set up alerting in the form of notifications in Zabbix.

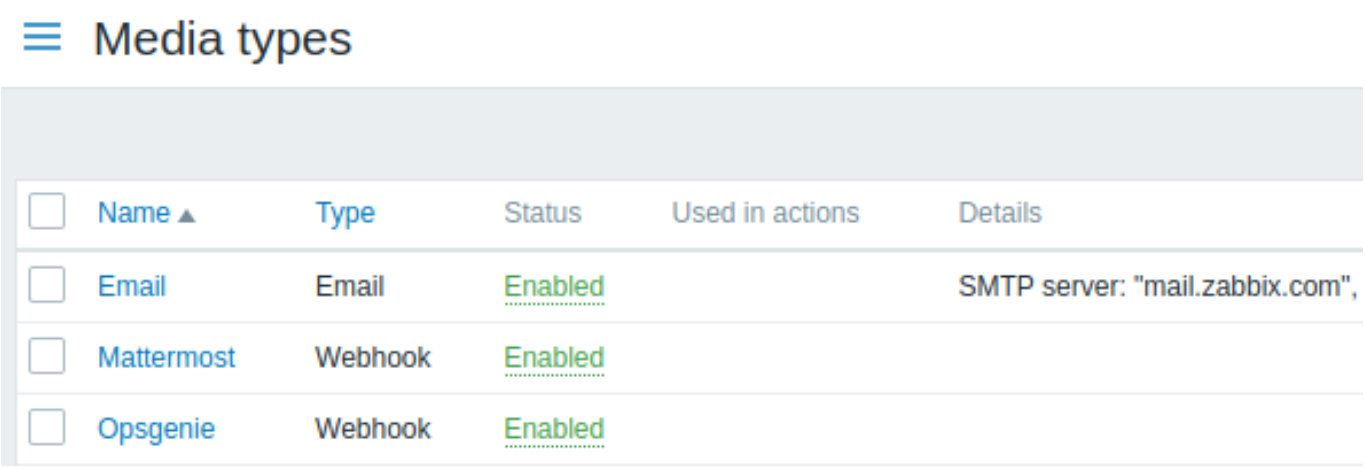
With items collecting data and triggers designed to ”fire” upon problem situations, it would also be useful to have some alerting mechanism in place that would notify us about important events even when we are not directly looking at Zabbix frontend.

This is what notifications do. Email being the most popular delivery method for problem notifications, we will learn how to set up an email notification.

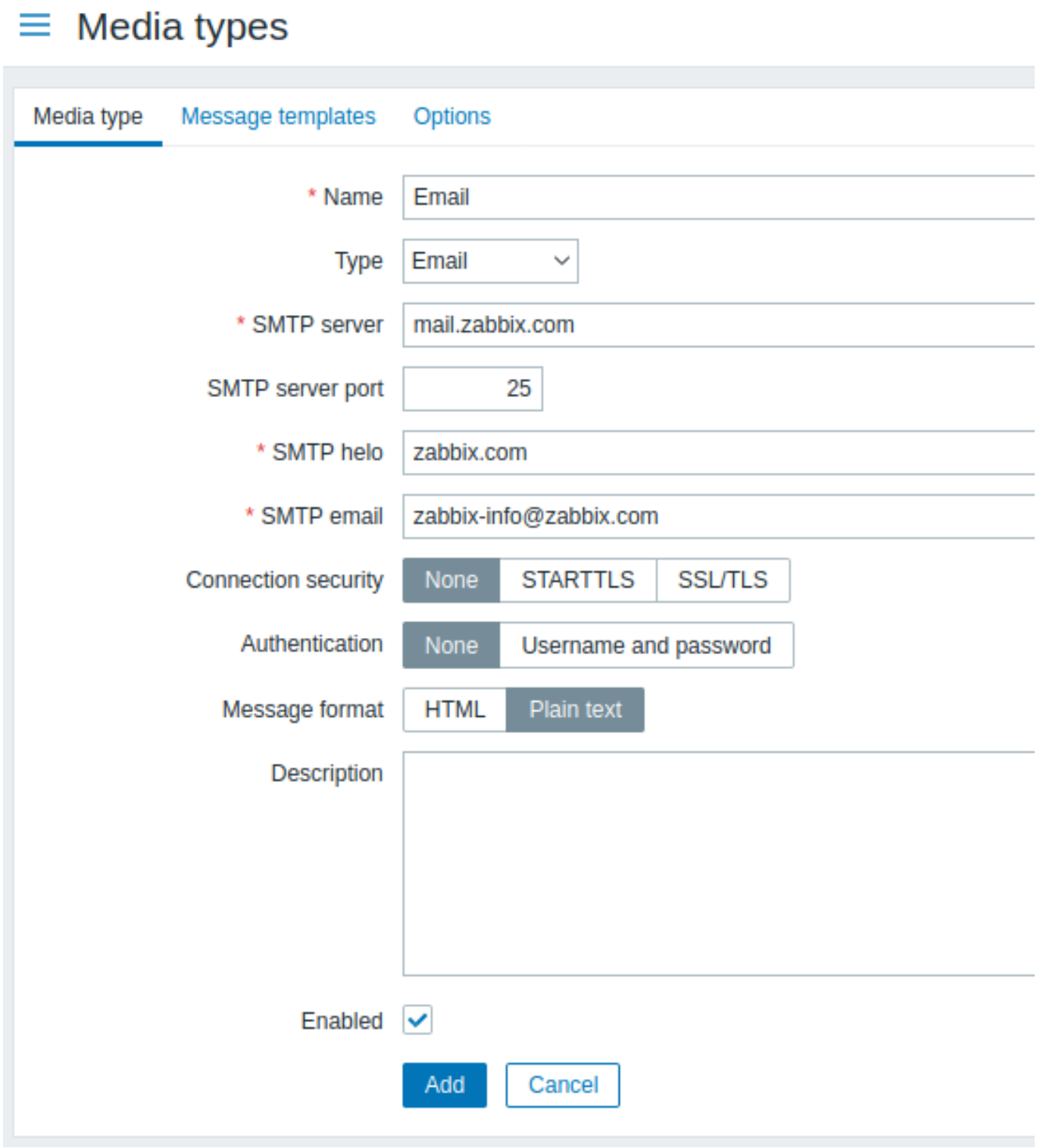
Email settings

Initially there are several predefined notification delivery methods in Zabbix. Email is one of those.

To configure email settings, go to *Alerts* → *Media types* and click on *Email* in the list of pre-defined media types.



This will present us with the email settings definition form.



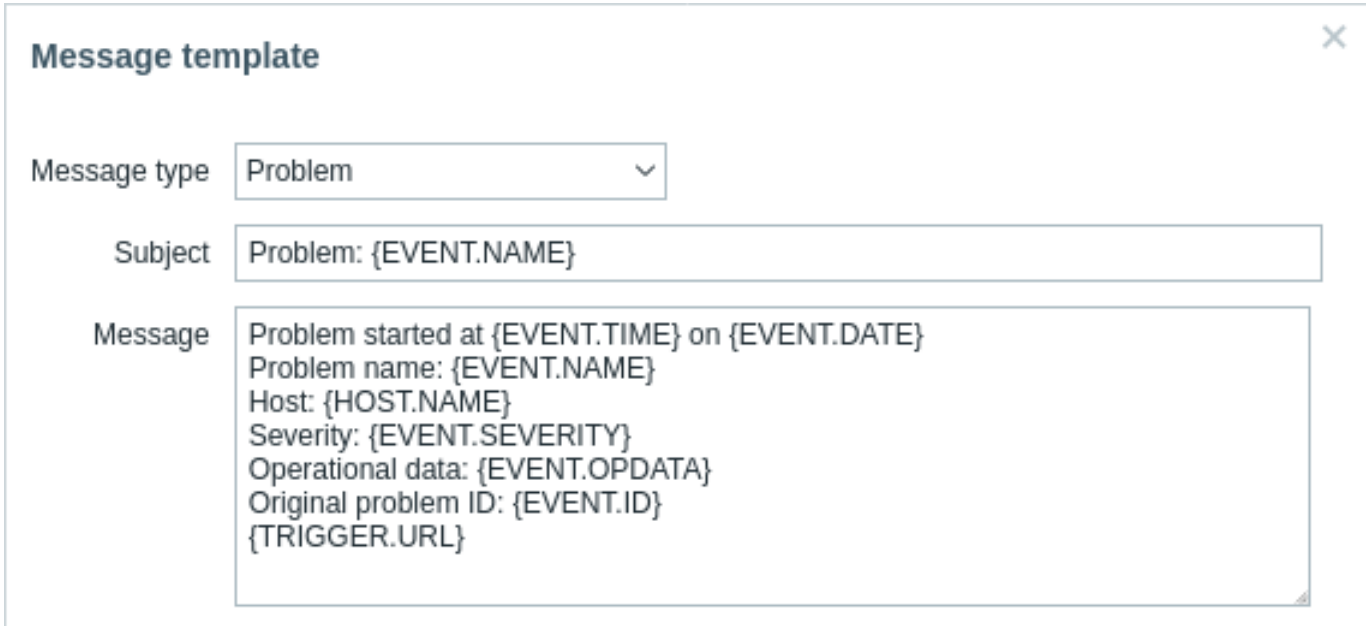
All mandatory input fields are marked with a red asterisk.

In the *Media type* tab, set the values of SMTP server, SMTP helo and SMTP email to the appropriate for your environment.

Note:
'SMTP email' will be used as the 'From' address for the notifications sent from Zabbix.

Next, it is required to define the content of the problem message. The content is defined by means of a message template, configured in the *Message templates* tab.

Click on *Add* to create a message template, and select *Problem* as the message type.

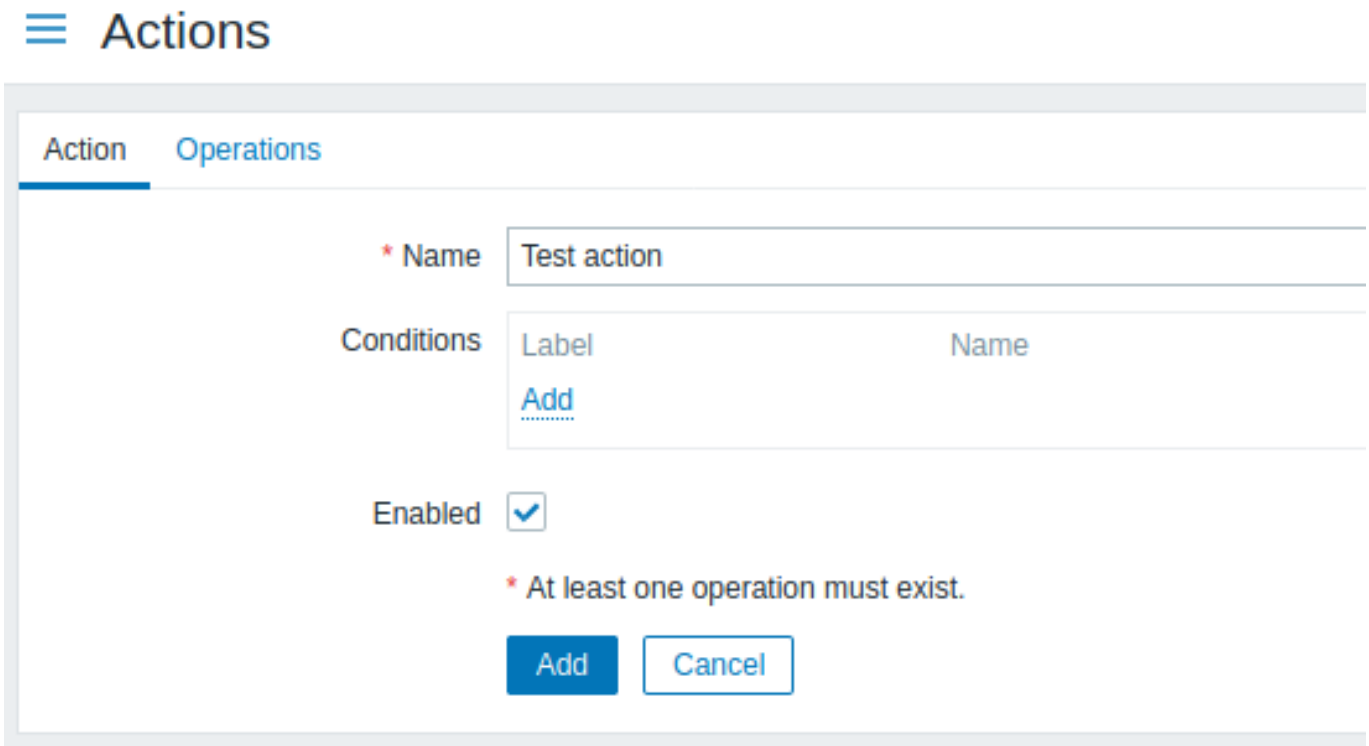


Click on *Add* when ready and save the form.

Now you have configured 'Email' as a working media type. The media type must also be linked to users by defining specific delivery addresses (like we did when configuring a new user), otherwise it will not be used.

New action

Delivering notifications is one of the things actions do in Zabbix. Therefore, to set up a notification, go to *Alerts* → *Actions* → *Trigger actions* and click on *Create action*.

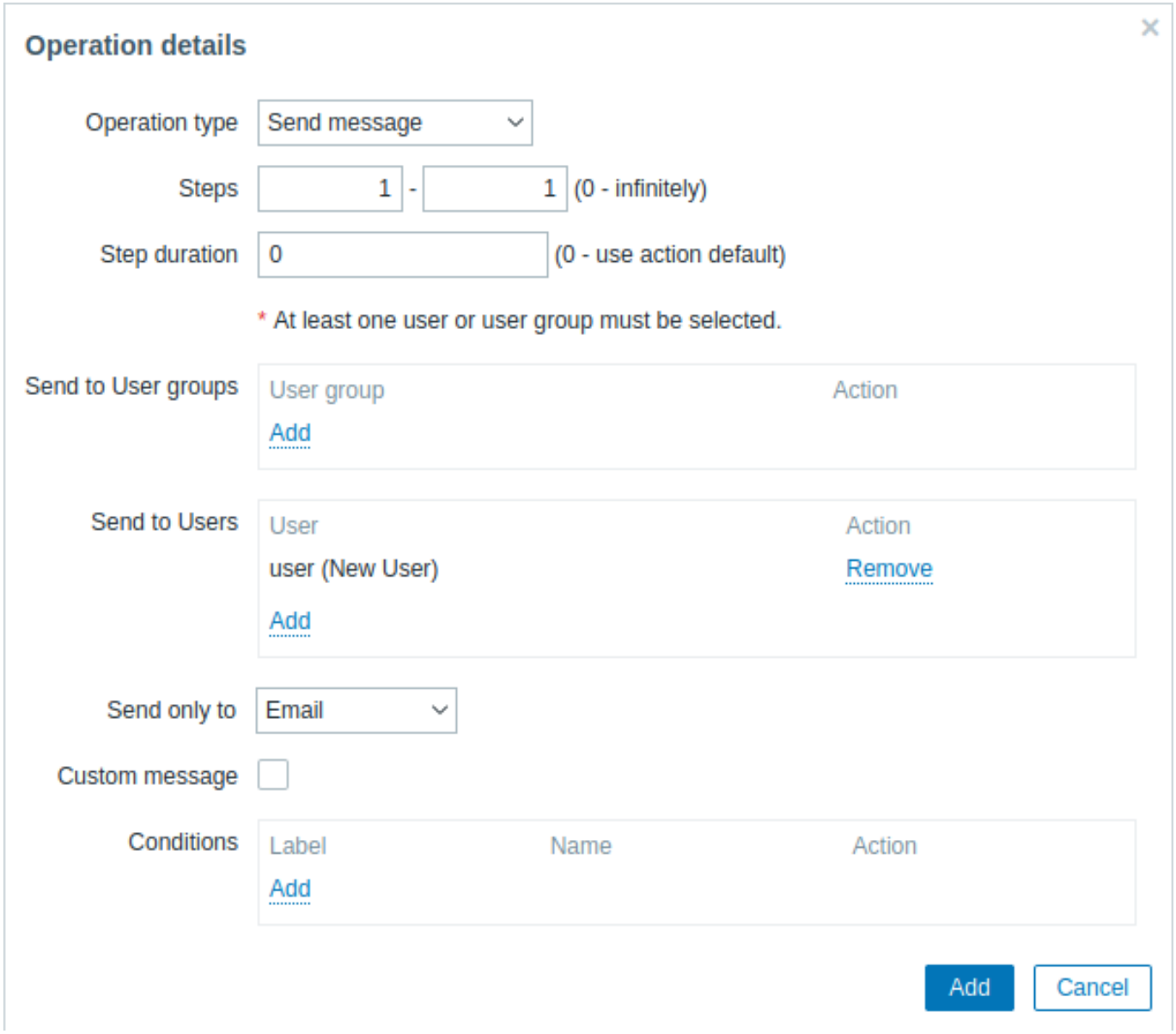


All mandatory input fields are marked with a red asterisk.

In this form, enter a name for the action.

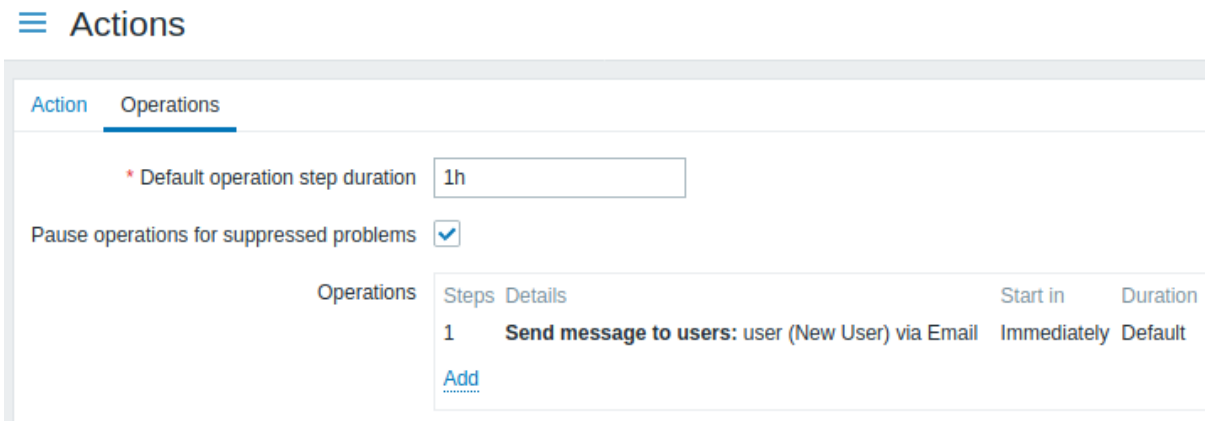
In the most simple case, if we do not add any more specific conditions, the action will be taken upon any trigger change from 'Ok' to 'Problem'.

We still should define what the action should do - and that is done in the *Operations* tab. Click on *Add* in the Operations block, which opens a new operation form.



All mandatory input fields are marked with a red asterisk.

Here, click on *Add* in the *Send to Users* block and select the user ('user') we have defined. Select 'Email' as the value of *Send only to*. When done with this, click on *Add*, and the operation should be added:



That is all for a simple action configuration, so click *Add* in the action form.

Receiving notification

Now, with delivering notifications configured, it would be fun to actually receive one. To help with that, we might on purpose increase the load on our host - so that our trigger "fires" and we receive a problem notification.

Open the console on your host and run:

```
cat /dev/urandom | md5sum
```

You may run one or several of these processes.

Now go to *Monitoring* → *Latest data* and see how the values of 'CPU Load' have increased. Remember, for our trigger to *fire*, the 'CPU Load' value has to go over '2' for 3 minutes running. Once it does:

- in *Monitoring* → *Problems* you should see the trigger with a flashing 'Problem' status
- you should receive a problem notification in your email

Attention:
If notifications do not work:

- verify once again that both the email settings and the action have been configured properly
- make sure the user you created has at least read permissions on the host which generated the event, as noted in the *Adding user* step. The user, being part of the 'Zabbix administrators' user group must have at least read access to 'Linux servers' host group that our host belongs to.
- Additionally, you can check out the action log by going to *Reports* → *Action log*.

6 New template

Overview

In this section you will learn how to set up a template.

Previously we learned how to set up an item, a trigger and how to get a problem notification for the host.

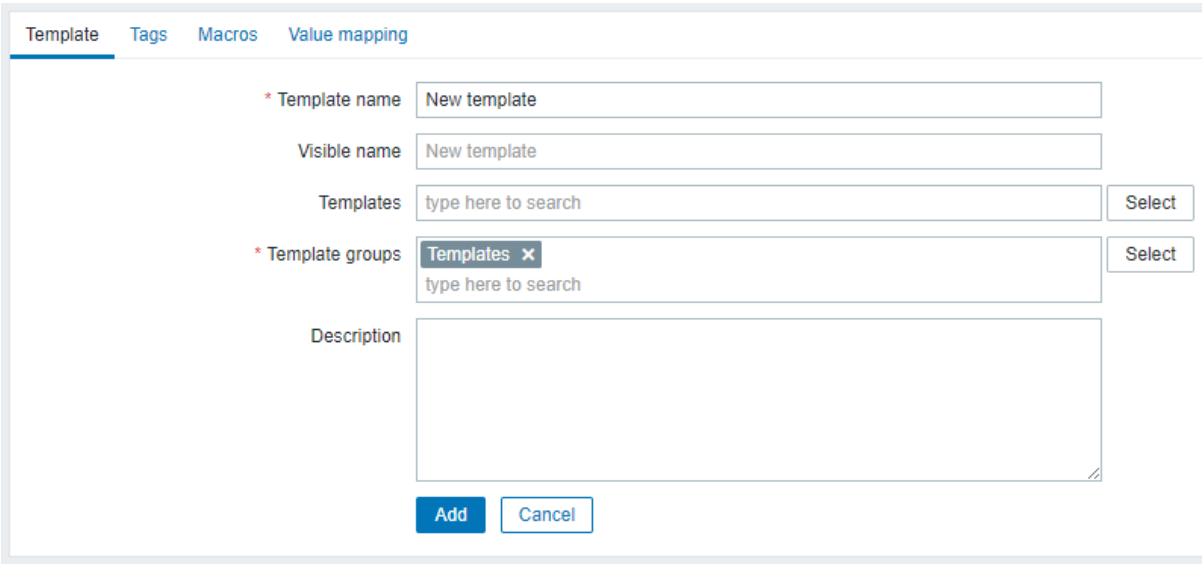
While all of these steps offer a great deal of flexibility in themselves, it may appear like a lot of steps to take if needed for, say, a thousand hosts. Some automation would be handy.

This is where templates come to help. Templates allow to group useful items, triggers and other entities so that those can be reused again and again by applying to hosts in a single step.

When a template is linked to a host, the host inherits all entities of the template. So, basically a pre-prepared bunch of checks can be applied very quickly.

Adding template

To start working with templates, we must first create one. To do that, in *Data collection* → *Templates* click on *Create template*. This will present us with a template configuration form.



All mandatory input fields are marked with a red asterisk.

The required parameters to enter here are:

Template name

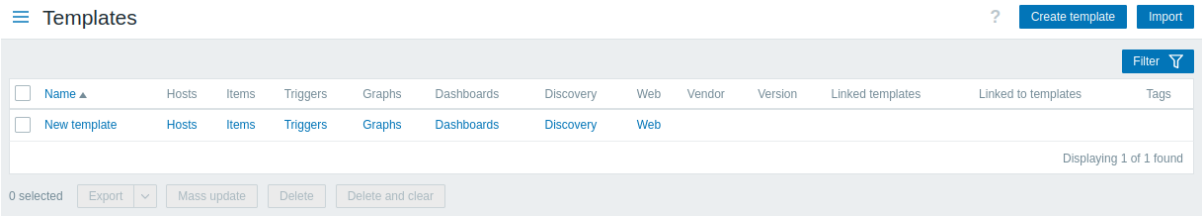
- Enter a template name. Alpha-numericals, spaces and underscores are allowed.

Template groups

- Select one or several groups by clicking *Select* button. The template must belong to a group.

Note:
Access permissions to template groups are assigned in the user group configuration on the **Template permissions** tab in the same way as host permissions. All access permissions are assigned to groups, not individual templates, that's why including the template into at least one group is mandatory.

When done, click *Add*. Your new template should be visible in the list of templates. You can also use the filter to find your template.



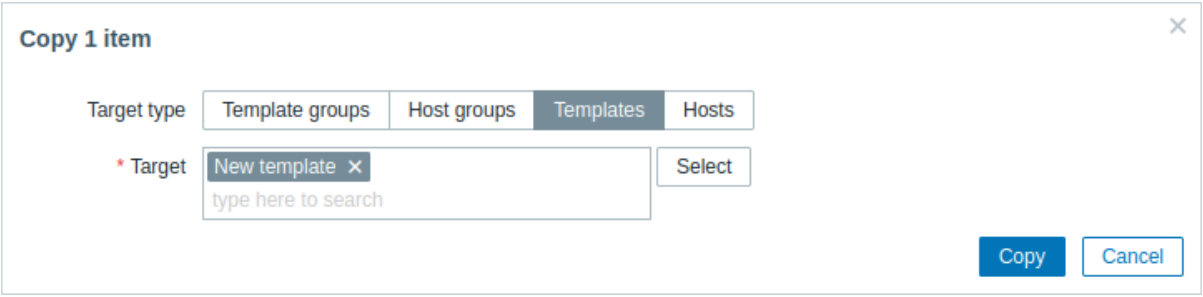
As you may see, the template is there, but it holds nothing in it - no items, triggers or other entities.

Adding item to template

To add an item to the template, go to the item list for 'New host'. In *Data collection* → *Hosts* click on *Items* next to 'New host'.

Then:

- Mark the checkbox of the 'CPU Load' item in the list.
- Click on *Copy* below the list.
- Select the template to copy the item to.



All mandatory input fields are marked with a red asterisk.

- Click on *Copy*.

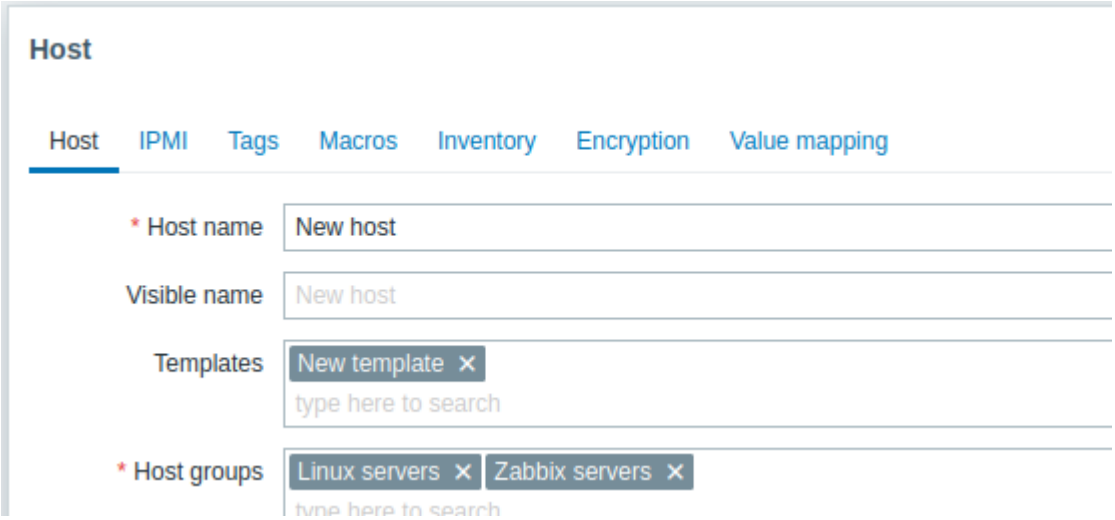
If you now go to *Data collection* → *Templates*, 'New template' should have one new item in it.

We will stop at one item only for now, but similarly you can add any other items, triggers or other entities to the template until it's a fairly complete set of entities for given purpose (monitoring OS, monitoring single application).

Linking template to host

With a template ready, it only remains to add it to a host. For that, go to *Data collection* → *Hosts*, click on 'New host' to open its property form and find the **Templates** field.

Start typing *New template* in the *Templates* field. The name of template we have created should appear in the dropdown list. Scroll down to select. See that it appears in the *Templates* field.



Click *Update* in the form to save the changes. The template is now added to the host, with all entities that it holds.

This way it can be applied to any other host as well. Any changes to the items, triggers and other entities at the template level will propagate to the hosts the template is linked to.

Linking pre-defined templates to hosts

As you may have noticed, Zabbix comes with a set of predefined templates for various OS, devices and applications. To get started with monitoring very quickly, you may link the appropriate one of them to a host, but beware that these templates need to be fine-tuned for your environment. Some checks may not be needed, and polling intervals may be way too frequent.

More information about templates is available.

6 Zabbix appliance

Overview As an alternative to setting up manually or reusing an existing server for Zabbix, users may download a Zabbix appliance or a Zabbix appliance installation CD image.

Zabbix appliance and installation CD versions are based on AlmaLinux 8 (x86_64).

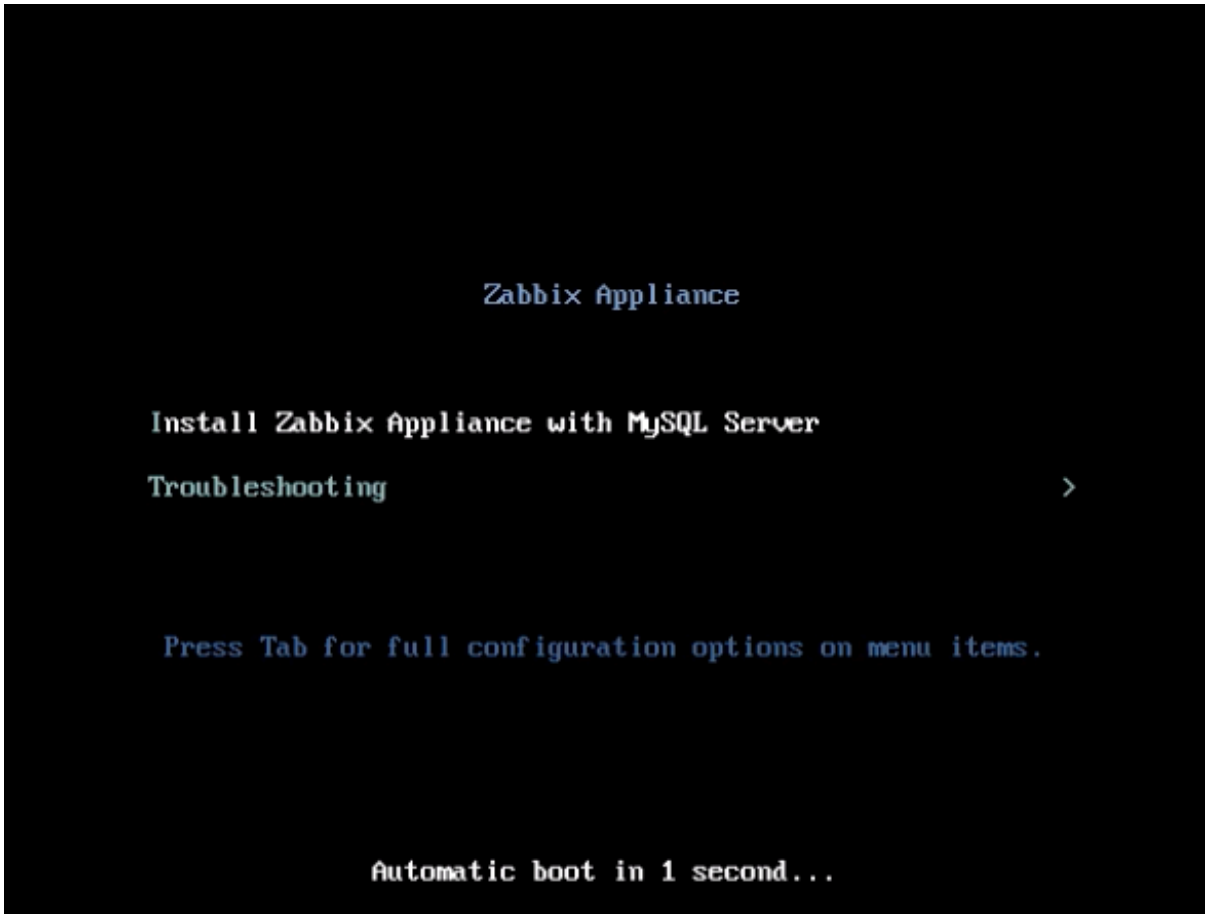
Zabbix appliance installation CD can be used for instant deployment of Zabbix server (MySQL).

Attention:
You can use this appliance to evaluate Zabbix. The appliance is not intended for serious production use.

System requirements:

- *RAM*: 1.5 GB
- *Disk space*: at least 8 GB should be allocated for the virtual machine
- *CPU*: 2 cores minimum

Zabbix installation CD/DVD boot menu:



Zabbix appliance contains a Zabbix server (configured and running on MySQL) and a frontend.

Zabbix virtual appliance is available in the following formats:

- VMware (.vmx)
- Open virtualization format (.ovf)
- Microsoft Hyper-V 2012 (.vhdx)
- Microsoft Hyper-V 2008 (.vhd)
- KVM, Parallels, QEMU, USB stick, VirtualBox, Xen (.raw)
- KVM, QEMU (.qcow2)

To get started, boot the appliance and point a browser at the IP the appliance has received over DHCP.

Attention:
DHCP must be enabled on the host.

To get the IP address from inside the virtual machine run:

```
ip addr show
```

To access Zabbix frontend, go to **http://<host_ip>** (for access from the host's browser bridged mode should be enabled in the VM network settings).

Note:
If the appliance fails to start up in Hyper-V, you may want to press `Ctrl+Alt+F2` to switch tty sessions.

1 Changes to AlmaLinux 8 configuration The appliance is based on AlmaLinux 8. There are some changes applied to the base AlmaLinux configuration.

1.1 Repositories

Official Zabbix repository has been added to */etc/yum.repos.d*:

```
[zabbix]
name=Zabbix Official Repository - $basearch
baseurl=http://repo.zabbix.com/zabbix/6.4/rhel/8/$basearch/
enabled=1
```

```
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-ZABBIX-A14FE591
```

1.2 Firewall configuration

The appliance uses iptables firewall with predefined rules:

- Opened SSH port (22 TCP);
- Opened Zabbix agent (10050 TCP) and Zabbix trapper (10051 TCP) ports;
- Opened HTTP (80 TCP) and HTTPS (443 TCP) ports;
- Opened SNMP trap port (162 UDP);
- Opened outgoing connections to NTP port (53 UDP);
- ICMP packets limited to 5 packets per second;
- All other incoming connections are dropped.

1.3 Using a static IP address

By default the appliance uses DHCP to obtain the IP address. To specify a static IP address:

- Log in as root user;
- Open */etc/sysconfig/network-scripts/ifcfg-eth0* file;
- Replace *BOOTPROTO=dhcp* with *BOOTPROTO=none*
- Add the following lines:
  - *IPADDR=<IP address of the appliance>*
  - *PREFIX=<CIDR prefix>*
  - *GATEWAY=<gateway IP address>*
  - *DNS1=<DNS server IP address>*
- Run **systemctl restart network** command.

Consult the official Red Hat documentation if needed.

1.4 Changing time zone

By default the appliance uses UTC for the system clock. To change the time zone, copy the appropriate file from */usr/share/zoneinfo* to */etc/localtime*, for example:

```
cp /usr/share/zoneinfo/Europe/Riga /etc/localtime
```

2 Zabbix configuration Zabbix appliance setup has the following passwords and configuration changes:

2.1 Credentials (login:password)

System:

- root:zabbix

Zabbix frontend:

- Admin:zabbix

Database:

- root:<random>
- zabbix:<random>

Note:
Database passwords are randomly generated during the installation process.
Root password is stored inside the /root/.my.cnf file. It is not required to input a password under the ”root” account.

To change the database user password, changes have to be made in the following locations:

- MySQL;
- /etc/zabbix/zabbix_server.conf;
- /etc/zabbix/web/zabbix.conf.php.

Note:
Separate users `zabbix_srv` and `zabbix_web` are defined for the server and the frontend respectively.

2.2 File locations

- Configuration files are located in **/etc/zabbix**.
- Zabbix server, proxy and agent logfiles are located in **/var/log/zabbix**.

- Zabbix frontend is located in **/usr/share/zabbix**.
- Home directory for the user **zabbix** is **/var/lib/zabbix**.

2.3 Changes to Zabbix configuration

- Frontend timezone is set to Europe/Riga (this can be modified in **/etc/php-fpm.d/zabbix.conf**);

3 Frontend access By default, access to the frontend is allowed from anywhere.

The frontend can be accessed at *http://<host>*.

This can be customized in **/etc/nginx/conf.d/zabbix.conf**. Nginx has to be restarted after modifying this file. To do so, log in using SSH as **root** user and execute:

```
systemctl restart nginx
```

4 Firewall By default, only the ports listed in the configuration changes above are open. To open additional ports, modify *"/etc/sysconfig/iptables"* file and reload firewall rules:

```
systemctl reload iptables
```

5 Upgrading The Zabbix appliance packages may be upgraded. To do so, run:

```
dnf update zabbix*
```

6 System Services Systemd services are available:

```
systemctl list-units zabbix*
```

7 Format-specific notes 7.1 VMware

The images in *vmdk* format are usable directly in VMware Player, Server and Workstation products. For use in ESX, ESXi and vSphere they must be converted using VMware vCenter Converter (authentication required for download). If you use VMWare vCenter Converter, you may encounter issues with the hybrid network adapter. In that case, you can try specifying the E1000 adapter during the conversion process. Alternatively, after the conversion is complete, you can delete the existing adapter and add an E1000 adapter.

7.2 HDD/flash image (raw)

```
dd if=./zabbix_appliance_6.4.0.raw of=/dev/sdc bs=4k conv=fdatasync
```

Replace */dev/sdc* with your Flash/HDD disk device.

7 Configuration

Please use the sidebar to access content in the Configuration section.

1 Configuring a template

Overview

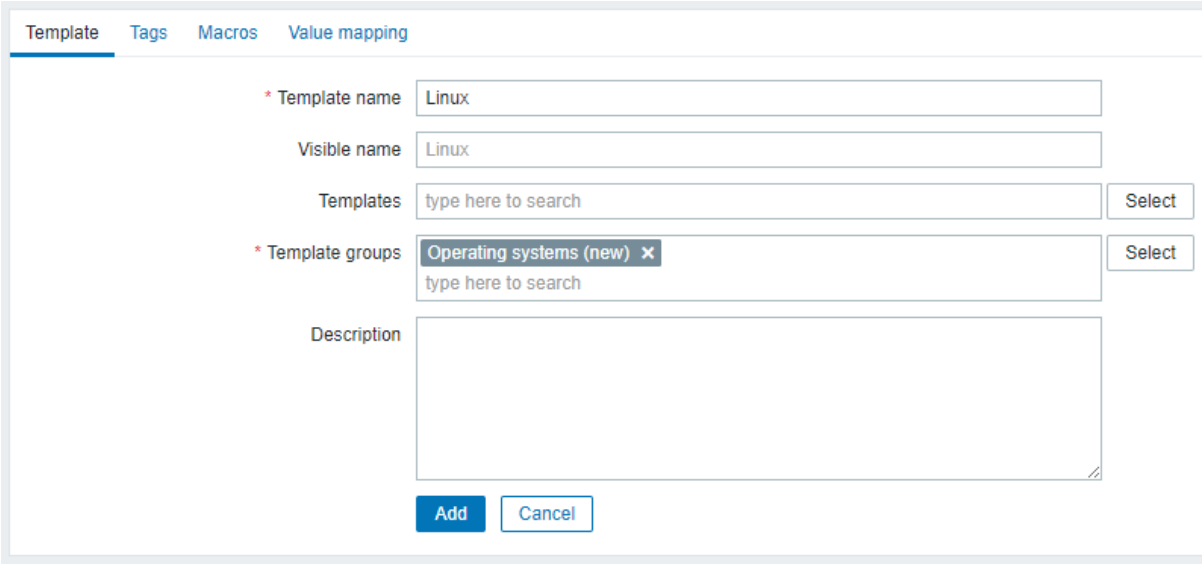
Configuring a template requires that you first create a template by defining its general parameters and then you add entities (items, triggers, graphs, etc.) to it.

Creating a template

To create a template, do the following:

1. Go to *Data collection → Templates*.
2. Click on *Create template*.
3. Edit template attributes.

The **Template** tab contains general template attributes.

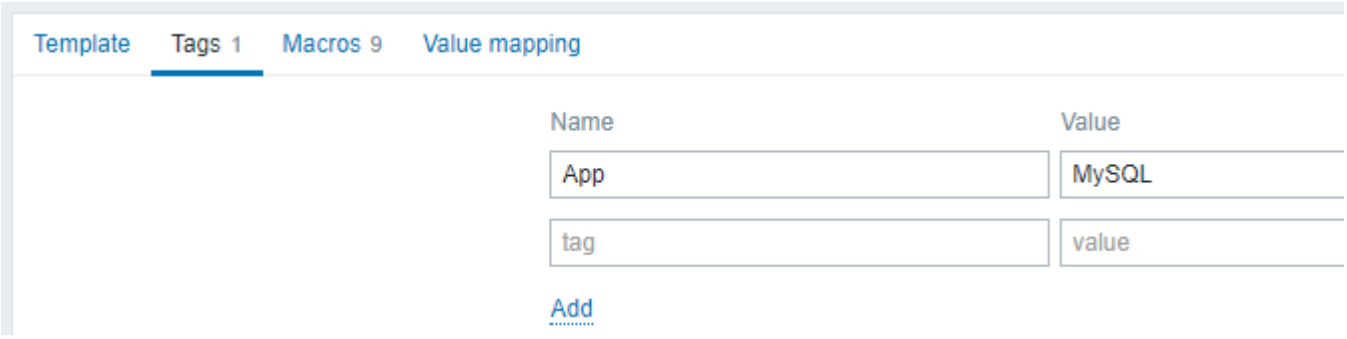


All mandatory input fields are marked with a red asterisk.

Template attributes:

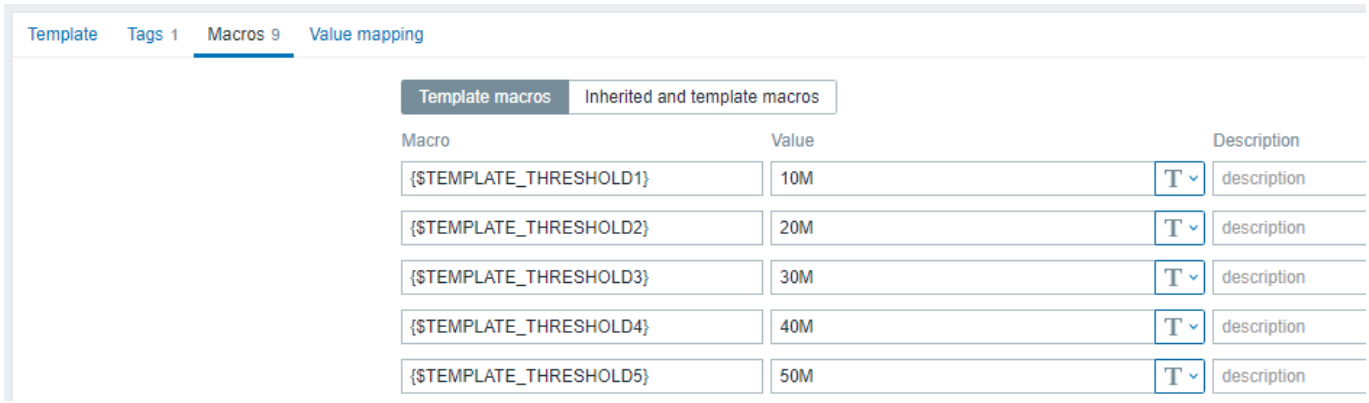
Parameter	Description
Template name	Unique template name. Alphanumerics, spaces, dots, dashes, and underscores are allowed. Leading and trailing spaces are not allowed.
Visible name	If you set this name, it will be the one visible in lists, maps, etc.
Templates	Link one or more templates to this template. All entities (items, triggers, etc.) will be inherited from the linked templates. To link a new template, start typing the template name in the *Link new templates* field. A list of matching templates will appear; scroll down to select. Alternatively, you may click on *Select* next to the field and select templates from the list in a pop-up window. The templates that are selected in the *Link new templates* field will be linked to the template when the template configuration form is saved or updated. To unlink a template, use one of the two options in the *Linked templates* block: *Unlink* - unlink the template, but preserve its entities (items, triggers, etc.); *Unlink and clear* - unlink the template and remove all of its entities (items, triggers, etc.).
Template groups	Template groups the template belongs to.
Description	Template description.
Vendor and version	Template vendor and version; displayed only when updating existing templates (out-of-the-box templates provided by Zabbix, imported templates, or templates modified through the Template API) if the template configuration contains such information. Cannot be modified in Zabbix frontend. For out-of-the-box templates, version is displayed as follows: major version of Zabbix, delimiter (”-”), revision number (increased with each new version of the template, and reset with each major version of Zabbix). For example, 6.4-0, 6.4-5, 7.0-0, 7.0-3.

The **Tags** tab allows you to define template-level tags. All problems of hosts linked to this template will be tagged with the values entered here.

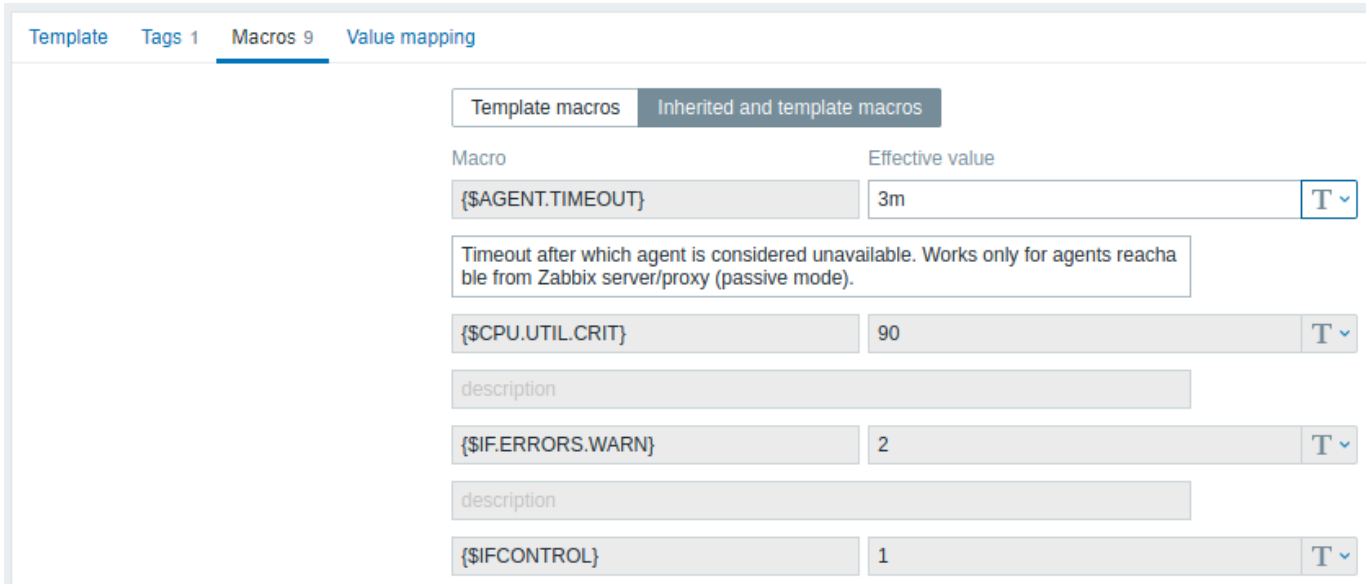


User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags.

The **Macros** tab allows you to define template-level user macros as a name-value pairs. Note that macro values can be kept as plain text, secret text, or Vault secret. Adding a description is also supported.



If you select the *Inherited and template macros* option, you may also view macros from linked templates and global macros that the template will inherit, as well as the values that the macros will resolve to.



For convenience, links to the respective templates, as well as a link to global macro configuration is provided. It is also possible to edit a linked template macro or global macro on the template level, effectively creating a copy of the macro on the template.

The **Value mapping** tab allows to configure human-friendly representation of item data in value mappings.

Buttons:

Add	Add the template. The added template should appear in the list.
Update	Update the properties of an existing template.
Clone	Create another template based on the properties of the current template, including the entities (items, triggers, etc.) inherited from linked templates, but excluding template vendor and version.
Full clone	Create another template based on the properties of the current template, including the entities (items, triggers, etc.) both inherited from linked templates and directly attached to the current template, but excluding template vendor and version.
Delete	Delete the template; entities of the template (items, triggers, etc.) remain with the linked hosts.
Delete and clear	Delete the template and all its entities from linked hosts.
Cancel	Cancel the editing of template properties.

Adding items, triggers, graphs

> **Attention:**
> Items have to be added to a template first. Triggers and graphs cannot be added without the corresponding item.

There are two ways to add items to the template:

1. To create new items, follow the guidelines for Creating an item.

2. To add existing items to the template:

- Go to *Data collection* → *Hosts* (or *Templates*).
- Click on *Items* in the row of the required host/template.
- Mark the checkboxes of items you want to add to the template.
- Click on *Copy* below the item list.
- Select the template (or group of templates) the items should be copied to and click on *Copy*.
  All the selected items should be copied to the template.

Adding triggers and graphs is done in a similar fashion (from the list of triggers and graphs respectively), again, keeping in mind that they can only be added if the required items are added first.

Adding dashboards

To add dashboards to a template in *Data collection* → *Templates*, do the following:

1. Click on *Dashboards* in the row of the template.
2. Configure a dashboard following the guidelines of configuring dashboards.

> **Attention:**
> The widgets that can be included in a template dashboard are: *Clock*, *Graph (classic)*, *Graph prototype*, *Item value*, *Plain text*, *URL*.

> **Note:**
> For details on accessing host dashboards that are created from template dashboards, see the host dashboard section.

Configuring low-level discovery rules

See the low-level discovery section of the manual.

Adding web scenarios

To add web scenarios to a template in *Data collection* → *Templates*, do the following:

1. Click on *Web* in the row of the template.
2. Configure a web scenario following the usual method of configuring web scenarios.

2 Configuring a template group

Overview

Template groups are used for the logical grouping of templates and assigning user permissions to them.

Each template must have at least one template group assigned. A template may belong to multiple template groups, and each template group may contain multiple templates.

Note that in Zabbix, all permissions are based on groups: user groups, host groups, and template groups. So, even if a single user needs access to a single template, it is granted by adding the user to a user group that has permission to access the template group containing that template.

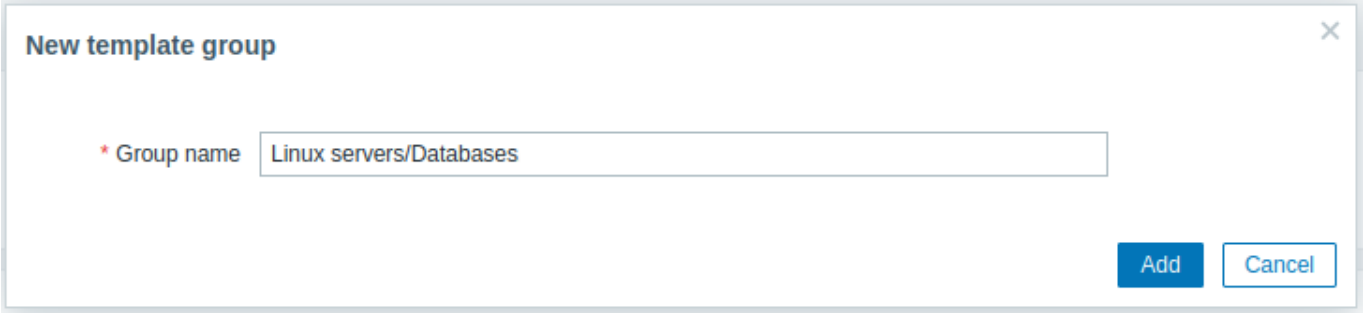
Configuration

> **Attention:**
> Only Super admin users can create template groups.

There are two options of creating a template group in Zabbix frontend.

Option one:

- Go to: *Data collection → Template groups*
- Click on *Create template group* in the upper right corner of the screen
- Enter the group name in the form



Option two: when configuring a template, enter a non-existing group name in the *Template groups* input field.

Once the template group is created, you can click on the template name in the list under *Data collection → Template groups* to edit the group name, clone the group, or delete the group.

Deleting a template group only deletes the logical group, not the templates in the group. It is not possible to delete a template group that is the only group for any existing template.

Creating template subgroups

A template subgroup (or nested template group) is a child of the parent template group that contains it.

A subgroup is created by using the forward slash '/' in the group name input field to denote its relation to the parent group(s). For example:

- inputting `Linux servers/Databases` creates the `Linux servers/Databases` subgroup of the parent group `Linux servers`.
- inputting `Linux servers/Databases/MySQL/Tokyo` creates the respective subgroup within the nested parent groups `Linux servers`, `Linux servers/Databases`, `Linux servers/Databases/MySQL`.

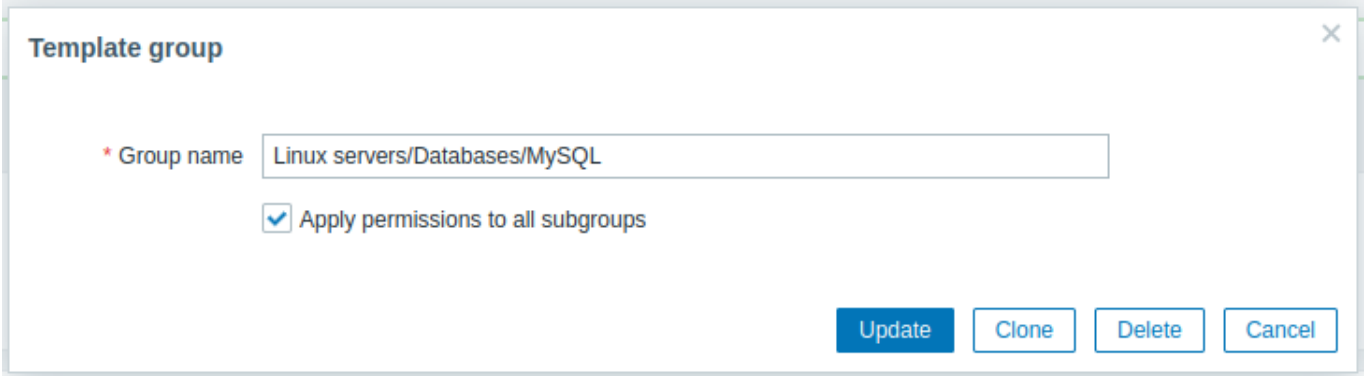
When creating a subgroup, using leading or trailing slashes, or several slashes in a row is not allowed. Escaping of '/' is not supported.

It is not required to create any parent template group(s) before creating a subgroup. You can choose whether to start by creating a subgroup (for example, `Linux servers/Databases`) or any parent template group(s) (in our example, `Linux servers`). If you start by creating a subgroup, parent template group(s) will **not** be created automatically.

Permissions to template groups

- When creating a subgroup to an existing parent template group (for example, creating `Linux servers/Databases` when `Linux servers` already exists), user group permissions to the subgroup are inherited from the parent.
- When creating a parent template group to an existing subgroup (for example, creating `Linux servers` when `Linux servers/Databases` already exists), no permissions to the parent are set.

When editing any template group, you can also set an additional option, *Apply permissions to all subgroups*.



Marking this checkbox and clicking on *Update* will apply the same level of permissions to all current and future subgroups of the template group being edited.

So, if any user groups have been given varying permissions to the subgroups of the template group being edited, marking the checkbox will grant all current and future subgroups the same user permissions as the group being edited.

Note that this option is not saved in the database and will override existing permissions. Any changes made through this option can be reverted only manually.

3 Linking/unlinking

Overview

Linking is a process whereby templates are applied to hosts, whereas unlinking removes the association with the template from a host.

Linking a template

To link a template to the host, do the following:

1. Go to *Data collection* → *Hosts*.
2. Click on the required host.
3. Start typing the template name in the *Templates* field. A list of matching templates will appear; scroll down to select. Alternatively, you may click on *Select* next to the field and select one or several templates from the list in a popup window.
4. Click on *Add/Update* in the host attributes form.

The host will now have all the entities of the template. This includes items, triggers, graphs, low-level discovery rules, web scenarios, as well as dashboards.

Attention:
Linking multiple templates to the same host will fail if those templates contain items with the same item key. And, as triggers and graphs use items, they cannot be linked to a single host from multiple templates either, if using identical item keys.

When entities (items, triggers, etc.) are added from the template:

- previously existing identical entities on the host are updated as entities of the template, and **any existing host-level customizations to the entity are lost**;
- entities from the template are added;
- any directly linked entities that, prior to template linkage, existed only on the host remain untouched.

In the lists, all entities from the template now are prefixed by the template name, indicating that these belong to the particular template. The template name itself (in gray text) is a link allowing to access the list of those entities on the template level.

Note:
For some items, such as external checks, HTTP agent checks, simple checks, SSH checks and Telnet checks, a host interface is optional. If, at the time of linking a template, the host does not have an interface defined these items will be added without an interface. If you add a host interface later it will not be assigned automatically to already existing items. To assign the newly added host interface to all template items at once, unlink the template from the host and then link it back again. To preserve item history use the option *Unlink*, do not use *Unlink and clear*.

If some entity is not prefixed by the template name, it means that it existed on the host before and was not added by the template.

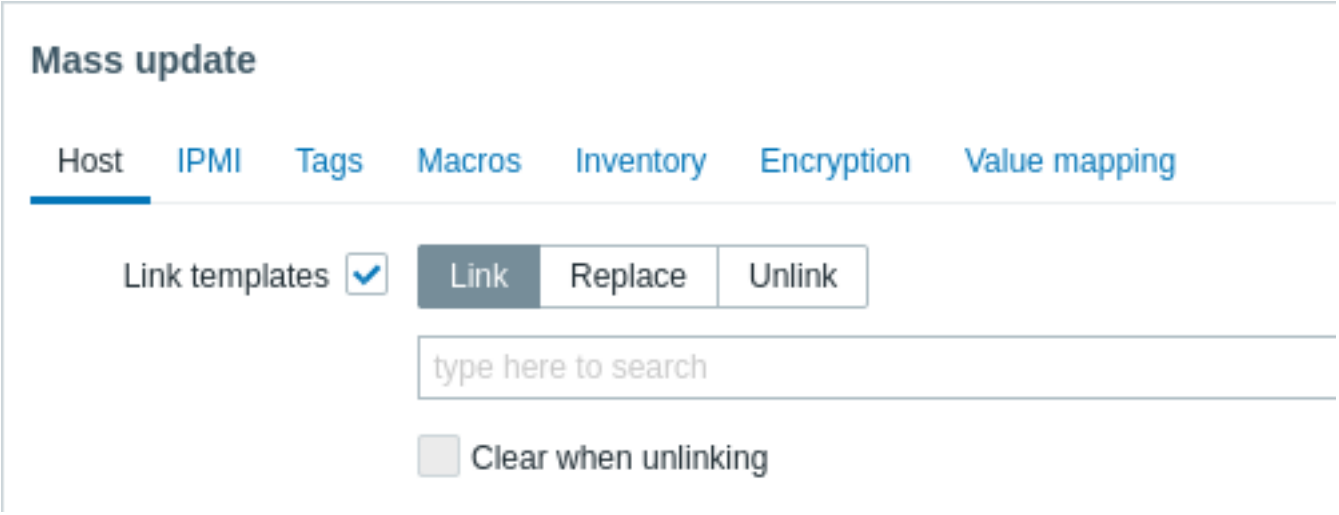
Entity uniqueness criteria

When adding entities (items, triggers, etc.) from a template it is important to know what of those entities already exist on the host and need to be updated and what entities differ. The uniqueness criteria for deciding upon the sameness/difference are:

- for items - the item key;
- for triggers - trigger name and expression;
- for custom graphs - graph name and its items.

Linking templates to several hosts

To update template linkage of many hosts, in *Data collection* → *Hosts* select some hosts by marking their checkboxes, then click on **Mass update** below the list and then select *Link templates*:



To link additional templates, start typing the template name in the auto-complete field until a dropdown appears offering the matching templates. Just scroll down to select the template to link.

The *Replace* option will allow to link a new template while unlinking any template that was linked to the hosts before. The *Unlink* option will allow to specify which templates to unlink. The *Clear when unlinking* option will allow to not only unlink any previously linked templates, but also remove all entities inherited from them (items, triggers, etc.).

Note:
Zabbix offers a sizable set of predefined templates. You can use these for reference, but beware of using them unchanged in production as they may contain too many items and poll for data too often. If you feel like using them, finetune them to fit you real needs.

Editing linked entities

If you try to edit an item or a trigger that was linked from the template, you may realize that many key options are disabled for editing. This makes sense as the idea of templates is that things are edited in a one-touch manner on the template level. However, you still can, for example, enable/disable an item on individual hosts and set the update interval, history length and some other parameters.

If you want to edit the entity fully, you have to edit it on the template level (template level shortcut is displayed in the form name), keeping in mind that these changes will affect all hosts that have this template linked to them.

Attention:
Any customizations to the entities implemented on a template-level will override the previous customizations of the entities on a host-level.

Unlinking a template

To unlink a template from a host, do the following:

1. Go to *Data collection → Hosts*.
2. Click on the required host and find the *Templates* field.
3. Click on *Unlink* or *Unlink and clear* next to the template to unlink the template.
4. Click on *Update* in the host attributes form.

Choosing the *Unlink* option will simply remove association with the template, while leaving all its entities with the host. This includes items, triggers, graphs, low-level discovery rules, and web scenarios, but excludes dashboards. Note that value maps and tags inherited from linked templates will also be removed.

Choosing the *Unlink and clear* option will remove both the association with the template and all its entities (items, triggers, etc.).

4 Nesting

Overview

Nesting is a way of one template encompassing one or more other templates.

As it makes sense to separate out entities on individual templates for various services, applications, etc., you may end up with quite a few templates all of which may need to be linked to quite a few hosts. To simplify the picture, it is possible to link some templates together in a single template.

The benefit of nesting is that you have to link only one template to the host, and the host will automatically inherit all entities from the templates that are linked to the one template. For example, if we link *T1* and *T2* to *T3*, we supplement *T3* with all entities from *T1* and *T2*, but not vice versa. If we link *T1* to *T2* and *T3*, we supplement *T2* and *T3* with entities from *T1*.

Configuring nested templates

To link templates, you need to take an existing template (or create a new one), and then:

1. Open the template configuration form.
2. Find the *Templates* field.
3. Click on *Select* to open the *Templates* pop-up window.
4. In the pop-up window, choose the required templates, and then click on *Select* to add the templates to the list.
5. Click on *Add* or *Update* in the template configuration form.

Thus, all entities of the configured template, as well as all entities of linked templates will now appear in the template configuration. This includes items, triggers, graphs, low-level discovery rules, and web scenarios, but excludes dashboards. However, linked template dashboards will, nevertheless, be inherited by hosts.

To unlink any of the linked templates, click on *Unlink* or *Unlink and clear* in the template configuration form, and then click on *Update*.

The *Unlink* option will simply remove the association with the linked template, while not removing any of its entities (items, triggers, etc.).

The *Unlink and clear* option will remove both the association with the linked template, as well as all its entities (items, triggers, etc.).

5 Mass update

Overview

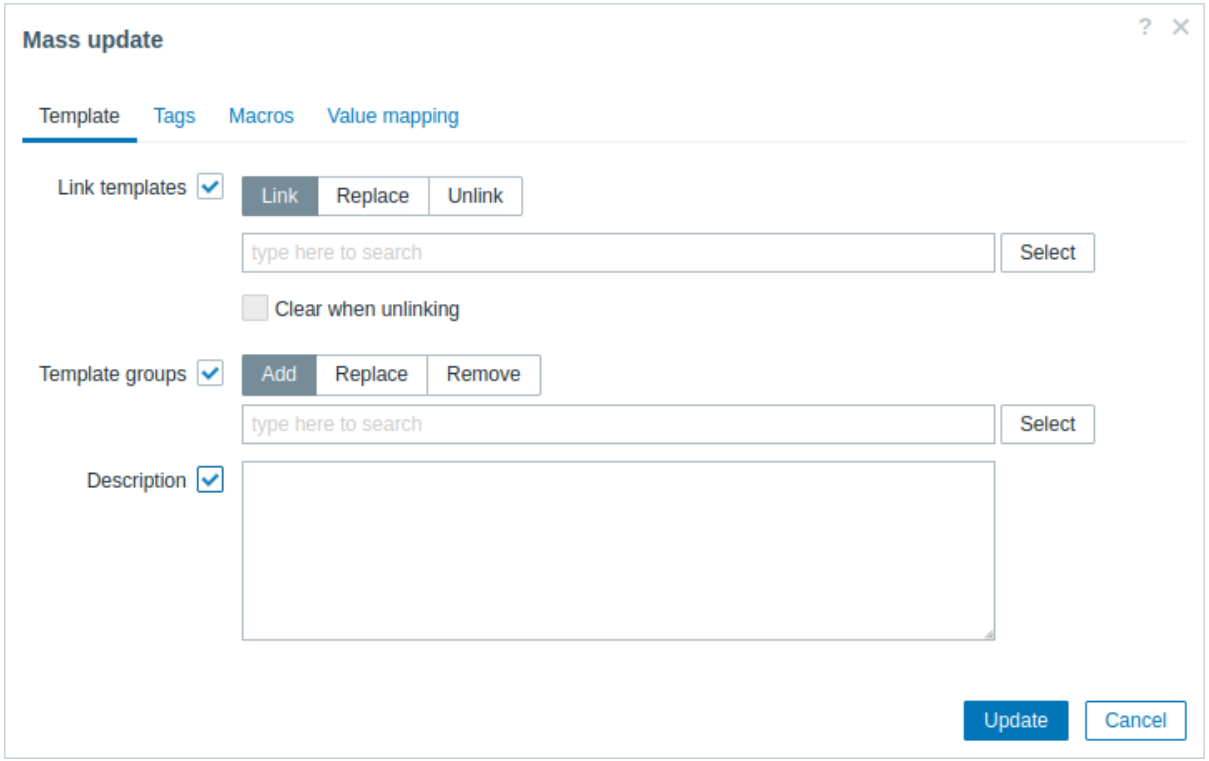
Sometimes you may want to change some attribute for a number of templates at once. Instead of opening each individual template for editing, you may use the mass update function for that.

Using mass update

To mass-update some templates, do the following:

1. Mark the checkboxes before the templates you want to update in the template list.
2. Click on *Mass update* below the list.
3. Navigate to the tab with required attributes (*Template*, *Tags*, *Macros* or *Value mapping*).
4. Mark the checkboxes of any attribute to update and enter a new value for them.

The **Template** tab contains general template mass update options.



The following options are available when selecting the respective button for the *Link templates* update:

- *Link* - specify which additional templates to link;
- *Replace* - specify which templates to link while at the same time unlinking any previously linked templates;
- *Unlink* - specify which templates to unlink.

To specify the templates to link/unlink, start typing the template name in the auto-complete field until a dropdown appears offering the matching templates. Just scroll down to select the required templates.

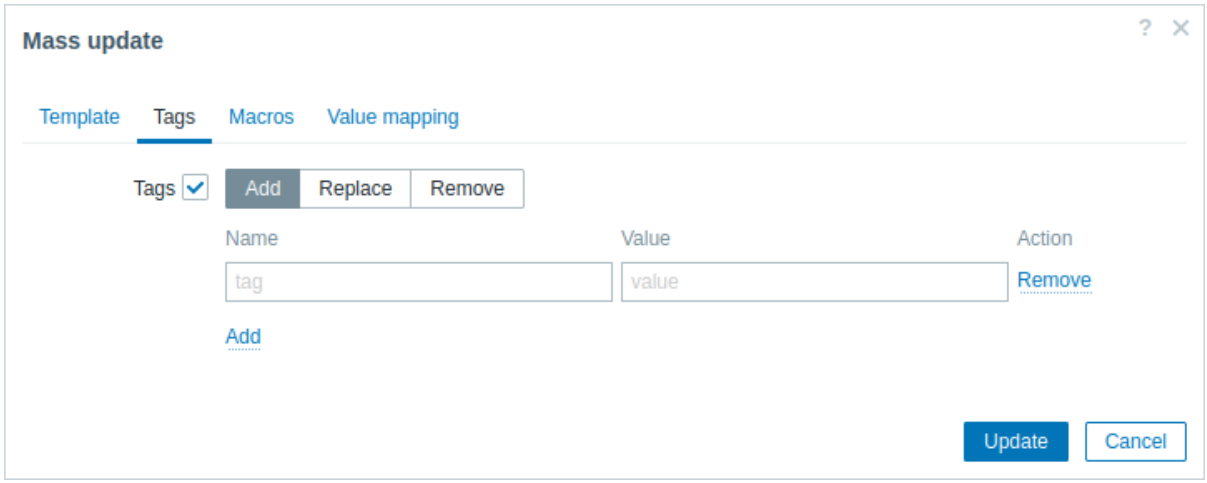
The *Clear when unlinking* option will allow to unlink any previously linked templates, as well as to remove all elements inherited from them (items, triggers, graphs, etc.).

The following options are available when selecting the respective button for the *Template groups* update:

- *Add* - allows to specify additional template groups from the existing ones or enter completely new template groups for the templates;
- *Replace* - will remove the template from any existing template groups and replace them with the one(s) specified in this field (existing or new template groups);
- *Remove* - will remove specific template groups from templates.

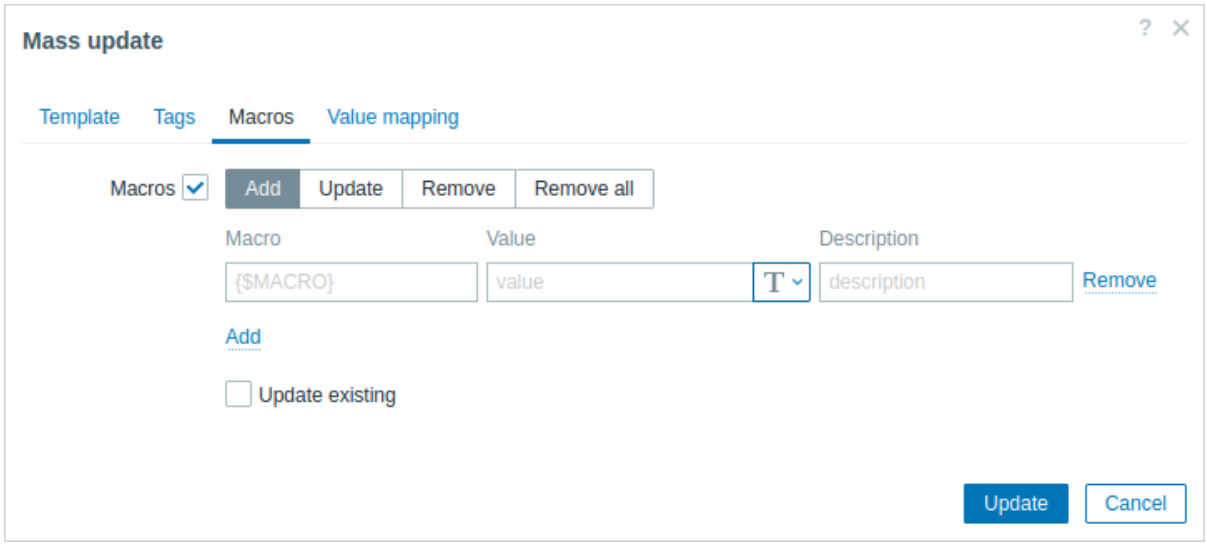
These fields are auto-complete - starting to type in them offers a dropdown of matching template groups. If the template group is new, it also appears in the dropdown and it is indicated by *(new)* after the string. Just scroll down to select.

The **Tags** tab allows you to mass update template-level tags.



User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags. Note that tags with the same name, but different values are not considered 'duplicates' and can be added to the same template.

The **Macros** tab allows you to mass update template-level macros.



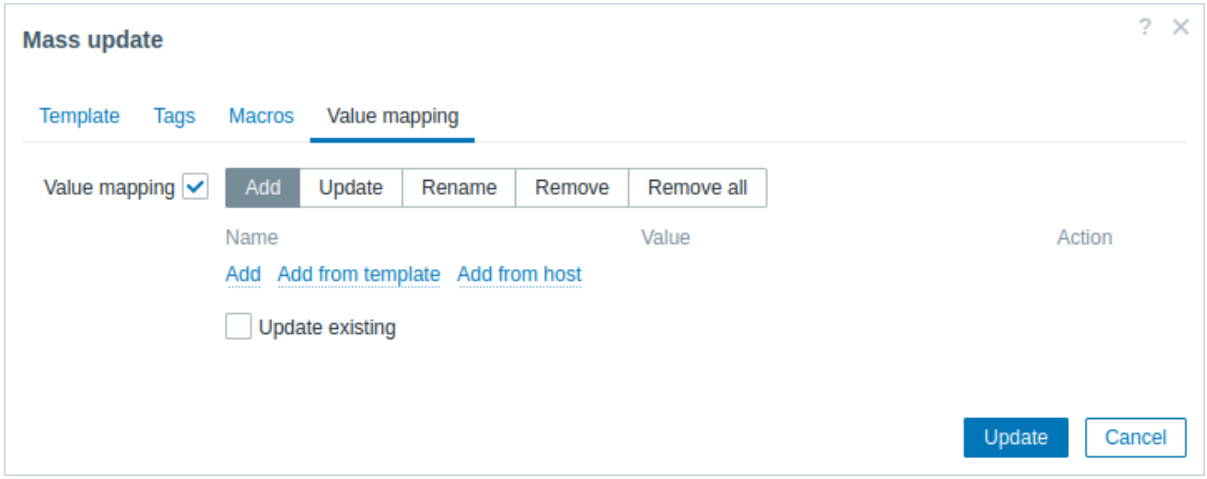
The following options are available when selecting the respective button for macros update:

- *Add* - allows to specify additional user macros for the templates. If *Update existing* checkbox is checked, value, type and description for the specified macro name will be updated. If unchecked, if a macro with that name already exist on the

template(s), it will not be updated.

- *Update* - will replace values, types and descriptions of macros specified in this list. If *Add missing* checkbox is checked, macro that didn't previously exist on a template will be added as new macro. If unchecked, only macros that already exist on a template will be updated.
- *Remove* - will remove specified macros from templates. If *Except selected* box is checked, all macros except specified in the list will be removed. If unchecked, only macros specified in the list will be removed.
- *Remove all* - will remove all user macros from templates. If *I confirm to remove all macros* checkbox is not checked, a new popup window will open asking to confirm removal of all macros.

The **Value mapping** tab allows you to mass update value mappings.



Buttons with the following options are available for value map update:

- *Add* - add value maps to the templates. If you mark *Update existing*, all properties of the value map with this name will be updated. Otherwise, if a value map with that name already exists, it will not be updated.
- *Update* - update existing value maps. If you mark *Add missing*, a value map that didn't previously exist on a template will be added as a new value map. Otherwise only the value maps that already exist on a template will be updated.
- *Rename* - give new name to an existing value map.
- *Remove* - remove the specified value maps from the templates. If you mark *Except selected*, all value maps will be removed **except** the ones that are specified.
- *Remove all* - remove all value maps from the templates. If the *I confirm to remove all value maps* checkbox is not marked, a new popup window will open asking to confirm the removal.

Add from template and *Add from host* options are available for value mapping add/update operations. They allow to select value mappings from a template or a host respectively.

When done with all required changes, click on *Update*. The attributes will be updated accordingly for all the selected templates.

1 Hosts and host groups

What is a "host"?

In Zabbix, a "host" refers to any physical or virtual device, application, service, or any other logically-related collection of monitored parameters.

Creating hosts is one of the first monitoring tasks in Zabbix. For example, if you want to monitor some parameters on a server "x", you must first create a host called, say, "Server X" and then you can look to add monitoring items to it.

Hosts are organized into host groups.

Proceed to creating and configuring a host.

1 Configuring a host

Overview

To configure a host in Zabbix frontend, do the following:

- Go to: *Data collection → Hosts* or *Monitoring → Hosts*
- Click on *Create host* to the right (or on the host name to edit an existing host)

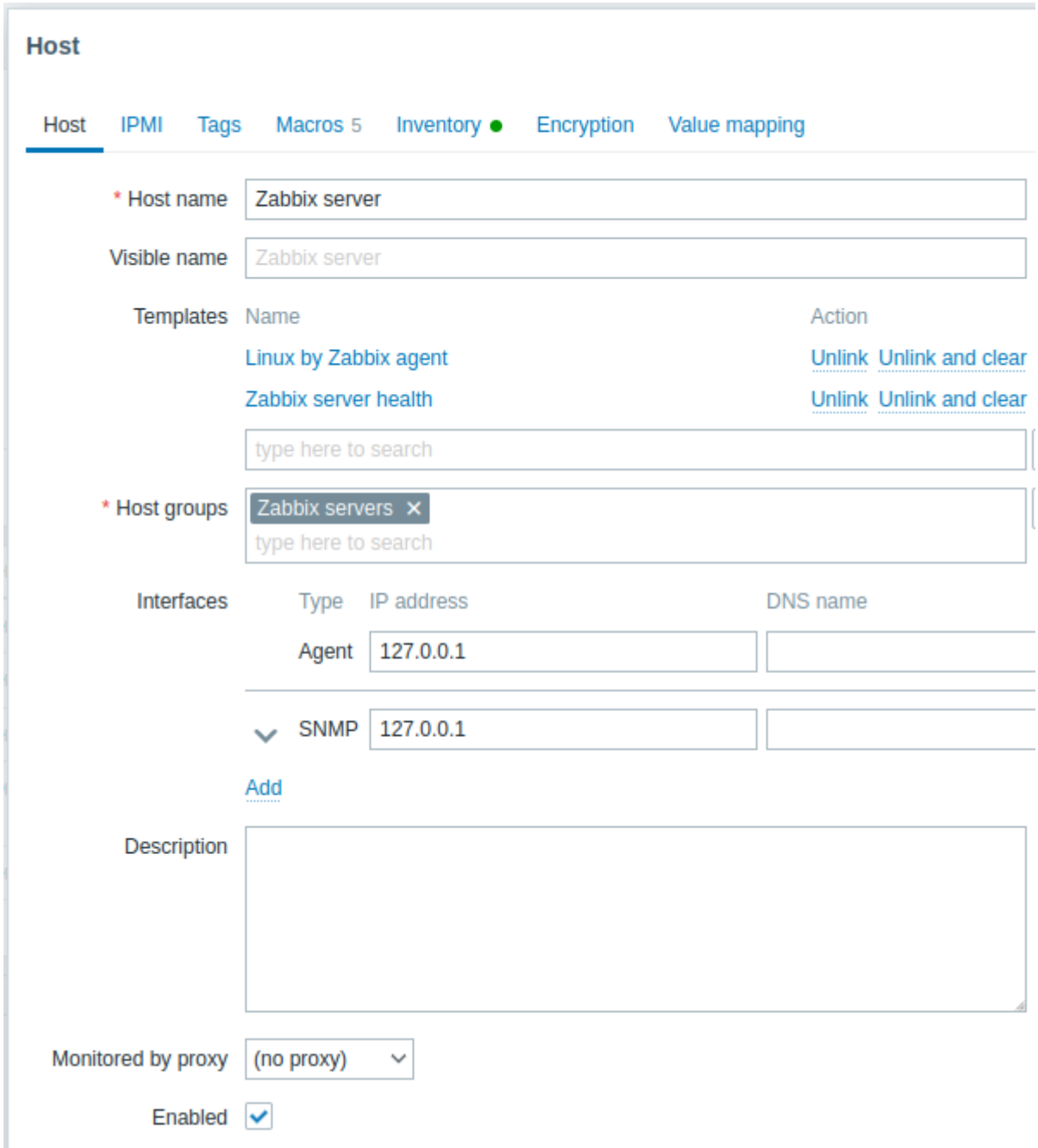
- Enter parameters of the host in the form

You can also use the *Clone* and *Full clone* buttons in the form of an existing host to create a new host. Clicking on *Clone* will retain all host parameters and template linkage (keeping all entities from those templates). *Full clone* will additionally retain directly attached entities (items, triggers, graphs, low-level discovery rules and web scenarios).

Note: When a host is cloned, it will retain all template entities as they are originally on the template. Any changes to those entities made on the existing host level (such as changed item interval, modified regular expression or added prototypes to the low-level discovery rule) will not be cloned to the new host; instead they will be as on the template.

Configuration

The **Host** tab contains general host attributes:



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Host name	Enter a unique host name. Alphanumerics, spaces, dots, dashes and underscores are allowed. However, leading and trailing spaces are disallowed. *Note:* With Zabbix agent running on the host you are configuring, the agent configuration file parameter *Hostname* must have the same value as the host name entered here. The name in the parameter is needed in the processing of active checks.

Parameter	Description
Visible name	Enter a unique visible name for the host. If you set this name, it will be the one visible in lists, maps, etc instead of the technical host name. This attribute has UTF-8 support.
Templates	Link templates to the host. All entities (items, triggers, etc.) will be inherited from the template. To link a new template, start typing the template name in the text input field. A list of matching templates will appear; scroll down to select. Alternatively, you may click on *Select* next to the field and select templates from the list in a popup window. All selected templates will be linked to the host when the host configuration form is saved or updated. To unlink a template, use one of the two options in the *Linked templates* block: *Unlink* - unlink the template, but preserve its entities (items, triggers, etc.); *Unlink and clear* - unlink the template and remove all its entities (items, triggers, etc.). Listed template names are clickable links leading to the template configuration form.
Host groups	Select host groups the host belongs to. A host must belong to at least one host group. A new group can be created and linked to the host by adding a non-existing group name.
Interfaces	Several host interface types are supported for a host: *Agent*, *SNMP*, *JMX* and *IPMI*. No interfaces are defined by default. To add a new interface, click on *Add* in the *Interfaces* block, select the interface type and enter *IP/DNS*, *Connect to* and *Port* info. *Note:* Interfaces that are used in any items cannot be removed and link *Remove* is grayed out for them. See Configuring SNMP monitoring for additional details on configuring an SNMP interface (v1, v2 and v3).
IP address	Host IP address (optional).
DNS name	Host DNS name (optional).
Connect to	Clicking the respective button will tell Zabbix server what to use to retrieve data from agents: **IP** - Connect to the host IP address (recommended) **DNS** - Connect to the host DNS name
Port	TCP/UDP port number. Default values are: 10050 for Zabbix agent, 161 for SNMP agent, 12345 for JMX and 623 for IPMI.
Default	Check the radio button to set the default interface.
Description	Enter the host description.
Monitored by proxy	The host can be monitored either by Zabbix server or one of Zabbix proxies: **(no proxy)** - host is monitored by Zabbix server **Proxy name** - host is monitored by Zabbix proxy "Proxy name"
Enabled	When the checkbox is checked, the host is enabled - ready for monitoring. When the checkbox is unchecked, the host is disabled - not monitored: For passive data requests initiated by Zabbix server/proxy (for example, Zabbix agent, SNMP agent, simple checks), monitoring stops as soon as you disable the host. For Zabbix agent active checks, monitoring stops within the time frame (approx. 5 seconds) that Zabbix agent receives information about the host having been disabled. During this brief interval, the host will continue to locally collect data for the active checks and try sending it to the server/proxy; however, since the host is marked as *Disabled*, the server/proxy will reject the data.

The **IPMI** tab contains IPMI management attributes.

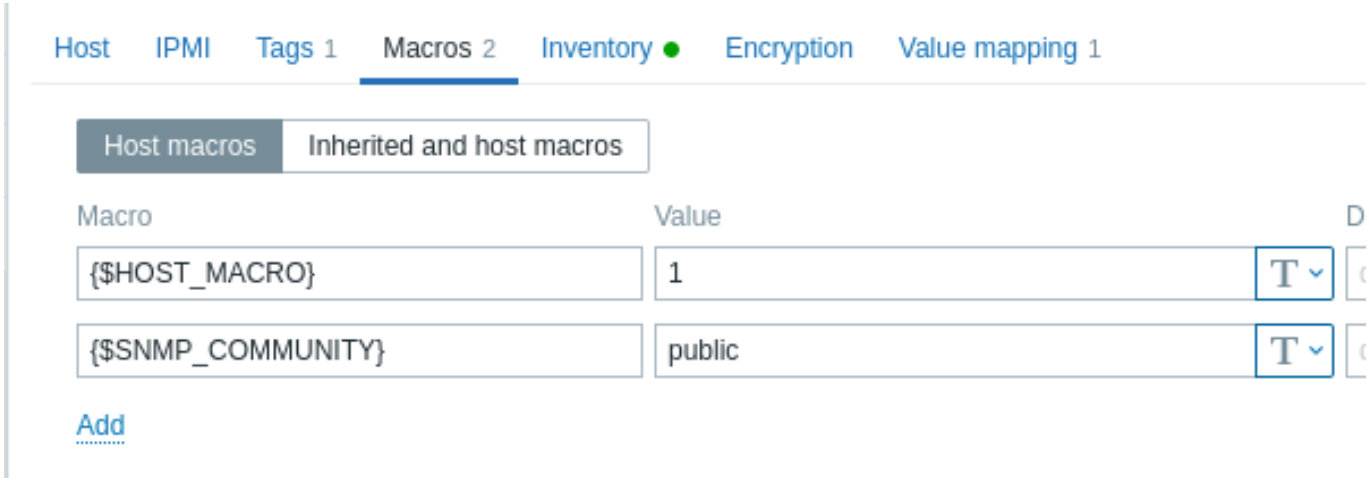
Parameter	Description
Authentication algorithm	Select the authentication algorithm.
Privilege level	Select the privilege level.
Username	User name for authentication. User macros may be used.
Password	Password for authentication. User macros may be used.

The **Tags** tab allows you to define host-level tags. All problems of this host will be tagged with the values entered here.

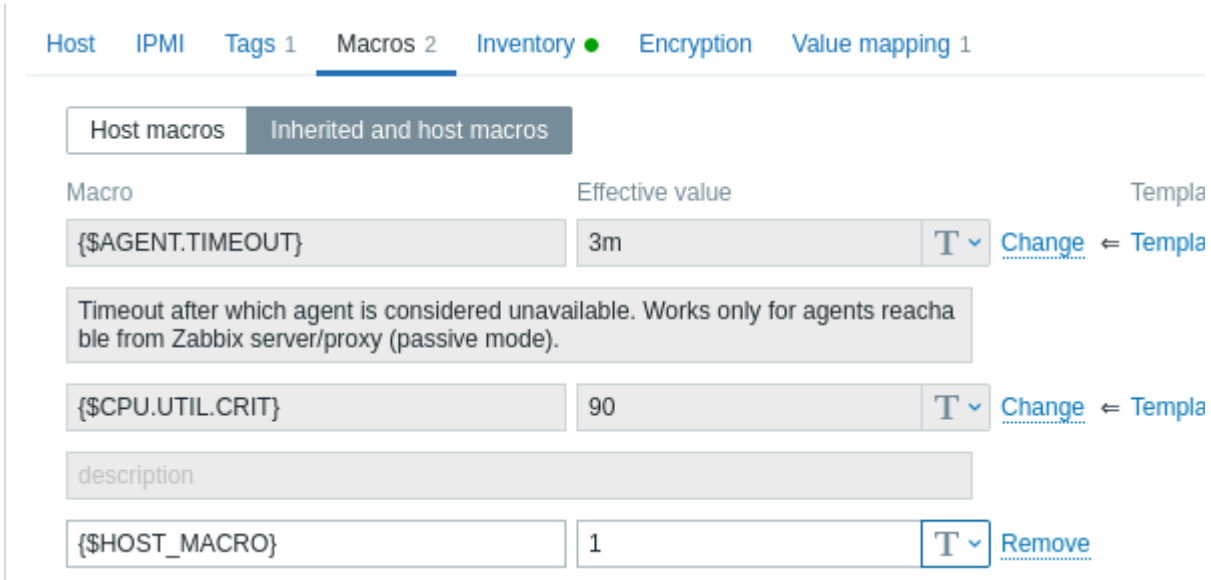


User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags.

The **Macros** tab allows you to define host-level user macros as a name-value pairs. Note that macro values can be kept as plain text, secret text or Vault secret. Adding a description is also supported.

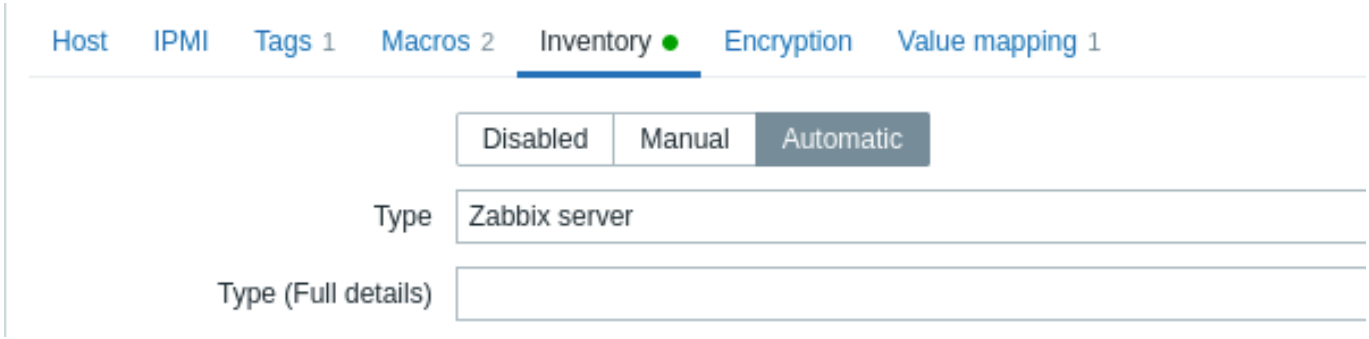


You may also view here template-level and global user macros if you select the *Inherited and host macros* option. That is where all defined user macros for the host are displayed with the value they resolve to as well as their origin.



For convenience, links to respective templates and global macro configuration are provided. It is also possible to edit a template/global macro on the host level, effectively creating a copy of the macro on the host.

The **Inventory** tab allows you to manually enter inventory information for the host. You can also select to enable *Automatic* inventory population, or disable inventory population for this host.



If inventory is enabled (manual or automatic), a green dot is displayed with the tab name.

Encryption

The **Encryption** tab allows you to require encrypted connections with the host.

Parameter	Description
Connections to host	How Zabbix server or proxy connects to Zabbix agent on a host: no encryption (default), using PSK (pre-shared key) or certificate.
Connections from host	Select what type of connections are allowed from the host (i.e. from Zabbix agent and Zabbix sender). Several connection types can be selected at the same time (useful for testing and switching to other connection type). Default is "No encryption".
Issuer	Allowed issuer of certificate. Certificate is first validated with CA (certificate authority). If it is valid, signed by the CA, then the *Issuer* field can be used to further restrict allowed CA. This field is intended to be used if your Zabbix installation uses certificates from multiple CAs. If this field is empty then any CA is accepted.
Subject	Allowed subject of certificate. Certificate is first validated with CA. If it is valid, signed by the CA, then the *Subject* field can be used to allow only one value of *Subject* string. If this field is empty then any valid certificate signed by the configured CA is accepted.
PSK identity	Pre-shared key identity string. Do not put sensitive information in the PSK identity, it is transmitted unencrypted over the network to inform a receiver which PSK to use.
PSK	Pre-shared key (hex-string). Maximum length: 512 hex-digits (256-byte PSK) if Zabbix uses GnuTLS or OpenSSL library, 64 hex-digits (32-byte PSK) if Zabbix uses mbed TLS (PolarSSL) library. Example: 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952

Value mapping

The **Value mapping** tab allows to configure human-friendly representation of item data in value mappings.

2 Configuring a host group

Overview

Host groups are used for the logical grouping of hosts and assigning user permissions to them.

Each host must have at least one host group assigned. A host may belong to multiple host groups, and each host group may contain multiple hosts.

Note that in Zabbix, all permissions are based on groups: user groups, host groups, and template groups. So, even if a single user needs access to a single host, it is granted by adding the user to a user group that has permission to access the host group containing that host.

Configuration

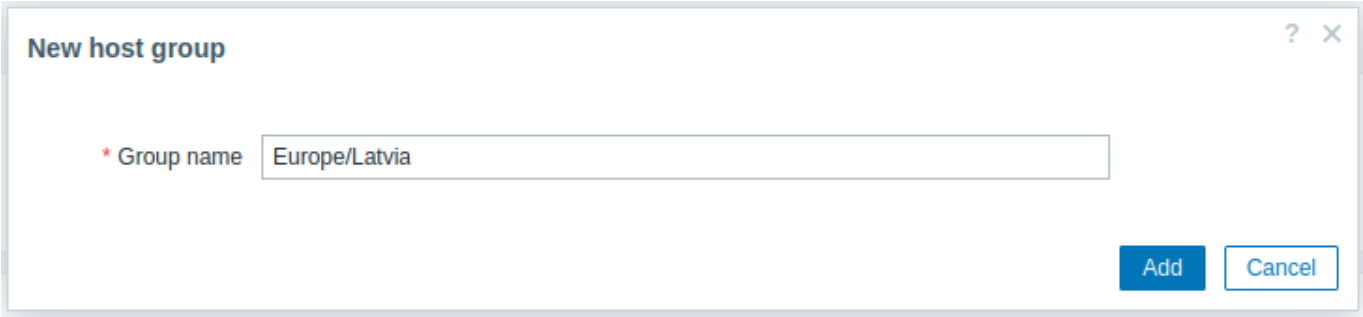
> **Attention:**
> Only Super admin users can create host groups.

There are two options of creating a host group in Zabbix frontend.

Option one:

- Go to: *Data collection → Host groups*
- Click on *Create host group* in the upper right corner of the screen

- Enter the group name in the form



Option two: when configuring a host, enter a non-existing group name in the *Host groups* input field.

Once the host group is created, you can click on the group name in the list under *Data collection* → *Host groups* to edit the group name, clone the group, or delete the group.

Deleting a host group only deletes the logical group, not the hosts in the group. It is not possible to delete a host group that is the only group for any existing host.

Creating host subgroups

A host subgroup (or nested host group) is a child of the parent host group that contains it.

A subgroup is created by using the forward slash '/' in the group name input field to denote its relation to the parent group(s). For example:

- inputting `Europe/Latvia` creates the `Europe/Latvia` subgroup of the parent group `Europe`.
- inputting `Europe/Latvia/Riga/Zabbix servers` creates the respective subgroup within the nested parent groups `Europe`, `Europe/Latvia`, `Europe/Latvia/Riga`.

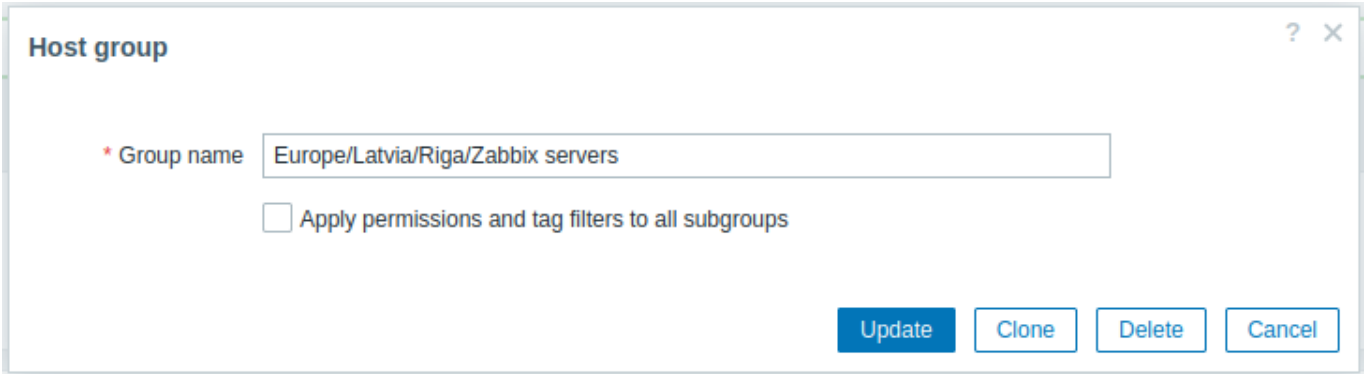
When creating a subgroup, using leading or trailing slashes, or several slashes in a row is not allowed. Escaping of '/' is not supported.

It is not required to create any parent host group(s) before creating a subgroup. You can choose whether to start by creating a subgroup (for example, `Europe/Latvia`) or any parent host group(s) (in our example, `Europe`). If you start by creating a subgroup, parent host group(s) will **not** be created automatically.

Permissions to host groups

- When creating a subgroup to an existing parent host group (for example, creating `Europe/Latvia` when `Europe` already exists), user group permissions to the subgroup are inherited from the parent.
- When creating a parent host group to an existing subgroup (for example, creating `Europe` when `Europe/Latvia` already exists), no permissions to the parent are set.

When editing any host group, you can also set an additional option, *Apply permissions and tag filters to all subgroups*.



Marking this checkbox and clicking on *Update* will apply the same level of permissions and tag filters to all current and future subgroups of the host group being edited.

So, if any user groups have been given varying permissions to the subgroups of the host group being edited, marking the checkbox will grant all current and future subgroups the same user permissions and tag-based permissions as the group being edited.

Note that this option is not saved in the database and will override existing permissions. Any changes made through this option can be reverted only manually.

3 Inventory

Overview

You can keep the inventory of networked devices in Zabbix.

There is a special *Inventory* menu in the Zabbix frontend. However, you will not see any data there initially and it is not where you enter data. Building inventory data is done manually when configuring a host or automatically by using some automatic population options.

Building inventory

Manual mode

When configuring a host, in the *Inventory* tab you can enter such details as the type of device, serial number, location, responsible person, etc - data that will populate inventory information.

If a URL is included in host inventory information and it starts with 'http' or 'https', it will result in a clickable link in the *Inventory* section.

Automatic mode

Host inventory can also be populated automatically. For that to work, when configuring a host the inventory mode in the *Inventory* tab must be set to *Automatic*.

Then you can configure host items to populate any host inventory field with their value, indicating the destination field with the respective attribute (called *Item will populate host inventory field*) in item configuration.

Items that are especially useful for automated inventory data collection:

- system.hw.chassis[full|type|vendor|model|serial] - default is [full], root permissions needed
- system.hw.cpu[all|cpunum,full|maxfreq|vendor|model|curfreq] - default is [all,full]
- system.hw.devices[pci|usb] - default is [pci]
- system.hw.macaddr[interface,short|full] - default is [all,full], interface is regexp
- system.sw.arch
- system.sw.os[name|short|full] - default is [name]
- system.sw.packages[regexp,manager,short|full] - default is [all,all,full]

Inventory mode selection

Inventory mode can be selected in the host configuration form.

Inventory mode by default for new hosts is selected based on the *Default host inventory mode* setting in *Administration* → *General* → *Other*.

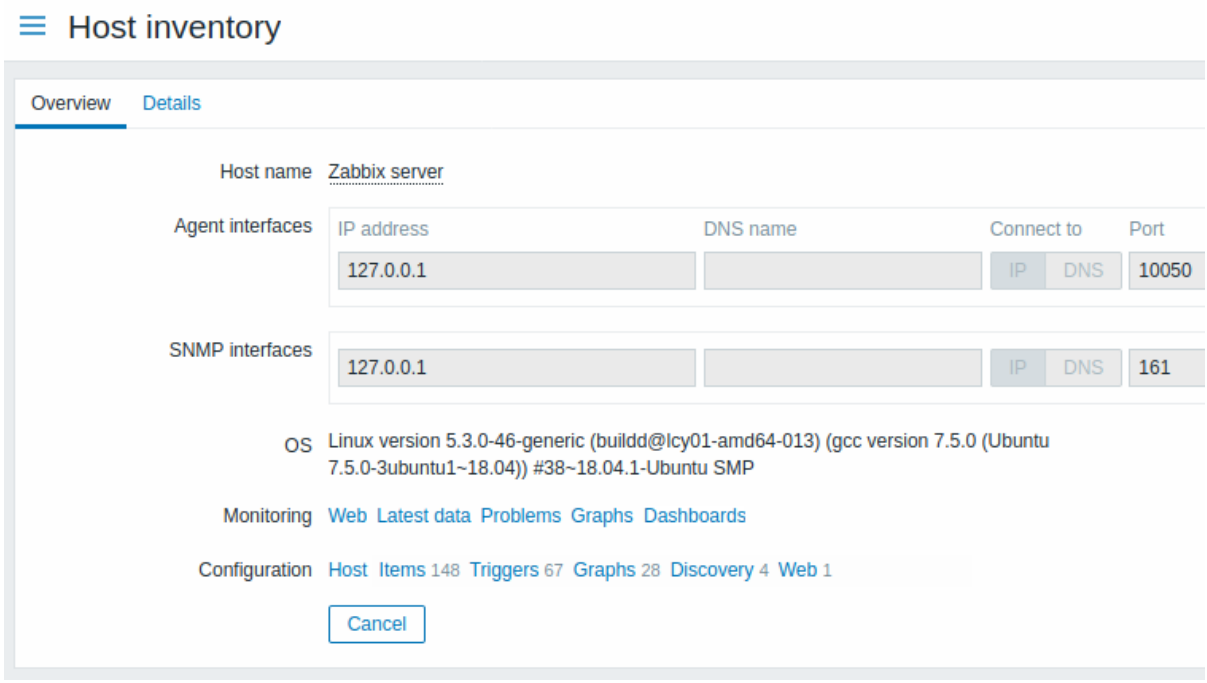
For hosts added by network discovery or autoregistration actions, it is possible to define a *Set host inventory mode* operation selecting manual or automatic mode. This operation overrides the *Default host inventory mode* setting.

Inventory overview

The details of all existing inventory data are available in the *Inventory* menu.

In *Inventory* → *Overview* you can get a host count by various fields of the inventory.

In *Inventory* → *Hosts* you can see all hosts that have inventory information. Clicking on the host name will reveal the inventory details in a form.



The **Overview** tab shows:

Parameter	Description
Host name	Name of the host. Clicking on the name opens a menu with the scripts defined for the host. Host name is displayed with an orange icon, if the host is in maintenance.
Visible name	Visible name of the host (if defined).
Host (Agent, SNMP, JMX, IPMI) interfaces	This block provides details of the interfaces configured for the host.
OS	Operating system inventory field of the host (if defined).
Hardware	Host hardware inventory field (if defined).
Software	Host software inventory field (if defined).
Description	Host description.
Monitoring	Links to monitoring sections with data for this host: *Web*, *Latest data*, *Problems*, *Graphs*, *Dashboards*.
Configuration	Links to configuration sections for this host: *Host*, *Items*, *Triggers*, *Graphs*, *Discovery*, *Web*. The amount of configured entities is listed after each link.

The **Details** tab shows all inventory fields that are populated (are not empty).

Inventory macros

There are host inventory macros {INVENTORY.*} available for use in notifications, for example:

"Server in {INVENTORY.LOCATION1} has a problem, responsible person is {INVENTORY.CONTACT1}, phone number {INVENTORY.POC.PRIMARY.PHONE.A1}."

For more details, see the supported macro page.

4 Mass update

Overview

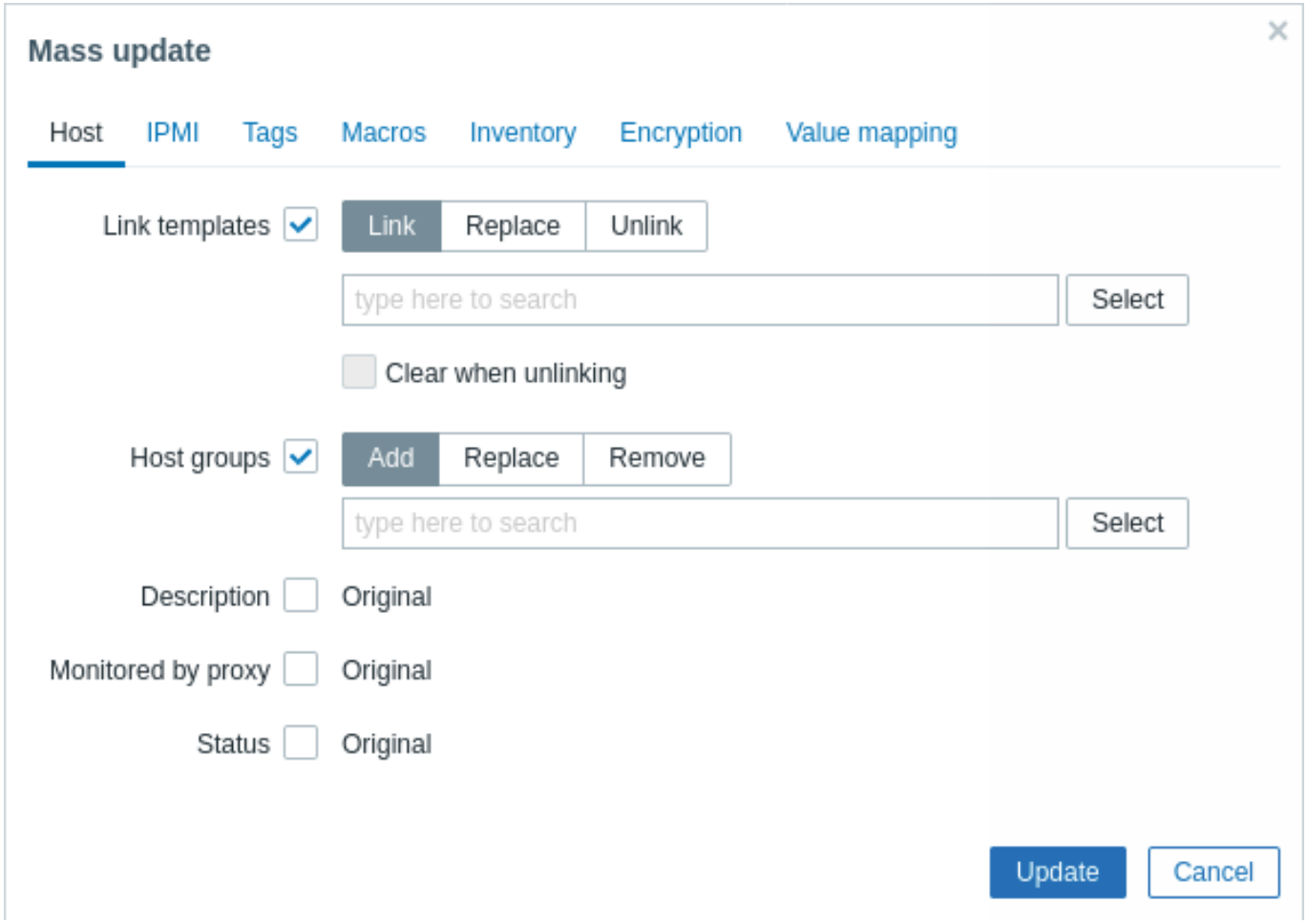
Sometimes you may want to change some attribute for a number of hosts at once. Instead of opening each individual host for editing, you may use the mass update function for that.

Using mass update

To mass-update some hosts, do the following:

- Mark the checkboxes before the hosts you want to update in the host list
- Click on *Mass update* below the list
- Navigate to the tab with required attributes (*Host*, *IPMI*, *Tags*, *Macros*, *Inventory*, *Encryption* or *Value mapping*)

- Mark the checkboxes of any attribute to update and enter a new value for them



The following options are available when selecting the respective button for **template** linkage update:

- *Link* - specify which additional templates to link
- *Replace* - specify which templates to link while unlinking any template that was linked to the hosts before
- *Unlink* - specify which templates to unlink

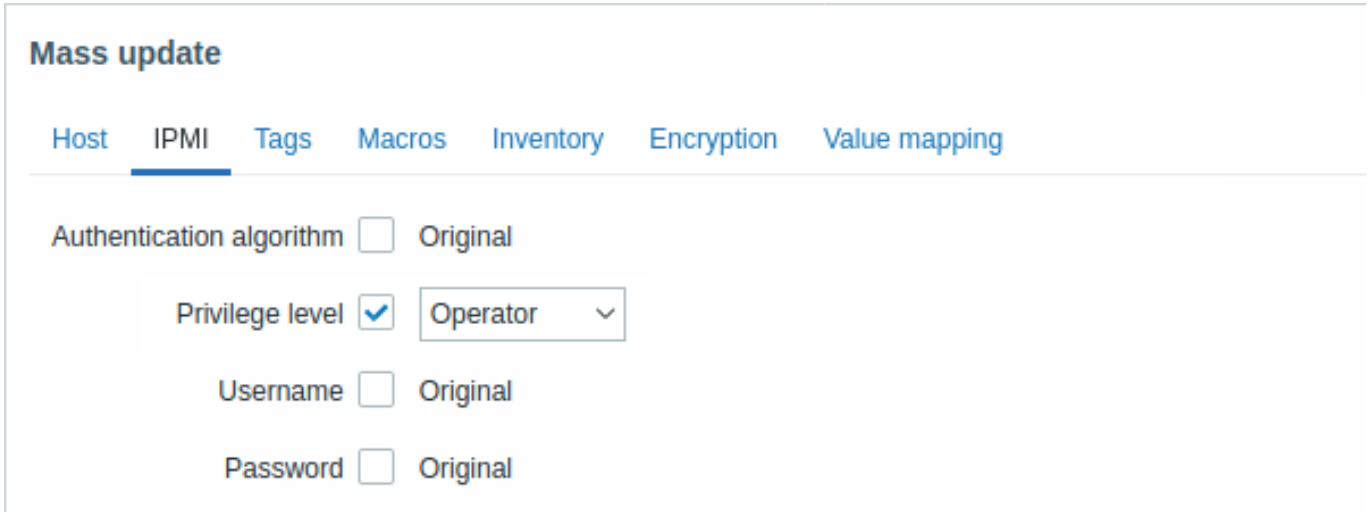
To specify the templates to link/unlink start typing the template name in the auto-complete field until a dropdown appears offering the matching templates. Just scroll down to select the required template.

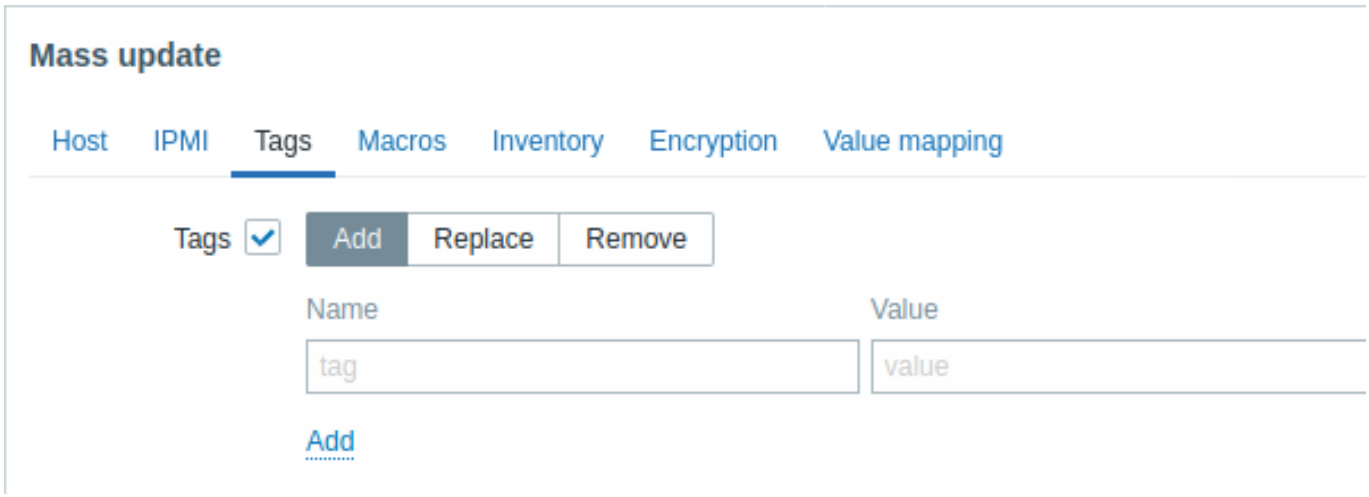
The *Clear when unlinking* option will allow to not only unlink any previously linked templates, but also remove all elements inherited from them (items, triggers, etc.).

The following options are available when selecting the respective button for **host group** update:

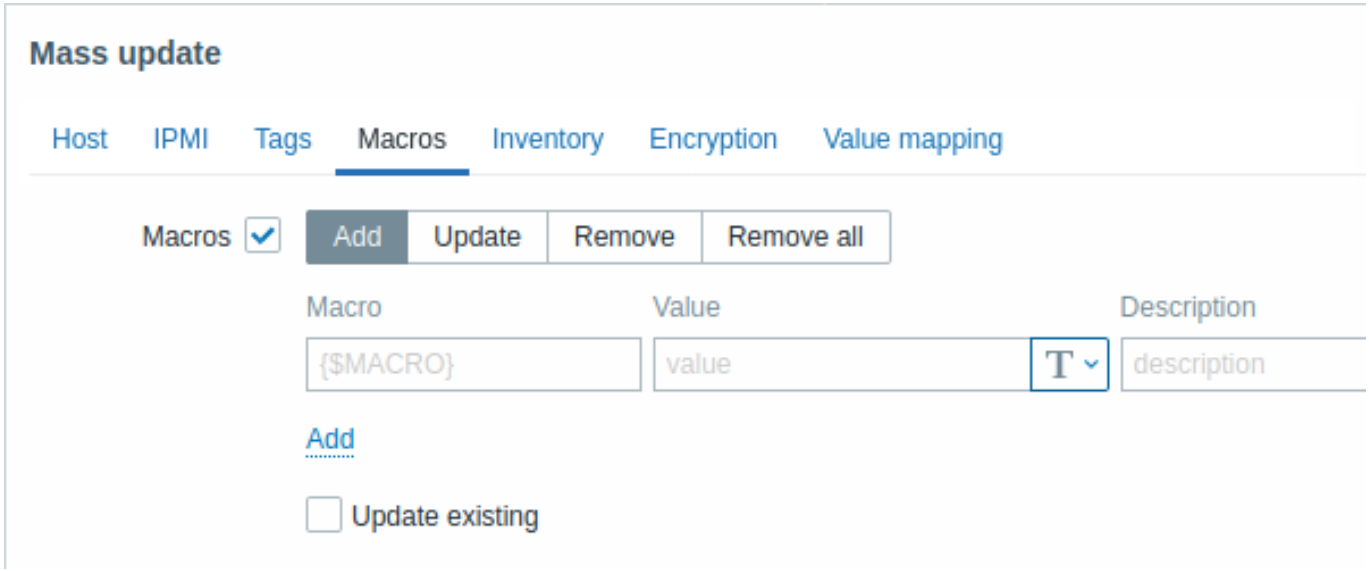
- *Add* - allows to specify additional host groups from the existing ones or enter completely new host groups for the hosts
- *Replace* - will remove the host from any existing host groups and replace them with the one(s) specified in this field (existing or new host groups)
- *Remove* - will remove specific host groups from hosts

These fields are auto-complete - starting to type in them offers a dropdown of matching host groups. If the host group is new, it also appears in the dropdown and it is indicated by *(new)* after the string. Just scroll down to select.





User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags. Note that tags with the same name but different values are not considered 'duplicates' and can be added to the same host.

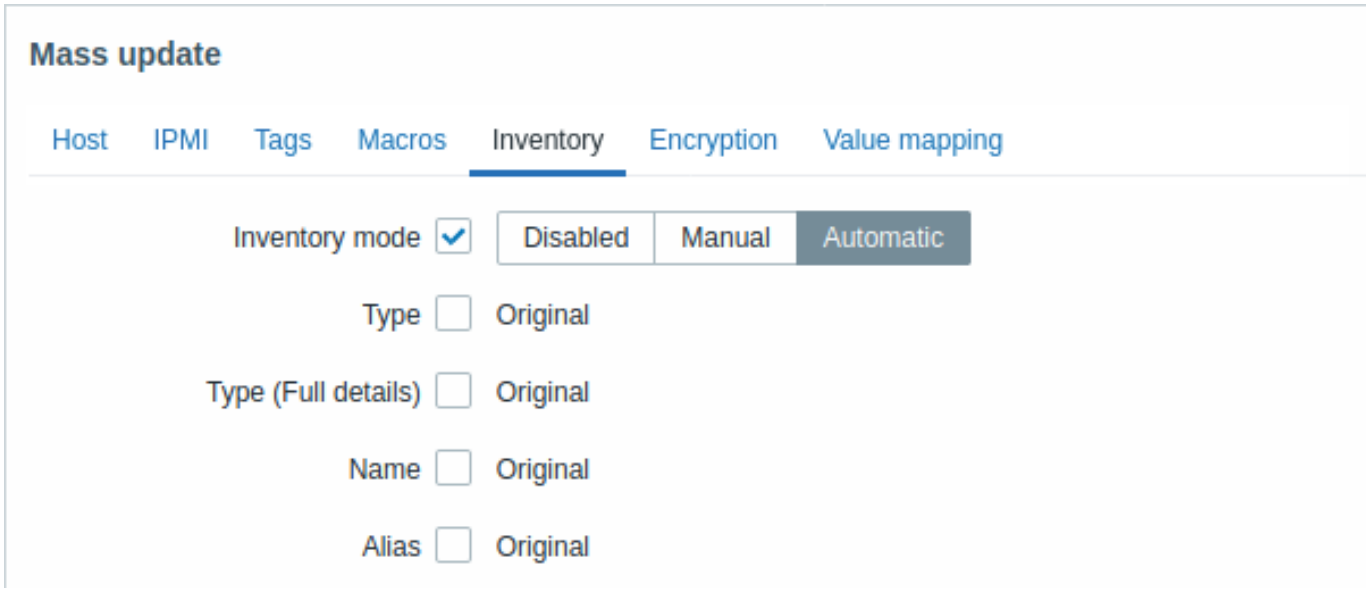


The following options are available when selecting the respective button for macros update:

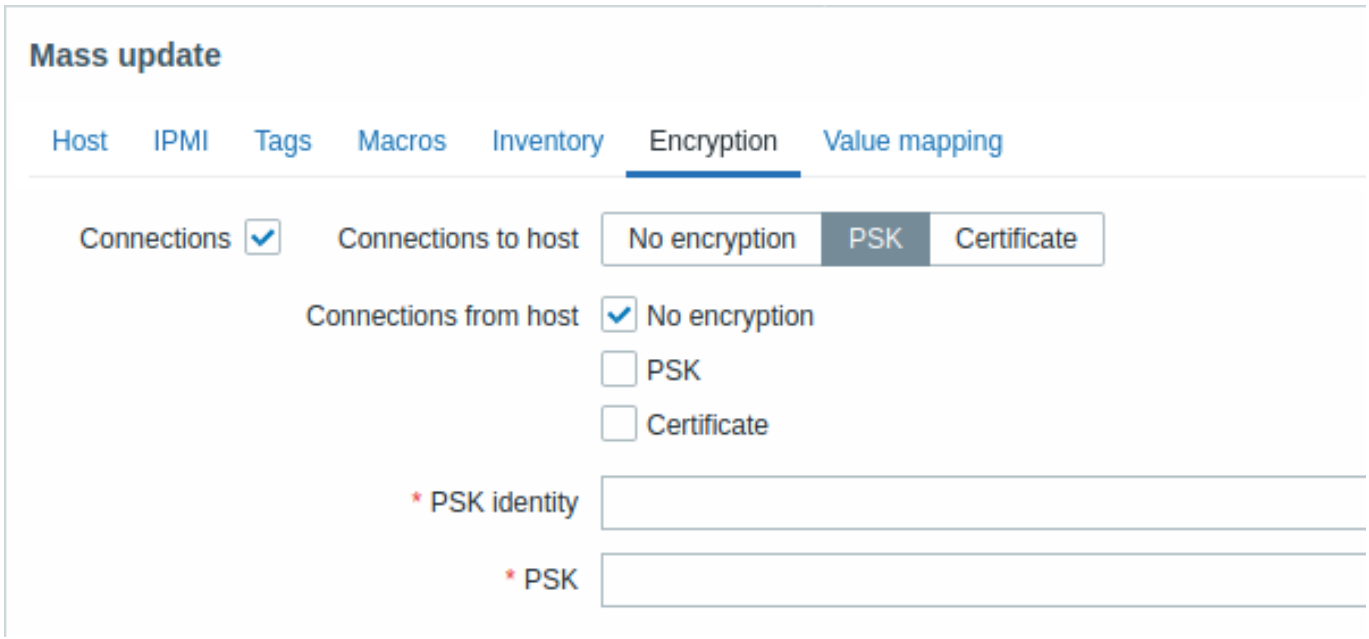
- *Add* - allows to specify additional user macros for the hosts. If *Update existing* checkbox is checked, value, type and description for the specified macro name will be updated. If unchecked, if a macro with that name already exist on the host(s), it will not be updated.
- *Update* - will replace values, types and descriptions of macros specified in this list. If *Add missing* checkbox is checked, macro that didn't previously exist on a host will be added as new macro. If unchecked, only macros that already exist on a host will be updated.
- *Remove* - will remove specified macros from hosts. If *Except selected* box is checked, all macros except specified in the list

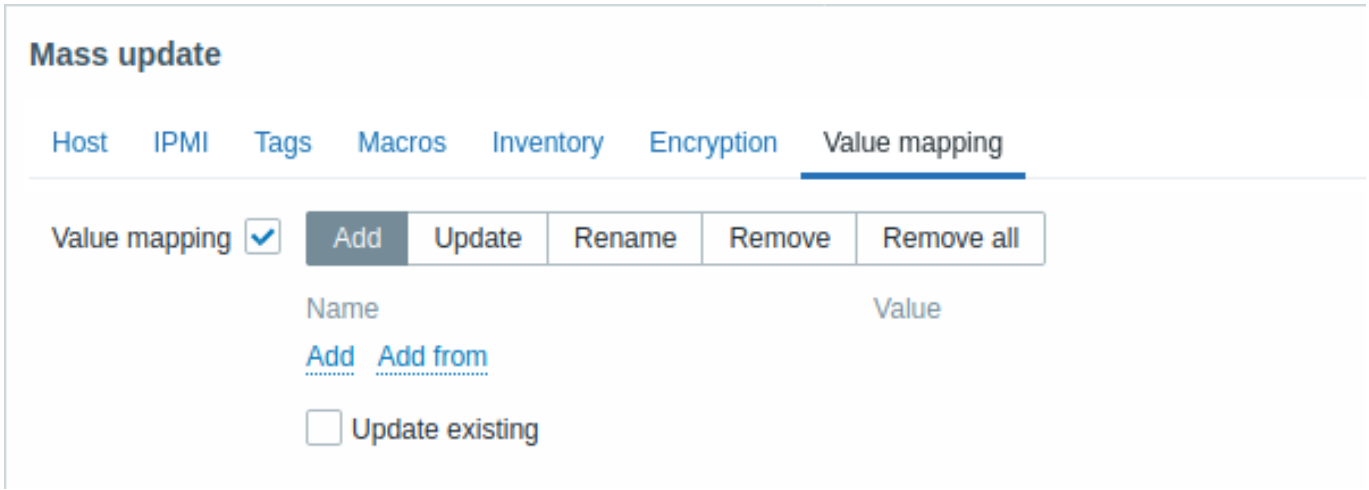
will be removed. If unchecked, only macros specified in the list will be removed.

- *Remove all* - will remove all user macros from hosts. If *I confirm to remove all macros* checkbox is not checked, a new popup window will open asking to confirm removal of all macros.



To be able to mass update inventory fields, the *Inventory mode* should be set to 'Manual' or 'Automatic'.





Buttons with the following options are available for value map update:

- *Add* - add value maps to the hosts. If you mark *Update existing*, all properties of the value map with this name will be

updated. Otherwise, if a value map with that name already exists, it will not be updated.

- *Update* - update existing value maps. If you mark *Add missing*, a value map that didn't previously exist on a host will be added as a new value map. Otherwise only the value maps that already exist on a host will be updated.
- *Rename* - give new name to an existing value map
- *Remove* - remove the specified value maps from the hosts. If you mark *Except selected*, all value maps will be removed **except** the ones that are specified.
- *Remove all* - remove all value maps from the hosts. If the *I confirm to remove all value maps* checkbox is not marked, a new popup window will open asking to confirm the removal.

When done with all required changes, click on *Update*. The attributes will be updated accordingly for all the selected hosts.

2 Items

Overview

An item is an individual metric.

Items are used for collecting data. Once you have configured a host, you must add items to get actual data. One way of quickly adding many items is to attach one of the predefined templates to a host. However, for optimized system performance, you may need to fine-tune the templates to have as many items and as frequent monitoring as necessary.

To specify what sort of data to collect from a host, use the item key. For example, an item with the key name **system.cpu.load** will collect processor load data, while an item with the key name **net.if.in** will collect incoming traffic information.

Additional parameters can be specified in square brackets after the key name. For example, system.cpu.load**[avg5]** will return the processor load average for the last 5 minutes, while net.if.in**[eth0]** will show incoming traffic in the interface "eth0".

> **Note:**
> See individual sections of item types for all supported item types and item keys.

Proceed to creating and configuring an item.

1 Creating an item

Overview

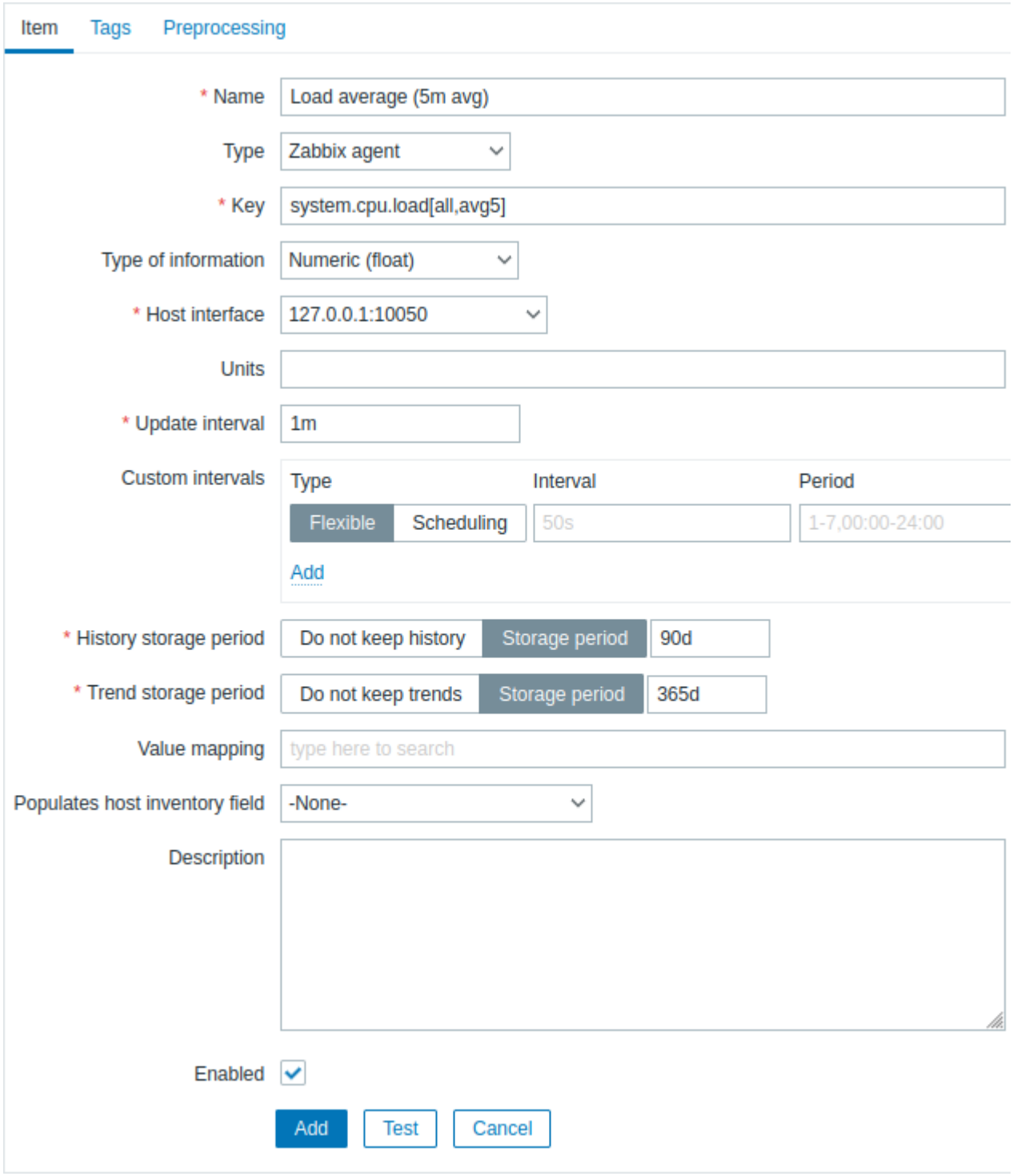
To create an item in Zabbix frontend, do the following:

- Go to: *Data collection → Hosts*
- Click on *Items* in the row of the host
- Click on *Create item* in the upper right corner of the screen
- Enter parameters of the item in the form

You can also create an item by opening an existing one, pressing the *Clone* button and then saving under a different name.

Configuration

The **Item** tab contains general item attributes.



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Item name.
Type	Item type. See individual item type sections.
Key	Item key (up to 2048 characters). The supported item keys can be found in individual item type sections. The key must be unique within a single host. If key type is 'Zabbix agent', 'Zabbix agent (active)' or 'Simple check', the key value must be supported by Zabbix agent or Zabbix server. See also: the correct key format.

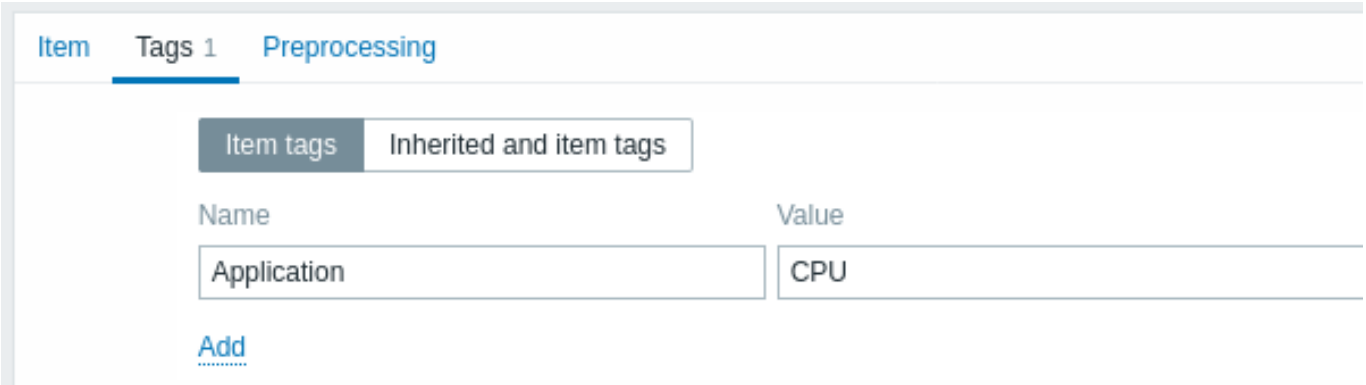
Parameter	Description
Type of information	Type of data as stored in the database after performing conversions, if any. **Numeric (unsigned)** - 64-bit unsigned integer **Numeric (float)** - 64-bit floating point number This type will allow precision of approximately 15 digits and range from approximately -1.79E+308 to 1.79E+308 (with exception of PostgreSQL 11 and earlier versions). Receiving values in scientific notation is also supported. E.g., 1.23E+7, 1e308, 1.1E-4. **Character** - short text data **Log** - long text data with optional log related properties (timestamp, source, severity, logeventid) **Text** - long text data. See also text data limits. For item keys that return data only in one specific format, matching type of information is selected automatically.
Host interface	Select the host interface. This field is available when editing an item on the host level.
Units	If a unit symbol is set, Zabbix will add postprocessing to the received value and display it with the set unit postfix. By default, if the raw value exceeds 1000, it is divided by 1000 and displayed accordingly. For example, if you set *bps* and receive a value of 881764, it will be displayed as 881.76 Kbps. The JEDEC memory standard is used for processing **B** (byte), **Bps** (bytes per second) units, which are divided by 1024. Thus, if units are set to **B** or **Bps** Zabbix will display: 1 as 1B/1Bps 1024 as 1KB/1KBps 1536 as 1.5KB/1.5KBps Special processing is used if the following time-related units are used: **unixtime** - translated to "yyyy.mm.dd hh:mm:ss". To translate correctly, the received value must be a *Numeric (unsigned)* type of information. **uptime** - translated to "hh:mm:ss" or "N days, hh:mm:ss" For example, if you receive the value as 881764 (seconds), it will be displayed as "10 days, 04:56:04" **s** - translated to "yyy mmm ddd hhh mmm sss ms"; parameter is treated as number of seconds. For example, if you receive the value as 881764 (seconds), it will be displayed as "10d 4h 56m" Only 3 upper major units are shown, like "1m 15d 5h" or "2h 4m 46s". If there are no days to display, only two levels are displayed - "1m 5h" (no minutes, seconds or milliseconds are shown). Will be translated to "< 1 ms" if the value is less than 0.001. *Note* that if a unit is prefixed with `!`, then no unit prefixes/processing is applied to item values. See unit conversion.
Update interval	Retrieve a new value for this item every N seconds. Maximum allowed update interval is 86400 seconds (1 day). Time suffixes are supported, e.g., 30s, 1m, 2h, 1d. User macros are supported. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. *Note*: The update interval can only be set to '0' if custom intervals exist with a non-zero value. If set to '0', and a custom interval (flexible or scheduled) exists with a non-zero value, the item will be polled during the custom interval duration. *Note* that the first item poll after the item became active or after update interval change might occur earlier than the configured value. New items will be checked within 60 seconds of their creation, unless they have Scheduling or Flexible update interval and the *Update interval* is set to 0. An existing passive item can be polled for value immediately by pushing the *Execute now* button.
Custom intervals	You can create custom rules for checking the item: **Flexible** - create an exception to the *Update interval* (interval with different frequency) **Scheduling** - create a custom polling schedule. For detailed information see Custom intervals. Time suffixes are supported in the *Interval* field, e.g., 30s, 1m, 2h, 1d. User macros are supported. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Scheduling is supported since Zabbix 3.0.0. *Note*: custom intervals for active checks are supported by Zabbix agent 2 only.

Parameter	Description
History storage period	Select either: **Do not keep history** - item history is not stored. Useful for master items if only dependent items need to keep history. This setting cannot be overridden by global housekeeper settings. **Storage period** - specify the duration of keeping detailed history in the database (1 hour to 25 years). Older data will be removed by the housekeeper. Stored in seconds. Time suffixes are supported, e.g., 2h, 1d. User macros are supported. The *Storage period* value can be overridden globally in *Administration* → *Housekeeping*. If a global overriding setting exists, a green  info icon is displayed. If you position your mouse on it, a warning message is displayed, e.g., *Overridden by global housekeeper settings (1d)*. It is recommended to keep the recorded values for the smallest possible time to reduce the size of value history in the database. Instead of keeping a long history of values, you can keep longer data of trends. See also History and trends.
Trend storage period	Select either: **Do not keep trends** - trends are not stored. This setting cannot be overridden by global housekeeper settings. **Storage period** - specify the duration of keeping aggregated (hourly min, max, avg, count) history in the database (1 day to 25 years). Older data will be removed by the housekeeper. Stored in seconds. Time suffixes are supported, e.g., 24h, 1d. User macros are supported. The *Storage period* value can be overridden globally in *Administration* → *Housekeeping*. If a global overriding setting exists, a green  info icon is displayed. If you position your mouse on it, a warning message is displayed, e.g., *Overridden by global housekeeper settings (7d)*. *Note:* Keeping trends is not available for non-numeric data - character, log and text. See also History and trends.
Value mapping	Apply value mapping to this item. Value mapping does not change received values, it is for displaying data only. It works with *Numeric(unsigned)*, *Numeric(float)* and *Character* items. For example, ”Windows service states”.
Log time format	Available for items of type **Log** only. Supported placeholders: * **y**: *Year (1970-2038)* * **M**: *Month (01-12)* * **d**: *Day (01-31)* * **h**: *Hour (00-23)* * **m**: *Minute (00-59)* * **s**: *Second (00-59)* If left blank, the timestamp will be set to 0 in Unix time, representing January 1, 1970. For example, consider the following line from the Zabbix agent log file: ” 23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211).” It begins with six character positions for PID, followed by date, time, and the rest of the message. The log time format for this line would be ”ppppp:yyyyMMdd:hhmmss”. Note that ”p” and ”:” characters are placeholders and can be any characters except ”yMdhms”.
Populates host inventory field	You can select a host inventory field that the value of item will populate. This will work if automatic inventory population is enabled for the host. This field is not available if *Type of information* is set to ’Log’.
Description	Enter an item description. User macros are supported.
Enabled	Mark the checkbox to enable the item so it will be processed.
Latest data	Click on the link to view the latest data for the item. This link is only available when editing an already existing item.

Note:
Item type specific fields are described on corresponding pages.

Note:
When editing an existing template level item on a host level, a number of fields are read-only. You can use the link in the form header and go to the template level and edit them there, keeping in mind that the changes on a template level will change the item for all hosts that the template is linked to.

The **Tags** tab allows to define item-level tags.



Item value preprocessing

The **Preprocessing** tab allows to define transformation rules for the received values.

Testing

Attention:
To perform item testing, ensure that the system time on the server and the proxy is synchronized. In the case when the server time is behind, item testing may return an error message "The task has been expired." Having set different time zones on the server and the proxy, however, won't affect the testing result.

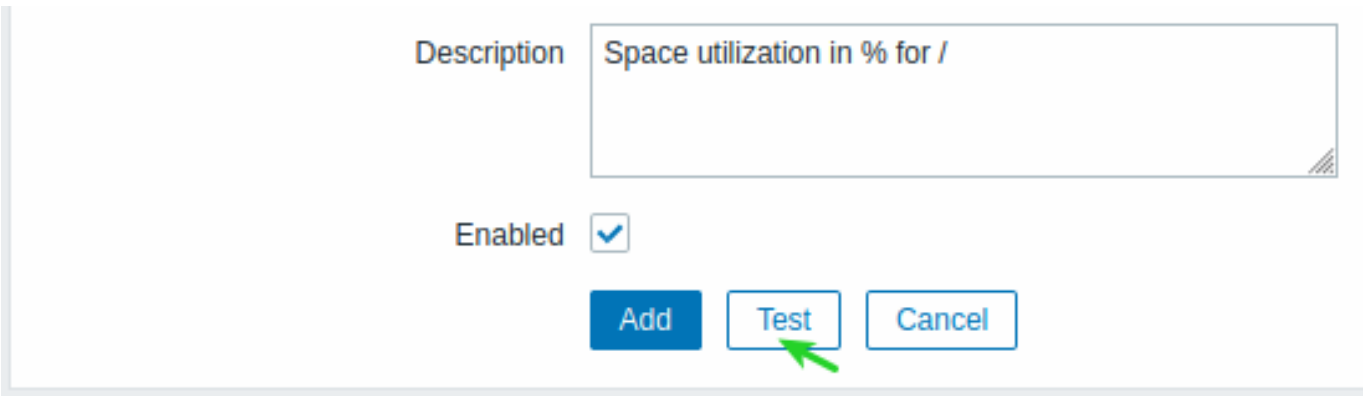
It is possible to test an item and, if configured correctly, get a real value in return. Testing can occur even before an item is saved.

Testing is available for host and template items, item prototypes and low-level discovery rules. Testing is not available for active items.

Item testing is available for the following passive item types:

- Zabbix agent
- SNMP agent (v1, v2, v3)
- IPMI agent
- SSH checks
- Telnet checks
- JMX agent
- Simple checks (except `icmpping*`, `vmware.*` items)
- Zabbix internal
- Calculated items
- External checks
- Database monitor
- HTTP agent
- Script

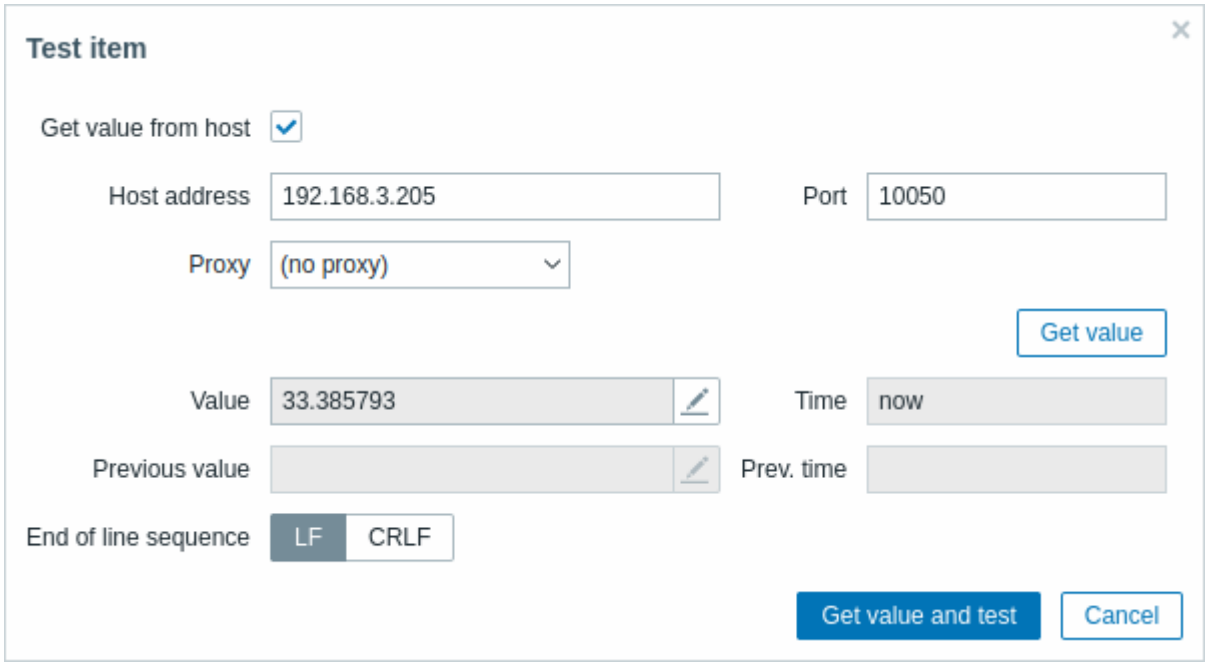
To test an item, click on the *Test* button at the bottom of the item configuration form. Note that the *Test* button will be disabled for items that cannot be tested (like active checks, excluded simple checks).



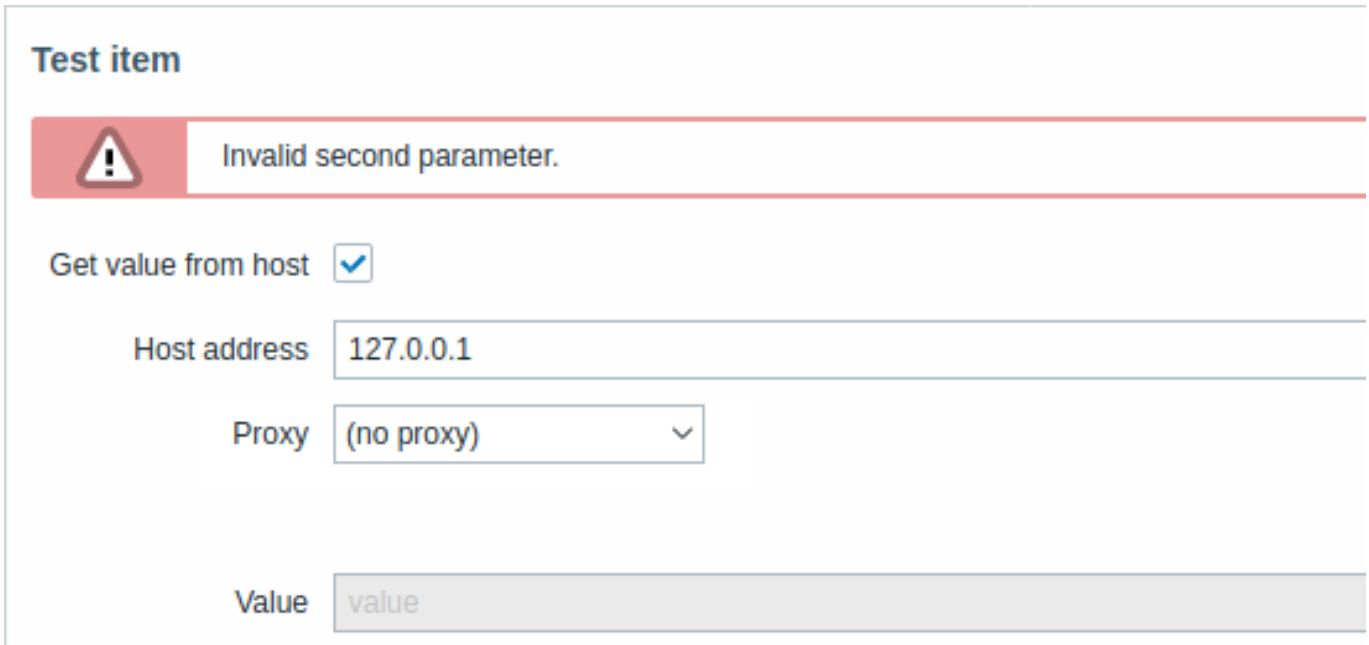
The item testing form has fields for the required host parameters (host address, port, proxy name/no proxy) and item-specific details (such as SNMPv2 community or SNMPv3 security credentials). These fields are context aware:

- The values are pre-filled when possible, i.e., for items requiring an agent, by taking the information from the selected agent interface of the host
- The values have to be filled manually for template items
- Plain-text macro values are resolved
- Fields where the value (or part of the value) is a secret or Vault macro are empty and have to be entered manually. If any item parameter contains a secret macro value, the following warning message is displayed: "Item contains user-defined macros with secret values. Values of these macros should be entered manually."
- The fields are disabled when not needed in the context of the item type (e.g., the host address field and the proxy field are disabled for calculated items)

To test the item, click on *Get value*. If the value is retrieved successfully, it will fill the *Value* field, moving the current value (if any) to the *Previous value* field while also calculating the *Prev. time* field, i.e., the time difference between the two values (clicks) and trying to detect an EOL sequence and switch to CRLF if detecting "\n\r" in retrieved value.



If the configuration is incorrect, an error message is displayed describing the possible cause.



A successfully retrieved value from host can also be used to test preprocessing steps.

Form buttons

Buttons at the bottom of the form allow to perform several operations.

Add	Add an item. This button is only available for new items.
Update	Update the properties of an item.
Clone	Create another item based on the properties of the current item.
Execute now	Execute a check for a new item value immediately. Supported for **passive** checks only (see more details). *Note* that when checking for a value immediately, configuration cache is not updated, thus the value will not reflect very recent changes to item configuration.
Test	Test if item configuration is correct by getting a value.
Clear history and trends	Delete the item history and trends.
Delete	Delete the item.
Cancel	Cancel the editing of item properties.

Unit conversion

By default, specifying a unit for an item results in a multiplier prefix being added - for example, an incoming value '2048' with unit 'B' would be displayed as '2KB'.

To prevent a unit from conversion, use the `!` prefix, for example, `!B`. To better understand how the conversion works with and without the exclamation mark, see the following examples of values and units:

```
1024 !B → 1024 B
1024 B → 1 KB
61 !s → 61 s
61 s → 1m 1s
0 !uptime → 0 uptime
0 uptime → 00:00:00
0 !! → 0 !
0 ! → 0
```

Note:
Before Zabbix 4.0, there was a hardcoded unit stoplist consisting of `ms`, `rpm`, `RPM`, `%`. This stoplist has been deprecated, thus the correct way to prevent converting such units is `!ms`, `!rpm`, `!RPM`, `!%`.

Text data limits

Text data limits depend on the database backend. Before storing text values in the database they get truncated to match the database value type limit:

Database	Type of information		
	Character	**Log**	**Text**
MySQL	255 characters	65536 bytes	65536 bytes
PostgreSQL	255 characters	65536 characters	65536 characters
Oracle	255 characters	65536 characters	65536 characters
SQLite (only Zabbix proxy)	255 characters	65536 characters	65536 characters

Custom script limit

Available custom script length depends on the database used:

Database	Limit in characters	Limit in bytes
MySQL	65535	65535
PostgreSQL	65535	not limited

Database	Limit in characters	Limit in bytes
Oracle	2048	4000
SQLite (only Zabbix proxy)	65535	not limited

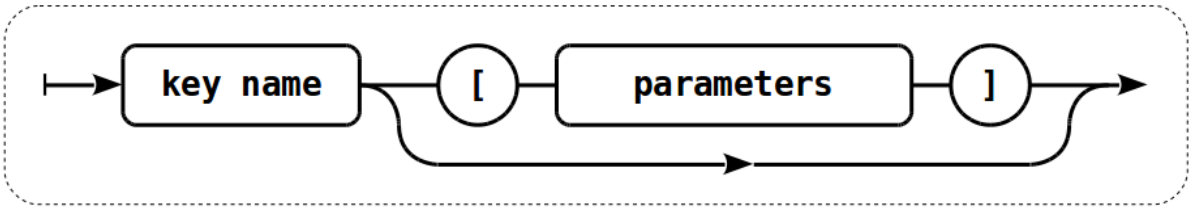
Unsupported items

An item can become unsupported if its value cannot be retrieved for some reason. Such items are still rechecked at their standard *Update interval*.

Unsupported items are reported as having a NOT SUPPORTED state.

1 Item key format

Item key format, including key parameters, must follow syntax rules. The following illustrations depict the supported syntax. Allowed elements and characters at each point can be determined by following the arrows - if some block can be reached through the line, it is allowed, if not - it is not allowed.



To construct a valid item key, one starts with specifying the key name, then there's a choice to either have parameters or not - as depicted by the two lines that could be followed.

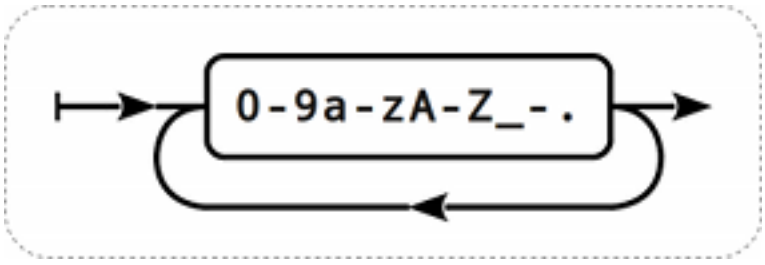
Key name

The key name itself has a limited range of allowed characters, which just follow each other. Allowed characters are:

`0-9a-zA-Z_-.`

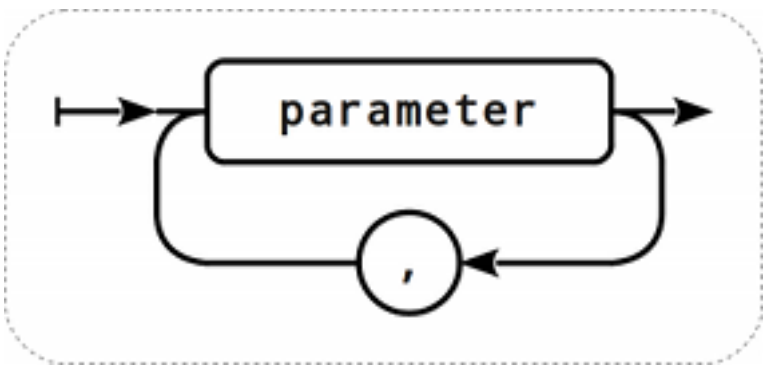
Which means:

- all numbers;
- all lowercase letters;
- all uppercase letters;
- underscore;
- dash;
- dot.

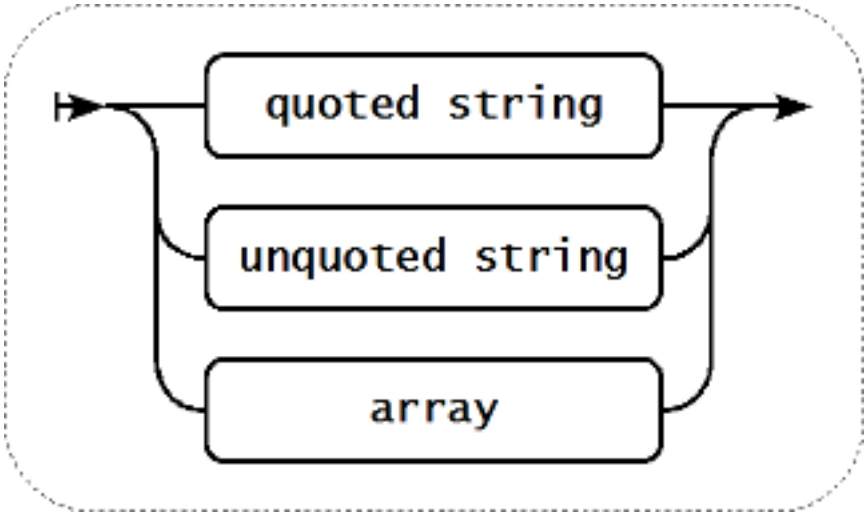


Key parameters

An item key can have multiple parameters that are comma separated.



Each key parameter can be either a quoted string, an unquoted string or an array.

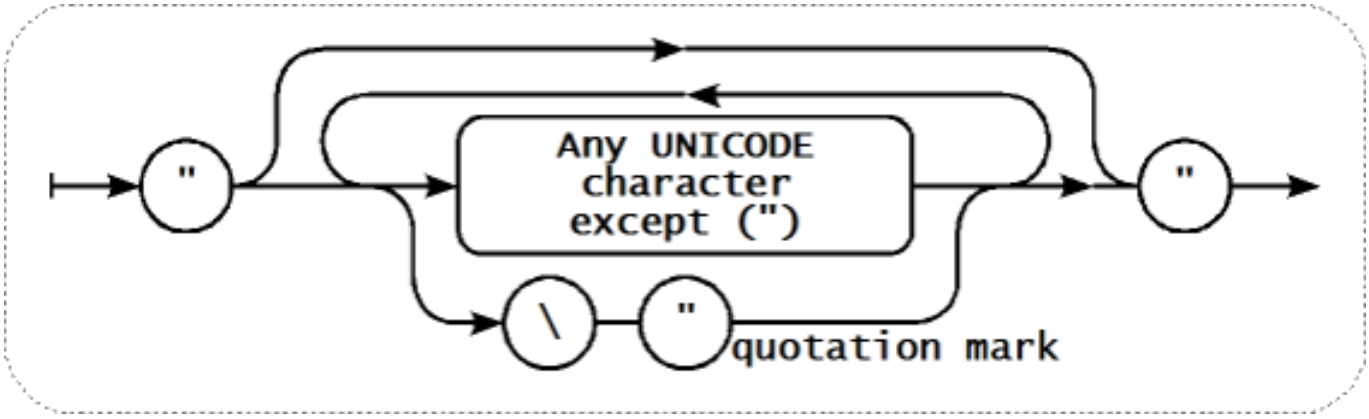


The parameter can also be left empty, thus using the default value. In that case, the appropriate number of commas must be added if any further parameters are specified. For example, item key **icmpping[„200„500]** would specify that the interval between individual pings is 200 milliseconds, timeout - 500 milliseconds, and all other parameters are left at their defaults.

It is possible to include macros in the parameters. Those can be user macros or some of the built-in macros. To see what particular built-in macros are supported in item key parameters, search the page Supported macros for ”item key parameters”.

Parameter - quoted string

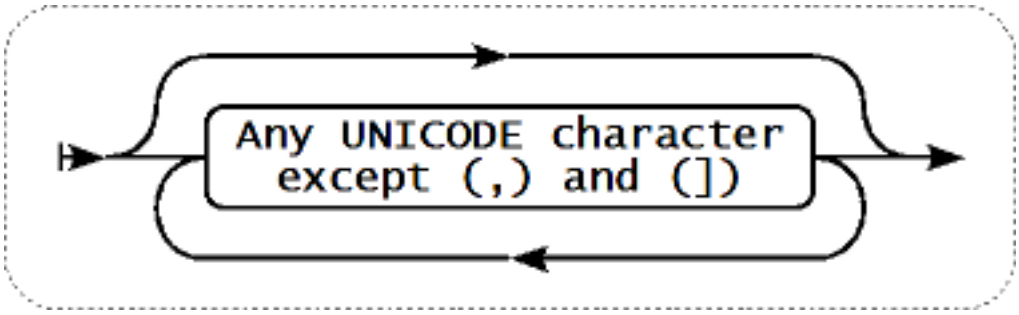
If the key parameter is a quoted string, any Unicode character is allowed. If the key parameter string contains a quotation mark, this parameter has to be quoted, and each quotation mark which is a part of the parameter string has to be escaped with a backslash (\) character. If the key parameter string contains comma, this parameter has to be quoted.



Warning:
To quote item key parameters, use double quotes only. Single quotes are not supported.

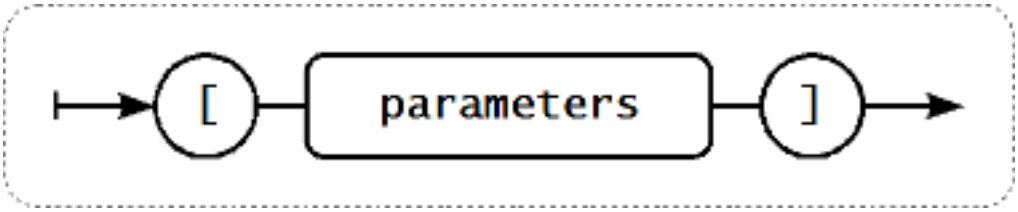
Parameter - unquoted string

If the key parameter is an unquoted string, any Unicode character is allowed except comma and right square bracket (]). Unquoted parameter cannot start with left square bracket ([).



Parameter - array

If the key parameter is an array, it is again enclosed in square brackets, where individual parameters come in line with the rules and syntax of specifying multiple parameters.



Attention:
Multi-level parameter arrays, e.g. `[a,[b,[c,d]],e]`, are not allowed.

2 Custom intervals

Overview

It is possible to create custom rules regarding the times when an item is checked. The two methods for that are *Flexible intervals*, which allow to redefine the default update interval, and *Scheduling*, whereby an item check can be executed at a specific time or sequence of times.

Note:
Zabbix agent 2 supports custom intervals for both passive and active checks, whereas Zabbix agent supports custom intervals only for passive checks. See Zabbix agent vs agent 2 comparison.

Flexible intervals

Flexible intervals allow to redefine the default update interval for specific time periods. A flexible interval is defined with *Interval* and *Period* where:

- *Interval* – the update interval for the specified time period
- *Period* – the time period when the flexible interval is active (see the time periods for detailed description of the *Period* format)

If multiple flexible intervals overlap, the smallest *Interval* value is used for the overlapping period. Note that if the smallest value of overlapping flexible intervals is '0', no polling will take place. Outside the flexible intervals the default update interval is used.

Note that if the flexible interval equals the length of the period, the item will be checked exactly once. If the flexible interval is greater than the period, the item might be checked once or it might not be checked at all (thus such configuration is not advisable). If the flexible interval is less than the period, the item will be checked at least once.

If the flexible interval is set to '0', the item is not polled during the flexible interval period and resumes polling according to the default *Update interval* once the period is over. Examples:

Interval	Period	Description
10	1-5,09:00-18:00	Item will be checked every 10 seconds during working hours.
0	1-7,00:00-7:00	Item will not be checked during the night.
0	7-7,00:00-24:00	Item will not be checked on Sundays.
60	1-7,12:00-12:01	Item will be checked at 12:00 every day. Note that this was used as a workaround for scheduled checks and starting with Zabbix 3.0 it is recommended to use scheduling intervals for such checks.

Scheduling intervals

Scheduling intervals are used to check items at specific times. While flexible intervals are designed to redefine the default item update interval, the scheduling intervals are used to specify an independent checking schedule, which is executed in parallel.

A scheduling interval is defined as: `md<filter>wd<filter>h<filter>m<filter>s<filter>` where:

- **md** - month days
- **wd** - week days
- **h** - hours
- **m** - minutes
- **s** – seconds

<filter> is used to specify values for its prefix (days, hours, minutes, seconds) and is defined as: [<from>[-<to>]][/<step>][,<filter
where:

- <from> and <to> define the range of matching values (included). If <to> is omitted then the filter matches a <from> - <from> range. If <from> is also omitted then the filter matches all possible values.
- <step> defines the skips of the number value through the range. By default <step> has the value of 1, which means that all values of the defined range are matched.

While the filter definitions are optional, at least one filter must be used. A filter must either have a range or the *<step>* value defined.

An empty filter matches either '0' if no lower-level filter is defined or all possible values otherwise. For example, if the hour filter is omitted then only '0' hour will match, provided minute and seconds filters are omitted too, otherwise an empty hour filter will match all hour values.

Valid <from> and <to> values for their respective filter prefix are:

Prefix	Description	*<from>*	*<to>*
md	Month days	1-31	1-31
wd	Week days	1-7	1-7
h	Hours	0-23	0-23
m	Minutes	0-59	0-59
s	Seconds	0-59	0-59

The <from> value must be less or equal to <to> value. The <step> value must be greater or equal to 1 and less or equal to <to> - <from>.

Single digit month days, hours, minutes and seconds values can be prefixed with 0. For example md01-31 and h/02 are valid intervals, but md01-031 and wd01-07 are not.

In Zabbix frontend, multiple scheduling intervals are entered in separate rows. In Zabbix API, they are concatenated into a single string with a semicolon ; as a separator.

If a time is matched by several intervals it is executed only once. For example, wd1h9;h9 will be executed only once on Monday at 9am.

Examples:

Interval	Will be executed
m0-59	every minute
h9-17/2	every 2 hours starting with 9:00 (9:00, 11:00 ...)
m0,30 or m/30	hourly at hh:00 and hh:30
m0,5,10,15,20,25,30,35,40,45,50,55 or m/5	every five minutes
wd1-5h9	every Monday till Friday at 9:00
wd1-5h9-18	every Monday till Friday at 9:00,10:00,...,18:00
h9,10,11 or h9-11	every day at 9:00, 10:00 and 11:00
md1h9m30	every 1st day of each month at 9:30
md1wd1h9m30	every 1st day of each month at 9:30 if it is Monday
h9m/30	every day at 9:00, 9:30
h9m0-59/30	every day at 9:00, 9:30
h9,10m/30	every day at 9:00, 9:30, 10:00, 10:30
h9-10m30	every day at 9:30, 10:30
h9m10-40/30	every day at 9:10, 9:40
h9,10m10-40/30	every day at 9:10, 9:40, 10:10, 10:40
h9-10m10-40/30	every day at 9:10, 9:40, 10:10, 10:40
h9m10-40	every day at 9:10, 9:11, 9:12, ... 9:40
h9m10-40/1	every day at 9:10, 9:11, 9:12, ... 9:40
h9-12,15	every day at 9:00, 10:00, 11:00, 12:00, 15:00
h9-12,15m0	every day at 9:00, 10:00, 11:00, 12:00, 15:00
h9-12,15m0s30	every day at 9:00:30, 10:00:30, 11:00:30, 12:00:30, 15:00:30
h9-12s30	every day at 9:00:30, 9:01:30, 9:02:30 ... 12:58:30, 12:59:30
h9m/30;h10 (*API-specific syntax*)	every day at 9:00, 9:30, 10:00

Interval	Will be executed
h9m/30 h10 (*add this as another row in frontend*)	every day at 9:00, 9:30, 10:00

Aligning time zones for proxies and agent 2

Note that Zabbix proxies and agent 2 use their local time zones when processing scheduling intervals.

For this reason, when scheduling intervals are applied to items monitored by Zabbix proxy or agent 2 active items, it is recommended to set the time zone of the respective proxies or agent 2 the same as Zabbix server, otherwise the queue may report item delays incorrectly.

The time zone for Zabbix proxy or agent 2 can be set using the environment variable `TZ` in the `systemd` unit file:

```
[Service]
...
Environment="TZ=Europe/Amsterdam"
```

2 Item value preprocessing

Overview

Preprocessing allows to define transformation rules for the received item values. One or several transformations are possible before saving to the database.

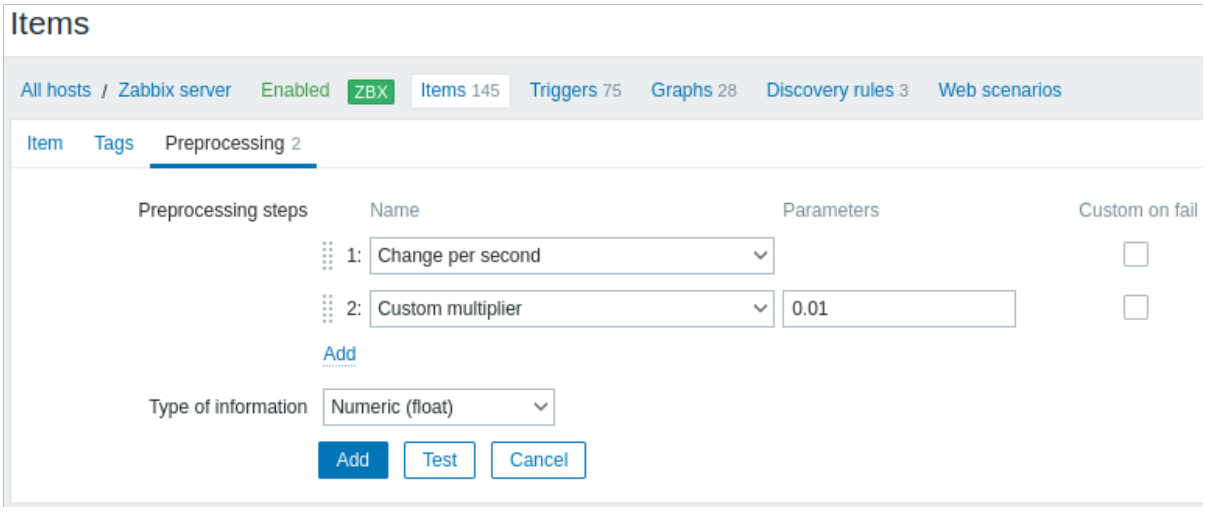
Transformations are executed in the order in which they are defined. Preprocessing is done by Zabbix server or proxy (if items are monitored by proxy).

Note that all values passed to preprocessing are of the string type, conversion to desired value type (as defined in item configuration) is performed at the end of the preprocessing pipeline; conversions, however, may also take place if required by the corresponding preprocessing step. See preprocessing details for more technical information.

See also: Usage examples

Configuration

Preprocessing rules are defined in the **Preprocessing** tab of the item configuration form.



Attention:
An item will become unsupported if any of the preprocessing steps fail, unless *Custom on fail* error-handling (available for supported transformations) has been configured to discard the value or to set a specified value.

For log items, log metadata (without value) will always reset item unsupported state and make item supported again, even if the initial error occurred after receiving a log value from agent.

User macros and user macros with context are supported in:

- preprocessing step parameters, including JavaScript code;
- custom error-handling parameters (*Set value to* and *Set error to* fields; *since Zabbix 6.4.10*).

Note:
Context is ignored when a macro is replaced with its value. Macro value is inserted in the code as is, it is not possible to add additional escaping before placing the value in the JavaScript code. Please be advised, that this can cause JavaScript errors in some cases.

Type	*Transformation*	Description
Text	*Regular expression*	Match the value to the <pattern> regular expression and replace value with <output>. The regular expression supports extraction of maximum 10 captured groups with the \N sequence. Failure to match the input value will make the item unsupported. Parameters: **pattern** - regular expression **output** - output formatting template. An \N (where N=1...9) escape sequence is replaced with the Nth matched group. A \0 escape sequence is replaced with the matched text. Please refer to regular expressions section for some existing examples. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
	Replace	Find the search string and replace it with another (or nothing). All occurrences of the search string will be replaced. Parameters: **search string** - the string to find and replace, case-sensitive (required) **replacement** - the string to replace the search string with. The replacement string may also be empty effectively allowing to delete the search string when found. It is possible to use escape sequences to search for or replace line breaks, carriage return, tabs and spaces "\n \r \t \s"; backslash can be escaped as "\\" and escape sequences can be escaped as "\\n". Escaping of line breaks, carriage return, tabs is automatically done during low-level discovery.
	Trim	Remove specified characters from the beginning and end of the value.
	Right trim	Remove specified characters from the end of the value.
	Left trim	Remove specified characters from the beginning of the value.

Structured data	*XML XPath*	Extract value or fragment from XML data using XPath functionality.
		For this option to work, Zabbix server must be compiled with libxml support.
		Examples:
		`number(/document/item/value)` will extract 10 from `<document><item><value>10</value></item></document>`
		`number(/document/item/@attribute)` will extract 10 from `<document><item attribute="10"></item></document>`
		`/document/item` will extract `<item><value>10</value></item>` from `<document><item><value>10</value></item></document>`
		Note that namespaces are not supported.
		If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.

	JSON Path	Extract value or fragment from JSON data using JSONPath functionality. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
	CSV to JSON	Convert CSV file data into JSON format. For more information, see: CSV to JSON preprocessing.
	XML to JSON	Convert data in XML format to JSON. For more information, see: Serialization rules. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.

Type	
SNMP	
SNMP walk value	Extract value by the specified OID/MIB name and apply formatting options: **Unchanged** - return hex-string as unescaped hex string (*note* that display hints are still applied); **UTF-8 from hex-STRING** - convert hex-string to UTF-8 string; **MAC from hex-STRING** - validate hex-string as MAC address and return a proper MAC address string (where `' '` are replaced by `':'`); **Integer from BITS** - convert the first 8 bytes of a bit string expressed as a sequence of hex characters (e.g. ”1A 2B 3C 4D”) into a 64-bit unsigned integer. In bit strings longer than 8 bytes, consequent bytes will be ignored. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
SNMP walk to JSON	Convert SNMP values to JSON. Specify a field name in the JSON and the corresponding SNMP OID path. Field values will be populated by values in the specified SNMP OID path. You may use this preprocessing step for SNMP OID discovery. Similar value formatting options as in the *SNMP walk value* step are available. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Arithmetic	
Custom multiplier	Multiply the value by the specified integer or floating-point value. Use this option to convert values received in KB, MBps, etc into B, Bps. Otherwise Zabbix cannot correctly set prefixes (K, M, G etc). *Note* that if the item type of information is *Numeric (unsigned)*, incoming values with a fractional part will be trimmed (i.e. ’0.9’ will become ’0’) before the custom multiplier is applied. Supported: scientific notation, for example, 1e+70 (since version 2.2); user macros and LLD macros (since version 4.0); strings that include macros, for example, `{#MACRO}e+10`, `{$MACRO1}e+{$MACRO2}`(since version 5.2.3) The macros must resolve to an integer or a floating-point number. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Change	
Simple change	Calculate the difference between the current and previous value. Evaluated as **value**-**prev_value**, where *value* - current value; *prev_value* - previously received value This setting can be useful to measure a constantly growing value. If the current value is smaller than the previous value, Zabbix discards that difference (stores nothing) and waits for another value. Only one change operation per item is allowed. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.

Type	
Change per second	Calculate the value change (difference between the current and previous value) speed per second. Evaluated as (**value**-**prev_value**)/(**time**-**prev_time**), where *value* - current value; *prev_value* - previously received value; *time* - current timestamp; *prev_time* - timestamp of previous value. This setting is useful for calculating the speed per second for a constantly growing value. If the current value is smaller than the previous value, Zabbix discards that difference (stores nothing) and waits for another value. This helps to work correctly with, for instance, a wrapping (overflow) of 32-bit SNMP counters. *Note*: As this calculation may produce floating-point numbers, it is recommended to set the 'Type of information' to *Numeric (float)*, even if the incoming raw values are integers. This is especially relevant for small numbers where the decimal part matters. If the floating-point values are large and may exceed the 'float' field length in which case the entire value may be lost, it is actually suggested to use *Numeric (unsigned)* and thus trim only the decimal part. Only one change operation per item is allowed. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Numeral systems	
Boolean to decimal	Convert the value from boolean format to decimal. The textual representation is translated into either 0 or 1. Thus, 'TRUE' is stored as 1 and 'FALSE' is stored as 0. All values are matched in a case-insensitive way. Currently recognized values are, for: *TRUE* - true, t, yes, y, on, up, running, enabled, available, ok, master *FALSE* - false, f, no, n, off, down, unused, disabled, unavailable, err, slave Additionally, any non-zero numeric value is considered to be TRUE and zero is considered to be FALSE. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Octal to decimal	Convert the value from octal format to decimal. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Hexadecimal to decimal	Convert the value from hexadecimal format to decimal. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Custom scripts	
JavaScript	Enter JavaScript code in the block that appears when clicking in the parameter field or on a pencil icon. Note that available JavaScript length depends on the database used. For more information, see: Javascript preprocessing.
Validation	
In range	Define a range that a value should be in by specifying minimum/maximum values (inclusive). Numeric values are accepted (including any number of digits, optional decimal part and optional exponential part, negative values). User macros and low-level discovery macros can be used. The minimum value should be less than the maximum. At least one value must exist. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.

Type	
Matches regular expression	Specify a regular expression that a value must match. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Does not match regular expression	Specify a regular expression that a value must not match. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Check for error in JSON	Check for an application-level error message located at JSONPath. Stop processing if succeeded and the message is not empty; otherwise, continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to the user as is, without adding preprocessing step information. No error will be reported in case of failing to parse invalid JSON. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Check for error in XML	Check for an application-level error message located at XPath. Stop processing if succeeded and the message is not empty; otherwise, continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to the user as is, without adding preprocessing step information. No error will be reported in case of failing to parse invalid XML. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Check for error using a regular expression	Check for an application-level error message using a regular expression. Stop processing if succeeded and the message is not empty; otherwise, continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to the user as is, without adding preprocessing step information. Parameters: **pattern** - regular expression **output** - output formatting template. An \N (where N=1…9) escape sequence is replaced with the Nth matched group. A \0 escape sequence is replaced with the matched text. If you mark the *Custom on fail* checkbox, it is possible to select custom error-handling options: either to discard the value, set a specified value, or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected.
Check for not supported value	Check if no item value could be retrieved. Normally that would lead to the item becoming unsupported, but you may modify that behavior by specifying the *Custom on fail* error-handling options: to discard the value, to set a specified value (that can also be used in triggers) or set a specified error message. In case of a failed preprocessing step, the item will not become unsupported if the option to discard the value or set a specified value is selected. Note that for this preprocessing step, the *Custom on fail* checkbox is grayed out and always marked. This preprocessing step only checks if no item value could be retrieved. It does not check, for example, if the type of the retrieved value (e.g., string) matches the item's type of information (e.g., numeric); for details, see Preprocessing examples. If there is a type mismatch, the item may still become unsupported after all preprocessing steps are executed. To check for a type mismatch, you may use, for example, the *Custom multiplier* preprocessing step; see Preprocessing examples. This step is always executed as the first preprocessing step and is placed above all others after saving changes to the item. It can be used only once. Supported since 5.2.0.
Throttling	

Type	
Discard unchanged	Discard a value if it has not changed. If a value is discarded, it is not saved in the database and Zabbix server has no knowledge that this value was received. No trigger expressions will be evaluated, as a result, no problems for related triggers will be created/resolved. Functions will work only based on data that is actually saved in the database. As trends are built based on data in the database, if there is no value saved for an hour then there will also be no trends data for that hour. Only one throttling option can be specified for an item. Note that in case of a very small difference (less than 0.000001) between the values of the items monitored by Zabbix proxy, it is possible that discarding will not be performed correctly by the proxy, and the values will be stored in the history as the same value, unless Zabbix server database is upgraded.
Discard unchanged with heartbeat	Discard a value if it has not changed within the defined time period (in seconds). Positive integer values are supported to specify the seconds (minimum - 1 second). Time suffixes can be used in this field (e.g. 30s, 1m, 2h, 1d). User macros and low-level discovery macros can be used in this field. If a value is discarded, it is not saved in the database and Zabbix server has no knowledge that this value was received. No trigger expressions will be evaluated, as a result, no problems for related triggers will be created/resolved. Functions will work only based on data that is actually saved in the database. As trends are built based on data in the database, if there is no value saved for an hour then there will also be no trends data for that hour. Only one throttling option can be specified for an item. Note that in case of a very small difference (less than 0.000001) between the values of the items monitored by Zabbix proxy, it is possible that discarding will not be performed correctly by the proxy, and the values will be stored in the history as the same value, unless Zabbix server database is upgraded.
Prometheus	
Prometheus pattern	Use the following query to extract required data from Prometheus metrics. See Prometheus checks for more details.
Prometheus to JSON	Convert required Prometheus metrics to JSON. See Prometheus checks for more details.

Attention:
For change and throttling preprocessing steps Zabbix has to remember the last value to calculate/compare the new value as required. These previous values are handled by the preprocessing manager. If Zabbix server or proxy is restarted or there is any change made to preprocessing steps the last value of the corresponding item is reset, resulting in:

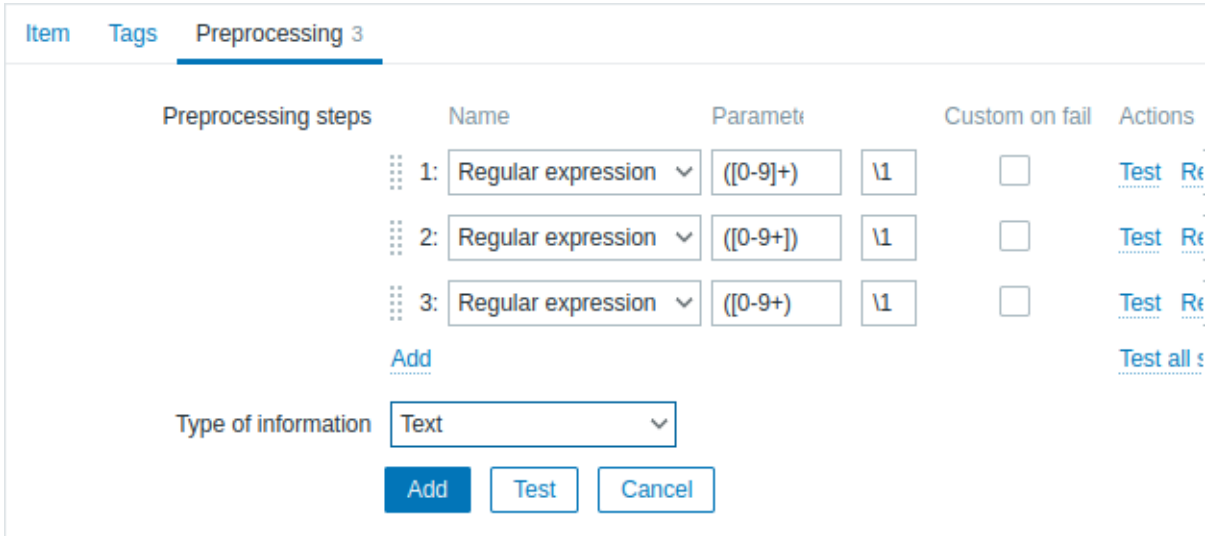
- for *Simple change*, *Change per second* steps - the next value will be ignored because there is no previous value to calculated change from;
- for *Discard unchanged*. *Discard unchanged with heartbeat* steps - the next value will never be discarded, even if it should have been because of discarding rules.

Item's *Type of information* parameter is displayed at the bottom of the tab when at least one preprocessing step is defined. If required, it is possible to change the type of information without leaving the *Preprocessing* tab. See Creating an item for the detailed parameter description.

Note:
If you use a custom multiplier or store value as *Change per second* for items with the type of information set to *Numeric (unsigned)* and the resulting calculated value is actually a float number, the calculated value is still accepted as a correct one by trimming the decimal part and storing the value as an integer.

Testing

Testing preprocessing steps is useful to make sure that complex preprocessing pipelines yield the results that are expected from them, without waiting for the item value to be received and preprocessed.

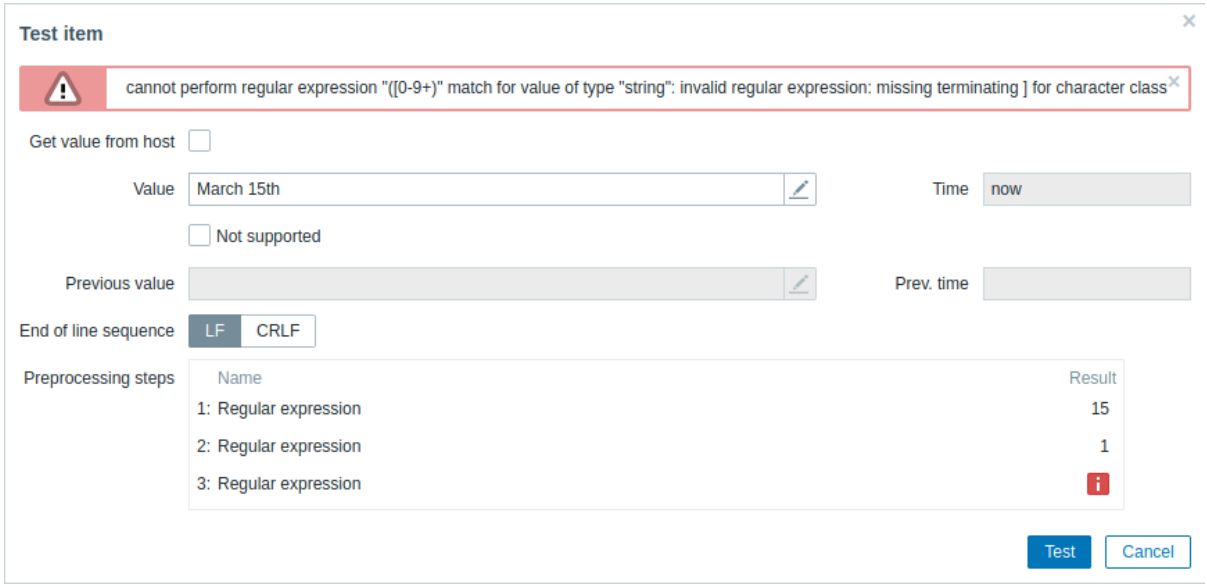


It is possible to test:

- against a hypothetical value
- against a real value from a host

Each preprocessing step can be tested individually as well as all steps can be tested together. When you click on the *Test* or *Test all steps* button respectively in the Actions block, a testing window is opened.

Testing hypothetical value



Parameter	Description
Get value from host	If you want to test a hypothetical value, leave this checkbox unmarked. See also: Testing real value.
Value	Enter the input value to test. Clicking in the parameter field or on the view/edit button will open a text area window for entering the value or code block.
Not supported	Mark this checkbox to test an unsupported value. This option is useful to test the *Check for not supported value* preprocessing step.
Time	Time of the input value is displayed: `now` (read-only).
Previous value	Enter a previous input value to compare to. Only for *Change* and *Throttling* preprocessing steps.
Previous time	Enter the previous input value time to compare to. Only for *Change* and *Throttling* preprocessing steps. The default value is based on the 'Update interval' field value of the item (if '1m', then this field is filled with `now-1m`). If nothing is specified or the user has no access to the host, the default is `now-30s`.
Macros	If any macros are used, they are listed along with their values. The values are editable for testing purposes, but the changes will only be saved within the testing context.

Parameter	Description
End of line sequence	Select the end of line sequence for multiline input values: **LF** - LF (line feed) sequence **CRLF** - CRLF (carriage-return line-feed) sequence.
Preprocessing steps	Preprocessing steps are listed; the testing result is displayed for each step after the *Test* button is clicked. Since Zabbix 6.4.16, test results are truncated to a maximum size of 512KB when sent to the frontend. If a result is truncated, a warning icon is displayed. The warning description is displayed on mouseover. Note that data larger than 512KB is still processed fully by Zabbix server. If the step failed in testing, an error icon is displayed. The error description is displayed on mouseover. In case "Custom on fail" is specified for the step and that action is performed, a new line appears right after the preprocessing test step row, showing what action was done and what outcome it produced (error or value).
Result	The final result of testing preprocessing steps is displayed in all cases when all steps are tested together (when you click on the *Test all steps* button). The type of conversion to the value type of the item is also displayed, for example `Result converted to Numeric (unsigned)`. Since Zabbix 6.4.16, test results are truncated to a maximum size of 512KB when sent to the frontend. If a result is truncated, a warning icon is displayed. The warning description is displayed on mouseover. Note that data larger than 512KB is still processed fully by Zabbix server.

Click on *Test* to see the result after each preprocessing step.

Test values are stored between test sessions for either individual steps or all steps, allowing the user to change preprocessing steps or item configuration and then return to the testing window without having to re-enter information. Values are lost on a page refresh though.

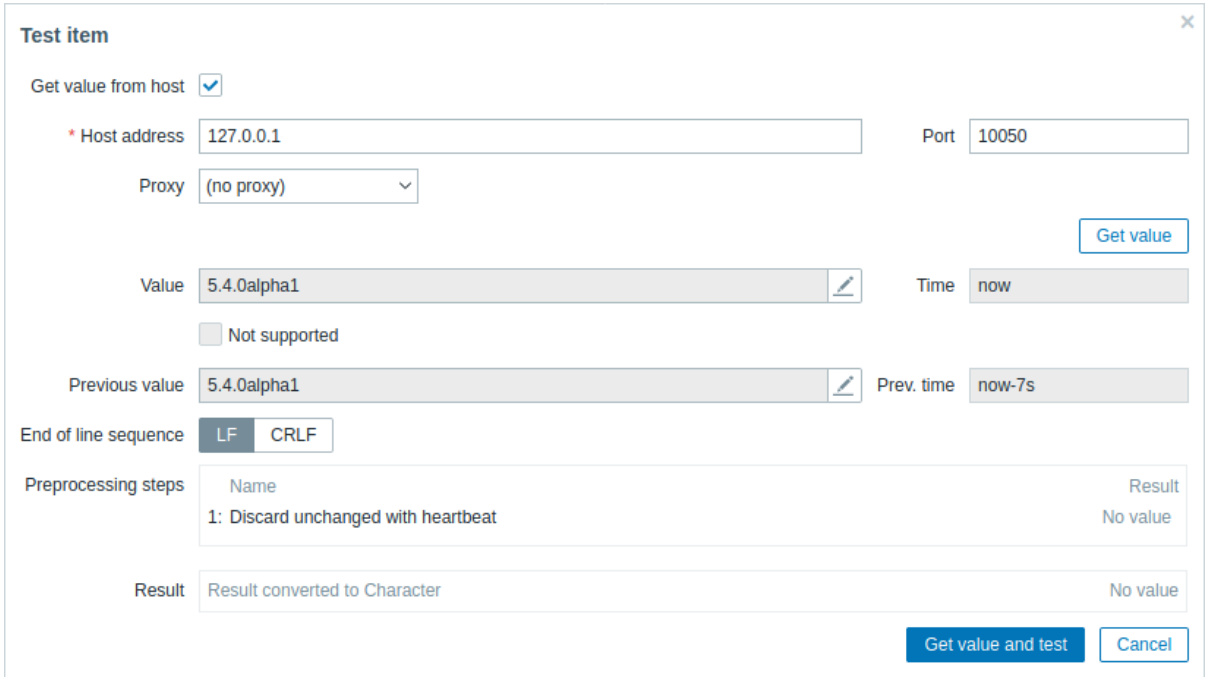
The testing is done by Zabbix server. The frontend sends a corresponding request to the server and waits for the result. The request contains the input value and preprocessing steps (with expanded user macros). For *Change* and *Throttling* steps, an optional previous value and time can be specified. The server responds with results for each preprocessing step.

All technical errors or input validation errors are displayed in the error box at the top of the testing window.

Testing real value

To test preprocessing against a real value:

- Mark the *Get value from host* checkbox
- Enter or verify host parameters (host address, port, proxy name/no proxy) and item-specific details (such as SNMPv2 community or SNMPv3 security credentials). These fields are context-aware:
  - The values are pre-filled when possible, i.e. for items requiring an agent, by taking the information from the selected agent interface of the host
  - The values have to be filled manually for template items
  - Plain-text macro values are resolved
  - Fields where the value (or part of the value) is a secret or Vault macro are empty and have to be entered manually. If any item parameter contains a secret macro value, the following warning message is displayed: "Item contains user-defined macros with secret values. Values of these macros should be entered manually."
  - The fields are disabled when not needed in the context of the item type (e.g. the host address and the proxy fields are disabled for calculated items)
- Click on *Get value and test* to test the preprocessing



If you have specified a value mapping in the item configuration form ('Show value' field), the item test dialog will show another line after the final result, named 'Result with value map applied'.

Parameters that are specific to getting a real value from a host:

Parameter	Description
Get value from host	Mark this checkbox to get a real value from the host.
Host address	Enter the host address. This field is automatically filled by the address of the item host interface.
Port	Enter the host port. This field is automatically filled by the port of item host interface.
Additional fields for SNMP interfaces (SNMP version, SNMP community, Context name, etc)	See Configuring SNMP monitoring for additional details on configuring an SNMP interface (v1, v2 and v3). These fields are automatically filled from the item host interface.
Proxy	Specify the proxy if the host is monitored by a proxy. This field is automatically filled by the proxy of the host (if any).

For the rest of the parameters, see Testing hypothetical value above.

1 Usage examples

Overview

This section presents examples of using preprocessing steps to accomplish some practical tasks.

Filtering VMware event log records

This example uses the Matches regular expression preprocessing step to filter unnecessary events from the VMware event log.

1. On a working VMware Hypervisor host, check that the event log item vmware.eventlog is present and working properly. Note that the event log item could already be present on the hypervisor if a VMware template has been linked during the host creation.

2. On the VMware Hypervisor host, create a dependent item of "Log" type and set the event log item as its master.

3. In the *Preprocessing* tab of the dependent item, select the "Matches regular expression" preprocessing step and specify, for example, one of the following parameters:

```
##### Filters all log events:
pattern: .* logged in .*
```

```
##### Filters lines containing usernames after "User":
pattern: \bUser\s+\K\S+
```

Attention:
If the regular expression is not matched, then the dependent item becomes unsupported with a corresponding error message. To avoid this, mark the "Custom on fail" checkbox and select an option such as discarding the value or setting a custom one. Please note that discarded values are not stored in the database; as a result, triggers are not evaluated and trend data is not generated.

Alternatively, you may use the Regular expression preprocessing step to extract matching groups and control output. For example:

```
##### Extracts and outputs the entire log event containing "logged in":
pattern: .*logged in.*
output: \0

##### Extracts and outputs usernames following "User":
pattern: User (.*?)(?=\ )
output: \1
```

Checking retrieved value type

This example uses the Custom multiplier preprocessing step to check if the retrieved item value type is numeric.

In the *Preprocessing* tab of an item, select the "Custom multiplier" preprocessing step and set the following parameter:

```
##### Multiplies the retrieved value by 1:
number: 1
```

Attention:
If preprocessing fails (e.g., input is not numeric), then the item becomes unsupported with a corresponding error message. To avoid this, mark the "Custom on fail" checkbox and select an option such as discarding the value or setting a custom one. Please note that discarded values are not stored in the database; as a result, triggers are not evaluated and trend data is not generated.

Checking for not supported value

This example uses the Check for not supported value preprocessing step to check if the item value could not be retrieved.

When a Zabbix server/proxy poller process attempts to collect an item value, it may:

- Return a valid result.
- Return a result that initially seems valid but may become unsupported later (e.g., due to a value type mismatch after preprocessing).
- Return an error of collecting the value, causing the item to become unsupported. Common causes include:
  - Unknown item key (for Zabbix agent, Simple check, or Zabbix internal items)
  - Unknown OID (SNMP agent), unknown sensor (IPMI agent), or no JMX metric (JMX agent)
  - Cannot read trap file (SNMP trap)
  - Script not found (External check)
  - No such URL (HTTP agent)
  - Login failed (SSH agent, TELNET agent)
  - Invalid formula syntax (Calculated), JavaScript syntax error (Script), or invalid SQL (Database monitor)

To detect and handle errors of collecting item values, you can use the "Check for not supported value" preprocessing step. Note that this step is always executed first and only detects errors that occur before preprocessing begins.

In the *Preprocessing* tab of an item, select the "Check for not supported value" preprocessing step.

Then, use the *Custom on fail* option to discard the value (in this case, the error), set a custom value, or return a custom error message. Please note that discarded values are not stored in the database; as a result, triggers are not evaluated and trend data is not generated.

2 Preprocessing details

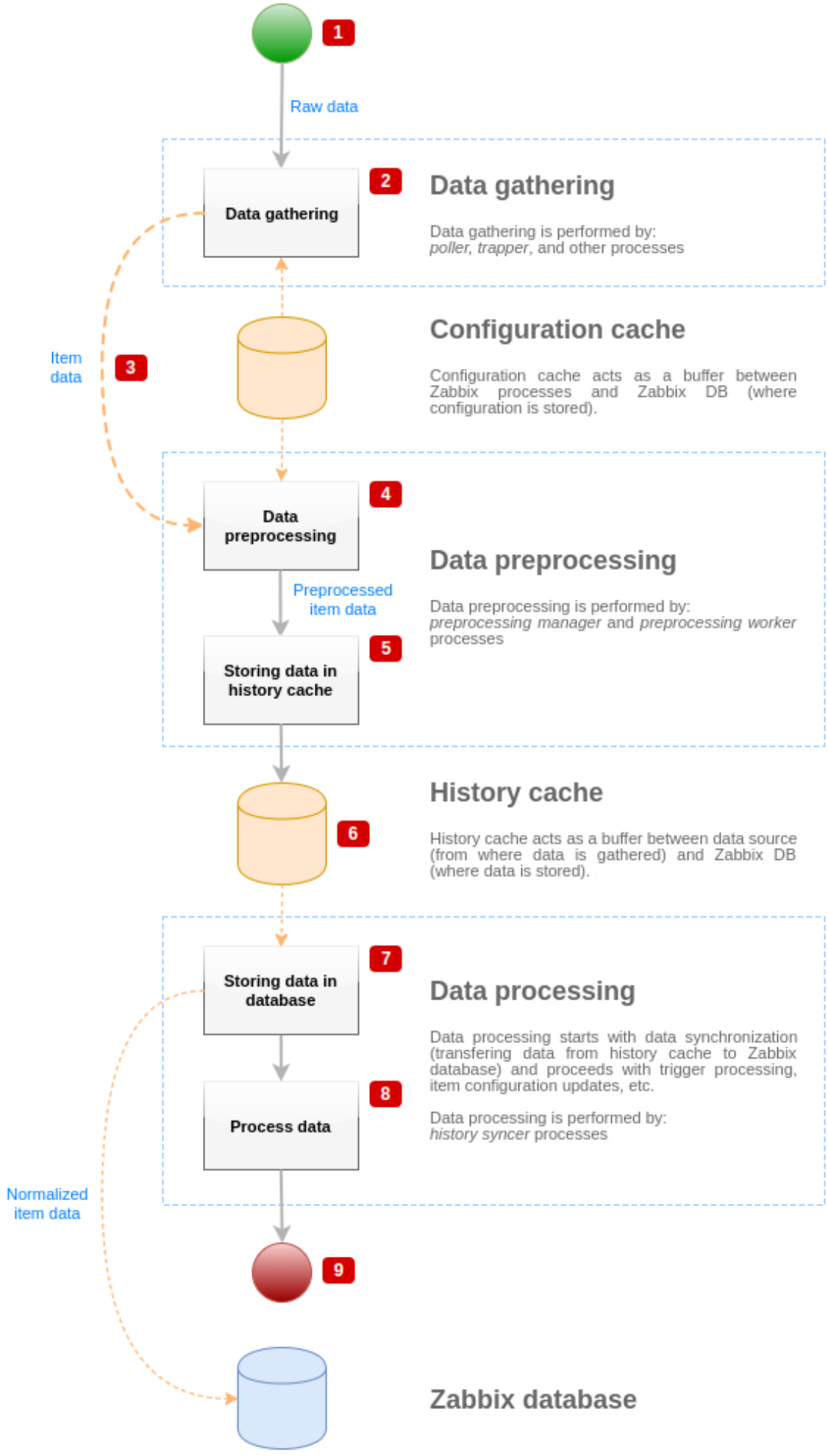
Overview

This section provides item value preprocessing details. The item value preprocessing allows to define and execute transformation rules for the received item values.

Preprocessing is managed by the preprocessing manager process along with preprocessing workers that perform the preprocessing steps. All values (with or without preprocessing) from different data gatherers pass through the preprocessing manager before being added to the history cache. Socket-based IPC communication is used between data gatherers (pollers, trappers, etc.) and the preprocessing process. Either Zabbix server or Zabbix proxy (for items monitored by the proxy) is performing preprocessing steps.

Item value processing

To visualize the data flow from data source to the Zabbix database, we can use the following simplified diagram:


1
Raw data
Data gathering
2
Data gathering
Data gathering is performed by:
poller, trapper, and other processes
Item data
3
Configuration cache
Configuration cache acts as a buffer between Zabbix processes and Zabbix DB (where configuration is stored).
Data preprocessing
4
Data preprocessing
Data preprocessing is performed by:
preprocessing manager and preprocessing worker processes
Preprocessed item data
Storing data in history cache
5
History cache
6
History cache acts as a buffer between data source (from where data is gathered) and Zabbix DB (where data is stored).
Storing data in database
7
Data processing
Data processing starts with data synchronization (transfering data from history cache to Zabbix database) and proceeds with trigger processing, item configuration updates, etc.
Process data
8
Data processing is performed by:
history syncer processes
Normalized item data
9
Zabbix database

The diagram above shows only processes, objects and actions related to item value processing in a **simplified** form. The diagram does not show conditional direction changes, error handling or loops. The local data cache of the preprocessing manager is not shown either because it doesn't affect the data flow directly. The aim of this diagram is to show processes involved in the item value processing and the way they interact.

- Data gathering starts with raw data from a data source. At this point, the data contains only ID, timestamp and value (can be multiple values as well)
- No matter what type of data gatherer is used, the idea is the same for active or passive checks, for trapper items, etc, as it only changes the data format and the communication starter (either data gatherer is waiting for a connection and data, or data gatherer initiates the communication and requests the data). The raw data is validated, the item configuration is retrieved from the configuration cache (data is enriched with the configuration data).
- A socket-based IPC mechanism is used to pass data from data gatherers to the preprocessing manager. At this point the data gatherer continues to gather data without waiting for the response from preprocessing manager.
- Data preprocessing is performed. This includes the execution of preprocessing steps and dependent item processing.

Note:
An item can change its state to NOT SUPPORTED while preprocessing is performed if any of preprocessing steps fail.

- The history data from the local data cache of the preprocessing manager is being flushed into the history cache.
- At this point the data flow stops until the next synchronization of history cache (when the history syncer process performs data synchronization).
- The synchronization process starts with data normalization before storing data in Zabbix database. The data normalization performs conversions to the desired item type (type defined in item configuration), including truncation of textual data based on predefined sizes allowed for those types (HISTORY_STR_VALUE_LEN for string, HISTORY_TEXT_VALUE_LEN for text and HISTORY_LOG_VALUE_LEN for log values). The data is being sent to the Zabbix database after the normalization is done.

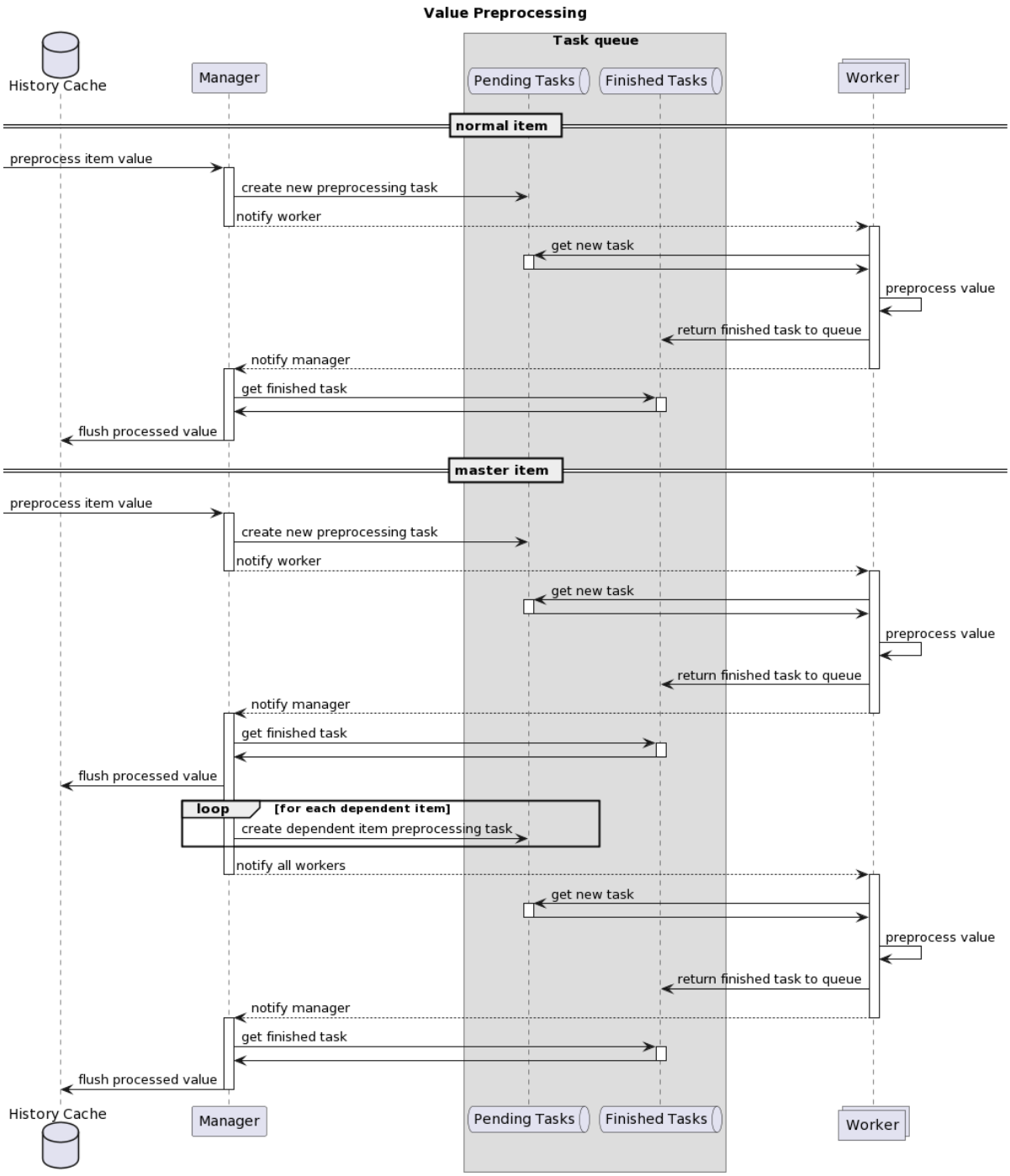
Note:
An item can change its state to NOT SUPPORTED if data normalization fails (for example, when a textual value cannot be converted to number).

- The gathered data is being processed - triggers are checked, the item configuration is updated if item becomes NOT SUPPORTED, etc.
- This is considered the end of data flow from the point of view of item value processing.

Item value preprocessing

Data preprocessing is performed in the following steps:

- The item value is passed to the preprocessing manager using a UNIX socket-based IPC mechanism.
- If the item has neither preprocessing nor dependent items, its value is either added to the history cache or sent to the LLD manager. Otherwise:
  - A preprocessing task is created and added to the queue and preprocessing workers are notified about the new task.
  - At this point the data flow stops until there is at least one unoccupied (i.e. not executing any tasks) preprocessing worker.
  - When a preprocessing worker is available it takes the next task from the queue.
  - After the preprocessing is done (both failed and successful execution of preprocessing steps), the preprocessed value is added to the finished task queue and the manager is notified about a new finished task.
  - The preprocessing manager converts the result to desired format (defined by item value type) and either adds it to the history cache or sends to the LLD manager.
  - If there are dependent items for the processed item, then dependent items are added to the preprocessing queue with the preprocessed master item value. Dependent items are enqueued bypassing the normal value preprocessing requests, but only for master items with the value set and not in a NOT SUPPORTED state.



Note that in the diagram the master item preprocessing is slightly simplified by skipping the preprocessing caching.

Preprocessing queue

The preprocessing queue is organized as:

- the list of pending tasks:
  - tasks created directly from value preprocessing requests in the order they were received.
- the list of immediate tasks (processed before pending tasks):
  - testing tasks (created in response to item/preprocessing testing requests by the frontend)
  - dependent item tasks
  - sequence tasks (tasks that must be executed in a strict order):
    - having preprocessing steps using the last value:
      - change
      - throttling
      - JavaScript (bytecode caching)
    - dependent item preprocessing caching
- the list of finished tasks

Preprocessing caching

Preprocessing caching was introduced to improve the preprocessing performance for multiple dependent items having similar preprocessing steps (which is a common LLD outcome).

Caching is done by preprocessing one dependent item and reusing some of the internal preprocessing data for the rest of the dependent items. The preprocessing cache is supported only for the first preprocessing step of the following types:

- Prometheus pattern (indexes input by metrics)
- JSONPath (parses the data into object tree and indexes the first expression `[?(@.path == "value")]`)

Preprocessing workers

The Zabbix server configuration file allows users to set the count of preprocessing worker threads. The StartPreprocessors configuration parameter should be used to set the number of pre-started instances of preprocessing workers. The optimal number of preprocessing workers can be determined by many factors, including the count of "preprocessable" items (items that require to execute any preprocessing steps), the count of data gathering processes, the average step count for item preprocessing, etc.

But assuming that there are no heavy preprocessing operations like parsing large XML/JSON chunks, the number of preprocessing workers can match the total number of data gatherers. This way, there will mostly (except for the cases when data from the gatherer comes in bulk) be at least one unoccupied preprocessing worker for collected data.

Warning:
Too many data gathering processes (pollers, unreachable pollers, ODBC pollers, HTTP pollers, Java pollers, pingers, trappers, proxypollers) together with IPMI manager, SNMP trapper and preprocessing workers can exhaust the per-process file descriptor limit for the preprocessing manager. This will cause Zabbix server to stop (usually shortly after the start, but sometimes it can take more time). The configuration file should be revised or the limit should be raised to avoid this situation.

Value processing pipeline

Item value processing is executed in multiple steps (or phases) by multiple processes. This can cause:

- A dependent item can receive values, while THE master value cannot. This can be achieved by using the following use case:
  - Master item has value type `UINT`, (trapper item can be used), dependent item has value type `TEXT`.
  - No preprocessing steps are required for both master and dependent items.
  - Textual value (like, "abc") should be passed to master item.
  - As there are no preprocessing steps to execute, preprocessing manager checks if master item is not in NOT SUPPORTED state and if value is set (both are true) and enqueues dependent item with the same value as master item (as there are no preprocessing steps).
  - When both master and dependent items reach history synchronization phase, master item becomes NOT SUPPORTED, because of the value conversion error (textual data cannot be converted to unsigned integer).

As a result, the dependent item receives a value, while the master item changes its state to NOT SUPPORTED.

- A dependent item receives value that is not present in the master item history. The use case is very similar to the previous one, except for the master item type. For example, if `CHAR` type is used for master item, then master item value will be truncated at the history synchronization phase, while dependent items will receive their value from the initial (not truncated) value of master item.

3 JSONPath functionality

Overview

This section provides details of supported JSONPath functionality in item value preprocessing steps.

JSONPath consists of segments separated with dots. A segment can be either a simple word like a JSON value name, `*` or a more complex construct enclosed within square brackets `[]`. The separating dot before bracket segment is optional and can be omitted. For example:

Path	Description
`$.object.name`	Return the object.name contents.
`$.object['name']`	Return the object.name contents.
`$.object.['name']`	Return the object.name contents.
`$["object"]['name']`	Return the object.name contents.
`$.['object'].["name"]`	Return the object.name contents.
`$.object.history.length()`	Return the number of object.history array elements.
`$[?(@.name == 'Object')].price.first()`	Return the price field of the first object with name 'Object'.

Path	Description
`$[?(@.name == 'Object')].history.first().length()`	Return the number of history array elements of the first object with name 'Object'.
`$[?(@.price > 10)].length()`	Return the number of objects with price being greater than 10.

See also: Escaping special characters from LLD macro values in JSONPath.

Supported segments

Segment	Description
`<name>`	Match object property by name.
`*`	Match all object properties.
`['<name>']`	Match object property by name.
`['<name>', '<name>', ...]`	Match object property by any of the listed names.
`[<index>]`	Match array element by the index.
`[<number>, <number>, ...]`	Match array element by any of the listed indexes.
`[*]`	Match all object properties or array elements.
`[<start>:<end>]`	Match array elements by the defined range: **<start>** - the first index to match (including). If not specified matches all array elements from the beginning. If negative specifies starting offset from the end of array. **<end>** - the last index to match (excluding). If not specified matches all array elements to the end. If negative specifies starting offset from the end of array.
`[?(<expression>)]`	Match objects/array elements by applying a filter expression.

To find a matching segment ignoring its ancestry (detached segment) it must be prefixed with '..' , for example `$..name` or `$..['name']` return values of all 'name' properties.

Matched element names can be extracted by adding a ~ suffix to the JSONPath. It returns the name of the matched object or an index in string format of the matched array item. The output format follows the same rules as other JSONPath queries - definite path results are returned 'as is' and indefinite path results are returned in array. However there is not much point of extracting the name of an element matching a definite path - it's already known.

Filter expression

The filter expression is an arithmetical expression in infix notation.

Supported operands:

Operand	Description	Example
`"<text>"`	Text constant.	'value: \'1\" "value: '1'"
`'<text>'`		
`<number>`	Numeric constant supporting scientific notation.	123
`<jsonpath starting with $>`	Value referred to by the JSONPath from the input document root node; only definite paths are supported.	$.object.name
`<jsonpath starting with @>`	Value referred to by the JSONPath from the current object/element; only definite paths are supported.	@.name

Supported operators:

Operator	Type	Description	Result
-	binary	Subtraction.	Number.
+	binary	Addition.	Number.
/	binary	Division.	Number.
*	binary	Multiplication.	Number.
==	binary	Is equal to.	Boolean (1 or 0).

Operator	Type	Description	Result
!=	binary	Is not equal to.	Boolean (1 or 0).
	binary	Is less than.	Boolean (1 or 0).
<=	binary	Is less than or equal to.	Boolean (1 or 0).
>	binary	Is greater than.	Boolean (1 or 0).
>=	binary	Is greater than or equal to.	Boolean (1 or 0).
=~	binary	Matches regular expression.	Boolean (1 or 0).
!	unary	Boolean not.	Boolean (1 or 0).
||	binary	Boolean or.	Boolean (1 or 0).
&&	binary	Boolean and.	Boolean (1 or 0).

Functions

Functions can be used at the end of JSONPath. Multiple functions can be chained if the preceding function returns value that is accepted by the following function.

Supported functions:

Function	Description	Input	Output
`avg`	Average value of numbers in input array.	Array of numbers.	Number.
`min`	Minimum value of numbers in input array.	Array of numbers.	Number.
`max`	Maximum value of numbers in input array.	Array of numbers.	Number.
`sum`	Sum of numbers in input array.	Array of numbers.	Number.
`length`	Number of elements in input array.	Array.	Number.
`first`	The first array element.	Array.	A JSON construct (object, array, value) depending on input array contents.

Quoted numeric values are accepted by the JSONPath aggregate functions. It means that the values are converted from string type to numeric if aggregation is required.

Incompatible input will cause the function to generate error.

Output value

JSONPaths can be divided in definite and indefinite paths. A definite path can return only null or a single match. An indefinite path can return multiple matches, basically JSONPaths with detached, multiple name/index list, array slice or expression segments. However, when a function is used the JSONPath becomes definite, as functions always output single value.

A definite path returns the object/array/value it's referencing, while indefinite path returns an array of the matched objects/arrays/values.

Attention:
The property order in JSONPath query results may not align with the original JSON property order due to internal optimization methods. For example, the JSONPath `$.books[1]["author", "title"]` may return `["title", "author"]`. If preserving the original property order is essential, alternative post-query processing methods should be considered.

Whitespace

Whitespace (space, tab characters) can be freely used in bracket notation segments and expressions, for example, `$[ 'a' ][ 0 ][ ?( $.b == 'c' ) ][ : -1 ].first( )`.

Strings

Strings should be enclosed with single ' or double " quotes. Inside the strings, single or double quotes (depending on which are used to enclose it) and backslashes \ are escaped with the backslash \ character.

Examples

Input data

```
{
  "books": [
    {
      "category": "reference",
      "author": "Nigel Rees",
```

```
      "title": "Sayings of the Century",
      "price": 8.95,
      "id": 1
    },
    {
      "category": "fiction",
      "author": "Evelyn Waugh",
      "title": "Sword of Honour",
      "price": 12.99,
      "id": 2
    },
    {
      "category": "fiction",
      "author": "Herman Melville",
      "title": "Moby Dick",
      "isbn": "0-553-21311-3",
      "price": 8.99,
      "id": 3
    },
    {
      "category": "fiction",
      "author": "J. R. R. Tolkien",
      "title": "The Lord of the Rings",
      "isbn": "0-395-19395-8",
      "price": 22.99,
      "id": 4
    }
  ],
  "services": {
    "delivery": {
      "servicegroup": 1000,
      "description": "Next day delivery in local town",
      "active": true,
      "price": 5
    },
    "bookbinding": {
      "servicegroup": 1001,
      "description": "Printing and assembling book in A5 format",
      "active": true,
      "price": 154.99
    },
    "restoration": {
      "servicegroup": 1002,
      "description": "Various restoration methods",
      "active": false,
      "methods": [
        {
          "description": "Chemical cleaning",
          "price": 46
        },
        {
          "description": "Pressing pages damaged by moisture",
          "price": 24.5
        },
        {
          "description": "Rebinding torn book",
          "price": 99.49
        }
      ]
    }
  },
  "filters": {
```

```
    "price": 10,
    "category": "fiction",
    "no filters": "no \"filters\""
  },
  "closed message": "Store is closed",
  "tags": [
    "a",
    "b",
    "c",
    "d",
    "e"
  ]
}
```

JSONPath	Type	Result	Comments
`$.filters.price`	definite	10	
`$.filters.category`	definite	fiction	
`$.filters['no filters']`	definite	no ”filters”	
`$.filters`	definite	{ ”price”: 10, ”category”: ”fiction”, ”no filters”: ”no \”filters\”” }	
`$.books[1].title`	definite	Sword of Honour	
`$.books[-1].author`	definite	J. R. R. Tolkien	
`$.books.length()`	definite	4	
`$.tags[:]`	indefinite	[”a”, ”b”, ”c”, ”d”, ”e”]	
`$.tags[2:]`	indefinite	[”c”, ”d”, ”e”]	
`$.tags[:3]`	indefinite	[”a”, ”b”, ”c”]	
`$.tags[1:4]`	indefinite	[”b”, ”c”, ”d”]	
`$.tags[-2:]`	indefinite	[”d”, ”e”]	
`$.tags[:-3]`	indefinite	[”a”, ”b”]	
`$.tags[:-3].length()`	definite	2	
`$.books[0, 2].title`	indefinite	[”Moby Dick”, ”Sayings of the Century”]	
`$.books[1]['author', "title"]`	indefinite	[”Sword of Honour”, ”Evelyn Waugh”]	
`$..id`	indefinite	[1, 2, 3, 4]	
`$.services..price`	indefinite	[154.99, 5, 46, 24.5, 99.49]	
`$.books[?(@.id == 4 - 0.4 * 5)].title`	indefinite	[”Sword of Honour”]	This query shows that arithmetical operations can be used in queries. Of course this query can be simplified to `$.books[?(@.id == 2)].title`
`$.books[?(@.id == 2 || @.id == 4)].title`	indefinite	[”Sword of Honour”, ”The Lord of the Rings”]	
`$.books[?(!(@.id == 2))].title`	indefinite	[”Sayings of the Century”, ”Moby Dick”, ”The Lord of the Rings”]	
`$.books[?(@.id != 2)].title`	indefinite	[”Sayings of the Century”, ”Moby Dick”, ”The Lord of the Rings”]	
`$.books[?(@.title =~ " of ")].title`	indefinite	[”Sayings of the Century”, ”Sword of Honour”, ”The Lord of the Rings”]	
`$.books[?(@.price > 12.99)].title`	indefinite	[”The Lord of the Rings”]	

JSONPath	Type	Result	Comments
`$.books[?(@.author > "Herman Melville")].title`	indefinite	[”Sayings of the Century”, ”The Lord of the Rings”]	
`$.books[?(@.price > $.filters.price)].title`	indefinite	[”Sword of Honour”, ”The Lord of the Rings”]	
`$.books[?(@.category == $.filters.category)].title`	indefinite	[”Sword of Honour”,”Moby Dick”,”The Lord of the Rings”]	
`$.books[?(@.category == "fiction" && @.price < 10)].title`	indefinite	[”Moby Dick”]	

`$..[?(@.id)]`	indefinite	[
		{
		”price”: 8.95,
		”id”: 1,
		”category”: ”reference”,
		”author”: ”Nigel Rees”,
		”title”: ”Sayings of the Century”
		},
		{
		”price”: 12.99,
		”id”: 2,
		”category”: ”fiction”,
		”author”: ”Evelyn Waugh”,
		”title”: ”Sword of Honour”
		},
		{
		”price”: 8.99,
		”id”: 3,
		”category”: ”fiction”,
		”author”: ”Herman Melville”,
		”title”: ”Moby Dick”,
		”isbn”: ”0-553-21311-3”
		},
		{
		”price”: 22.99,
		”id”: 4,
		”category”: ”fiction”,
		”author”: ”J. R. R. Tolkien”,
		”title”: ”The Lord of the Rings”,
		”isbn”: ”0-395-19395-8”
		}
		]	

`$.services..[?(@.price > 50)].description`	indefinite	’[”Printing and assembling book in A5 format”, ”Rebinding torn book”]	
`$..id.length()`	definite	4	
`$.books[?(@.id == 2)].title.first()`	definite	Sword of Honour	
`$..tags.first().length()`	definite	5	$..tags is indefinite path, so it returns an array of matched elements - [[”a”, ”b”, ”c”, ”d”, ”e”]], first() returns the first element - [”a”, ”b”, ”c”, ”d”, ”e”] and finally length() calculates its length - 5.
`$.books[*].price.min()`	definite	8.95	
`$..price.max()`	definite	154.99	

JSONPath	Type	Result	Comments
`$.books[?(@.category == "fiction")].price.avg()`	definite	14.99	
`$.books[?(@.category == $.filters.xyz)].title`	indefinite		A query without match returns NULL for definite and indefinite paths.
`$.services[?(@.isDefault=="true")].servicegroup`	definite	[1001,1000]	Text constants must be used in boolean value comparisons.
`$.services[?(@.isDefault=="false")].servicegroup`	definite	[1002]	Text constants must be used in boolean value comparisons.
`$.services[?(@.servicegroup=="1002")]~.first()`	definite	1002	

1 Escaping special characters from LLD macro values in JSONPath

When low-level discovery macros are used in JSONPath preprocessing and their values are resolved, the following rules of escaping special characters are applied:

- only backslash (\) and double quote (”) characters are considered for escaping;
- if the resolved macro value contains these characters, each of them is escaped with a backslash;
- if they are already escaped with a backslash, it is not considered as escaping and both the backslash and the following special characters are escaped once again.

For example:

JSONPath	LLD macro value	After substitution
$.[?(@.value == ”{#MACRO}”)]	special ”value”	$.[?(@.value == ”special \”value\””)]
	c:\temp	$.[?(@.value == ”c:\\temp”)]
	a\\b	$.[?(@.value == ”a\\\\b”)]

When used in the expression the macro that may have special characters should be enclosed in double quotes:

JSONPath	LLD macro value	After substitution	Result
$.[?(@.value == ”{#MACRO}”)]	special ”value”	$.[?(@.value == ”special \”value\””)]	OK
$.[?(@.value == {#MACRO})]		$.[?(@.value == special \”value\”)]	**Bad JSONPath expression**

When used in the path the macro that may have special characters should be enclosed in square brackets **and** double quotes:

JSONPath	LLD macro value	After substitution	Result
$.[”{#MACRO}”].value	c:\temp	$.[”c:\\temp”].value	OK
$.{#MACRO}.value		$.c:\\temp.value	**Bad JSONPath expression**

4 JavaScript preprocessing

Overview

This section provides details of preprocessing by JavaScript.

JavaScript preprocessing

JavaScript preprocessing is done by invoking JavaScript function with a single parameter ’value’ and user provided function body. The preprocessing step result is the value returned from this function, for example, to perform Fahrenheit to Celsius conversion user must enter:

```
return (value - 32)  * 5 / 9
```

in JavaScript preprocessing parameters, which will be wrapped into a JavaScript function by server:

```
function (value)
{
   return (value - 32) * 5 / 9
}
```

The input parameter 'value' is always passed as a string. The return value is automatically coerced to string via ToString() method (if it fails then the error is returned as string value), with a few exceptions:

- returning undefined value will result in an error;
- returning null value will cause the input value to be discarded, much like 'Discard value' preprocessing on 'Custom on fail' action.

Errors can be returned by throwing values/objects (normally either strings or Error objects).

For example:

```
if (value == 0)
    throw "Zero input value"
return 1/value
```

Each script has a 10 second execution timeout (depending on the script it might take longer for the timeout to trigger); exceeding it will return error. A 512-megabyte heap limit is enforced (64 megabytes before Zabbix 6.4.4).

The JavaScript preprocessing step bytecode is cached and reused when the step is applied next time. Any changes to the item's preprocessing steps will cause the cached script to be reset and recompiled later.

Consecutive runtime failures (3 in a row) will cause the engine to be reinitialized to mitigate the possibility of one script breaking the execution environment for the next scripts (this action is logged with DebugLevel 4 and higher).

JavaScript preprocessing is implemented with Duktape (https://duktape.org/) JavaScript engine.

See also: Additional JavaScript objects and global functions

Using macros in scripts

It is possible to use user macros in JavaScript code. If a script contains user macros, these macros are resolved by server/proxy before executing specific preprocessing steps. Note that when testing preprocessing steps in the frontend, macro values will not be pulled and need to be entered manually.

> **Note:**
> Context is ignored when a macro is replaced with its value. Macro value is inserted in the code as is, it is not possible to add additional escaping before placing the value in the JavaScript code. Please be advised, that this can cause JavaScript errors in some cases.

In an example below, if received value exceeds a {$THRESHOLD} macro value, the threshold value (if present) will be returned instead:

```
var threshold = '{$THRESHOLD}';
return (!isNaN(threshold) && value > threshold) ? threshold : value;
```

Examples

The following examples illustrate how you can use JavaScript preprocessing.

Each example contains a brief description, a function body for JavaScript preprocessing parameters, and the preprocessing step result - value returned by the function.

Example 1: Convert a number (scientific notation to integer)

Convert the number "2.62128e+07" from scientific notation to an integer.

```
return (Number(value))
```

Value returned by the function: 26212800.

Example 2: Convert a number (binary to decimal)

Convert the binary number "11010010" to a decimal number.

```
return(parseInt(value,2))
```

Value returned by the function: 210.

Example 3: Round a number

Round the number ”18.2345” to 2 digits.

```
return(Math.round(value* 100) / 100)
```

Value returned by the function: 18.23.

Example 4: Count letters in a string

Count the number of letters in the string ”Zabbix”.

```
return (value.length)
```

Value returned by the function: 6.

Example 5: Get time remaining

Get the remaining time (in seconds) until the expiration date of a certificate (Feb 12 12:33:56 2022 GMT).

```
var split = value.split(' '),
    MONTHS_LIST = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'],
    month_index = ('0' + (MONTHS_LIST.indexOf(split[0]) + 1)).slice(-2),
    ISOdate = split[3] + '-' + month_index + '-' + split[1] + 'T' + split[2],
    now = Date.now();

return parseInt((Date.parse(ISOdate) - now) / 1000);
```

Value returned by the function: 44380233.

Example 6: Remove JSON properties

Modify the JSON data structure by removing any properties with the key "data_size" or "index_size".

```
var obj=JSON.parse(value);

for (i = 0; i < Object.keys(obj).length; i++) {
    delete obj[i]["data_size"];
    delete obj[i]["index_size"];
}

return JSON.stringify(obj)
```

Value accepted by the function:

```
[
    {
        "table_name":"history",
        "data_size":"326.05",
        "index_size":"174.34"
    },
    {
        "table_name":"history_log",
        "data_size":"6.02",
        "index_size":"3.45"
    }
]
```

Value returned by the function:

```
[
    {
        "table_name":"history"
    },
    {
        "table_name":"history_log"
    }
]
```

Example 7: Convert Apache status to JSON

Convert the value received from a web.page.get Zabbix agent item (e.g., web.page.get[http://127.0.0.1:80/server-status?auto]) to a JSON object.

```
// Convert Apache status to JSON

// Split the value into substrings and put these substrings into an array
var lines = value.split('\n');

// Create an empty object "output"
var output = {};

// Create an object "workers" with predefined properties
var workers = {
    '_': 0, 'S': 0, 'R': 0, 'W': 0,
    'K': 0, 'D': 0, 'C': 0, 'L': 0,
    'G': 0, 'I': 0, '.': 0
};

// Add the substrings from the "lines" array to the "output" object as properties (key-value pairs)
for (var i = 0; i < lines.length; i++) {
    var line = lines[i].match(/([A-z0-9 ]+): (.*)/);

    if (line !== null) {
        output[line[1]] = isNaN(line[2]) ? line[2] : Number(line[2]);
    }
}

// Multiversion metrics
output.ServerUptimeSeconds = output.ServerUptimeSeconds || output.Uptime;
output.ServerVersion = output.ServerVersion || output.Server;

// Parse "Scoreboard" property to get the worker count
if (typeof output.Scoreboard === 'string') {
    for (var i = 0; i < output.Scoreboard.length; i++) {
        var char = output.Scoreboard[i];

        workers[char]++;
    }
}

// Add worker data to the "output" object
output.Workers = {
    waiting: workers['_'], starting: workers['S'], reading: workers['R'],
    sending: workers['W'], keepalive: workers['K'], dnslookup: workers['D'],
    closing: workers['C'], logging: workers['L'], finishing: workers['G'],
    cleanup: workers['I'], slot: workers['.']
};

// Return JSON string
return JSON.stringify(output);
```

Value accepted by the function:

```
HTTP/1.1 200 OK
Date: Mon, 27 Mar 2023 11:08:39 GMT
Server: Apache/2.4.52 (Ubuntu)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 405
Content-Type: text/plain; charset=ISO-8859-1

127.0.0.1
ServerVersion: Apache/2.4.52 (Ubuntu)
ServerMPM: prefork
```

```
Server Built: 2023-03-08T17:32:01
CurrentTime: Monday, 27-Mar-2023 14:08:39 EEST
RestartTime: Monday, 27-Mar-2023 12:19:59 EEST
ParentServerConfigGeneration: 1
ParentServerMPMGeneration: 0
ServerUptimeSeconds: 6520
ServerUptime: 1 hour 48 minutes 40 seconds
Load1: 0.56
Load5: 0.33
Load15: 0.28
Total Accesses: 2476
Total kBytes: 8370
Total Duration: 52718
CPUUser: 8.16
CPUSystem: 3.44
CPUChildrenUser: 0
CPUChildrenSystem: 0
CPULoad: .177914
Uptime: 6520
ReqPerSec: .379755
BytesPerSec: 3461.58
BytesPerReq: 3461.58
DurationPerReq: 21.2916
BusyWorkers: 2
IdleWorkers: 6
Scoreboard: ____KW__.....................................................................................
```

Value returned by the function:


```
{
    "Date": "Mon, 27 Mar 2023 11:08:39 GMT",
    "Server": "Apache/2.4.52 (Ubuntu)",
    "Vary": "Accept-Encoding",
    "Encoding": "gzip",
    "Length": 405,
    "Type": "text/plain; charset=ISO-8859-1",
    "ServerVersion": "Apache/2.4.52 (Ubuntu)",
    "ServerMPM": "prefork",
    "Server Built": "2023-03-08T17:32:01",
    "CurrentTime": "Monday, 27-Mar-2023 14:08:39 EEST",
    "RestartTime": "Monday, 27-Mar-2023 12:19:59 EEST",
    "ParentServerConfigGeneration": 1,
    "ParentServerMPMGeneration": 0,
    "ServerUptimeSeconds": 6520,
    "ServerUptime": "1 hour 48 minutes 40 seconds",
    "Load1": 0.56,
    "Load5": 0.33,
    "Load15": 0.28,
    "Total Accesses": 2476,
    "Total kBytes": 8370,
    "Total Duration": 52718,
    "CPUUser": 8.16,
    "CPUSystem": 3.44,
    "CPUChildrenUser": 0,
    "CPUChildrenSystem": 0,
    "CPULoad": 0.177914,
    "Uptime": 6520,
    "ReqPerSec": 0.379755,
    "BytesPerSec": 1314.55,
    "BytesPerReq": 3461.58,
    "DurationPerReq": 21.2916,
    "BusyWorkers": 2,
    "IdleWorkers": 6,
    "Scoreboard": "____KW__.....................................................................................
```

```
    "Workers": {
        "waiting": 6,
        "starting": 0,
        "reading": 0,
        "sending": 1,
        "keepalive": 1,
        "dnslookup": 0,
        "closing": 0,
        "logging": 0,
        "finishing": 0,
        "cleanup": 0,
        "slot": 142
    }
}
```

1 Additional JavaScript objects

Overview

This section describes Zabbix additions to the JavaScript language implemented with Duktape and supported global JavaScript functions.

Built-in objects

Zabbix

The Zabbix object provides interaction with the internal Zabbix functionality.

Method	Description
`log(loglevel, message)`	Writes <message> into Zabbix log using <loglevel> log level (see configuration file DebugLevel parameter).

Example:

```
Zabbix.log(3, "this is a log entry written with 'Warning' log level")
```

You may use the following aliases:

Alias	Alias to
console.log(object)	Zabbix.log(4, JSON.stringify(object))
console.warn(object)	Zabbix.log(3, JSON.stringify(object))
console.error(object)	Zabbix.log(2, JSON.stringify(object))

Attention:
The total size of all logged messages is limited to 8 MB per script execution.

Method	Description
`sleep(delay)`	Delay JavaScript execution by `delay` milliseconds.

Example (delay execution by 15 seconds):

```
Zabbix.sleep(15000)
```

HttpRequest

This object encapsulates cURL handle allowing to make simple HTTP requests. Errors are thrown as exceptions.

Attention:
The initialization of multiple `HttpRequest` objects is limited to 10 per script execution.

Method	Description
`addHeader(value)`	Adds HTTP header field. This field is used for all following requests until cleared with the `clearHeader()` method. The total length of header fields that can be added to a single `HttpRequest` object is limited to 128 Kbytes (special characters and header names included).
`clearHeader()`	Clears HTTP header. If no header fields are set, `HttpRequest` will set Content-Type to application/json if the data being posted is JSON-formatted; text/plain otherwise.
`connect(url)`	Sends HTTP CONNECT request to the URL and returns the response.
`customRequest(method, url, data)`	Allows to specify any HTTP method in the first parameter. Sends the method request to the URL with optional *data* payload and returns the response.
`delete(url, data)`	Sends HTTP DELETE request to the URL with optional *data* payload and returns the response.
`getHeaders(<asArray>)`	Returns the object of received HTTP header fields. The `asArray` parameter may be set to "true" (e.g. `getHeaders(true)`), "false" or be undefined. If set to "true" the received HTTP header field values will be returned as arrays; this should be used to retrieve the field values of multiple same-name headers. If not set or set to "false", the received HTTP header field values will be returned as strings.
`get(url, data)`	Sends HTTP GET request to the URL with optional *data* payload and returns the response.
`head(url)`	Sends HTTP HEAD request to the URL and returns the response.
`options(url)`	Sends HTTP OPTIONS request to the URL and returns the response.
`patch(url, data)`	Sends HTTP PATCH request to the URL with optional *data* payload and returns the response.
`put(url, data)`	Sends HTTP PUT request to the URL with optional *data* payload and returns the response.
`post(url, data)`	Sends HTTP POST request to the URL with optional *data* payload and returns the response.
`getStatus()`	Returns the status code of the last HTTP request.
`setProxy(proxy)`	Sets HTTP proxy to "proxy" value. If this parameter is empty then no proxy is used.

`setHttpAuth(bitmask, username, password)`	Sets enabled HTTP authentication methods (HTTPAUTH_BASIC, HTTPAUTH_DIGEST, HTTPAUTH_NEGOTIATE, HTTPAUTH_NTLM, HTTPAUTH_NONE) in the 'bitmask' parameter.
	The HTTPAUTH_NONE flag allows to disable HTTP authentication.
	Examples:
	`request.setHttpAuth(HTTPAUTH_NTLM | HTTPAUTH_BASIC, username, password)`
	`request.setHttpAuth(HTTPAUTH_NONE)`

`trace(url, data)`	Sends HTTP TRACE request to the URL with optional *data* payload and returns the response.

Example:

```
try {
    Zabbix.log(4, 'jira webhook script value='+value);

    var result = {
        'tags': {
            'endpoint': 'jira'
        }
    },
    params = JSON.parse(value),
    req = new HttpRequest(),
    fields = {},
    resp;

    req.addHeader('Content-Type: application/json');
    req.addHeader('Authorization: Basic '+params.authentication);

    fields.summary = params.summary;
    fields.description = params.description;
    fields.project = {"key": params.project_key};
    fields.issuetype = {"id": params.issue_id};
    resp = req.post('https://jira.example.com/rest/api/2/issue/',
        JSON.stringify({"fields": fields})
    );

    if (req.getStatus() != 201) {
        throw 'Response code: '+req.getStatus();
    }
```

```
    resp = JSON.parse(resp);
    result.tags.issue_id = resp.id;
    result.tags.issue_key = resp.key;
} catch (error) {
    Zabbix.log(4, 'jira issue creation failed json : '+JSON.stringify({"fields": fields}));
    Zabbix.log(4, 'jira issue creation failed : '+error);

    result = {};
}

return JSON.stringify(result);
```

XML

The XML object allows the processing of XML data in the item and low-level discovery preprocessing and webhooks.

Attention:
In order to use XML object, server/proxy must be compiled with libxml2 support.

Method	Description
`XML.query(data, expression)`	Retrieves node content using XPath. Returns null if node is not found. **expression** - an XPath expression; **data** - XML data as a string.
`XML.toJson(data)`	Converts data in XML format to JSON.
`XML.fromJson(object)`	Converts data in JSON format to XML.

Example:

Input:

```
<menu>
    <food type = "breakfast">
        <name>Chocolate</name>
        <price>$5.95</price>
        <description></description>
        <calories>650</calories>
    </food>
</menu>
```

Output:

```
{
    "menu": {
        "food": {
            "@type": "breakfast",
            "name": "Chocolate",
            "price": "$5.95",
            "description": null,
            "calories": "650"
        }
    }
}
```

Serialization rules

XML to JSON conversion will be processed according to the following rules (for JSON to XML conversions reversed rules are applied):

1. XML attributes will be converted to keys that have their names prepended with '@'.

Example:

Input:

```
 <xml foo="FOO">
   <bar>
     <baz>BAZ</baz>
```

```
  </bar>
</xml>
```

Output:

```
{
  "xml": {
    "@foo": "FOO",
    "bar": {
      "baz": "BAZ"
    }
  }
}
```

2. Self-closing elements (<foo/>) will be converted as having 'null' value.

Example:

Input:

```
<xml>
  <foo/>
</xml>
```

Output:

```
{
  "xml": {
    "foo": null
  }
}
```

3. Empty attributes (with "" value) will be converted as having empty string ('') value.

Example:

Input:

```
<xml>
  <foo bar="" />
</xml>
```

Output:

```
{
  "xml": {
    "foo": {
      "@bar": ""
    }
  }
}
```

4. Multiple child nodes with the same element name will be converted to a single key that has an array of values as its value.

Example:

Input:

```
<xml>
  <foo>BAR</foo>
  <foo>BAZ</foo>
  <foo>QUX</foo>
</xml>
```

Output:

```
{
  "xml": {
    "foo": ["BAR", "BAZ", "QUX"]
  }
}
```

5. If a text element has no attributes and no children, it will be converted as a string.

Example:

Input:

```
<xml>
    <foo>BAZ</foo>
</xml>
```

Output:

```
{
  "xml": {
    "foo": "BAZ"
   }
}
```

6. If a text element has no children, but has attributes: text content will be converted to an element with the key '#text' and content as a value; attributes will be converted as described in the serialization rule 1.

Example:

Input:

```
<xml>
  <foo bar="BAR">
    BAZ
  </foo>
</xml>
```

Output:

```
{
  "xml": {
    "foo": {
      "@bar": "BAR",
      "#text": "BAZ"
    }
  }
}
```

Global JavaScript functions

Additional global JavaScript functions have been implemented with Duktape:

- btoa(data) - encodes the data to Base64 string;
- atob(base64_string) - since version 6.4.16, decodes Base64 string as Uint8Array buffer (in Zabbix 6.4.0 - 6.4.15 this function returns a decoded string).

```
try {
    b64 = btoa("test string");
    buffer = atob(b64);

    // Note that decoding logic depends on the data format of the buffer.
    decoded = String.fromCharCode.apply(this, [].slice.call(buffer));
}
catch (error) {
    return {'error.name' : error.name, 'error.message' : error.message};
}
```

- md5(data) - calculates the MD5 hash of the data
- sha256(data) - calculates the SHA256 hash of the data
- hmac('<hash type>',key,data) - returns HMAC hash as hex formatted string; MD5 and SHA256 hash types are supported; key and data parameters support binary data. Examples:
  - `hmac('md5',key,data)`
  - `hmac('sha256',key,data)`
- sign(hash,key,data) - returns calculated signature (RSA signature with SHA-256) as a string, where:
 hash - only 'sha256' is allowed, otherwise an error is thrown;
 key - the private key. It should correspond to PKCS#1 or PKCS#8 standard. The key can be provided in different forms:

- with spaces instead of newlines;
- with escaped or non-escaped '\n's instead of newlines;
- without any newlines as a single-line string;
- as a JSON-formatted string.

The key also can be loaded from a user macro/secret macro/vault.

data - the data that will be signed. It can be a string (binary data also supported) or buffer (Uint8Array/ArrayBuffer).
OpenSSL or GnuTLS is used to calculate the signatures. If Zabbix was built without any of these encryption libraries, an error will be thrown ('missing OpenSSL or GnuTLS library').
This function is supported since Zabbix 6.4.1.

5 CSV to JSON preprocessing

Overview

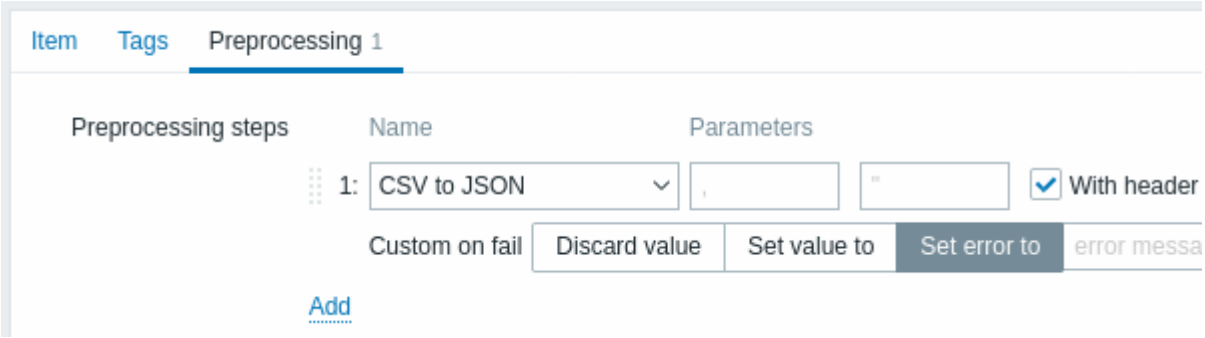
In this preprocessing step it is possible to convert CSV file data into JSON format. It's supported in:

- items (item prototypes)
- low-level discovery rules

Configuration

To configure a CSV to JSON preprocessing step:

- Go to the Preprocessing tab in item/discovery rule configuration
- Click on *Add*
- Select the *CSV to JSON* option



The first parameter allows to set a custom delimiter. Note that if the first line of CSV input starts with "Sep=" and is followed by a single UTF-8 character then that character will be used as the delimiter in case the first parameter is not set. If the first parameter is not set and a delimiter is not retrieved from the "Sep=" line, then a comma is used as a separator.

The second optional parameter allows to set a quotation symbol.

If the *With header row* checkbox is marked, the header line values will be interpreted as column names (see Header processing for more information).

If the *Custom on fail* checkbox is marked, the item will not become unsupported in case of a failed preprocessing step. Additionally custom error handling options may be set: discard the value, set a specified value or set a specified error message.

Header processing

The CSV file header line can be processed in two different ways:

- If the *With header row* checkbox is marked - header line values are interpreted as column names. In this case the column names must be unique and the data row should not contain more columns than the header row;
- If the *With header row* checkbox is not marked - the header line is interpreted as data. Column names are generated automatically (1,2,3,4...)

CSV file example:

```
Nr,Item name,Key,Qty
1,active agent item,agent.hostname,33
"2","passive agent item","agent.version","44"
3,"active,passive agent items",agent.ping,55
```

Note:
A quotation character within a quoted field in the input must be escaped by preceding it with another quotation character.

Processing header line

JSON output when a header line is expected:

```
[
   {
      "Nr":"1",
      "Item name":"active agent item",
      "Key":"agent.hostname",
      "Qty":"33"
   },
   {
      "Nr":"2",
      "Item name":"passive agent item",
      "Key":"agent.version",
      "Qty":"44"
   },
   {
      "Nr":"3",
      "Item name":"active,passive agent items",
      "Key":"agent.ping",
      "Qty":"55"
   }
]
```

No header line processing

JSON output when a header line is not expected:

```
[
   {
      "1":"Nr",
      "2":"Item name",
      "3":"Key",
      "4":"Qty"
   },
   {
      "1":"1",
      "2":"active agent item",
      "3":"agent.hostname",
      "4":"33"
   },
   {
      "1":"2",
      "2":"passive agent item",
      "3":"agent.version",
      "4":"44"
   },
   {
      "1":"3",
      "2":"active,passive agent items",
      "3":"agent.ping",
      "4":"55"
   }
]
```

3 Item types

Overview

Item types cover various methods of acquiring data from your system. Each item type comes with its own set of supported item keys and required parameters.

The following items types are currently offered by Zabbix:

- Zabbix agent checks

- SNMP agent checks
- SNMP traps
- IPMI checks
- Simple checks
  - VMware monitoring
- Log file monitoring
- Calculated items
  - Aggregate calculations
- Zabbix internal checks
- SSH checks
- Telnet checks
- External checks
- Trapper items
- JMX monitoring
- ODBC checks
- Dependent items
- HTTP checks
- Prometheus checks
- Script items

Details for all item types are included in the subpages of this section. Even though item types offer a lot of options for data gathering, there are further options through user parameters or loadable modules.

Some checks are performed by Zabbix server alone (as agent-less monitoring) while others require Zabbix agent or even Zabbix Java gateway (with JMX monitoring).

Attention:
If a particular item type requires a particular interface (like an IPMI check needs an IPMI interface on the host) that interface must exist in the host definition.

Multiple interfaces can be set in the host definition: Zabbix agent, SNMP agent, JMX and IPMI. If an item can use more than one interface, it will search the available host interfaces (in the order: Agent→SNMP→JMX→IPMI) for the first appropriate one to be linked with.

All items that return text (character, log, text types of information) can return whitespace only as well (where applicable) setting the return value to an empty string (supported since 2.0).

1 Zabbix agent

Overview

This section provides details on the item keys that use communication with Zabbix agent for data gathering.

There are passive and active agent checks. When configuring an item, you can select the required type:

- *Zabbix agent* - for passive checks
- *Zabbix agent (active)* - for active checks

Note that all item keys supported by Zabbix agent on Windows are also supported by the new generation Zabbix agent 2. See the additional item keys that you can use with the agent 2 only.

Supported item keys

The item keys that you can use with Zabbix agent are listed below.

The item keys are listed without parameters and additional information. Click on the item key to see the full details.

Item key	Description	Item group
kernel.maxfiles	The maximum number of opened files supported by OS.	Kernel
kernel.maxproc	The maximum number of processes supported by OS.	
kernel.openfiles	The number of currently open file descriptors.	
log	The monitoring of a log file.	Log monitoring
log.count	The count of matched lines in a monitored log file.	
logrt	The monitoring of a log file that is rotated.	
logrt.count	The count of matched lines in a monitored log file that is rotated.	
modbus.get	Reads Modbus data.	Modbus

Item key	Description	Item group
net.dns	Checks if the DNS service is up.	Network
net.dns.record	Performs a DNS query.	
net.if.collisions	The number of out-of-window collisions.	
net.if.discovery	The list of network interfaces.	
net.if.in	The incoming traffic statistics on a network interface.	
net.if.out	The outgoing traffic statistics on a network interface.	
net.if.total	The sum of incoming and outgoing traffic statistics on a network interface.	
net.tcp.listen	Checks if this TCP port is in LISTEN state.	
net.tcp.port	Checks if it is possible to make a TCP connection to the specified port.	
net.tcp.service	Checks if a service is running and accepting TCP connections.	
net.tcp.service.perf	Checks the performance of a TCP service.	
net.tcp.socket.count	Returns the number of TCP sockets that match parameters.	
net.udp.listen	Checks if this UDP port is in LISTEN state.	
net.udp.service	Checks if a service is running and responding to UDP requests.	
net.udp.service.perf	Checks the performance of a UDP service.	
net.udp.socket.count	Returns the number of UDP sockets that match parameters.	
proc.cpu.util	The process CPU utilization percentage.	Processes
proc.get	The list of OS processes and their parameters.	
proc.mem	The memory used by the process in bytes.	
proc.num	The number of processes.	
sensor	Hardware sensor reading.	Sensors
system.boottime	The system boot time.	System
system.cpu.discovery	The list of detected CPUs/CPU cores.	
system.cpu.intr	The device interrupts.	
system.cpu.load	The CPU load.	
system.cpu.num	The number of CPUs.	
system.cpu.switches	The count of context switches.	
system.cpu.util	The CPU utilization percentage.	
system.hostname	The system host name.	
system.hw.chassis	The chassis information.	
system.hw.cpu	The CPU information.	
system.hw.devices	The listing of PCI or USB devices.	
system.hw.macaddr	The listing of MAC addresses.	
system.localtime	The system time.	
system.run	Run the specified command on the host.	
system.stat	The system statistics.	
system.sw.arch	The software architecture information.	
system.sw.os	The operating system information.	
system.sw.os.get	Detailed information about the operating system (version, type, distribution name, minor and major version, etc).	
system.sw.packages	The listing of installed packages.	
system.sw.packages.get	A detailed listing of installed packages.	
system.swap.in	The swap-in (from device into memory) statistics.	
system.swap.out	The swap-out (from memory onto device) statistics.	
system.swap.size	The swap space size in bytes or in percentage from total.	
system.uname	Identification of the system.	
system.uptime	The system uptime in seconds.	
system.users.num	The number of users logged in.	
vfs.dev.discovery	The list of block devices and their type.	Virtual file systems
vfs.dev.read	The disk read statistics.	
vfs.dev.write	The disk write statistics.	
vfs.dir.count	The directory entry count.	
vfs.dir.get	The directory entry list.	
vfs.dir.size	The directory size.	
vfs.file.cksum	The file checksum, calculated by the UNIX cksum algorithm.	
vfs.file.contents	Retrieving the contents of a file.	
vfs.file.exists	Checks if the file exists.	
vfs.file.get	Returns information about a file.	
vfs.file.md5sum	The MD5 checksum of file.	
vfs.file.owner	Retrieves the owner of a file.	

Item key	Description	Item group
vfs.file.permissions	Returns a 4-digit string containing the octal number with UNIX permissions.	
vfs.file.regexp	Retrieve a string in the file.	
vfs.file.regmatch	Find a string in the file.	
vfs.file.size	The file size.	
vfs.file.time	The file time information.	
vfs.fs.discovery	The list of mounted filesystems with their type and mount options.	
vfs.fs.get	The list of mounted filesystems with their type, available disk space, inode statistics and mount options.	
vfs.fs.inode	The number or percentage of inodes.	
vfs.fs.size	The disk space in bytes or in percentage from total.	
vm.memory.size	The memory size in bytes or in percentage from total.	Virtual memory
web.page.get	Get the content of a web page.	Web monitoring
web.page.perf	The loading time of a full web page.	
web.page.regexp	Find a string on the web page.	
agent.hostmetadata	The agent host metadata.	Zabbix
agent.hostname	The agent host name.	
agent.ping	The agent availability check.	
agent.variant	The variant of Zabbix agent (Zabbix agent or Zabbix agent 2).	
agent.version	The version of Zabbix agent.	
zabbix.stats	Returns a set of Zabbix server or proxy internal metrics remotely.	
zabbix.stats	Returns the number of monitored items in the queue which are delayed on Zabbix server or proxy remotely.	

Supported platforms

Except where specified differently in the item details, the agent items (and all parameters) are supported on:

- **Linux**
- **FreeBSD**
- **Solaris**
- **HP-UX**
- **AIX**
- **Tru64**
- **MacOS X**
- **OpenBSD**
- **NetBSD**

Many agent items are also supported on **Windows**. See the Windows agent item page for details.

Item key details

Parameters without angle brackets are mandatory. Parameters marked with angle brackets **< >** are optional.

kernel.maxfiles

 The maximum number of opened files supported by OS.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, MacOS X, OpenBSD, NetBSD.

kernel.maxproc

 The maximum number of processes supported by OS.
 Return value: *Integer*.
 Supported platforms: Linux 2.6 and later, FreeBSD, Solaris, MacOS X, OpenBSD, NetBSD.

kernel.openfiles

 The number of currently open file descriptors.
 Return value: *Integer*.
 Supported platforms: Linux (the item may work on other UNIX-like platforms).

log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent dir>]

 The monitoring of a log file.
 Return value: *Log*.
 See supported platforms.

Parameters:

- **file** - the full path and name of a log file;
- **regexp** - a regular expression describing the required pattern;

- **encoding** - the code page identifier;
- **maxlines** - the maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in zabbix_agentd.conf;
- **mode** - possible values: *all* (default) or *skip* - skip processing of older data (affects only newly created items);
- **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an **\N** (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups);
- **maxdelay** - the maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0.0 - ignore older lines in order to get the most recent lines analyzed within "maxdelay" seconds. Read the maxdelay notes before using it!
- **options** - additional options:
mtime-noreread - non-unique records, reread only if the file size changes (ignore modification time change). (This parameter is deprecated since 5.0.2, because now mtime is ignored.)
- **persistent dir** (only in zabbix_agentd on Unix systems; not supported in Zabbix agent 2) - the absolute pathname of directory where to store persistent files. See also additional notes on persistent files.

Comments:

- The item must be configured as an active check;
- If the file is missing or permissions do not allow access, the item turns unsupported;
- If `output` is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the `output` parameter is ignored.
- Content extraction using the `output` parameter takes place on the agent.

Examples:

```
log[/var/log/syslog]
log[/var/log/syslog,error]
log[/home/zabbix/logs/logfile,,,100]
```

Example of using the `output` parameter for extracting a number from log record:

```
log[/app1/app.log,"task run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors",,,,\1] #this item will
```

Example of using the `output` parameter for rewriting a log record before sending to server:

```
log[/app1/app.log,"([0-9 :-]+) task run ([0-9.]+) sec, processed ([0-9]+) records, ([0-9]+) errors",,,,"\1
```

log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent dir>]

 The count of matched lines in a monitored log file.
 Return value: *Integer*.
 See supported platforms.

Parameters:

- **file** - the full path and name of log file;
- **regexp** - a regular expression describing the required pattern;
- **encoding** - the code page identifier;
- **maxproclines** - the maximum number of new lines per second the agent will analyze (cannot exceed 10000). The default value is 10*'MaxLinesPerSecond' in zabbix_agentd.conf.
- **mode** - possible values: *all* (default) or *skip* - skip processing of older data (affects only newly created items).
- **maxdelay** - the maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0.0 - ignore older lines in order to get the most recent lines analyzed within "maxdelay" seconds. Read the maxdelay notes before using it!
- **options** - additional options:
mtime-noreread - non-unique records, reread only if the file size changes (ignore modification time change). (This parameter is deprecated since 5.0.2, because now mtime is ignored.)
- **persistent dir** (only in zabbix_agentd on Unix systems; not supported in Zabbix agent 2) - the absolute pathname of directory where to store persistent files. See also additional notes on persistent files.

Comments:

- The item must be configured as an active check;
- Matching lines are counted in the new lines since the last log check by the agent, and thus depend on the item update interval;
- If the file is missing or permissions do not allow access, the item turns unsupported.

logrt[file regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent dir>]

 The monitoring of a log file that is rotated.
 Return value: *Log*.
 See supported platforms.

Parameters:

- **file regexp** - the absolute path to file, with the file name specified using a regular expression. Note that the regular expression applies only to the file name and does not need to match the entire name (e.g., /path/to/agent will match zabbix_agentd.log).
- **regexp** - a regular expression describing the required content pattern.
- **encoding** - the code page identifier.
- **maxlines** - the maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of 'MaxLinesPerSecond' in zabbix_agentd.conf.
- **mode** - possible values: *all* (default) or *skip* - skip processing of older data (affects only newly created items).
- **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an **\N** (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).
- **maxdelay** - the maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0.0 - ignore older lines in order to get the most recent lines analyzed within "maxdelay" seconds. Read the maxdelay notes before using it!
- **options** - the type of log file rotation and other options. Possible values:
rotate (default),
copytruncate - note that *copytruncate* cannot be used together with *maxdelay*. In this case *maxdelay* must be 0 or not specified; see copytruncate notes,
mtime-reread - non-unique records, reread if modification time or size changes (default),
mtime-noreread - non-unique records, reread only if the size changes (ignore modification time change).
- **persistent dir** (only in zabbix_agentd on Unix systems; not supported in Zabbix agent 2) - the absolute pathname of directory where to store persistent files. See also additional notes on persistent files.

Comments:

- The item must be configured as an active check;
- Log rotation is based on the last modification time of files;
- Note that logrt is designed to work with one currently active log file, with several other matching inactive files rotated. If, for example, a directory has many active log files, a separate logrt item should be created for each one. Otherwise if one logrt item picks up too many files it may lead to exhausted memory and a crash of monitoring.
- If `output` is left empty - the whole line containing the matched text is returned. Note that all global regular expression types except 'Result is TRUE' always return the whole matched line and the `output` parameter is ignored.
- Content extraction using the `output` parameter takes place on the agent.

Examples:

```
logrt["/home/zabbix/logs/^logfile[0-9]{1,3}$",,,100] #this item will match a file like "logfile1" (will no
logrt["/home/user/^logfile_.*_[0-9]{1,3}$","pattern_to_match","UTF-8",100] #this item will collect data fr
```

Example of using the `output` parameter for extracting a number from log record:

```
logrt[/app1/^test.*log$,"task run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors",,,,\1] #this ite
```

Example of using the `output` parameter for rewriting a log record before sending to server:

```
logrt[/app1/^test.*log$,"([0-9 :-]+) task run ([0-9.]+) sec, processed ([0-9]+) records, ([0-9]+) errors",
```

logrt.count[file regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent dir>]

 The count of matched lines in a monitored log file that is rotated.
 Return value: *Integer*.
 See supported platforms.

Parameters:

- **file regexp** - the absolute path to file, with the file name specified using a regular expression. Note that the regular expression applies only to the file name and does not need to match the entire name (e.g., /path/to/agent will match zabbix_agentd.log).
- **regexp** - a regular expression describing the required pattern.
- **encoding** - the code page identifier.
- **maxproclines** - the maximum number of new lines per second the agent will analyze (cannot exceed 10000). The default value is 10*'MaxLinesPerSecond' in zabbix_agentd.conf.
- **mode** - possible values: *all* (default) or *skip* - skip processing of older data (affects only newly created items).
- **maxdelay** - the maximum delay in seconds. Type: float. Values: 0 - (default) never ignore log file lines; > 0.0 - ignore older lines in order to get the most recent lines analyzed within "maxdelay" seconds. Read the maxdelay notes before using it!
- **options** - the type of log file rotation and other options. Possible values:
rotate (default),
copytruncate - note that *copytruncate* cannot be used together with *maxdelay*. In this case *maxdelay* must be 0 or not specified; see copytruncate notes,
mtime-reread - non-unique records, reread if modification time or size changes (default),
mtime-noreread - non-unique records, reread only if the size changes (ignore modification time change).
- **persistent dir** (only in zabbix_agentd on Unix systems; not supported in Zabbix agent 2) - the absolute pathname of directory where to store persistent files. See also additional notes on persistent files.

Comments:

- The item must be configured as an active check;
- Matching lines are counted in the new lines since the last log check by the agent, and thus depend on the item update interval;
- Log rotation is based on the last modification time of files..

modbus.get[endpoint,<slave id>,<function>,<address>,<count>,<type>,<endianness>,<offset>]

 Reads Modbus data.
 Return value: *JSON object*.
 Supported platforms: Linux.

Parameters:

- **endpoint** - the endpoint defined as `protocol://connection_string`;
- **slave id** - the slave ID;
- **function** - the Modbus function;
- **address** - the address of first registry, coil or input;
- **count** - the number of records to read;
- **type** - the type of data;
- **endianness** - the endianness configuration;
- **offset** - the number of registers, starting from 'address', the results of which will be discarded.

See a detailed description of parameters.

net.dns[<ip>,name,<type>,<timeout>,<count>,<protocol>]

 Checks if the DNS service is up.
 Return values: 0 - DNS is down (server did not respond or DNS resolution failed); 1 - DNS is up.
 See supported platforms.

Parameters:

- **ip** - the IP address of DNS server (leave empty for the default DNS server, ignored on Windows unless using Zabbix agent 2);
- **name** - the DNS name to query;
- **type** - the record type to be queried (default is *SOA*);
- **timeout** (ignored on Windows unless using Zabbix agent 2) - the timeout for the request in seconds (default is 1 second);
- **count** (ignored on Windows unless using Zabbix agent 2) - the number of tries for the request (default is 2);
- **protocol** - the protocol used to perform DNS queries: *udp* (default) or *tcp*.

Comments:

- The possible values for `type` are: *ANY*, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (not supported for Zabbix agent on Windows, Zabbix agent 2 on all OS), *HINFO*, *MINFO*, *TXT*, *SRV*
- Internationalized domain names are not supported, please use IDNA encoded names instead.

Example:

`net.dns[198.51.100.1,example.com,MX,2,1]`

net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]

 Performs a DNS query.
 Return value: a character string with the required type of information.
 See supported platforms.

Parameters:

- **ip** - the IP address of DNS server (leave empty for the default DNS server, ignored on Windows unless using Zabbix agent 2);
- **name** - the DNS name to query;
- **type** - the record type to be queried (default is *SOA*);
- **timeout** (ignored on Windows unless using Zabbix agent 2) - the timeout for the request in seconds (default is 1 second);
- **count** (ignored on Windows unless using Zabbix agent 2) - the number of tries for the request (default is 2);
- **protocol** - the protocol used to perform DNS queries: *udp* (default) or *tcp*.

Comments:

- The possible values for `type` are:
ANY, *A*, *NS*, *CNAME*, *MB*, *MG*, *MR*, *PTR*, *MD*, *MF*, *MX*, *SOA*, *NULL*, *WKS* (not supported for Zabbix agent on Windows, Zabbix agent 2 on all OS), *HINFO*, *MINFO*, *TXT*, *SRV*
- Internationalized domain names are not supported, please use IDNA encoded names instead.

Example:

`net.dns.record[198.51.100.1,example.com,MX,2,1]`

net.if.collisions[if]

 The number of out-of-window collisions.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, Solaris, AIX, MacOS X, OpenBSD, NetBSD. Root privileges are required on NetBSD.

Parameter:

- **if** - network interface name

net.if.discovery

 The list of network interfaces. Used for low-level discovery.
 Return value: *JSON object*.
 Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, OpenBSD, NetBSD.

net.if.in[if,<mode>]

 The incoming traffic statistics on a network interface.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, Solaris[5], HP-UX, AIX, MacOS X, OpenBSD, NetBSD. Root privileges are required on NetBSD.

Parameters:

- **if** - network interface name (Unix); network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows);
- **mode** - possible values:
bytes - number of bytes (default)
packets - number of packets
errors - number of errors
dropped - number of dropped packets
overruns (fifo) - the number of FIFO buffer errors
frame - the number of packet framing errors
compressed - the number of compressed packets received by the device driver
multicast - the number of multicast frames received by the device driver

Comments:

- You may use this key with the *Change per second* preprocessing step in order to get the bytes-per-second statistics;
- The *dropped* mode is supported only on Linux, FreeBSD, HP-UX, MacOS X, OpenBSD, NetBSD;
- The *overruns*, *frame*, *compressed*, *multicast* modes are supported only on Linux;
- On HP-UX this item does not provide details on loopback interfaces (e.g. lo0).

Examples:

```
net.if.in[eth0]
net.if.in[eth0,errors]
```

net.if.out[if,<mode>]

 The outgoing traffic statistics on a network interface.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, Solaris[5], HP-UX, AIX, MacOS X, OpenBSD, NetBSD. Root privileges are required on NetBSD.

Parameters:

- **if** - network interface name (Unix); network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows);
- **mode** - possible values:
bytes - number of bytes (default)
packets - number of packets
errors - number of errors
dropped - number of dropped packets
overruns (fifo) - the number of FIFO buffer errors
collisions (colls) - the number of collisions detected on the interface
carrier - the number of carrier losses detected by the device driver
compressed - the number of compressed packets transmitted by the device driver

Comments:

- You may use this key with the *Change per second* preprocessing step in order to get the bytes-per-second statistics;
- The *dropped* mode is supported only on Linux, HP-UX;
- The *overruns*, *collision*, *carrier*, *compressed* modes are supported only on Linux;
- On HP-UX this item does not provide details on loopback interfaces (e.g. lo0).

Examples:

```
net.if.out[eth0]
net.if.out[eth0,errors]
```

net.if.total[if,<mode>]

 The sum of incoming and outgoing traffic statistics on a network interface.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, Solaris[5], HP-UX, AIX, MacOS X, OpenBSD, NetBSD. Root privileges are required on NetBSD.

Parameters:

- **if** - network interface name (Unix); network interface full description or IPv4 address; or, if in braces, network interface GUID (Windows);

- **mode** - possible values:
bytes - number of bytes (default)
packets - number of packets
errors - number of errors
dropped - number of dropped packets
overruns (fifo) - the number of FIFO buffer errors
collisions (colls) - the number of collisions detected on the interface
compressed - the number of compressed packets transmitted or received by the device driver

Comments:

- You may use this key with the *Change per second* preprocessing step in order to get the bytes-per-second statistics;
- The *dropped* mode is supported only on Linux, HP-UX. Dropped packets are supported only if both `net.if.in` and `net.if.out` work for dropped packets on your platform.
- The *overruns*, *collision*, *compressed* modes are supported only on Linux;
- On HP-UX this item does not provide details on loopback interfaces (e.g. lo0).

Examples:

```
net.if.total[eth0]
net.if.total[eth0,errors]
```

net.tcp.listen[port]

 Checks if this TCP port is in LISTEN state.
 Return values: 0 - it is not in LISTEN state; 1 - it is in LISTEN state.
 Supported platforms: Linux, FreeBSD, Solaris, MacOS X.

Parameter:

- **port** - TCP port number

On Linux kernels 2.6.14 and above, the information about listening TCP sockets is obtained from the kernel's NETLINK interface, if possible. Otherwise, the information is retrieved from /proc/net/tcp and /roc/net/tcp6 files.

Example:

```
net.tcp.listen[80]
```

net.tcp.port[<ip>,port]

 Checks if it is possible to make a TCP connection to the specified port.
 Return values: 0 - cannot connect; 1 - can connect.
 See supported platforms.

Parameters:

- **ip** - the IP address or DNS name (default is 127.0.0.1);
- **port** - the port number.

Comments:

- For simple TCP performance testing use `net.tcp.service.perf[tcp,<ip>,<port>]`;
- These checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually).

Example:

```
net.tcp.port[,80] #this item can be used to test the web server availability running on port 80
```

net.tcp.service[service,<ip>,<port>]

 Checks if a service is running and accepting TCP connections.
 Return values: 0 - service is down; 1 - service is running.
 See supported platforms.

Parameters:

- **service** - *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, or *telnet* (see details);
- **ip** - the IP address or DNS name (default is 127.0.0.1);
- **port** - the port number (by default the standard service port number is used).

Comments:

- These checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually);
- Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use `net.tcp.port[]` for checks like these.
- Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2;
- The telnet check looks for a login prompt (':' at the end).

Example:

```
net.tcp.service[ftp,,45] #this item can be used to test the availability of FTP server on TCP port 45
```

net.tcp.service.perf[service,<ip>,<port>]

 Checks the performance of a TCP service.
 Return values: *Float* (0 - service is down; seconds - the number of seconds spent while connecting to the service).
 See supported platforms.

Parameters:

- **service** - *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, or *telnet* (see details);
- **ip** - the IP address or DNS name (default is 127.0.0.1);
- **port** - the port number (by default the standard service port number is used).

Comments:

- Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use `net.tcp.service.perf[tcp,<ip>,<port>]` for checks like these.
- The telnet check looks for a login prompt (':' at the end).

Example:

```
net.tcp.service.perf[ssh] #this item can be used to test the speed of initial response from the SSH server
```

net.tcp.socket.count[<laddr>,<lport>,<raddr>,<rport>,<state>]

 Returns the number of TCP sockets that match parameters.
 Return value: *Integer*.
 Supported platforms: Linux.

Parameters:

- **laddr** - the local IPv4/6 address or CIDR subnet;
- **lport** - the local port number or service name;
- **raddr** - the remote IPv4/6 address or CIDR subnet;
- **rport** - the remote port number or service name;
- **state** - the connection state (*established*, *syn_sent*, *syn_recv*, *fin_wait1*, *fin_wait2*, *time_wait*, *close*, *close_wait*, *last_ack*, *listen*, *closing*).

Example:

```
net.tcp.socket.count[,80,,,established] #the number of connections to local TCP port 80 in the established
```

net.udp.listen[port]

 Checks if this UDP port is in LISTEN state.
 Return values: 0 - it is not in LISTEN state; 1 - it is in LISTEN state.
 Supported platforms: Linux, FreeBSD, Solaris, MacOS X.

Parameter:

- **port** - UDP port number

Example:

```
net.udp.listen[68]
```

net.udp.service[service,<ip>,<port>]

 Checks if a service is running and responding to UDP requests.
 Return values: 0 - service is down; 1 - service is running.
 See supported platforms.

Parameters:

- **service** - *ntp* (see details);
- **ip** - the IP address or DNS name (default is 127.0.0.1);
- **port** - the port number (by default the standard service port number is used).

Example:

```
net.udp.service[ntp,,45] #this item can be used to test the availability of NTP service on UDP port 45
```

net.udp.service.perf[service,<ip>,<port>]

 Checks the performance of a UDP service.
 Return values: *Float* (0 - service is down; seconds - the number of seconds spent waiting for response from the service).
 See supported platforms.

Parameters:

- **service** - *ntp* (see details);
- **ip** - the IP address or DNS name (default is 127.0.0.1);
- **port** - the port number (by default the standard service port number is used).

Example:

`net.udp.service.perf[ntp] #this item can be used to test response time from NTP service`

net.udp.socket.count[<laddr>,<lport>,<raddr>,<rport>,<state>]

 Returns the number of UDP sockets that match parameters.
 Return value: *Integer*.
 Supported platforms: Linux.

Parameters:

- **laddr** - the local IPv4/6 address or CIDR subnet;
- **lport** - the local port number or service name;
- **raddr** - the remote IPv4/6 address or CIDR subnet;
- **rport** - the remote port number or service name;
- **state** - the connection state (*established*, *unconn*).

Example:

`net.udp.socket.count[,,,,established] #returns the number of UDP sockets in the connected state`

proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]

 The process CPU utilization percentage.
 Return value: *Float*.
 Supported platforms: Linux, Solaris[6].

Parameters:

- **name** - the process name (default is *all processes*);
- **user** - the user name (default is *all users*);
- **type** - the CPU utilization type: *total* (default), *user*, or *system*;
- **cmdline** - filter by command line (it is a regular expression);
- **mode** - the data gathering mode: *avg1* (default), *avg5*, or *avg15*;
- **zone** - the target zone: *current* (default) or *all*. This parameter is supported on Solaris only.

Comments:

- The returned value is based on a single CPU core utilization percentage. For example, the CPU utilization of a process fully using two cores is 200%.
- The process CPU utilization data is gathered by a collector which supports the maximum of 1024 unique (by name, user and command line) queries. Queries not accessed during the last 24 hours are removed from the collector.
- When setting the `zone` parameter to *current* (or default) in case the agent has been compiled on a Solaris without zone support, but running on a newer Solaris where zones are supported, then the agent will return NOTSUPPORTED (the agent cannot limit results to only the current zone). However, *all* is supported in this case.

Examples:

```
proc.cpu.util[,root] #CPU utilization of all processes running under the "root" user
proc.cpu.util[zabbix_server,zabbix] #CPU utilization of all zabbix_server processes running under the zabb
```

proc.get[<name>,<user>,<cmdline>,<mode>]

 The list of OS processes and their parameters. Can be used for low-level discovery.
 Return value: *JSON object*.
 Supported platforms: Linux, FreeBSD, Windows, OpenBSD, NetBSD.

Parameters:

- **name** - the process name (default *all processes*);
- **user** - the user name (default *all users*);
- **cmdline** - filter by command line (it is a regular expression). This parameter is not supported for Windows; on other platforms it is not supported if mode is set to 'summary'.
- **mode** - possible values:
process (default), *thread* (not supported for NetBSD), *summary*. See a list of process parameters returned for each mode and OS.

Comments:

- If a value cannot be retrieved, for example, because of an error (process already died, lack of permissions, system call failure), -1 will be returned;
- See notes on selecting processes with `name` and `cmdline` parameters (Linux-specific).

Examples:

```
proc.get[zabbix_server,zabbix,,process] #list of all zabbix_server processes running under the zabbix user
proc.get[java,,,thread] #list of all Java processes, returns one entry per thread
proc.get[,zabbix,,summary] #combined data for processes of each type running under the zabbix user, return
```

proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]

 The memory used by the process in bytes.
 Return value: *Integer* - with `mode` as *max*, *min*, *sum*; *Float* - with `mode` as *avg*
 Supported platforms: Linux, FreeBSD, Solaris, AIX, Tru64, OpenBSD, NetBSD.

Parameters:

- **name** - the process name (default is *all processes*);
- **user** - the user name (default is *all users*);
- **mode** - possible values: *avg*, *max*, *min*, or *sum* (default);
- **cmdline** - filter by command line (it is a regular expression);
- **memtype** - the type of memory used by process

Comments:

- The `memtype` parameter is supported only on Linux, FreeBSD, Solaris[6], AIX;
- When several processes use shared memory, the sum of memory used by processes may result in large, unrealistic values.

See notes on selecting processes with `name` and `cmdline` parameters (Linux-specific).

When this item is invoked from the command line and contains a command line parameter (e.g. using the agent test mode: `zabbix_agentd -t proc.mem[,,,apache2]`), one extra process will be counted, as the agent will count itself.

Examples:

```
proc.mem[,root] #the memory used by all processes running under the "root" user
proc.mem[zabbix_server,zabbix] #the memory used by all zabbix_server processes running under the zabbix us
proc.mem[,oracle,max,oracleZABBIX] #the memory used by the most memory-hungry process running under Oracle
```

proc.num[<name>,<user>,<state>,<cmdline>,<zone>]

 The number of processes.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, Solaris[6], HP-UX, AIX, Tru64, OpenBSD, NetBSD.

Parameters:

- **name** - the process name (default is *all processes*);
- **user** - the user name (default is *all users*);
- **state** - possible values:
all (default),
disk - uninterruptible sleep,
run - running,
sleep - interruptible sleep,
trace - stopped,
zomb - zombie;
- **cmdline** - filter by command line (it is a regular expression);
- **zone** - the target zone: *current* (default), or *all*. This parameter is supported on Solaris only.

Comments:

- The *disk* and *trace* state parameters are supported only on Linux, FreeBSD, OpenBSD, NetBSD;
- When this item is invoked from the command line and contains a command line parameter (e.g. using the agent test mode: `zabbix_agentd -t proc.num[,,,apache2]`), one extra process will be counted, as the agent will count itself;
- When setting the `zone` parameter to *current* (or default) in case the agent has been compiled on a Solaris without zone support, but running on a newer Solaris where zones are supported, then the agent will return NOTSUPPORTED (the agent cannot limit results to only the current zone). However, *all* is supported in this case.
- See notes on selecting processes with `name` and `cmdline` parameters (Linux-specific).

Examples:

```
proc.num[,mysql] #the number of processes running under the mysql user
proc.num[apache2,www-data] #the number of apache2 processes running under the www-data user
proc.num[,oracle,sleep,oracleZABBIX] #the number of processes in sleep state running under Oracle having c
```

sensor[device,sensor,<mode>]

 Hardware sensor reading.
 Return value: *Float*.
 Supported platforms: Linux, OpenBSD.

Parameters:

- **device** - the device name, can be a regular expression if mode is omitted;
- **sensor** - the sensor name, can be a regular expression if mode is omitted;
- **mode** - possible values: *avg*, *max*, or *min* (if this parameter is omitted, device and sensor are treated verbatim).

Comments:

- Reads /proc/sys/dev/sensors on Linux 2.4;
- Reads /sys/class/hwmon on Linux 2.6+. See a more detailed description of sensor item on Linux.
- Reads the *hw.sensors* MIB on OpenBSD.

Example:

```
sensor[w83781d-i2c-0-2d,temp1]
sensor[cpu0,temp0] #the temperature of one CPU
sensor["cpu[0-2]$",temp,avg] #the average temperature of the first three CPUs
```

system.boottime

 The system boot time.
 Return value: *Integer (Unix timestamp)*.
 Supported platforms: Linux, FreeBSD, Solaris, MacOS X, OpenBSD, NetBSD.

system.cpu.discovery

 The list of detected CPUs/CPU cores. Used for low-level discovery.
 Return value: *JSON object*.
 See supported platforms.

system.cpu.intr

 The device interrupts.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD, NetBSD.

system.cpu.load[<cpu>,<mode>]

 The CPU load.
 Return value: *Float*.
 See supported platforms.

Parameters:

- **cpu** - possible values: *all* (default) or *percpu* (the total load divided by online CPU count);
- **mode** - possible values: *avg1* (one-minute average, default), *avg5*, or *avg15*.

The *percpu* parameter is not supported on Tru64.

Example:

```
system.cpu.load[,avg5]
```

system.cpu.num[<type>]

 The number of CPUs.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD.

Parameter:

- **type** - possible values: *online* (default) or *max*

The *max* type parameter is supported only on Linux, FreeBSD, Solaris, MacOS X.

Example:

```
system.cpu.num
```

system.cpu.switches

 The count of context switches.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD, NetBSD.

system.cpu.util[<cpu>,<type>,<mode>,<logical or physical>]

 The CPU utilization percentage.
 Return value: *Float*.
 Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, Tru64, OpenBSD, NetBSD.

Parameters:

- **cpu** - *<CPU number>* or *all* (default);
- **type** - possible values: *user* (default), *idle*, *nice*, *system*, *iowait*, *interrupt*, *softirq*, *steal*, *guest* (on Linux kernels 2.6.24 and above), or *guest_nice* (on Linux kernels 2.6.33 and above);
- **mode** - possible values: *avg1* (one-minute average, default), *avg5*, or *avg15*;
- **logical or physical** - possible values: *logical* (default) or *physical*. This parameter is supported on AIX only.

Comments:

- The *nice* type parameter is supported only on Linux, FreeBSD, HP-UX, Tru64, OpenBSD, NetBSD.
- The *iowait* type parameter is supported only on Linux 2.6 and later, Solaris, AIX.
- The *interrupt* type parameter is supported only on Linux 2.6 and later, FreeBSD, OpenBSD.
- The *softirq*, *steal*, *guest*, *guest_nice* type parameters are supported only on Linux 2.6 and later.
- The *avg5* and *avg15* mode parameters are supported on Linux, FreeBSD, Solaris, HP-UX, AIX, OpenBSD, NetBSD.

Example:

```
system.cpu.util[0,user,avg5]
```

system.hostname[<type>,<transform>]

 The system host name.
 Return value: *String*.
 See supported platforms.

Parameters:

- **type** - possible values: *netbios* (default on Windows), *host* (default on Linux) or *shorthost* (since version 5.4.7; returns part of the hostname before the first dot, a full string for names without dots);
- **transform** - possible values: *none* (default) or *lower* (convert to lowercase).

The value is acquired by taking `nodename` from the uname() system API output.

Examples of returned values:

```
system.hostname → linux-w7x1
system.hostname → example.com
system.hostname[shorthost] → example
```

system.hw.chassis[<info>]

 The chassis information.
 Return value: *String*.
 Supported platforms: Linux.

Parameter:

- **info** - possible values: *full* (default), *model*, *serial*, *type*, or *vendor*

Comments:

- This item key depends on the availability of the SMBIOS table;
- It will try to read the DMI table from sysfs, if sysfs access fails then try reading directly from memory;
- **Root permissions** are required because the value is acquired by reading from sysfs or memory.

Example:

```
system.hw.chassis[full] → Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXXX Desktop
```

system.hw.cpu[<cpu>,<info>]

 The CPU information.
 Return value: *String* or *Integer*.
 Supported platforms: Linux.

Parameters:

- **cpu** - <*CPU number*> or *all* (default);
- **info** - possible values: *full* (default), *curfreq*, *maxfreq*, *model* or *vendor*.

Comments:

- Gathers info from `/proc/cpuinfo` and `/sys/devices/system/cpu/[cpunum]/cpufreq/cpuinfo_max_freq`;
- If a CPU number and *curfreq* or *maxfreq* is specified, a numeric value is returned (Hz).

Example:

```
system.hw.cpu[0,vendor] → AuthenticAMD
```

system.hw.devices[<type>]

 The listing of PCI or USB devices.
 Return value: *Text*.
 Supported platforms: Linux.

Parameter:

- **type** - *pci* (default) or *usb*

Returns the output of either the lspci or lsusb utility (executed without any parameters).

Example:

```
system.hw.devices → 00:00.0 Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge
```

system.hw.macaddr[<interface>,<format>]

 The listing of MAC addresses.
 Return value: *String*.
 Supported platforms: Linux.

Parameters:

- **interface** - *all* (default) or a regular expression;
- **format** - *full* (default) or *short*

Comments:

- Lists MAC addresses of the interfaces whose name matches the given `interface` regular expression (*all* lists for all interfaces);
- If `format` is specified as *short*, interface names and identical MAC addresses are not listed.

Example:

```
system.hw.macaddr["eth0$",full] → [eth0] 00:11:22:33:44:55
```

system.localtime[<type>]

 The system time.
 Return value: *Integer* - with `type` as *utc*; *String* - with `type` as *local*.
 See supported platforms.

Parameters:

- **type** - possible values: *utc* - (default) the time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds or *local* - the time in the 'yyyy-mm-dd,hh:mm:ss.nnn,+hh:mm' format

Must be used as a passive check only.

Example:

```
system.localtime[local] #create an item using this key and then use it to display the host time in the *Cl
```

system.run[command,<mode>]

 Run the specified command on the host.
 Return value: *Text* result of the command or 1 - with `mode` as *nowait* (regardless of the command result).
 See supported platforms.

Parameters:

- **command** - command for execution;
- **mode** - possible values: *wait* - wait end of execution (default) or *nowait* - do not wait.

Comments:

- This item is disabled by default. Learn how to enable them;
- The return value of the item is a standard output together with a standard error produced by the command. Exit code checking is not performed;
- To be processed correctly, the return value of the command must be of `text` data type. An empty result is also allowed;
- The return value is limited to 16MB (including trailing whitespace that is truncated); database limits also apply;
- See also: Command execution.

Example:

```
system.run[ls -l /] #return a detailed file list of the root directory
```

system.stat[resource,<type>]

 The system statistics.
 Return value: *Integer* or *float*.
 Supported platforms: AIX.

Parameters:

- **ent** - the number of processor units this partition is entitled to receive (float);
- **kthr,<type>** - information about kernel thread states:
r - average number of runnable kernel threads (float)
b - average number of kernel threads placed in the Virtual Memory Manager wait queue (float)
- **memory,<type>** - information about the usage of virtual and real memory:
avm - active virtual pages (integer)
fre - size of the free list (integer)
- **page,<type>** - information about page faults and paging activity:
fi - file page-ins per second (float)
fo - file page-outs per second (float)
pi - pages paged in from paging space (float)
po - pages paged out to paging space (float)
fr - pages freed (page replacement) (float)
sr - pages scanned by page-replacement algorithm (float)
- **faults,<type>** - trap and interrupt rate:
in - device interrupts (float)
sy - system calls (float)
cs - kernel thread context switches (float)
- **cpu,<type>** - breakdown of percentage usage of processor time:
us - user time (float)
sy - system time (float)
id - idle time (float)
wa - idle time during which the system had outstanding disk/NFS I/O request(s) (float)
pc - number of physical processors consumed (float)
ec - the percentage of entitled capacity consumed (float)
lbusy - indicates the percentage of logical processor(s) utilization that occurred while executing at the user and system level (float)
app - indicates the available physical processors in the shared pool (float)
- **disk,<type>** - disk statistics:
bps - indicates the amount of data transferred (read or written) to the drive in bytes per second (integer)
tps - indicates the number of transfers per second that were issued to the physical disk/tape (float)|

Comments:

- Take note of the following limitations in these items:
 `system.stat[cpu,app]` - supported only on AIX LPAR of type "Shared"
 `system.stat[cpu,ec]` - supported on AIX LPAR of type "Shared" and "Dedicated" ("Dedicated"

always returns 100 (percent))
 `system.stat[cpu,lbusy]` - supported only on AIX LPAR of type "Shared"
 `system.stat[cpu,pc]` - supported on AIX LPAR of type "Shared" and "Dedicated"
 `system.stat[ent]` - supported on AIX LPAR of type "Shared" and "Dedicated"

system.sw.arch

 The software architecture information.
 Return value: *String*.
 See supported platforms.

The info is acquired from the `uname()` function.

Example:

`system.sw.arch → i686`

system.sw.os[<info>]

 The operating system information.
 Return value: *String*.
 Supported platforms: Linux, Windows. Supported on Windows since Zabbix 6.4.

Parameter:

- **info** - possible values: *full* (default), *short*, or *name*

The info is acquired from (note that not all files and options are present in all distributions):

- `/proc/version` (*full*) on Linux;
- `/proc/version_signature` (*short*) on Linux;
- the PRETTY_NAME parameter from `/etc/os-release` on Linux-systems supporting it or `/etc/issue.net` (*name*);
- the `HKLM\\SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion` registry key on Windows.

Examples:

```
system.sw.os[short] → Ubuntu 2.6.35-28.50-generic 2.6.35.11
system.sw.os[full] → [s|Windows 10 Enterprise 22621.1.amd64fre.ni_release.220506-1250 Build 22621.963]
```

system.sw.os.get

 Detailed information about the operating system (version, type, distribution name, minor and major version, etc).
 Return value: *JSON object*.
 Supported platforms: Linux, Windows. Supported since Zabbix 6.4.

system.sw.packages[<regexp>,<manager>,<format>]

 The listing of installed packages.
 Return value: *Text*.
 Supported platforms: Linux.

Parameters:

- **regexp** - *all* (default) or a regular expression;
- **manager** - *all* (default) or a package manager;
- **format** - *full* (default) or *short*.

Comments:

- Lists (alphabetically) installed packages whose name matches the given regular expression (*all* lists them all);
- Supported package managers (executed command):
dpkg (dpkg --get-selections)
pkgtool (ls /var/log/packages)
rpm (rpm -qa)
pacman (pacman -Q)
- If `format` is specified as *full*, packages are grouped by package managers (each manager on a separate line beginning with its name in square brackets);
- If `format` is specified as *short*, packages are not grouped and are listed on a single line.

Example:

`system.sw.packages[mini,dpkg,short] → python-minimal, python2.6-minimal, ubuntu-minimal`

system.sw.packages.get[<regexp>,<manager>]

 A detailed listing of installed packages.
 Return value: *JSON object*.
 Supported platforms: Linux. Supported since Zabbix 6.4.

Parameters:

- **regexp** - *all* (default) or a regular expression;
- **manager** - *all* (default) or a package manager (possible values: *rpm*, *dpkg*, *pkgtool*, or *pacman*).

Comments:

- Returns unformatted JSON with the installed packages whose name matches the given regular expression;

- The output is an array of objects each containing the following keys: name, manager, version, size, architecture, buildtime and installtime (see more details).

system.swap.in[<device>,<type>]

 The swap-in (from device into memory) statistics.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, OpenBSD.

Parameters:

- **device** - specify the device used for swapping (Linux only) or *all* (default);
- **type** - possible values: *count* (number of swapins, default on non-Linux platforms), *sectors* (sectors swapped in), or *pages* (pages swapped in, default on Linux).

Comments:

- The source of this information is:
/proc/swaps, /proc/partitions, /proc/stat (Linux 2.4)
/proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
- Note that *pages* will only work if device was not specified;
- The *sectors* type parameter is supported only on Linux.

Example:

`system.swap.in[,pages]`

system.swap.out[<device>,<type>]

 The swap-out (from memory onto device) statistics.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, OpenBSD.

Parameters:

- **device** - specify the device used for swapping (Linux only) or *all* (default);
- **type** - possible values: *count* (number of swapouts, default on non-Linux platforms), *sectors* (sectors swapped out), or *pages* (pages swapped out, default on Linux).

Comments:

- The source of this information is:
`/proc/swaps`, `/proc/partitions`, `/proc/stat` (Linux 2.4)
`/proc/swaps`, `/proc/diskstats`, `/proc/vmstat` (Linux 2.6)
- Note that *pages* will only work if device was not specified;
- The *sectors* type parameter is supported only on Linux.

Example:

`system.swap.out[,pages]`

system.swap.size[<device>,<type>]

 The swap space size in bytes or in percentage from total.
 Return value: *Integer* - for bytes; *Float* - for percentage.
 Supported platforms: Linux, FreeBSD, Solaris, AIX, Tru64, OpenBSD.

Parameters:

- **device** - specify the device used for swapping (FreeBSD only) or *all* (default);
- **type** - possible values: *free* (free swap space, default), *pfree* (free swap space, in percent), *pused* (used swap space, in percent), *total* (total swap space), or *used* (used swap space).

Comments:

- Note that *pfree*, *pused* are not supported on Windows if swap size is 0;
- If device is not specified Zabbix agent will only take into account swap devices (files), the physical memory will be ignored. For example, on Solaris systems the `swap -s` command includes a portion of physical memory and swap devices (unlike `swap -l`).

Example:

`system.swap.size[,pfree]` → `free swap space percentage`

system.uname

 Identification of the system.
 Return value: *String*.
 See supported platforms.

Comments:

- On UNIX the value for this item is obtained with the uname() system call;
- On Windows the item returns the OS architecture, whereas on UNIX it returns the CPU architecture.|

Example (UNIX):

```
system.uname → FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386
```

system.uptime

 The system uptime in seconds.
 Return value: *Integer*.
 Supported platforms: Linux, FreeBSD, Solaris, AIX, MacOS X, OpenBSD, NetBSD. The support on Tru64 is unknown.

In item configuration, use **s** or **uptime** units to get readable values.|

system.users.num

 The number of users logged in.
 Return value: *Integer*.
 See supported platforms.

The **who** command is used on the agent side to obtain the value.

vfs.dev.discovery

 The list of block devices and their type. Used for low-level discovery.
 Return value: *JSON object*.
 Supported platforms: Linux.

vfs.dev.read[<device>,<type>,<mode>]

 The disk read statistics.
 Return value: *Integer* - with `type` in *sectors*, *operations*, *bytes*; *Float* - with `type` in *sps*, *ops*, *bps*.
 Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD.

Parameters:

- **device** - disk device (default is *all* [3]);
- **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, or *bps* (*sps*, *ops*, *bps* stand for: sectors, operations, bytes per second, respectively);
- **mode** - possible values: *avg1* (one-minute average, default), *avg5*, or *avg15*. This parameter is supported only with `type` in: sps, ops, bps.

Comments:

- If using an update interval of three hours or more[2], this item will always return '0';
- The *sectors* and *sps* type parameters are supported only on Linux;
- The *ops* type parameter is supported only on Linux and FreeBSD;
- The *bps* type parameter is supported only on FreeBSD;
- The *bytes* type parameter is supported only on FreeBSD, Solaris, AIX, OpenBSD;
- The `mode` parameter is supported only on Linux, FreeBSD;
- You may use relative device names (for example, `sda`) as well as an optional /dev/ prefix (for example, `/dev/sda`);
- LVM logical volumes are supported;
- The default values of 'type' parameter for different OSes:
AIX - operations
FreeBSD - bps
Linux - sps
OpenBSD - operations
Solaris - bytes
- *sps*, *ops* and *bps* on supported platforms is limited to 1024 devices (1023 individual and one for *all*).

Example:

```
vfs.dev.read[,operations]
```

vfs.dev.write[<device>,<type>,<mode>]

 The disk write statistics.
 Return value: *Integer* - with `type` in *sectors*, *operations*, *bytes*; *Float* - with `type` in *sps*, *ops*, *bps*.
 Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD.

Parameters:

- **device** - disk device (default is *all* [3]);
- **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, or *bps* (*sps*, *ops*, *bps* stand for: sectors, operations, bytes per second, respectively);
- **mode** - possible values: *avg1* (one-minute average, default), *avg5*, or *avg15*. This parameter is supported only with `type` in: sps, ops, bps.

Comments:

- If using an update interval of three hours or more[2], this item will always return '0';
- The *sectors* and *sps* type parameters are supported only on Linux;
- The *ops* type parameter is supported only on Linux and FreeBSD;
- The *bps* type parameter is supported only on FreeBSD;
- The *bytes* type parameter is supported only on FreeBSD, Solaris, AIX, OpenBSD;

- The `mode` parameter is supported only on Linux, FreeBSD;
- You may use relative device names (for example, `sda`) as well as an optional /dev/ prefix (for example, `/dev/sda`);
- LVM logical volumes are supported;
- The default values of 'type' parameter for different OSes:
AIX - operations
FreeBSD - bps
Linux - sps
OpenBSD - operations
Solaris - bytes
- *sps*, *ops* and *bps* on supported platforms is limited to 1024 devices (1023 individual and one for *all*).

Example:

```
vfs.dev.write[,operations]
```

vfs.dir.count[dir,<regex incl>,<regex excl>,<types incl>,<types excl>,<max depth>,<min size>,<max size>,<min age>,<max age>,<regex excl dir>]

 The directory entry count.
 Return value: *Integer*.
 See supported platforms.

Parameters:

- **dir** - the absolute path to directory;
- **regex incl** - a regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value);
- **regex excl** - a regular expression describing the name pattern of the entity (file, directory, symbolic link) to exclude; don't exclude any if empty (default value);
- **types incl** - directory entry types to count, possible values: *file* - regular file, *dir* - subdirectory, *sym* - symbolic link, *sock* - socket, *bdev* - block device, *cdev* - character device, *fifo* - FIFO, *dev* - synonymous with "bdev,cdev", *all* - all types (default), i.e. "file,dir,sym,sock,bdev,cdev,fifo". Multiple types must be separated with comma and quoted.
- **types excl** - directory entry types (see `types incl`) to NOT count. If some entry type is in both `types incl` and `types excl`, directory entries of this type are NOT counted.
- **max depth** - the maximum depth of subdirectories to traverse:
-1 (default) - unlimited,
0 - no descending into subdirectories.
- **min size** - the minimum size (in bytes) for file to be counted. Smaller files will not be counted. Memory suffixes can be used.
- **max size** - the maximum size (in bytes) for file to be counted. Larger files will not be counted. Memory suffixes can be used.
- **min age** - the minimum age (in seconds) of directory entry to be counted. More recent entries will not be counted. Time suffixes can be used.
- **max age** - the maximum age (in seconds) of directory entry to be counted. Entries so old and older will not be counted (modification time). Time suffixes can be used.
- **regex excl dir** - a regular expression describing the name pattern of the directory to exclude. All content of the directory will be excluded (in contrast to regex_excl)

Comments:

- Environment variables, e.g. %APP_HOME%, $HOME and %TEMP% are not supported;
- Pseudo-directories "." and ".." are never counted;
- Symbolic links are never followed for directory traversal;
- Both `regex incl` and `regex excl` are being applied to files and directories when calculating the entry count, but are ignored when picking subdirectories to traverse (if `regex incl` is "(?i)^.+\.zip$" and `max depth` is not set, then all subdirectories will be traversed, but only the files of type zip will be counted).
- The execution time is limited by the timeout value in agent configuration (3 sec). Since large directory traversal may take longer than that, no data will be returned and the item will turn unsupported. Partial count will not be returned.
- When filtering by size, only regular files have meaningful sizes. Under Linux and BSD, directories also have non-zero sizes (a few Kb typically). Devices have zero sizes, e.g. the size of **/dev/sda1** does not reflect the respective partition size. Therefore, when using <`min_size`> and <`max_size`>, it is advisable to specify <`types_incl`> as "*file*", to avoid surprises.

Examples:

```
vfs.dir.count[/dev] #monitors the number of devices in /dev (Linux)
```

vfs.dir.get[dir,<regex incl>,<regex excl>,<types incl>,<types excl>,<max depth>,<min size>,<max size>,<min age>,<max age>,<regex excl dir>]

 The directory entry list.
 Return value: *JSON object*.
 See supported platforms.

Parameters:

- **dir** - the absolute path to directory;
- **regex incl** - a regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value);
- **regex excl** - a regular expression describing the name pattern of the entity (file, directory, symbolic link) to exclude; don't exclude any if empty (default value);

- **types incl** - directory entry types to list, possible values: *file* - regular file, *dir* - subdirectory, *sym* - symbolic link, *sock* - socket, *bdev* - block device, *cdev* - character device, *fifo* - FIFO, *dev* - synonymous with "bdev,cdev", *all* - all types (default), i.e. "file,dir,sym,sock,bdev,cdev,fifo". Multiple types must be separated with comma and quoted.
- **types excl** - directory entry types (see `types incl`) to NOT list. If some entry type is in both `types incl` and `types excl`, directory entries of this type are NOT listed.
- **max depth** - the maximum depth of subdirectories to traverse:
-1 (default) - unlimited,
0 - no descending into subdirectories.
- **min size** - the minimum size (in bytes) for file to be listed. Smaller files will not be listed. Memory suffixes can be used.
- **max size** - the maximum size (in bytes) for file to be listed. Larger files will not be listed. Memory suffixes can be used.
- **min age** - the minimum age (in seconds) of directory entry to be listed. More recent entries will not be listed. Time suffixes can be used.
- **max age** - the maximum age (in seconds) of directory entry to be listed. Entries so old and older will not be listed (modification time). Time suffixes can be used.
- **regex excl dir** - a regular expression describing the name pattern of the directory to exclude. All content of the directory will be excluded (in contrast to `regex excl`)

Comments:

- Environment variables, e.g. %APP_HOME%, $HOME and %TEMP% are not supported;
- Pseudo-directories "." and ".." are never listed;
- Symbolic links are never followed for directory traversal;
- Both `regex incl` and `regex excl` are being applied to files and directories when generating the entry list, but are ignored when picking subdirectories to traverse (if `regex incl` is "(?i)^.+\.zip$" and `max depth` is not set, then all subdirectories will be traversed, but only the files of type zip will be counted).
- The execution time is limited by the timeout value in agent configuration. Since large directory traversal may take longer than that, no data will be returned and the item will turn unsupported. Partial list will not be returned.
- When filtering by size, only regular files have meaningful sizes. Under Linux and BSD, directories also have non-zero sizes (a few Kb typically). Devices have zero sizes, e.g. the size of **/dev/sda1** does not reflect the respective partition size. Therefore, when using `min size` and `max size`, it is advisable to specify `types incl` as "*file*", to avoid surprises.

Examples:

```
vfs.dir.get[/dev] #retrieves the device list in /dev (Linux)
```

vfs.dir.size[dir,<regex incl>,<regex excl>,<mode>,<max depth>,<regex excl dir>]

 The directory size (in bytes).
 Return value: *Integer*.
 Supported platforms: Linux. The item may work on other UNIX-like platforms.

Parameters:

- **dir** - the absolute path to directory;
- **regex incl** - a regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value);
- **regex excl** - a regular expression describing the name pattern of the entity (file, directory, symbolic link) to exclude; don't exclude any if empty (default value);
- **mode** - possible values: *apparent* (default) - gets apparent file sizes rather than disk usage (acts as `du -sb dir`), *disk* - gets disk usage (acts as `du -s -B1 dir`). Unlike the du command, the vfs.dir.size item takes hidden files in account when calculating the directory size (acts as `du -sb .[^.]* *` within dir).
- **max depth** - the maximum depth of subdirectories to traverse: **-1** (default) - unlimited, **0** - no descending into subdirectories.
- **regex excl dir** - a regular expression describing the name pattern of the directory to exclude. All content of the directory will be excluded (in contrast to `regex excl`)

Comments:

- Only directories with at least the read permission for *zabbix* user are calculated. For directories with read permission only, the size of the directory itself is calculated. Directories with read & execute permissions are calculated including contents.
- With large directories or slow drives this item may time out due to the Timeout setting in agent and server/proxy configuration files. Increase the timeout values as necessary.
- The file size limit depends on large file support.

Examples:

```
vfs.dir.size[/tmp,log] #calculates the size of all files in /tmp containing 'log' in their names
vfs.dir.size[/tmp,log,^.+\.old$] #calculates the size of all files in /tmp containing 'log' in their names
```

vfs.file.cksum[file,<mode>]

 The file checksum, calculated by the UNIX cksum algorithm.
 Return value: *Integer* - with `mode` as *crc32*, *String* - with `mode` as *md5*, *sha256*.
 See supported platforms.

Parameters:

- **file** - the full path to file;
- **mode** - *crc32* (default), *md5*, or *sha256*.

The file size limit depends on large file support.

Example:

```
vfs.file.cksum[/etc/passwd]
```

Example of returned values (crc32/md5/sha256 respectively):

```
675436101
9845acf68b73991eb7fd7ee0ded23c44
ae67546e4aac995e5c921042d0cf0f1f7147703aa42bfbfb65404b30f238f2dc
```

vfs.file.contents[file,<encoding>]

 Retrieving the contents of a file[7].
 Return value: *Text*.
 See supported platforms.

Parameters:

- **file** - the full path to file;
- **encoding** - the code page identifier.

Comments:

- The return value is limited to 16MB (including trailing whitespace that is truncated); database limits also apply;
- An empty string is returned if the file is empty or contains LF/CR characters only;
- The byte order mark (BOM) is excluded from the output.

Example:

```
vfs.file.contents[/etc/passwd]
```

vfs.file.exists[file,<types incl>,<types excl>]

 Checks if the file exists.
 Return value: 0 - not found; 1 - file of the specified type exists.
 See supported platforms.

Parameters:

- **file** - the full path to file;
- **types incl** - the list of file types to include, possible values: *file* (regular file, default (if types_excl is not set)), *dir* (directory), *sym* (symbolic link), *sock* (socket), *bdev* (block device), *cdev* (character device), *fifo* (FIFO), *dev* (synonymous with "bdev,cdev"), *all* (all mentioned types, default if types_excl is set).
- **types excl** - the list of file types to exclude, see types_incl for possible values (by default no types are excluded)

Comments:

- Multiple types must be separated with a comma and the entire set enclosed in quotes "";
- If the same type is in both <types_incl> and <types_excl>, files of this type are excluded;
- The file size limit depends on large file support.

Examples:

```
vfs.file.exists[/tmp/application.pid]
vfs.file.exists[/tmp/application.pid,"file,dir,sym"]
vfs.file.exists[/tmp/application_dir,dir]
```

vfs.file.get[file]

 Returns information about a file.
 Return value: *JSON object*.
 See supported platforms.

Parameter:

- **file** - the full path to file

Comments:

- Supported file types on UNIX-like systems: regular file, directory, symbolic link, socket, block device, character device, FIFO.
- The file size limit depends on large file support.

Example:

```
vfs.file.get[/etc/passwd] #return a JSON with information about the /etc/passwd file (type, user, permissi
```

vfs.file.md5sum[file]

 The MD5 checksum of file.
 Return value: Character string (MD5 hash of the file).
 See supported platforms.

Parameter:

- **file** - the full path to file

The file size limit depends on large file support.

Example:

```
vfs.file.md5sum[/usr/local/etc/zabbix_agentd.conf]
```

Example of returned value:

```
b5052decb577e0fffd622d6ddc017e82
```

vfs.file.owner[file,<ownertype>,<resulttype>]

 Retrieves the owner of a file.
 Return value: *String*.
 See supported platforms.

Parameters:

- **file** - the full path to file;
- **ownertype** - *user* (default) or *group* (Unix only);
- **resulttype** - *name* (default) or *id*; for id - return uid/gid on Unix, SID on Windows.

The file size limit depends on large file support.

Example:

```
vfs.file.owner[/tmp/zabbix_server.log] #return the file owner of /tmp/zabbix_server.log
vfs.file.owner[/tmp/zabbix_server.log,,id] #return the file owner ID of /tmp/zabbix_server.log
```

vfs.file.permissions[file]

 Return a 4-digit string containing the octal number with UNIX permissions.
 Return value: *String*.
 Supported platforms: Linux. The item may work on other UNIX-like platforms.

Parameters:

- **file** - the full path to file

The file size limit depends on large file support.

Example:

```
vfs.file.permissions[/etc/passwd] #return permissions of /etc/passwd, for example, '0644'
```

vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]

 Retrieve a string in the file[7].
 Return value: The line containing the matched string, or as specified by the optional `output` parameter.
 See supported platforms.

Parameters:

- **file** - the full path to file;
- **regexp** - a regular expression describing the required pattern;
- **encoding** - the code page identifier;
- **start line** - the number of the first line to search (first line of file by default);
- **end line** - the number of the last line to search (last line of file by default);
- **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an **\N** (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).

Comments:

- The file size limit depends on large file support.
- Only the first matching line is returned;
- An empty string is returned if no line matched the expression;
- The byte order mark (BOM) is excluded from the output;
- Content extraction using the `output` parameter takes place on the agent.

Examples:

```
vfs.file.regexp[/etc/passwd,zabbix]
vfs.file.regexp[/path/to/some/file,"([0-9]+)$",,3,5,\1]
vfs.file.regexp[/etc/passwd,"^zabbix:.:([0-9]+)",,,,\1] → getting the ID of user *zabbix*
```

vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]

 Find a string in the file[7].
 Return values: 0 - match not found; 1 - found.
 See supported platforms.

Parameters:

- **file** - the full path to file;
- **regexp** - a regular expression describing the required pattern;
- **encoding** - the code page identifier;
- **start line** - the number of the first line to search (first line of file by default);
- **end line** - the number of the last line to search (last line of file by default).

Comments:

- The file size limit depends on large file support.
- The byte order mark (BOM) is ignored.

Example:

```
vfs.file.regmatch[/var/log/app.log,error]
```

vfs.file.size[file,<mode>]

 The file size (in bytes).
 Return value: *Integer*.
 See supported platforms.

Parameters:

- **file** - the full path to file;
- **mode** - possible values: *bytes* (default) or *lines* (empty lines are counted, too).

Comments:

- The file must have read permissions for user *zabbix*;
- The file size limit depends on large file support.

Example:

```
vfs.file.size[/var/log/syslog]
```

vfs.file.time[file,<mode>]

 The file time information.
 Return value: *Integer* (Unix timestamp).
 See supported platforms.

Parameters:

- **file** - the full path to file;
- **mode** - possible values:
modify (default) - the last time of modifying file content,
access - the last time of reading file,
change - the last time of changing file properties

The file size limit depends on large file support.

Example:

```
vfs.file.time[/etc/passwd,modify]
```

vfs.fs.discovery

 The list of mounted filesystems with their type and mount options. Used for low-level discovery.
 Return value: *JSON object*.
 Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD.

vfs.fs.get

 The list of mounted filesystems with their type, available disk space, inode statistics and mount options. Can be used for low-level discovery.
 Return value: *JSON object*.
 Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD.

Comments:

- File systems with the inode count equal to zero, which can be the case for file systems with dynamic inodes (e.g. btrfs), are also reported;
- See also: Discovery of mounted filesystems.

vfs.fs.inode[fs,<mode>]

 The number or percentage of inodes.
 Return value: *Integer* - for number; *Float* - for percentage.
 See supported platforms.

Parameters:

- **fs** - the filesystem;
- **mode** - possible values: *total* (default), *free*, *used*, *pfree* (free, percentage), or *pused* (used, percentage).

If the inode count equals zero, which can be the case for file systems with dynamic inodes (e.g. btrfs), the pfree/pused values will be reported as ”100” and ”0” respectively.

Example:

`vfs.fs.inode[/,pfree]`

vfs.fs.size[fs,<mode>]

 The disk space in bytes or in percentage from total.
 Return value: *Integer* - for bytes; *Float* - for percentage.
 See supported platforms.

Parameters:

- **fs** - the filesystem;
- **mode** - possible values: *total* (default), *free*, *used*, *pfree* (free, percentage), or *pused* (used, percentage).

Comments:

- If the filesystem is not mounted, returns the size of a local filesystem where the mount point is located;
- The reserved space of a file system is taken into account and not included when using the *free* mode.

Example:

`vfs.fs.size[/tmp,free]`

vm.memory.size[<mode>]

 The memory size in bytes or in percentage from total.
 Return value: *Integer* - for bytes; *Float* - for percentage.
 See supported platforms.

Parameter:

- **mode** - possible values: *total* (default), *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *slab*, *wired*, *used*, *pused* (used, percentage), *available*, or *pavailable* (available, percentage).

Comments:

- This item accepts three categories of parameters:
1) *total* - total amount of memory
2) platform-specific memory types: *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *slab*, *wired*
3) user-level estimates on how much memory is used and available: *used*, *pused*, *available*, *pavailable*
- The *active* mode parameter is supported only on FreeBSD, HP-UX, MacOS X, OpenBSD, NetBSD;
- The *anon*, *exec*, *file* mode parameters are supported only on NetBSD;
- The *buffers* mode parameter is supported only on Linux, FreeBSD, OpenBSD, NetBSD;
- The *cached* mode parameter is supported only on Linux, FreeBSD, AIX, OpenBSD, NetBSD;
- The *inactive*, *wired* mode parameters are supported only on FreeBSD, MacOS X, OpenBSD, NetBSD;
- The *pinned* mode parameter is supported only on AIX;
- The *shared* mode parameter is supported only on Linux 2.4, FreeBSD, OpenBSD, NetBSD;
- See also additional details for this item.

Example:

`vm.memory.size[pavailable]`

web.page.get[host,<path>,<port>]

 Get the content of a web page.
 Return value: Web page source as text (including headers).
 See supported platforms.

Parameters:

- **host** - the hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[4]. A missing scheme will be treated as *http*. If a URL is specified `path` and `port` must be empty. Specifying user name/password when connecting to servers that require authentication, for example: `http://user:password@www.example.com` is only possible with cURL support [4]. Punycode is supported in hostnames.
- **path** - the path to an HTML document (default is /);

- **port** - the port number (default is 80 for HTTP)

Comments:

- This item turns unsupported if the resource specified in `host` does not exist or is unavailable;
- `host` can be a hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled.

Example:

```
web.page.get[www.example.com,index.php,80]
web.page.get[https://www.example.com]
web.page.get[https://blog.example.com/?s=zabbix]
web.page.get[localhost:80]
web.page.get["[::1]/server-status"]
```

web.page.perf[host,<path>,<port>]

 The loading time of a full web page (in seconds).
 Return value: *Float*.
 See supported platforms.

Parameters:

- **host** - the hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[4]. A missing scheme will be treated as *http*. If a URL is specified `path` and `port` must be empty. Specifying user name/password when connecting to servers that require authentication, for example: `http://user:password@www.example.com` is only possible with cURL support [4]. Punycode is supported in hostnames.
- **path** - the path to an HTML document (default is /);
- **port** - the port number (default is 80 for HTTP)

Comments:

- This item turns unsupported if the resource specified in `host` does not exist or is unavailable;
- `host` can be a hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled.

Example:

```
web.page.perf[www.example.com,index.php,80]
web.page.perf[https://www.example.com]
```

web.page.regexp[host,<path>,<port>,regexp,<length>,<output>]

 Find a string on the web page.
 Return value: The matched string, or as specified by the optional `output` parameter.
 See supported platforms.

Parameters:

- **host** - the hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[4]. A missing scheme will be treated as *http*. If a URL is specified `path` and `port` must be empty. Specifying user name/password when connecting to servers that require authentication, for example: `http://user:password@www.example.com` is only possible with cURL support [4]. Punycode is supported in hostnames.
- **path** - the path to an HTML document (default is /);
- **port** - the port number (default is 80 for HTTP)
- **regexp** - a regular expression describing the required pattern;
- **length** - the maximum number of characters to return;
- **output** - an optional output formatting template. The **\0** escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an **\N** (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured groups).

Comments:

- This item turns unsupported if the resource specified in `host` does not exist or is unavailable;
- `host` can be a hostname, domain name, IPv4 or IPv6 address. But for IPv6 address Zabbix agent must be compiled with IPv6 support enabled.
- Content extraction using the `output` parameter takes place on the agent.

Example:

```
web.page.regexp[www.example.com,index.php,80,OK,2]
web.page.regexp[https://www.example.com,,,OK,2]|
```

agent.hostmetadata

 The agent host metadata.
 Return value: *String*.
 See supported platforms.

Returns the value of HostMetadata or HostMetadataItem parameters, or empty string if none are defined.

agent.hostname

 The agent host name.
 Return value: *String*.
 See supported platforms.

Returns:

- As passive check - the name of the first host listed in the Hostname parameter of the agent configuration file;
- As active check - the name of the current hostname.

agent.ping

 The agent availability check.
 Return value: Nothing - unavailable; 1 - available.
 See supported platforms.

Use the **nodata()** trigger function to check for host unavailability.

agent.variant

 The variant of Zabbix agent (Zabbix agent or Zabbix agent 2).
 Return value: 1 - Zabbix agent; 2 - Zabbix agent 2.
 See supported platforms.

agent.version

 The version of Zabbix agent.
 Return value: *String*.
 See supported platforms.

Example of returned value:

`6.0.3`

zabbix.stats[<ip>,<port>]

 Returns a set of Zabbix server or proxy internal metrics remotely.
 Return value: *JSON object*.
 See supported platforms.

Parameters:

- **ip** - the IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1);
- **port** - the port of server/proxy to be remotely queried (default is 10051)

Comments:

- A selected set of internal metrics is returned by this item. For details, see Remote monitoring of Zabbix stats;
- Note that the stats request will only be accepted from the addresses listed in the 'StatsAllowedIP' server/proxy parameter on the target instance.

zabbix.stats[<ip>,<port>,queue,<from>,<to>]

 Returns the number of monitored items in the queue which are delayed on Zabbix server or proxy remotely.
 Return value: *JSON object*.
 See supported platforms.

Parameters:

- **ip** - the IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1);
- **port** - the port of server/proxy to be remotely queried (default is 10051)
- **queue** - constant (to be used as is)
- **from** - delayed by at least (default is 6 seconds)
- **to** - delayed by at most (default is infinity)

Note that the stats request will only be accepted from the addresses listed in the 'StatsAllowedIP' server/proxy parameter on the target instance.

Footnotes

[1] A Linux-specific note. Zabbix agent must have read-only access to filesystem */proc*. Kernel patches from www.grsecurity.org limit access rights of non-privileged users.

[2] `vfs.dev.read[]`, `vfs.dev.write[]`: Zabbix agent will terminate "stale" device connections if the item values are not accessed for more than 3 hours. This may happen if a system has devices with dynamically changing paths or if a device gets manually removed. Note also that these items, if using an update interval of 3 hours or more, will always return '0'.

[3] `vfs.dev.read[]`, `vfs.dev.write[]`: If default *all* is used for the first parameter then the key will return summary statistics, including all block devices like sda, sdb, and their partitions (sda1, sda2, sdb3...) and multiple devices (MD raid) based on those block devices/partitions and logical volumes (LVM) based on those block devices/partitions. In such cases returned values should be considered only as relative value (dynamic in time) but not as absolute values.

[4] SSL (HTTPS) is supported only if agent is compiled with cURL support. Otherwise the item will turn unsupported.

[5] The `bytes` and `errors` values are not supported for loopback interfaces on Solaris systems up to and including Solaris 10 6/06 as byte, error and utilization statistics are not stored and/or reported by the kernel. However, if you're monitoring a Solaris system via net-snmp, values may be returned as net-snmp carries legacy code from the cmu-snmp dated as old as 1997 that, upon failing to read byte values from the interface statistics returns the packet counter (which does exist on loopback interfaces) multiplied by an arbitrary value of 308. This makes the assumption that the average length of a packet is 308 octets, which is a very rough estimation as the MTU limit on Solaris systems for loopback interfaces is 8892 bytes. These values should not be assumed to be correct or even closely accurate. They are guestimates. The Zabbix agent does not do any guess work, but net-snmp will return a value for these fields.

[6] The command line on Solaris, obtained from /proc/pid/psinfo, is limited to 80 bytes and contains the command line as it was when the process was started.

[7] `vfs.file.contents[]`, `vfs.file.regexp[]`, `vfs.file.regmatch[]` items can be used for retrieving file contents. If you want to restrict access to specific files with sensitive information, run Zabbix agent under a user that has no access permissions to viewing these files.

Usage with command-line utilities

Note that when testing or using item keys with zabbix_agentd or zabbix_get from the command line you should consider shell syntax too.

For example, if a certain parameter of the key has to be enclosed in double quotes you have to explicitly escape double quotes, otherwise they will be trimmed by the shell as special characters and will not be passed to the Zabbix utility.

Examples:

```
$ zabbix_agentd -t 'vfs.dir.count[/var/log,,,"file,dir",,0]'

$ zabbix_agentd -t vfs.dir.count[/var/log,,,\"file,dir\",,0]
```

Encoding settings

To make sure that the acquired data are not corrupted you may specify the correct encoding for processing the check (e.g. 'vfs.file.contents') in the `encoding` parameter. The list of supported encodings (code page identifiers) may be found in documentation for libiconv (GNU Project) or in Microsoft Windows SDK documentation for "Code Page Identifiers".

If no encoding is specified in the `encoding` parameter the following resolution strategies are applied:

- If encoding is not specified (or is an empty string) it is assumed to be UTF-8, the data is processed "as-is";
- BOM analysis - applicable for items 'vfs.file.contents', 'vfs.file.regexp', 'vfs.file.regmatch'. An attempt is made to determine the correct encoding by using the byte order mark (BOM) at the beginning of the file. If BOM is not present - standard resolution (see above) is applied instead.

Troubleshooting agent items

- If used with the passive agent, *Timeout* value in server configuration may need to be higher than *Timeout* in the agent configuration file. Otherwise the item may not get any value because the server request to agent timed out first.

1 Zabbix agent 2

Zabbix agent 2 supports all item keys supported for Zabbix agent on Unix and Windows. This page provides details on the additional item keys, which you can use with Zabbix agent 2 only, grouped by the plugin they belong to.

The item keys are listed without parameters and additional information. Click on the item key to see the full details.

Item key	Description	Plugin
ceph.df.details	The cluster's data usage and distribution among pools.	Ceph
ceph.osd.stats	Aggregated and per OSD statistics.	
ceph.osd.discovery	The list of discovered OSDs.	
ceph.osd.dump	The usage thresholds and statuses of OSDs.	
ceph.ping	Tests whether a connection to Ceph can be established.	
ceph.pool.discovery	The list of discovered pools.	
ceph.status	The overall cluster's status.	
docker.container_info	Low-level information about a container.	Docker
docker.container_stats	The container resource usage statistics.	
docker.containers	Returns the list of containers.	
docker.containers.discovery	Returns the list of containers. Used for low-level discovery.	
docker.data.usage	Information about the current data usage.	

Item key	Description	Plugin
docker.images	Returns the list of images.	
docker.images.discovery	Returns the list of images. Used for low-level discovery.	
docker.info	The system information.	
docker.ping	Test if the Docker daemon is alive or not.	
memcached.ping	Test if a connection is alive or not.	Memcached
memcached.stats	Gets the output of the STATS command.	
mongodb.collection.stats	Returns a variety of storage statistics for a given collection.	MongoDB
mongodb.collections.discovery	Returns a list of discovered collections.	
mongodb.collections.usage	Returns the usage statistics for collections.	
mongodb.connpool.stats	Returns information regarding the open outgoing connections from the current database instance to other members of the sharded cluster or replica set.	
mongodb.db.stats	Returns the statistics reflecting a given database system state.	
mongodb.db.discovery	Returns a list of discovered databases.	
mongodb.jumbo_chunks.count	Returns the count of jumbo chunks.	
mongodb.oplog.stats	Returns the status of the replica set, using data polled from the oplog.	
mongodb.ping	Test if a connection is alive or not.	
mongodb.rs.config	Returns the current configuration of the replica set.	
mongodb.rs.status	Returns the replica set status from the point of view of the member where the method is run.	
mongodb.server.status	Returns the database state.	
mongodb.sh.discovery	Returns the list of discovered shards present in the cluster.	
mongodb.version	Returns the database server version.	
mqtt.get	Subscribes to a specific topic or topics (with wildcards) of the provided broker and waits for publications.	MQTT
mssql.availability.group.get	Returns availability groups.	MSSQL
mssql.custom.query	Returns the result of a custom query.	
mssql.db.get	Returns all available MSSQL databases.	
mssql.job.status.get	Returns the status of jobs.	
mssql.last.backup.get	Returns the last backup time for all databases.	
mssql.local.db.get	Returns databases that are participating in an Always On availability group and replica (primary or secondary) and are located on the server that the connection was established to.	
mssql.mirroring.get	Returns mirroring info.	
mssql.nonlocal.db.get	Returns databases that are participating in an Always On availability group and replica (primary or secondary) located on other servers (the database is not local to the SQL Server instance that the connection was established to).	
mssql.perfcounter.get	Returns the performance counters.	
mssql.ping	Test if a connection is alive or not.	
mssql.quorum.get	Returns the quorum info.	
mssql.quorum.member.get	Returns the quorum members.	
mssql.replica.get	Returns the replicas.	
mssql.version	Returns the MSSQL version.	
mysql.custom.query	Returns the result of a custom query.	MySQL
mysql.db.discovery	Returns the list of MySQL databases.	
mysql.db.size	The database size in bytes.	
mysql.get_status_variables	Values of the global status variables.	
mysql.ping	Test if a connection is alive or not.	
mysql.replication.discovery	Returns the list of MySQL replications.	
mysql.replication.get_slave_status	The replication status.	
mysql.version	The MySQL version.	
oracle.diskgroups.stats	Returns the Automatic Storage Management (ASM) disk groups statistics.	Oracle
oracle.diskgroups.discovery	Returns the list of ASM disk groups.	
oracle.archive.info	The archive logs statistics.	
oracle.cdb.info	The Container Databases (CDBs) information.	
oracle.custom.query	The result of a custom query.	
oracle.datafiles.stats	Returns the data files statistics.	
oracle.db.discovery	Returns the list of databases.	
oracle.fra.stats	Returns the Fast Recovery Area (FRA) statistics.	
oracle.instance.info	The instance statistics.	
oracle.pdb.info	The Pluggable Databases (PDBs) information.	
oracle.pdb.discovery	Returns the list of PDBs.	

Item key	Description	Plugin
oracle.pga.stats	Returns the Program Global Area (PGA) statistics.	
oracle.ping	Test whether a connection to Oracle can be established.	
oracle.proc.stats	Returns the processes statistics.	
oracle.redolog.info	The log file information from the control file.	
oracle.sga.stats	Returns the System Global Area (SGA) statistics.	
oracle.sessions.stats	Returns the sessions statistics.	
oracle.sys.metrics	Returns a set of system metric values.	
oracle.sys.params	Returns a set of system parameter values.	
oracle.ts.stats	Returns the tablespaces statistics.	
oracle.ts.discovery	Returns a list of tablespaces.	
oracle.user.info	Returns Oracle user information.	
oracle.version	Returns the database server version.	
pgsql.autovacuum.count	The number of autovacuum workers.	PostgreSQL
pgsql.archive	The information about archived files.	
pgsql.bgwriter	The combined number of checkpoints for the database cluster, broken down by checkpoint type.	
pgsql.cache.hit	The PostgreSQL buffer cache hit rate.	
pgsql.connections	Returns connections by type.	
pgsql.custom.query	Returns the result of a custom query.	
pgsql.db.age	The age of the oldest FrozenXID of the database.	
pgsql.db.bloating_tables	The number of bloating tables per database.	
pgsql.db.discovery	The list of PostgreSQL databases.	
pgsql.db.size	The database size in bytes.	
pgsql.dbstat	Collects the statistics per database.	
pgsql.dbstat.sum	The summarized data for all databases in a cluster.	
pgsql.locks	The information about granted locks per database.	
pgsql.oldest.xid	The age of the oldest XID.	
pgsql.ping	Test if a connection is alive or not.	
pgsql.queries	Query metrics by execution time.	
pgsql.replication.count	The number of standby servers.	
pgsql.replication.process	The flush lag, write lag and replay lag per each sender process.	
pgsql.replication.process.discovery	The replication process name discovery.	
pgsql.replication.recovery_role	The recovery status.	
pgsql.replication.status	The status of replication.	
pgsql.replication_lag.b	The replication lag in bytes.	
pgsql.replication_lag.sec	The replication lag in seconds.	
pgsql.uptime	The PostgreSQL uptime in milliseconds.	
pgsql.version	Returns PostgreSQL version.	
pgsql.wal.stat	The WAL statistics.	
redis.config	Gets the configuration parameters of a Redis instance that match the pattern.	Redis
redis.info	Gets the output of the INFO command.	
redis.ping	Test if a connection is alive or not.	
redis.slowlog.count	The number of slow log entries since Redis was started.	
smart.attribute.discovery	Returns a list of S.M.A.R.T. device attributes.	S.M.A.R.T.
smart.disk.discovery	Returns a list of S.M.A.R.T. devices.	
smart.disk.get	Returns all available properties of S.M.A.R.T. devices.	
systemd.unit.get	Returns all properties of a systemd unit.	Systemd
systemd.unit.info	Systemd unit information.	
systemd.unit.discovery	The list of systemd units and their details.	
web.certificate.get	Validates the certificates and returns certificate details.	Web certificates

See also:

- Built-in plugins
- Loadable plugins

Item key details

Parameters without angle brackets are mandatory. Parameters marked with angle brackets **<** **>** are optional.

ceph.df.details[connString,<user>,<apikey>]

 The cluster's data usage and distribution among pools.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the Ceph login credentials.

ceph.osd.stats[connString,<user>,<apikey>]

 Aggregated and per OSD statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the Ceph login credentials.

ceph.osd.discovery[connString,<user>,<apikey>]

 The list of discovered OSDs. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the Ceph login credentials.

ceph.osd.dump[connString,<user>,<apikey>]

 The usage thresholds and statuses of OSDs.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the Ceph login credentials.

ceph.ping[connString,<user>,<apikey>]

 Tests whether a connection to Ceph can be established.
 Return value: *0* - connection is broken (if there is any error presented including AUTH and configuration issues); *1* - connection is successful.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the Ceph login credentials.

ceph.pool.discovery[connString,<user>,<apikey>]

 The list of discovered pools. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the Ceph login credentials.

ceph.status[connString,<user>,<apikey>]

 The overall cluster's status.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the Ceph login credentials.

docker.container_info[<ID>,<info>]

 Low-level information about a container.
 Return value: The output of the ContainerInspect API call serialized as JSON.

Parameters:

- **ID** - the ID or name of the container;
- **info** - the amount of information returned. Supported values: *short* (default) or *full*.

The Agent 2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.

docker.container_stats[<ID>]

 The container resource usage statistics.
 Return value: The output of the ContainerStats API call and CPU usage percentage serialized as JSON.

Parameter:

- **ID** - the ID or name of the container.

The Agent 2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.

docker.containers

 The list of containers.
 Return value: The output of the ContainerList API call serialized as JSON.

The Agent 2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.

docker.containers.discovery[<options>]

 Returns the list of containers. Used for low-level discovery.
 Return value: *JSON object*.

Parameter:

- **options** - specify whether all or only running containers should be discovered. Supported values: *true* - return all containers; *false* - return only running containers (default).

The Agent 2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.

docker.data.usage

 Information about the current data usage.
 Return value: The output of the SystemDataUsage API call serialized as JSON.

The Agent 2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.

docker.images

 Returns the list of images.
 Return value: The output of the ImageList API call serialized as JSON.

The Agent 2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.

docker.images.discovery

 Returns the list of images. Used for low-level discovery.
 Return value: *JSON object*.

The Agent 2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.

docker.info

 The system information.
 Return value: The output of the SystemInfo API call serialized as JSON.

The Agent 2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.

docker.ping

 Test if the Docker daemon is alive or not.
 Return value: *1* - the connection is alive; *0* - the connection is broken.

The Agent 2 user (’zabbix’) must be added to the ’docker’ group for sufficient privileges. Otherwise the check will fail.

memcached.ping[connString,<user>,<password>]

 Test if a connection is alive or not.
 Return value: *1* - the connection is alive; *0* - the connection is broken (if there is any error presented including AUTH and configuration issues).

Parameters:

- **connString** - the URI or session name;
- **user, password** - the Memcached login credentials.

memcached.stats[connString,<user>,<password>,<type>]

 Gets the output of the STATS command.
 Return value: *JSON* - the output is serialized as JSON.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the Memcached login credentials;
- **type** - stat type to be returned: *items*, *sizes*, *slabs* or *settings* (empty by default, returns general statistics).

mongodb.collection.stats[connString,<user>,<password>,<database>,collection]

 Returns a variety of storage statistics for a given collection.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials;
- **database** - the database name (default: admin);
- **collection** - the collection name.

mongodb.collections.discovery[connString,<user>,<password>]

 Returns a list of discovered collections. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mongodb.collections.usage[connString,<user>,<password>]

 Returns the usage statistics for collections.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mongodb.connpool.stats[connString,<user>,<password>]

 Returns information regarding the open outgoing connections from the current database instance to other members of the sharded cluster or replica set.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials;
- **database** - the database name (default: admin);
- **collection** - the collection name.

mongodb.db.stats[connString,<user>,<password>,<database>]

 Returns the statistics reflecting a given database system state.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials;
- **database** - the database name (default: admin).

mongodb.db.discovery[connString,<user>,<password>]

 Returns a list of discovered databases. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mongodb.jumbo_chunks.count[connString,<user>,<password>]

 Returns the count of jumbo chunks.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mongodb.oplog.stats[connString,<user>,<password>]

 Returns the status of the replica set, using data polled from the oplog.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mongodb.ping[connString,<user>,<password>]

 Test if a connection is alive or not.
 Return value: *1* - the connection is alive; *0* - the connection is broken (if there is any error presented including AUTH and configuration issues).

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mongodb.rs.config[connString,<user>,<password>]

 Returns the current configuration of the replica set.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mongodb.rs.status[connString,<user>,<password>]

 Returns the replica set status from the point of view of the member where the method is run.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mongodb.server.status[connString,<user>,<password>]

 Returns the database state.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mongodb.sh.discovery[connString,<user>,<password>]

 Returns the list of discovered shards present in the cluster.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mongodb.version[connString,<user>,<password>]

 Returns the database server version.
 Return value: *String*.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MongoDB login credentials.

mqtt.get[<broker url>,topic,<user>,<password>]

 Subscribes to a specific topic or topics (with wildcards) of the provided broker and waits for publications.
 Return value: Depending on topic content. If wildcards are used, returns topic content as JSON.

Parameters:

- **broker url** - the MQTT broker URL in the format `protocol://host:port` without query parameters (supported protocols: `tcp`, `ssl`, `ws`). If no value is specified, the agent will use `tcp://localhost:1883`. If a protocol or port are omitted, default protocol (`tcp`) or port (1883) will be used;
- **topic** - the MQTT topic (mandatory). Wildcards (+,#) are supported;
- **user, password** - the authentication credentials (if required).

Comments:

- The item must be configured as an active check ('Zabbix agent (active)' item type);
- TLS encryption certificates can be used by saving them into a default location (e.g. `/etc/ssl/certs/` directory for Ubuntu). For TLS, use the `tls://` scheme.

mssql.availability.group.get[URI,<user>,<password>]

 Returns availability groups.
 Return value: *JSON object*.

Parameters:

- **URI** - MSSQL server URI (the only supported schema is *sqlserver://*). Embedded credentials will be ignored;
- **user, password** - username, password to send to protected MSSQL server.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.custom.query[URI,<user>,<password>,queryName,<args...>]

 Returns the result of a custom query.
 Return value: *JSON object*.

Parameters:

- **URI** - MSSQL server URI (the only supported schema is *sqlserver://*). Embedded credentials will be ignored;
- **user, password** - username, password to send to protected MSSQL server;
- **queryName** - name of a custom query configured in `Plugins.MSSQL.CustomQueriesDir` without the .sql extension;
- **args** - one or several comma-separated arguments to pass to a query.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.db.get

 Returns all available MSSQL databases.
 Return value: *JSON object*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.job.status.get

 Returns the status of jobs.
 Return value: *JSON object*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.last.backup.get

 Returns the last backup time for all databases.
 Return value: *JSON object*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.local.db.get

 Returns databases that are participating in an Always On availability group and replica (primary or secondary) and are located on the server that the connection was established to.
 Return value: *JSON object*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.mirroring.get

 Returns mirroring info.
 Return value: *JSON object*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.nonlocal.db.get

 Returns databases that are participating in an Always On availability group and replica (primary or secondary) located on other servers (the database is not local to the SQL Server instance that the connection was established to).
 Return value: *JSON object*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.perfcounter.get

 Returns the performance counters.
 Return value: *JSON object*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.ping

 Ping the database. Test if connection is correctly configured.
 Return value: *1* - alive, *0* - not alive.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.quorum.get

 Returns the quorum info.
 Return value: *JSON object*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.quorum.member.get

 Returns the quorum members.
 Return value: *JSON object*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.replica.get

 Returns the replicas.
 Return value: *JSON object*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mssql.version

 Returns the MSSQL version.
 Return value: *String*.

Supported since Zabbix 6.4.12. For more information see the MSSQL plugin readme.

mysql.custom.query[connString,<user>,<password>,queryName,<args...>]

 Returns the result of a custom query.
 Return value: *JSON object*.

Parameters:

- **connString** - URI or session name;
- **user, password** - MySQL login credentials;
- **queryName** - name of a custom query, must match SQL file name without an extension;
- **args** - one or several comma-separated arguments to pass to a query.

Supported since Zabbix 6.4.6. For more information see the MySQL plugin readme.

mysql.db.discovery[connString,<user>,<password>]

 Returns the list of MySQL databases. Used for low-level discovery.
 Return value: The result of the ”show databases” SQL query in LLD JSON format.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MySQL login credentials.

mysql.db.size[connString,<user>,<password>,<database name>]

 The database size in bytes.
 Return value: Result of the ”select coalesce(sum(data_length + index_length),0) as size from information_schema.tables where table_schema=?” SQL query for specific database in bytes.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MySQL login credentials;
- **database name** - the database name.

mysql.get_status_variables[connString,<user>,<password>]

 Values of the global status variables.
 Return value: Result of the ”show global status” SQL query in JSON format.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MySQL login credentials.

mysql.ping[connString,<user>,<password>]

 Test if a connection is alive or not.
 Return value: *1* - the connection is alive; *0* - the connection is broken (if there is any error presented including AUTH and configuration issues).

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MySQL login credentials.

mysql.replication.discovery[connString,<user>,<password>]

 Returns the list of MySQL replications. Used for low-level discovery.
 Return value: The result of the ”show slave status” SQL query in LLD JSON format.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MySQL login credentials.

mysql.replication.get_slave_status[connString,<user>,<password>,<master host>]

 The replication status.
 Return value: Result of the ”show slave status” SQL query in JSON format.

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MySQL login credentials;
- **master host** - the replication master host name. If none found, an error is returned. If this parameter is not specified, all hosts are returned.

Note that before Zabbix 6.4.8, the ”master host” parameter is ignored and always the first master host is returned.

mysql.version[connString,<user>,<password>]

 The MySQL version.
 Return value: *String* (with the MySQL instance version).

Parameters:

- **connString** - the URI or session name;
- **user, password** - the MySQL login credentials.

oracle.diskgroups.stats[connString,<user>,<password>,<service>,<diskgroup>]

 Returns the Automatic Storage Management (ASM) disk groups statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name;
- **diskgroup** - the name of the ASM disk group to query.

oracle.diskgroups.discovery[connString,<user>,<password>,<service>]

 Returns the list of ASM disk groups. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.archive.info[connString,<user>,<password>,<service>,<destination>]

 The archive logs statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name;
- **destination** - the name of the destination to query.

oracle.cdb.info[connString,<user>,<password>,<service>,<database>]

 The Container Databases (CDBs) information.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name;
- **destination** - the name of the database to query.

oracle.custom.query[connString,<user>,<password>,<service>,queryName,<args...>]

 The result of a custom query.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name;
- **queryName** - the name of a custom query, must match SQL file name without an extension;
- **args** - one or several comma-separated arguments to pass to the query.

oracle.datafiles.stats[connString,<user>,<password>,<service>]

 Returns the data files statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name;
- **diskgroup** - the name of the ASM disk group to query.

oracle.db.discovery[connString,<user>,<password>,<service>]

 Returns the list of databases. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.fra.stats[connString,<user>,<password>,<service>]

 Returns the Fast Recovery Area (FRA) statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.instance.info[connString,<user>,<password>,<service>]

 The instance statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.pdb.info[connString,<user>,<password>,<service>,<database>]

 The Pluggable Databases (PDBs) information.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name;
- **destination** - the name of the database to query.

oracle.pdb.discovery[connString,<user>,<password>,<service>]

 Returns the list of PDBs. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.pga.stats[connString,<user>,<password>,<service>]

 Returns the Program Global Area (PGA) statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.ping[connString,<user>,<password>,<service>]

 Test whether a connection to Oracle can be established.
 Return value: *1* - the connection is successful; *0* - the connection is broken (if there is any error presented including AUTH and configuration issues).

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.proc.stats[connString,<user>,<password>,<service>]

 Returns the processes statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.redolog.info[connString,<user>,<password>,<service>]

 The log file information from the control file.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.sga.stats[connString,<user>,<password>,<service>]

 Returns the System Global Area (SGA) statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.sessions.stats[connString,<user>,<password>,<service>,<lockMaxTime>]

 Returns the sessions statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name;
- **lockMaxTime** - the maximum session lock duration in seconds to count the session as a prolongedly locked. Default: 600 seconds.

oracle.sys.metrics[connString,<user>,<password>,<service>,<duration>]

 Returns a set of system metric values.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name;
- **duration** - the capturing interval (in seconds) of system metric values. Possible values: *60* — long duration (default), *15* — short duration.

oracle.sys.params[connString,<user>,<password>,<service>]

 Returns a set of system parameter values.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.ts.stats[connString,<user>,<password>,<service>,<tablespace>,<type>]

 Returns the tablespaces statistics.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name;
- **tablespace** - name of the tablespace to query. Default (if left empty and `type` is set):
- ”TEMP” (if `type` is set to ”TEMPORARY”);
- ”USERS” (if `type` is set to ”PERMANENT”).
- **type** - the type of the tablespace to query. Default (if `tablespace` is set): ”PERMANENT”.

oracle.ts.discovery[connString,<user>,<password>,<service>]

 Returns a list of tablespaces. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

oracle.user.info[connString,<user>,<password>,<service>,<username>]

 Returns Oracle user information.
 Return value: *JSON object*.

Parameters:

- **connString** - the URI or session name;
- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name;
- **username** - the username for which the information is needed. Lowercase usernames are not supported. Default: current user.

oracle.version[connString,<user>,<password>,<service>]

 Returns the database server version.
 Return value: *String*.

Parameters:

- **connString** - the URI or session name;

- **user** - the Oracle username, supports appending one of the login options `as sysdba`, `as sysoper`, or `as sysasm` in the format `user as sysdba` (a login option is case-insensitive, must not contain a trailing space);
- **password** - the Oracle password;
- **service** - the Oracle service name.

pgsql.autovacuum.count[uri,<username>,<password>,<database name>]

 The number of autovacuum workers.
 Return value: *Integer*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.archive[uri,<username>,<password>,<database name>]

 The information about archived files.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.bgwriter[uri,<username>,<password>,<database name>]

 The combined number of checkpoints for the database cluster, broken down by checkpoint type.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.cache.hit[uri,<username>,<password>,<database name>]

 The PostgreSQL buffer cache hit rate.
 Return value: *Float*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.connections[uri,<username>,<password>,<database name>]

 Returns connections by type.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.custom.query[uri,<username>,<password>,queryName,<args...>]

 Returns the result of a custom query.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **queryName** - the name of a custom query, must match the SQL file name without an extension;
- **args** - one or several comma-separated arguments to pass to a query.

pgsql.db.age[uri,<username>,<password>,<database name>]

 The age of the oldest FrozenXID of the database.
 Return value: *Integer*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.db.bloating_tables[uri,<username>,<password>,<database name>]

 The number of bloating tables per database.
 Return value: *Integer*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.db.discovery[uri,<username>,<password>,<database name>]

 The list of PostgreSQL databases. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.db.size[uri,<username>,<password>,<database name>]

 The database size in bytes.
 Return value: *Integer*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.dbstat[uri,<username>,<password>,<database name>]

 Collects the statistics per database. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.dbstat.sum[uri,<username>,<password>,<database name>]

 The summarized data for all databases in a cluster.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.locks[uri,<username>,<password>,<database name>]

 The information about granted locks per database. Used for low-level discovery.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.oldest.xid[uri,<username>,<password>,<database name>]

 The age of the oldest XID.
 Return value: *Integer*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.ping[uri,<username>,<password>,<database name>]

 Tests whether a connection is alive or not.
 Return value: *1* - the connection is alive; *0* - the connection is broken (if there is any error presented including AUTH and configuration issues).

Parameters:

- **uri** - the URI or session name;

- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.queries[uri,<username>,<password>,<database name>,<time period>]

 Queries metrics by execution time.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name;
- **timePeriod** - the execution time limit for the count of slow queries (must be a positive integer).

pgsql.replication.count[uri,<username>,<password>]

 The number of standby servers.
 Return value: *Integer*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials.

pgsql.replication.process[uri,<username>,<password>]

 The flush lag, write lag and replay lag per each sender process.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials.

pgsql.replication.process.discovery[uri,<username>,<password>]

 The replication process name discovery.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials.

pgsql.replication.recovery_role[uri,<username>,<password>]

 The recovery status.
 Return value: *0* - master mode; *1* - recovery is still in progress (standby mode).

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials.

pgsql.replication.status[uri,<username>,<password>]

 The status of replication.
 Return value: *0* - streaming is down; *1* - streaming is up; *2* - master mode.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials.

pgsql.replication_lag.b[uri,<username>,<password>]

 The replication lag in bytes.
 Return value: *Integer*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials.

pgsql.replication_lag.sec[uri,<username>,<password>]

 The replication lag in seconds.
 Return value: *Integer*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials.

pgsql.uptime[uri,<username>,<password>,<database name>]

 The PostgreSQL uptime in milliseconds.
 Return value: *Float*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.version[uri,<username>,<password>,<database name>]

 Returns PostgreSQL version.
 Return value: *String*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

pgsql.wal.stat[uri,<username>,<password>,<database name>]

 The WAL statistics.
 Return value: *JSON object*.

Parameters:

- **uri** - the URI or session name;
- **username, password** - the PostgreSQL credentials;
- **database name** - the database name.

redis.config[connString,<password>,<pattern>]

 Gets the configuration parameters of a Redis instance that match the pattern.
 Return value: *JSON* - if a glob-style pattern was used; single value - if a pattern did not contain any wildcard character.

Parameters:

- **connString** - the URI or session name;
- **password** - the Redis password;
- **pattern** - a glob-style pattern (* by default).

redis.info[connString,<password>,<section>]

 Gets the output of the INFO command.
 Return value: *JSON* - the output is serialized as JSON.

Parameters:

- **connString** - the URI or session name;
- **password** - the Redis password;
- **section** - the section of information (*default* by default).

redis.ping[connString,<password>]

 Test if a connection is alive or not.
 Return value: *1* - the connection is alive; *0* - the connection is broken (if there is any error presented including AUTH and configuration issues).

Parameters:

- **connString** - the URI or session name;
- **password** - the Redis password.

redis.slowlog.count[connString,<password>]

 The number of slow log entries since Redis was started.
 Return value: *Integer*.

Parameters:

- **connString** - the URI or session name;
- **password** - the Redis password.

smart.attribute.discovery

 Returns a list of S.M.A.R.T. device attributes.
 Return value: *JSON object*.

Comments:

- The following macros and their values are returned: {#NAME}, {#DISKTYPE}, {#ID}, {#ATTRNAME}, {#THRESH};

- HDD, SSD and NVME drive types are supported. Drives can be alone or combined in a RAID. {#NAME} will have an add-on in case of RAID, e.g: {"{#NAME}": "/dev/sda cciss,2"}.

smart.disk.discovery

 Returns a list of S.M.A.R.T. devices.
 Return value: *JSON object*.

Comments:

- The following macros and their values are returned: {#NAME}, {#DISKTYPE}, {#MODEL}, {#SN}, {#PATH}, {#ATTRIBUTES}, {#RAIDTYPE};
- HDD, SSD and NVME drive types are supported. If a drive does not belong to a RAID, the {#RAIDTYPE} will be empty. {#NAME} will have an add-on in case of RAID, e.g: {"{#NAME}": "/dev/sda cciss,2"}.

smart.disk.get[<path>,<raid type>]

 Returns all available properties of S.M.A.R.T. devices.
 Return value: *JSON object*.

Parameters:

- **path** - the disk path, the {#PATH} macro may be used as a value;
- **raid_type** - the RAID type, the {#RAID} macro may be used as a value

Comments:

- HDD, SSD and NVME drive types are supported. Drives can be alone or combined in a RAID;
- The data includes smartctl version and call arguments, and additional fields:
disk_name - holds the name with the required add-ons for RAID discovery, e.g: {"disk_name": "/dev/sda cciss,2"}
disk_type - holds the disk type HDD, SSD, or NVME, e.g: {"disk_type": "ssd"};
- If no parameters are specified, the item will return information about all disks.

systemd.unit.get[unit name,<interface>]

 Returns all properties of a systemd unit.
 Return value: *JSON object*.

Parameters:

- **unit name** - the unit name (you may want to use the {#UNIT.NAME} macro in item prototype to discover the name);
- **interface** - the unit interface type, possible values: *Unit* (default), *Service*, *Socket*, *Device*, *Mount*, *Automount*, *Swap*, *Target*, *Path*.

Comments:

- This item is supported on Linux platform only;
- LoadState, ActiveState and UnitFileState for Unit interface are returned as text and integer: `"ActiveState":{"state":1,"text":"a`

systemd.unit.info[unit name,<property>,<interface>]

 A systemd unit information.
 Return value: *String*.

Parameters:

- **unit name** - the unit name (you may want to use the {#UNIT.NAME} macro in item prototype to discover the name);
- **property** - unit property (e.g. ActiveState (default), LoadState, Description);
- **interface** - the unit interface type (e.g. Unit (default), Socket, Service).

Comments:

- This item is supported on Linux platform only;
- This item allows to retrieve a specific property from specific type of interface as described in dbus API.

Examples:

```
systemd.unit.info["{#UNIT.NAME}"] #collect active state (active, reloading, inactive, failed, activating,
systemd.unit.info["{#UNIT.NAME}",LoadState] #collect load state info on discovered systemd units
systemd.unit.info[mysqld.service,Id] #retrieve the service technical name (mysqld.service)
systemd.unit.info[mysqld.service,Description] #retrieve the service description (MySQL Server)
systemd.unit.info[mysqld.service,ActiveEnterTimestamp] #retrieve the last time the service entered the act
systemd.unit.info[dbus.socket,NConnections,Socket] #collect the number of connections from this socket uni
```

systemd.unit.discovery[<type>]

 List of systemd units and their details. Used for low-level discovery.
 Return value: *JSON object*.

Parameter:

- **type** - possible values: *all*, *automount*, *device*, *mount*, *path*, *service* (default), *socket*, *swap*, *target*.

This item is supported on Linux platform only.

web.certificate.get[hostname,<port>,<address>]

 Validates the certificates and returns certificate details.
 Return value: *JSON object*.

Parameters:

- **hostname** - can be either IP or DNS.
May contain the URL scheme (*https* only), path (it will be ignored), and port.
If a port is provided in both the first and the second parameters, their values must match.
If address (the 3rd parameter) is specified, the hostname is only used for SNI and hostname verification;
- **port** - the port number (default is 443 for HTTPS);
- **address** - can be either IP or DNS. If specified, it will be used for connection, and hostname (the 1st parameter) will be used for SNI, and host verification. In case, the 1st parameter is an IP and the 3rd parameter is DNS, the 1st parameter will be used for connection, and the 3rd parameter will be used for SNI and host verification.

Comments:

- This item turns unsupported if the resource specified in `host` does not exist or is unavailable or if TLS handshake fails with any error except an invalid certificate;
- Currently, AIA (Authority Information Access) X.509 extension, CRLs and OCSP (including OCSP stapling), Certificate Transparency, and custom CA trust store are not supported.

2 Windows Zabbix agent

Overview

The Windows Zabbix agent items are presented in two lists:

- Shared items - the item keys that are shared with the UNIX Zabbix agent;
- Windows-specific items - the item keys that are supported **only** on Windows.

Note that all item keys supported by Zabbix agent on Windows are also supported by the new generation Zabbix agent 2. See the additional item keys that you can use with the agent 2 only.

See also: Minimum permissions for Windows items

Shared items

The table below lists Zabbix agent items that are supported on Windows and are shared with the UNIX Zabbix agent:

- The item key is a link to full details of the UNIX Zabbix agent item
- Windows-relevant item comments are included

Item key	Description	Item group
log	The monitoring of a log file. This item is not supported for Windows Event Log. The `persistent_dir` parameter is not supported on Windows.	Log monitoring
log.count	The count of matched lines in a monitored log file. This item is not supported for Windows Event Log. The `persistent_dir` parameter is not supported on Windows.	
logrt	The monitoring of a log file that is rotated. This item is not supported for Windows Event Log. The `persistent_dir` parameter is not supported on Windows.	
logrt.count	The count of matched lines in a monitored log file that is rotated. This item is not supported for Windows Event Log. The `persistent_dir` parameter is not supported on Windows.	
modbus.get	Reads Modbus data.	Modbus
net.dns	Checks if the DNS service is up. The `ip`, `timeout` and `count` parameters are ignored on Windows.	Network
net.dns.record	Performs a DNS query. The `ip`, `timeout` and `count` parameters are ignored on Windows.	
net.if.discovery	The list of network interfaces. Some Windows versions (for example, Server 2008) might require the latest updates installed to support non-ASCII characters in interface names.	

Item key	Description	Item group
net.if.in	The incoming traffic statistics on a network interface. On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. Multi-byte interface names on Windows are supported. You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items.	
net.if.out	The outgoing traffic statistics on a network interface. On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. Multi-byte interface names on Windows are supported. You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items.	
net.if.total	The sum of incoming and outgoing traffic statistics on a network interface. On Windows, the item gets values from 64-bit counters if available. 64-bit interface statistic counters were introduced in Windows Vista and Windows Server 2008. If 64-bit counters are not available, the agent uses 32-bit counters. You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items.	
net.tcp.listen	Checks if this TCP port is in LISTEN state.	
net.tcp.port	Checks if it is possible to make a TCP connection to the specified port.	
net.tcp.service	Checks if a service is running and accepting TCP connections. Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2.	
net.tcp.service.perf	Checks the performance of a TCP service. Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2.	
net.tcp.socket.count	Returns the number of TCP sockets that match parameters. This item is supported on Linux by Zabbix agent, but on Windows it is supported only by Zabbix agent 2 on 64-bit Windows.	
net.udp.service	Checks if a service is running and responding to UDP requests.	
net.udp.service.perf	Checks the performance of a UDP service.	
net.udp.socket.count	Returns the number of UDP sockets that match parameters. This item is supported on Linux by Zabbix agent, but on Windows it is supported only by Zabbix agent 2 on 64-bit Windows.	
proc.num	The number of processes. On Windows, only the `name` and `user` parameters are supported.	Processes
system.cpu.discovery	The list of detected CPUs/CPU cores.	System
system.cpu.load	The CPU load.	
system.cpu.num	The number of CPUs.	
system.cpu.util	The CPU utilization percentage. The value is acquired using the *Processor Time* performance counter. Note that since Windows 8 its Task Manager shows CPU utilization based on the *Processor Utility* performance counter, while in previous versions it was the *Processor Time* counter. *system* is the only `type` parameter supported on Windows.	
system.hostname	The system host name. The value is acquired by either GetComputerName() (for **netbios**) or gethostname() (for **host**) functions on Windows. See also a more detailed description.	
system.localtime	The system time.	
system.run	Run the specified command on the host.	
system.sw.arch	The software architecture information.	
system.swap.size	The swap space size in bytes or in percentage from total. The `pused` type parameter is supported on Linux by Zabbix agent, but on Windows it is supported only by Zabbix agent 2. Note that this key might report incorrect swap space size/percentage on virtualized (VMware ESXi, VirtualBox) Windows platforms. In this case you may use the `perf_counter[\700(_Total)\702]` key to obtain correct swap space percentage.	

Item key	Description	Item group
system.uname	Identification of the system. On Windows the value for this item is obtained from Win32_OperatingSystem and Win32_Processor WMI classes. The OS name (including edition) might be translated to the user's display language. On some versions of Windows it contains trademark symbols and extra spaces.	
system.uptime	The system uptime in seconds.	
vfs.dir.count	The directory entry count. On Windows, directory symlinks are skipped and hard links are counted only once.	Virtual file systems
vfs.dir.get	The directory entry list. On Windows, directory symlinks are skipped and hard links are counted only once.	
vfs.dir.size	The directory size. On Windows any symlink is skipped and hard links are taken into account only once.	
vfs.file.cksum	The file checksum, calculated by the UNIX cksum algorithm.	
vfs.file.contents	Retrieving the contents of a file.	
vfs.file.exists	Checks if the file exists. On Windows the double quotes have to be backslash '\' escaped and the whole item key enclosed in double quotes when using the command line utility for calling zabbix_get.exe or agent2. Note that the item may turn unsupported on Windows if a directory is searched within a non-existing directory, e.g. `vfs.file.exists[C:\no\dir,dir]` (where 'no' does not exist).	
vfs.file.get	Returns information about a file. Supported file types on Windows: regular file, directory, symbolic link	
vfs.file.md5sum	The MD5 checksum of file.	
vfs.file.owner	Retrieves the owner of a file.	
vfs.file.regexp	Retrieve a string in the file.	
vfs.file.regmatch	Find a string in the file.	
vfs.file.size	The file size.	
vfs.file.time	The file time information. On Windows XP `vfs.file.time[file,change]` may be equal to `vfs.file.time[file,access]`.	
vfs.fs.discovery	The list of mounted filesystems with their type and mount options. The {#FSLABEL} macro is supported on Windows since Zabbix 6.0.	
vfs.fs.get	The list of mounted filesystems with their type, available disk space, inode statistics and mount options. The {#FSLABEL} macro is supported on Windows since Zabbix 6.0.	
vfs.fs.size	The disk space in bytes or in percentage from total.	
vm.memory.size	The memory size in bytes or in percentage from total.	Virtual memory
web.page.get	Get the content of a web page.	Web monitoring
web.page.perf	The loading time of a full web page.	
web.page.regexp	Find a string on the web page.	
agent.hostmetadata	The agent host metadata.	Zabbix
agent.hostname	The agent host name.	
agent.ping	The agent availability check.	
agent.variant	The variant of Zabbix agent (Zabbix agent or Zabbix agent 2).	
agent.version	The version of Zabbix agent.	
zabbix.stats	Returns a set of Zabbix server or proxy internal metrics remotely.	
zabbix.stats	Returns the number of monitored items in the queue which are delayed on Zabbix server or proxy remotely.	

Windows-specific items

The table provides details on the item keys that are supported **only** by the Windows Zabbix agent.

Windows-specific items sometimes are an approximate counterpart of a similar agent item, for example `proc_info`, supported on Windows, roughly corresponds to the `proc.mem` item, not supported on Windows.

The item key is a link to full item key details.

Item key	Description	Item group
eventlog	The Windows event log monitoring.	Log monitoring
net.if.list	The network interface list (includes interface type, status, IPv4 address, description).	Network
perf_counter	The value of any Windows performance counter.	Performance counters
perf_counter_en	The value of any Windows performance counter in English.	
perf_instance.discovery	The list of object instances of Windows performance counters.	
perf_instance_en.discovery	The list of object instances of Windows performance counters, discovered using the object names in English.	
proc_info	Various information about specific process(es).	Processes
registry.data	Return data for the specified value name in the Windows Registry key.	Registry
registry.get	The list of Windows Registry values or keys located at given key.	
service.discovery	The list of Windows services.	Services
service.info	Information about a service.	
services	The listing of services.	
vm.vmemory.size	The virtual memory size in bytes or in percentage from the total.	Virtual memory
wmi.get	Execute a WMI query and return the first selected object.	WMI
wmi.getall	Execute a WMI query and return the whole response.	

Item key details

Parameters without angle brackets are mandatory. Parameters marked with angle brackets **< >** are optional.

eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>,<mode>]

 The event log monitoring.
 Return value: *Log*.

Parameters:

- **name** - the name of the event log;
- **regexp** - a regular expression describing the required pattern (case sensitive);
- **severity** - a regular expression describing severity (case insensitive). This parameter accepts a regular expression based on the following values: ”Information”, ”Warning”, ”Error”, ”Critical”, ”Verbose” (running on Windows Vista or newer).
- **source** - a regular expression describing the source identifier (case insensitive);
- **eventid** - a regular expression describing the event identifier(s) (case sensitive);
- **maxlines** - the maximum number of new lines per second the agent will send to Zabbix server or proxy. This parameter overrides the value of ’MaxLinesPerSecond’ in zabbix_agentd.conf.
- **mode** - possible values: *all* (default) or *skip* - skip the processing of older data (affects only newly created items).

Comments:

- The item must be configured as an active check;
- The agent is unable to send in events from the ”Forwarded events” log;
- Windows Eventing 6.0 is supported;
- Selecting a non-Log type of information for this item will lead to the loss of local timestamp, as well as log severity and source information;
- See also additional information on log monitoring.

Examples:

```
eventlog[Application]
eventlog[Security,,"Failure Audit",,^(529|680)$]
eventlog[System,,"Warning|Error"]
eventlog[System,,,,^1$]
eventlog[System,,,,@TWOSHORT] #here a custom regular expression named `TWOSHORT` is referenced (defined as
```

net.if.list

 The network interface list (includes interface type, status, IPv4 address, description).
 Return value: *Text*.

Comments:

- Multi-byte interface names supported;
- Disabled interfaces are not listed;
- Enabling/disabling some components may change their ordering in the Windows interface name;

- Some Windows versions (for example, Server 2008) might require the latest updates installed to support non-ASCII characters in interface names.

perf_counter[counter,<interval>]

 The value of any Windows performance counter.
 Return value: *Integer*, *float*, *string* or *text* (depending on the request).

Parameters:

- **counter** - the path to the counter;
- **interval** - the last N seconds for storing the average value. The `interval` must be between 1 and 900 seconds (included) and the default value is 1.

Comments:

- `interval` is used for counters that require more than one sample (like CPU utilization), so the check returns an average value for last "interval" seconds every time;
- Performance Monitor can be used to obtain the list of available counters.
- See also: Windows performance counters.

perf_counter_en[counter,<interval>]

 The value of any Windows performance counter in English.
 Return value: *Integer*, *float*, *string* or *text* (depending on the request).

Parameters:

- **counter** - the path to the counter in English;
- **interval** - the last N seconds for storing the average value. The `interval` must be between 1 and 900 seconds (included) and the default value is 1.

Comments:

- `interval` is used for counters that require more than one sample (like CPU utilization), so the check returns an average value for last "interval" seconds every time;
- This item is only supported on **Windows Server 2008/Vista** and above;
- You can find the list of English strings by viewing the following registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009`.

perf_instance.discovery[object]

 The list of object instances of Windows performance counters. Used for low-level discovery.
 Return value: *JSON object*.

Parameter:

- **object** - the object name (localized).

perf_instance_en.discovery[object]

 The list of object instances of Windows performance counters, discovered using the object names in English. Used for low-level discovery.
 Return value: *JSON object*.

Parameter:

- **object** - the object name (in English).

proc_info[process,<attribute>,<type>]

 Various information about specific process(es).
 Return value: *Float*.

Parameters:

- **process** - the process name;
- **attribute** - the requested process attribute;
- **type** - the representation type (meaningful when more than one process with the same name exists)

Comments:

- The following `attributes` are supported:
vmsize (default) - size of process virtual memory in Kbytes
wkset - size of process working set (amount of physical memory used by process) in Kbytes
pf - number of page faults
ktime - process kernel time in milliseconds
utime - process user time in milliseconds
io_read_b - number of bytes read by process during I/O operations
io_read_op - number of read operation performed by process
io_write_b - number of bytes written by process during I/O operations
io_write_op - number of write operation performed by process
io_other_b - number of bytes transferred by process during operations other than read and write operations
io_other_op - number of I/O operations performed by process, other than read and write operations
gdiobj - number of GDI objects used by process
userobj - number of USER objects used by process;

- Valid types are:
avg (default) - average value for all processes named <process>
min - minimum value among all processes named <process>
max - maximum value among all processes named <process>
sum - sum of values for all processes named <process>;
- On a 64-bit system, a 64-bit Zabbix agent is required for this item to work correctly.

Examples:

```
proc_info[iexplore.exe,wkset,sum] #retrieve the amount of physical memory taken by all Internet Explorer p
proc_info[iexplore.exe,pf,avg] #retrieve the average number of page faults for Internet Explorer processes
```

registry.data[key,<value name>]

 Return data for the specified value name in the Windows Registry key.
 Return value: *Integer*, *string* or *text* (depending on the value type)

Parameters:

- **key** - the registry key including the root key; root abbreviations (e.g. HKLM) are allowed;
- **value name** - the registry value name in the key (empty string "" by default). The default value is returned if the value name is not supplied.

Comments:

- Supported root abbreviations:
HKCR - HKEY_CLASSES_ROOT
HKCC - HKEY_CURRENT_CONFIG
HKCU - HKEY_CURRENT_USER
HKCULS - HKEY_CURRENT_USER_LOCAL_SETTINGS
HKLM - HKEY_LOCAL_MACHINE
HKPD - HKEY_PERFORMANCE_DATA
HKPN - HKEY_PERFORMANCE_NLSTEXT
HKPT - HKEY_PERFORMANCE_TEXT
HKU - HKEY_USERS
- Keys with spaces must be double-quoted.

Examples:

```
registry.data["HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\Windows Error Reporting"] #return the data of
registry.data["HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting","EnableZip"] #return the data of t
```

registry.get[key,<mode>,<name regexp>]

 The list of Windows Registry values or keys located at given key.
 Return value: *JSON object*.

Parameters:

- **key** - the registry key including the root key; root abbreviations (e.g. HKLM) are allowed (see comments for registry.data[] to see full list of abbreviations);
- **mode** - possible values:
values (default) or *keys*;
- **name regexp** - only discover values with names that match the regexp (default - discover all values). Allowed only with *values* as `mode`.

Keys with spaces must be double-quoted.

Examples:

```
registry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall,values,"^DisplayName|DisplayVersion$
registry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall,values] #return the data of the all
registry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall,keys] #return all subkeys of this ke
```

service.discovery

 The list of Windows services. Used for low-level discovery.
 Return value: *JSON object*.

service.info[service,<param>]

 Information about a service.
 Return value: *Integer* - with `param` as *state*, *startup*; *String* - with `param` as *displayname*, *path*, *user*; *Text* - with `param` as *description*
Specifically for *state*: 0 - running, 1 - paused, 2 - start pending, 3 - pause pending, 4 - continue pending, 5 - stop pending, 6 - stopped, 7 - unknown, 255 - no such service
Specifically for *startup*: 0 - automatic, 1 - automatic delayed, 2 - manual, 3 - disabled, 4 - unknown, 5 - automatic trigger start, 6 - automatic delayed trigger start, 7 - manual trigger start

Parameters:

- **service** - a real service name or its display name as seen in the MMC Services snap-in;
- **param** - *state* (default), *displayname*, *path*, *user*, *startup*, or *description*.

Comments:

- Items like `service.info[service,state]` and `service.info[service]` will return the same information;
- Only with `param` as *state* this item returns a value for non-existing services (255).

Examples:

```
service.info[SNMPTRAP] - state of the SNMPTRAP service;
service.info[SNMP Trap] - state of the same service, but with the display name specified;
service.info[EventLog,startup] - the startup type of the EventLog service
```

services[<type>,<state>,<exclude>]

 The listing of services.
 Return value: *0* - if empty; *Text* - the list of services separated by a newline.

Parameters:

- **type** - *all* (default), *automatic*, *manual*, or *disabled*;
- **state** - *all* (default), *stopped*, *started*, *start_pending*, *stop_pending*, *running*, *continue_pending*, *pause_pending*, or *paused*;
- **exclude** - the services to exclude from the result. Excluded services should be listed in double quotes, separated by comma, without spaces.

Examples:

```
services[,started] #returns the list of started services;
services[automatic, stopped] #returns the list of stopped services that should be running;
services[automatic, stopped, "service1,service2,service3"] #returns the list of stopped services that shou
```

vm.vmemory.size[<type>]

 The virtual memory size in bytes or in percentage from the total.
 Return value: *Integer* - for bytes; *float* - for percentage.

Parameter:

- **type** - possible values: *available* (available virtual memory), *pavailable* (available virtual memory, in percent), *pused* (used virtual memory, in percent), *total* (total virtual memory, default), or *used* (used virtual memory)

Comments:

- The monitoring of virtual memory statistics is based on:
  - Total virtual memory on Windows (total physical + page file size);
  - The maximum amount of memory Zabbix agent can commit;
  - The current committed memory limit for the system or Zabbix agent, whichever is smaller.

Example:

```
vm.vmemory.size[pavailable] #return the available virtual memory, in percentage
```

wmi.get[<namespace>,<query>]

 Execute a WMI query and return the first selected object.
 Return value: *Integer*, *float*, *string* or *text* (depending on the request).

Parameters:

- **namespace** - the WMI namespace;
- **query** - the WMI query returning a single object.

WMI queries are performed with WQL.

Example:

```
wmi.get[root\cimv2,select status from Win32_DiskDrive where Name like '%PHYSICALDRIVE0%'] #returns the sta
```

wmi.getall[<namespace>,<query>]

 Execute a WMI query and return the whole response. Can be used for low-level discovery.
 Return value: *JSON object*

Parameters:

- **namespace** - the WMI namespace;
- **query** - the WMI query.

Comments:

- WMI queries are performed with WQL.
- JSONPath preprocessing can be used to point to more specific values in the returned JSON.

Example:

```
wmi.getall[root\cimv2,select * from Win32_DiskDrive where Name like '%PHYSICALDRIVE%'] #returns status inf
```

Monitoring Windows services

This tutorial provides step-by-step instructions for setting up the monitoring of Windows services. It is assumed that Zabbix server and agent are configured and operational.

Step 1

Get the service name.

You can get the service name by going to the MMC Services snap-in and bringing up the properties of the service. In the *General* tab you should see a field called "Service name". The value that follows is the name you will use when setting up an item for monitoring. For example, if you wanted to monitor the "workstation" service, then your service might be: **lanmanworkstation**.

Step 2

Configure an item for monitoring the service.

The item `service.info[service,<param>]` retrieves information about a particular service. Depending on the information you need, specify the `param` option which accepts the following values: *displayname*, *state*, *path*, *user*, *startup* or *description*. The default value is *state* if `param` is not specified (`service.info[service]`).

The type of return value depends on chosen `param`: integer for *state* and *startup*; character string for *displayname*, *path* and *user*; text for *description*.

Example:

- *Key:* `service.info[lanmanworkstation]`
- *Type of information:* Numeric (unsigned)

The item `service.info[lanmanworkstation]` will retrieve information about the state of the service as a numerical value. To map a numerical value to a text representation in the frontend ("0" as "Running", "1" as "Paused", etc.), you can configure value mapping on the host on which the item is configured. To do this, either link the template *Windows services by Zabbix agent* or *Windows services by Zabbix agent active* to the host, or configure on the host a new value map that is based on the *Windows service state* value map configured on the mentioned templates.

Note that both of the mentioned templates have a discovery rule configured that will discover services automatically. If you do not want this, you can disable the discovery rule on the host level once the template has been linked to the host.

Discovery of Windows services

Low-level discovery provides a way to automatically create items, triggers, and graphs for different entities on a computer. Zabbix can automatically start monitoring Windows services on your machine, without the need to know the exact name of a service or create items for each service manually. A filter can be used to generate real items, triggers, and graphs only for services of interest.

2 SNMP agent

Overview

You may want to use SNMP monitoring on devices such as printers, network switches, routers or UPS that usually are SNMP-enabled and on which it would be impractical to attempt setting up complete operating systems and Zabbix agents.

To be able to retrieve data provided by SNMP agents on these devices, Zabbix server must be initially configured with SNMP support by specifying the `--with-net-snmp` flag.

SNMP checks are performed over the UDP protocol only.

Zabbix server and proxy daemons log lines similar to the following if they receive an incorrect SNMP response:

```
SNMP response from host "gateway" does not contain all of the requested variable bindings
```

While they do not cover all the problematic cases, they are useful for identifying individual SNMP devices for which combined requests should be disabled.

Zabbix server/proxy will always retry at least one time after an unsuccessful query attempt: either through the SNMP library's retrying mechanism or through the internal combined processing mechanism.

> **Warning:**
> If monitoring SNMPv3 devices, make sure that msgAuthoritativeEngineID (also known as snmpEngineID or "Engine ID") is never shared by two devices. According to RFC 2571 (section 3.1.1.1) it must be unique for each device.

Warning:
RFC3414 requires the SNMPv3 devices to persist their engineBoots. Some devices do not do that, which results in their SNMP messages being discarded as outdated after being restarted. In such situation, SNMP cache needs to be manually cleared on a server/proxy (by using -R snmp_cache_reload) or the server/proxy needs to be restarted.

Configuring SNMP monitoring

To start monitoring a device through SNMP, the following steps have to be performed:

Step 1

Find out the SNMP string (or OID) of the item you want to monitor.

To get a list of SNMP strings, use the **snmpwalk** command (part of net-snmp software which you should have installed as part of the Zabbix installation) or equivalent tool:

```
shell> snmpwalk -v 2c -c public <host IP> .
```

As '2c' here stands for SNMP version, you may also substitute it with '1', to indicate SNMP Version 1 on the device.

This should give you a list of SNMP strings and their last value. If it doesn't then it is possible that the SNMP 'community' is different from the standard 'public' in which case you will need to find out what it is.

You can then go through the list until you find the string you want to monitor, e.g. if you wanted to monitor the bytes coming in to your switch on port 3 you would use the `IF-MIB::ifHCInOctets.3` string from this line:

```
IF-MIB::ifHCInOctets.3 = Counter64: 3409739121
```

You may now use the **snmpget** command to find out the numeric OID for 'IF-MIB::ifHCInOctets.3':

```
shell> snmpget -v 2c -c public -On <host IP> IF-MIB::ifHCInOctets.3
```

Note that the last number in the string is the port number you are looking to monitor. See also: Dynamic indexes.

This should give you something like the following:

```
.1.3.6.1.2.1.31.1.1.1.6.3 = Counter64: 3472126941
```

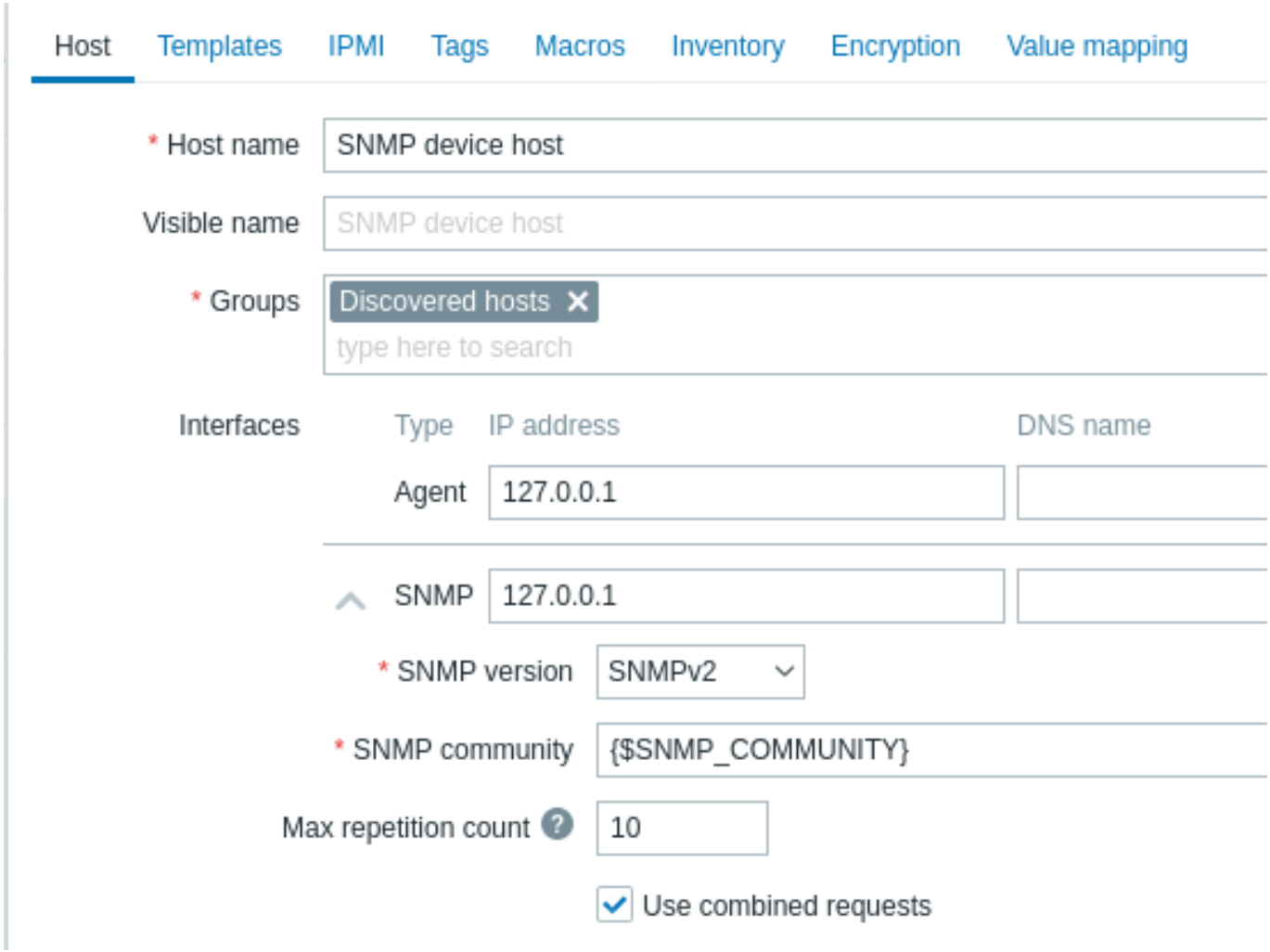
Again, the last number in the OID is the port number.

Note:
Some of the most used SNMP OIDs are translated automatically to a numeric representation by Zabbix.

In the last example above value type is "Counter64", which internally corresponds to ASN_COUNTER64 type. The full list of supported types is ASN_COUNTER, ASN_COUNTER64, ASN_UINTEGER, ASN_UNSIGNED64, ASN_INTEGER, ASN_INTEGER64, ASN_FLOAT, ASN_DOUBLE, ASN_TIMETICKS, ASN_GAUGE, ASN_IPADDRESS, ASN_OCTET_STR and ASN_OBJECT_ID. These types roughly correspond to "Counter32", "Counter64", "UInteger32", "INTEGER", "Float", "Double", "Timeticks", "Gauge32", "IpAddress", "OCTET STRING", "OBJECT IDENTIFIER" in **snmpget** output, but might also be shown as "STRING", "Hex-STRING", "OID" and other, depending on the presence of a display hint.

Step 2

Create a host corresponding to a device.



Add an SNMP interface for the host:

- Enter the IP address/DNS name and port number
- Select the *SNMP version* from the dropdown
- Add interface credentials depending on the selected SNMP version:
  - SNMPv1, v2 require only the community (usually 'public')
  - SNMPv3 requires more specific options (see below)
- Specify the max repetition value (default: 10) for native SNMP bulk requests (GetBulkRequest-PDUs); only for `discovery[]` and `walk[]` items in SNMPv2 and v3. Note that setting this value too high may cause the SNMP agent check timeout.
- Mark the *Use combined requests* checkbox to allow combined processing of SNMP requests (not related to native SNMP bulk requests)

SNMPv3 parameter	Description
Context name	Enter context name to identify item on SNMP subnet. *Context name* is supported for SNMPv3 items since Zabbix 2.2. User macros are resolved in this field.
Security name	Enter security name. User macros are resolved in this field.
Security level	Select security level: **noAuthNoPriv** - no authentication nor privacy protocols are used **AuthNoPriv** - authentication protocol is used, privacy protocol is not **AuthPriv** - both authentication and privacy protocols are used
Authentication protocol	Select authentication protocol - *MD5*, *SHA1*, *SHA224*, *SHA256*, *SHA384* or *SHA512*.
Authentication passphrase	Enter authentication passphrase. User macros are resolved in this field.
Privacy protocol	Select privacy protocol - *DES*, *AES128*, *AES192*, *AES256*, *AES192C* (Cisco) or *AES256C* (Cisco). Note that: - on some older systems net-snmp may not support AES256; - on some newer systems (for example, RHEL9) support of DES may be dropped for the net-snmp package.

SNMPv3 parameter	Description
Privacy passphrase	Enter privacy passphrase. User macros are resolved in this field.

In case of wrong SNMPv3 credentials (security name, authentication protocol/passphrase, privacy protocol):

- Zabbix receives an ERROR from net-snmp, except for wrong *Privacy passphrase* in which case Zabbix receives a TIMEOUT error from net-snmp;
- SNMP interface availability will switch to red (unavailable).

Warning:
Changes in *Authentication protocol*, *Authentication passphrase*, *Privacy protocol* or *Privacy passphrase*, made without changing the *Security name*, will take effect only after the cache on a server/proxy is manually cleared (by using -R snmp_cache_reload) or the server/proxy is restarted. In cases, where *Security name* is also changed, all parameters will be updated immediately.

You can use one of the provided SNMP templates that will automatically add a set of items. Before using a template, verify that it is compatible with the host.

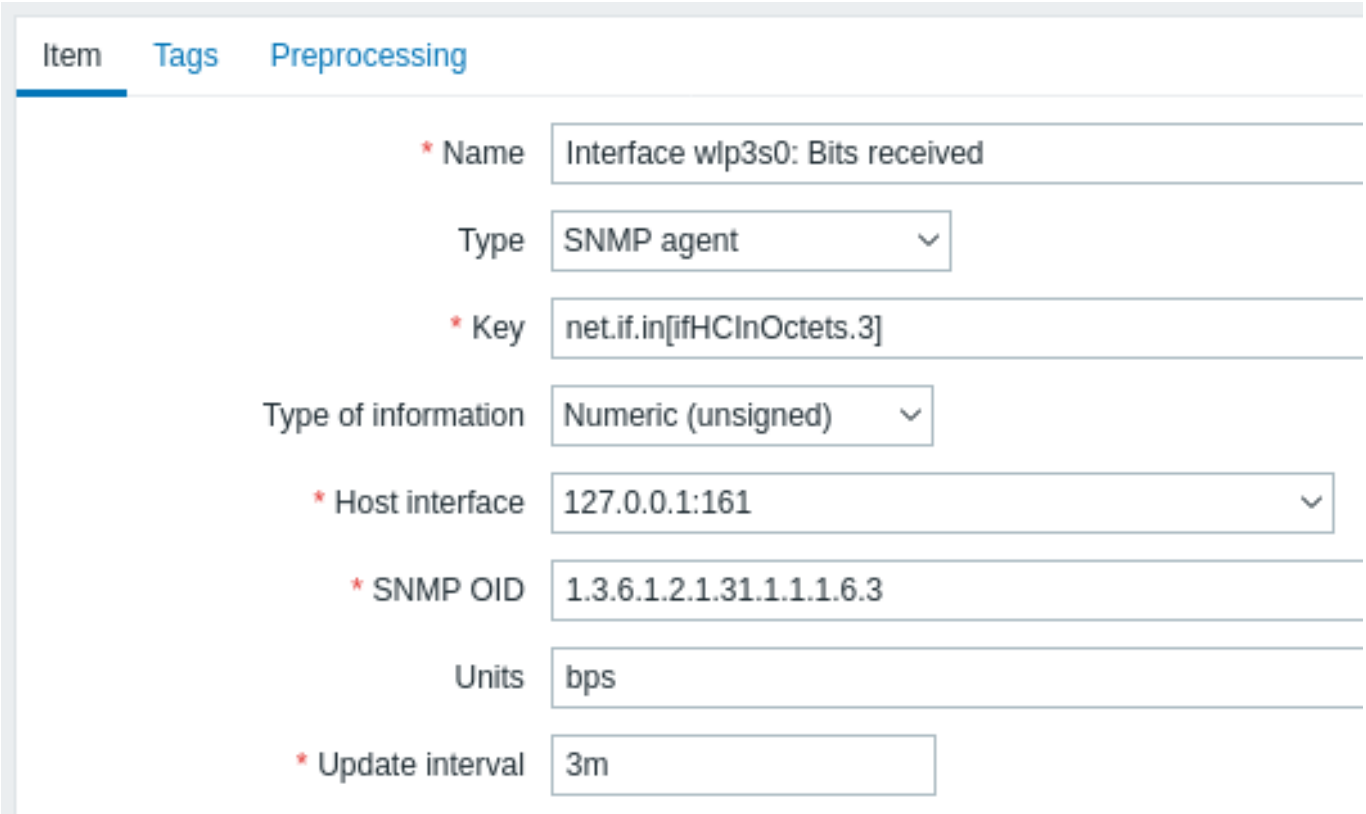
Click on *Add* to save the host.

Step 3

Create an item for monitoring.

So, now go back to Zabbix and click on *Items* for the SNMP host you created earlier. Depending on whether you used a template or not when creating your host, you will have either a list of SNMP items associated with your host or just an empty list. We will work on the assumption that you are going to create the item yourself using the information you have just gathered using snmpwalk and snmpget, so click on *Create item*.

Fill in the required parameters in the new item form:



Parameter	Description
Name	Enter the item name.
Type	Select **SNMP agent** here.
Key	Enter the key as something meaningful.
Host interface	Make sure to select the SNMP interface, e.g. of your switch/router.

Parameter	Description
SNMP OID	This field supports two options: **1)** Enter a single textual or numeric OID, for example: 1.3.6.1.2.1.31.1.1.1.6.3 (in this case, make sure to add a *Change per second* step in the *Preprocessing* tab; otherwise you will get cumulative values from the SNMP device instead of the latest change). **2)** Use the **walk[OID1,OID2,...]** item, which makes use of native SNMP bulk requests (GetBulkRequest-PDUs). You may use this as the master item, with dependent items that extract data from the master using preprocessing. For example, `walk[1.3.6.1.2.1.2.2.1.2,1.3.6.1.2.1.2.2.1.3]`. This item returns the output of the snmpwalk utility with -Oe -Ot -On parameters. MIB names are supported as parameters; thus `walk[1.3.6.1.2.1.2.2.1.2]` and `walk[ifDescr]` will return the same output. If several OIDs/MIBs are specified, i.e. `walk[ifDescr,ifType,ifPhysAddress]`, then the output is a concatenated list. This item uses GetBulk requests with SNMPv2 and v3 interfaces and GetNext for SNMPv1 interfaces; max repetitions for bulk requests are configured on the interface level. You may use this item as a master item in SNMP discovery.

All mandatory input fields are marked with a red asterisk.

Now save the item and go to *Monitoring* → *Latest data* for your SNMP data.

Example 1

General example:

Parameter	Description
OID	1.2.3.45.6.7.8.0 (or .1.2.3.45.6.7.8.0)
Key	<Unique string to be used as reference to triggers> For example, "my_param".

Note that OID can be given in either numeric or string form. However, in some cases, string OID must be converted to numeric representation. Utility snmpget may be used for this purpose:

```
shell> snmpget -On localhost public enterprises.ucdavis.memory.memTotalSwap.0
```

Example 2

Monitoring of uptime:

Parameter	Description
OID	MIB::sysUpTime.0
Key	router.uptime
Value type	Float
Units	uptime
Preprocessing step: Custom multiplier	0.01

Native SNMP bulk requests

The **walk[OID1,OID2,...]** item allows to use native SNMP functionality for bulk requests (GetBulkRequest-PDUs), available in SNMP versions 2/3.

A GetBulk request in SNMP executes multiple GetNext requests and returns the result in a single response. This may be used for regular SNMP items as well as for SNMP discovery to minimize network roundtrips.

The SNMP **walk[OID1,OID2,...]** item may be used as the master item that collects data in one request with dependent items that parse the response as needed using preprocessing.

Note that using native SNMP bulk requests is not related to the option of combining SNMP requests, which is Zabbix own way of combining multiple SNMP requests (see next section).

Internal workings of combined processing

Zabbix server and proxy may query SNMP devices for multiple values in a single request. This affects several types of SNMP items:

- regular SNMP items
- SNMP items with dynamic indexes
- SNMP low-level discovery rules

All SNMP items on a single interface with identical parameters are scheduled to be queried at the same time. The first two types of items are taken by pollers in batches of at most 128 items, whereas low-level discovery rules are processed individually, as before.

On the lower level, there are two kinds of operations performed for querying values: getting multiple specified objects and walking an OID tree.

For ”getting”, a GetRequest-PDU is used with at most 128 variable bindings. For ”walking”, a GetNextRequest-PDU is used for SNMPv1 and GetBulkRequest with ”max-repetitions” field of at most 128 is used for SNMPv2 and SNMPv3.

Thus, the benefits of combined processing for each SNMP item type are outlined below:

- regular SNMP items benefit from ”getting” improvements;
- SNMP items with dynamic indexes benefit from both ”getting” and ”walking” improvements: ”getting” is used for index verification and ”walking” for building the cache;
- SNMP low-level discovery rules benefit from ”walking” improvements.

However, there is a technical issue that not all devices are capable of returning 128 values per request. Some always return a proper response, but others either respond with a ”tooBig(1)” error or do not respond at all once the potential response is over a certain limit.

In order to find an optimal number of objects to query for a given device, Zabbix uses the following strategy. It starts cautiously with querying 1 value in a request. If that is successful, it queries 2 values in a request. If that is successful again, it queries 3 values in a request and continues similarly by multiplying the number of queried objects by 1.5, resulting in the following sequence of request sizes: 1, 2, 3, 4, 6, 9, 13, 19, 28, 42, 63, 94, 128.

However, once a device refuses to give a proper response (for example, for 42 variables), Zabbix does two things.

First, for the current item batch it halves the number of objects in a single request and queries 21 variables. If the device is alive, then the query should work in the vast majority of cases, because 28 variables were known to work and 21 is significantly less than that. However, if that still fails, then Zabbix falls back to querying values one by one. If it still fails at this point, then the device is definitely not responding and request size is not an issue.

The second thing Zabbix does for subsequent item batches is it starts with the last successful number of variables (28 in our example) and continues incrementing request sizes by 1 until the limit is hit. For example, assuming the largest response size is 32 variables, the subsequent requests will be of sizes 29, 30, 31, 32, and 33. The last request will fail and Zabbix will never issue a request of size 33 again. From that point on, Zabbix will query at most 32 variables for this device.

If large queries fail with this number of variables, it can mean one of two things. The exact criteria that a device uses for limiting response size cannot be known, but we try to approximate that using the number of variables. So the first possibility is that this number of variables is around the device’s actual response size limit in the general case: sometimes response is less than the limit, sometimes it is greater than that. The second possibility is that a UDP packet in either direction simply got lost. For these reasons, if Zabbix gets a failed query, it reduces the maximum number of variables to try to get deeper into the device’s comfortable range, but (starting from 2.2.8) only up to two times.

In the example above, if a query with 32 variables happens to fail, Zabbix will reduce the count to 31. If that happens to fail, too, Zabbix will reduce the count to 30. However, Zabbix will not reduce the count below 30, because it will assume that further failures are due to UDP packets getting lost, rather than the device’s limit.

If, however, a device cannot handle combined requests properly for other reasons and the heuristic described above does not work, since Zabbix 2.4 there is a ”Use combined requests” setting for each interface that allows to disable combined requests for that device.

1 Dynamic indexes

Overview

While you may find the required index number (for example, of a network interface) among the SNMP OIDs, sometimes you may not completely rely on the index number always staying the same.

Index numbers may be dynamic - they may change over time and your item may stop working as a consequence.

To avoid this scenario, it is possible to define an OID which takes into account the possibility of an index number changing.

For example, if you need to retrieve the index value to append to **ifInOctets** that corresponds to the **GigabitEthernet0/1** interface on a Cisco device, use the following OID:

```
ifInOctets["index","ifDescr","GigabitEthernet0/1"]
```

The syntax

A special syntax for OID is used:

<OID of data>["index","<base OID of index>","<string to search for>"]

Parameter	Description
OID of data	Main OID to use for data retrieval on the item.
index	Method of processing. Currently one method is supported: **index** – search for index and append it to the data OID
base OID of index	This OID will be looked up to get the index value corresponding to the string.
string to search for	The string to use for an exact match with a value when doing lookup. Case sensitive.

Example

Getting memory usage of *apache* process.

If using this OID syntax:

```
HOST-RESOURCES-MIB::hrSWRunPerfMem["index","HOST-RESOURCES-MIB::hrSWRunPath", "/usr/sbin/apache2"]
```

the index number will be looked up here:

```
...
HOST-RESOURCES-MIB::hrSWRunPath.5376 = STRING: "/sbin/getty"
HOST-RESOURCES-MIB::hrSWRunPath.5377 = STRING: "/sbin/getty"
HOST-RESOURCES-MIB::hrSWRunPath.5388 = STRING: "/usr/sbin/apache2"
HOST-RESOURCES-MIB::hrSWRunPath.5389 = STRING: "/sbin/sshd"
...
```

Now we have the index, 5388. The index will be appended to the data OID in order to receive the value we are interested in:

```
HOST-RESOURCES-MIB::hrSWRunPerfMem.5388 = INTEGER: 31468 KBytes
```

Index lookup caching

When a dynamic index item is requested, Zabbix retrieves and caches whole SNMP table under base OID for index, even if a match would be found sooner. This is done in case another item would refer to the same base OID later - Zabbix would look up index in the cache, instead of querying the monitored host again. Note that each poller process uses separate cache.

In all subsequent value retrieval operations only the found index is verified. If it has not changed, value is requested. If it has changed, cache is rebuilt - each poller that encounters a changed index walks the index SNMP table again.

2 Special OIDs

Some of the most used SNMP OIDs are translated automatically to a numeric representation by Zabbix. For example, **ifIndex** is translated to **1.3.6.1.2.1.2.2.1.1**, **ifIndex.0** is translated to **1.3.6.1.2.1.2.2.1.1.0**.

The table contains list of the special OIDs.

Special OID	Identifier	Description
ifIndex	1.3.6.1.2.1.2.2.1.1	A unique value for each interface.
ifDescr	1.3.6.1.2.1.2.2.1.2	A textual string containing information about the interface.This string should include the name of the manufacturer, the product name and the version of the hardware interface.
ifType	1.3.6.1.2.1.2.2.1.3	The type of interface, distinguished according to the physical/link protocol(s) immediately 'below' the network layer in the protocol stack.
ifMtu	1.3.6.1.2.1.2.2.1.4	The size of the largest datagram which can be sent / received on the interface, specified in octets.
ifSpeed	1.3.6.1.2.1.2.2.1.5	An estimate of the interface's current bandwidth in bits per second.
ifPhysAddress	1.3.6.1.2.1.2.2.1.6	The interface's address at the protocol layer immediately 'below' the network layer in the protocol stack.
ifAdminStatus	1.3.6.1.2.1.2.2.1.7	The current administrative state of the interface.
ifOperStatus	1.3.6.1.2.1.2.2.1.8	The current operational state of the interface.

Special OID	Identifier	Description
ifInOctets	1.3.6.1.2.1.2.2.1.10	The total number of octets received on the interface, including framing characters.
ifInUcastPkts	1.3.6.1.2.1.2.2.1.11	The number of subnetwork-unicast packets delivered to a higher-layer protocol.
ifInNUcastPkts	1.3.6.1.2.1.2.2.1.12	The number of non-unicast (i.e., subnetwork- broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol.
ifInDiscards	1.3.6.1.2.1.2.2.1.13	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol. One possible reason for discarding such a packet could be to free up buffer space.
ifInErrors	1.3.6.1.2.1.2.2.1.14	The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.
ifInUnknownProtos	1.3.6.1.2.1.2.2.1.15	The number of packets received via the interface which were discarded because of an unknown or unsupported protocol.
ifOutOctets	1.3.6.1.2.1.2.2.1.16	The total number of octets transmitted out of the interface, including framing characters.
ifOutUcastPkts	1.3.6.1.2.1.2.2.1.17	The total number of packets that higher-level protocols requested be transmitted, and which were not addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.
ifOutNUcastPkts	1.3.6.1.2.1.2.2.1.18	The total number of packets that higher-level protocols requested be transmitted, and which were addressed to a multicast or broadcast address at this sub-layer, including those that were discarded or not sent.
ifOutDiscards	1.3.6.1.2.1.2.2.1.19	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted. One possible reason for discarding such a packet could be to free up buffer space.
ifOutErrors	1.3.6.1.2.1.2.2.1.20	The number of outbound packets that could not be transmitted because of errors.
ifOutQLen	1.3.6.1.2.1.2.2.1.21	The length of the output packet queue (in packets).

3 MIB files

Introduction

MIB stands for the Management Information Base. MIB files allow to use textual representation of an OID (Object Identifier). It is possible to use raw OIDs when monitoring SNMP devices with Zabbix, but if you feel more comfortable using textual representation, you need to install MIB files.

For example,

`ifHCOutOctets`

is textual representation of the OID

`1.3.6.1.2.1.31.1.1.1.10`

Installing MIB files

On Debian-based systems:

```
# apt install snmp-mibs-downloader
# download-mibs
```

On RedHat-based systems:

```
# dnf install net-snmp-libs
```

Enabling MIB files

On RedHat-based systems, MIB files should be enabled by default. On Debian-based systems, you have to edit the file `/etc/snmp/snmp.conf` and comment out the line that says `mibs :`

```
# As the snmp packages come without MIB files due to license reasons, loading
# of MIBs is disabled by default. If you added the MIBs you can re-enable
# loading them by commenting out the following line.
#mibs :
```

Testing MIB files

Testing SNMP MIBs can be done using `snmpwalk` utility. If you don't have it installed, use the following instructions.

On Debian-based systems:

```
# apt install snmp
```

On RedHat-based systems:

```
# dnf install net-snmp-utils
```

After that, the following command must not give error when you query a network device:

```
$ snmpwalk -v 2c -c public <NETWORK DEVICE IP> ifInOctets
IF-MIB::ifInOctets.1 = Counter32: 176137634
IF-MIB::ifInOctets.2 = Counter32: 0
IF-MIB::ifInOctets.3 = Counter32: 240375057
IF-MIB::ifInOctets.4 = Counter32: 220893420
[...]
```

Using MIBs in Zabbix

The most important to keep in mind is that Zabbix processes do not get informed of the changes made to MIB files. So after every change you must restart Zabbix server or proxy, e. g.:

```
# systemctl restart zabbix-server
```

After that, the changes made to MIB files are in effect.

Using custom MIB files

There are standard MIB files coming with every GNU/Linux distribution. But some device vendors provide their own.

Let's say, you would like to use CISCO-SMI MIB file. The following instructions will download and install it:

```
# wget ftp://ftp.cisco.com/pub/mibs/v2/CISCO-SMI.my -P /tmp
# mkdir -p /usr/local/share/snmp/mibs
# grep -q '^mibdirs +/usr/local/share/snmp/mibs' /etc/snmp/snmp.conf 2>/dev/null || echo "mibdirs +/usr/lo
# cp /tmp/CISCO-SMI.my /usr/local/share/snmp/mibs
```

Now you should be able to use it. Try to translate the name of the object *ciscoProducts* from the MIB file to OID:

```
# snmptranslate -IR -On CISCO-SMI::ciscoProducts
.1.3.6.1.4.1.9.1
```

If you receive errors instead of the OID, ensure all the previous commands did not return any errors.

The object name translation worked, you are ready to use custom MIB file. Note the MIB name prefix (*CISCO-SMI::*) used in the query. You will need this when using command-line tools as well as Zabbix.

Don't forget to restart Zabbix server/proxy before using this MIB file in Zabbix.

> **Attention:**
> Keep in mind that MIB files can have dependencies. That is, one MIB may require another. In order to satisfy these dependencies you have to install all the affected MIB files.

3 SNMP traps

Overview

Receiving SNMP traps is the opposite to querying SNMP-enabled devices.

In this case, the information is sent from an SNMP-enabled device and is collected or "trapped" by Zabbix.

Usually, traps are sent upon some condition change and the agent connects to the server on port 162 (as opposed to port 161 on the agent side that is used for queries). Using traps may detect some short problems that occur amidst the query interval and may be missed by the query data.

Receiving SNMP traps in Zabbix is designed to work with **snmptrapd** and one of the mechanisms for passing the traps to Zabbix - either a Bash or Perl script or SNMPTT.

Note:
The simplest way to set up trap monitoring after configuring Zabbix is to use the Bash script solution, because Perl and SNMPTT are often missing in modern distributions and require more complex configuration. However, this solution uses a script configured as `traphandle`. For better performance on production systems, use the embedded Perl solution (either script with `do perl` option or SNMPTT).

The workflow of receiving a trap:

1. `snmptrapd` receives a trap
2. `snmptrapd` passes the trap to the receiver script (Bash, Perl) or SNMPTT
3. The receiver parses, formats and writes the trap to a file
4. Zabbix SNMP trapper reads and parses the trap file
5. For each trap Zabbix finds all "SNMP trapper" items with host interfaces matching the received trap address. Note that only the selected "IP" or "DNS" in host interface is used during the matching.
6. For each found item, the trap is compared to regexp in `snmptrap[regexp]`. The trap is set as the value of **all** matched items. If no matching item is found and there is an `snmptrap.fallback` item, the trap is set as the value of that.
7. If the trap was not set as the value of any item, Zabbix by default logs the unmatched trap. (This is configured by "Log unmatched SNMP traps" in Administration → General → Other.)

Configuring SNMP traps

Configuring the following fields in the frontend is specific for this item type:

- Your host must have an SNMP interface

In *Data collection* → *Hosts*, in the **Host interface** field set an SNMP interface with the correct IP or DNS address. The address from each received trap is compared to the IP and DNS addresses of all SNMP interfaces to find the corresponding hosts.

- Configure the item

In the **Key** field use one of the SNMP trap keys:

Key		
Description	Return value	Comments
snmptrap[regexp] Catches all SNMP traps that match the regular expression specified in **regexp**. If regexp is unspecified, catches any trap.	SNMP trap	This item can be set only for SNMP interfaces. User macros and global regular expressions are supported in the parameter of this item key.
snmptrap.fallback Catches all SNMP traps that were not caught by any of the snmptrap[] items for that interface.	SNMP trap	This item can be set only for SNMP interfaces.

Note:
Multiline regular expression matching is not supported at this time.

Set the **Type of information** to 'Log' for the timestamps to be parsed. Note that other formats such as 'Numeric' are also acceptable but might require a custom trap handler.

Note:
For SNMP trap monitoring to work, it must first be set up correctly (see below).

Setting up SNMP trap monitoring

Configuring Zabbix server/proxy

To read the traps, Zabbix server or proxy must be configured to start the SNMP trapper process and point to the trap file that is being written by SNMPTT or a Bash/Perl trap receiver. To do that, edit the configuration file (zabbix_server.conf or zabbix_proxy.conf):

```
StartSNMPTrapper=1
SNMPTrapperFile=[TRAP FILE]
```

> **Warning:**
> If systemd parameter **PrivateTmp** is used, this file is unlikely to work in */tmp*.

Configuring Bash trap receiver

Requirements: only snmptrapd.

A Bash trap receiver script can be used to pass traps to Zabbix server directly from snmptrapd. To configure it, add the `traphandle` option to snmptrapd configuration file (`snmptrapd.conf`), see example.

Configuring Perl trap receiver

Requirements: Perl, Net-SNMP compiled with --enable-embedded-perl (done by default since Net-SNMP 5.4)

A Perl trap receiver (look for misc/snmptrap/zabbix_trap_receiver.pl) can be used to pass traps to Zabbix server directly from snmptrapd. To configure it:

- add the Perl script to the snmptrapd configuration file (snmptrapd.conf), e.g.:

```
perl do "[FULL PATH TO PERL RECEIVER SCRIPT]";
```

- configure the receiver, e.g:

```
$SNMPTrapperFile = '[TRAP FILE]';
$DateTimeFormat = '[DATE TIME FORMAT]';
```

> **Note:**
> If the script name is not quoted, snmptrapd will refuse to start up with messages, similar to these:

> `Regexp modifiers "/l" and "/a" are mutually exclusive at (eval 2) line 1, at end of line`
> `Regexp modifier "/l" may not appear twice at (eval 2) line 1, at end of line`

Configuring SNMPTT

At first, snmptrapd should be configured to use SNMPTT.

> **Note:**
> For the best performance, SNMPTT should be configured as a daemon using **snmptthandler-embedded** to pass the traps to it. See instructions for configuring SNMPTT.

When SNMPTT is configured to receive the traps, configure `snmptt.ini`:

1. enable the use of the Perl module from the NET-SNMP package:

```
net_snmp_perl_enable = 1
```

2. log traps to the trap file which will be read by Zabbix:

```
log_enable = 1
log_file = [TRAP FILE]
```

3. set the date-time format:

```
date_time_format = %H:%M:%S %Y/%m/%d
```

> **Warning:**
> The "net-snmp-perl" package has been removed in RHEL 8.0-8.2; re-added in RHEL 8.3. For more information, see the known issues.

Now format the traps for Zabbix to recognize them (edit snmptt.conf):

1. Each FORMAT statement should start with "ZBXTRAP [address]", where [address] will be compared to IP and DNS addresses of SNMP interfaces on Zabbix. E.g.:

```
EVENT coldStart .1.3.6.1.6.3.1.1.5.1 "Status Events" Normal
FORMAT ZBXTRAP $aA Device reinitialized (coldStart)
```

2. See more about SNMP trap format below.

Attention:
Do not use unknown traps - Zabbix will not be able to recognize them. Unknown traps can be handled by defining a general event in snmptt.conf:


```
EVENT general .* "General event" Normal
```

SNMP trap format

All customized Perl trap receivers and SNMPTT trap configuration must format the trap in the following way:

```
[timestamp] [the trap, part 1] ZBXTRAP [address] [the trap, part 2]
```

where

- [timestamp] - the timestamp used for log items
- ZBXTRAP - header that indicates that a new trap starts in this line
- [address] - IP address used to find the host for this trap

Note that "ZBXTRAP" and "[address]" will be cut out from the message during processing. If the trap is formatted otherwise, Zabbix might parse the traps unexpectedly.

Example trap:

```
11:30:15 2011/07/27 .1.3.6.1.6.3.1.1.5.3 Normal "Status Events" localhost - ZBXTRAP 192.168.1.1 Link down
```

This will result in the following trap for SNMP interface with IP=192.168.1.1:

```
11:30:15 2011/07/27 .1.3.6.1.6.3.1.1.5.3 Normal "Status Events"
localhost - Link down on interface 2. Admin state: 1. Operational state: 2
```

System requirements

Large file support

Zabbix has large file support for SNMP trapper files. The maximum file size that Zabbix can read is 2^63 (8 EiB). Note that the filesystem may impose a lower limit on the file size.

Log rotation

Zabbix does not provide any log rotation system - that should be handled by the user. The log rotation should first rename the old file and only later delete it so that no traps are lost:

1. Zabbix opens the trap file at the last known location and goes to step 3
2. Zabbix checks if the currently opened file has been rotated by comparing the inode number to the defined trap file's inode number. If there is no opened file, Zabbix resets the last location and goes to step 1.
3. Zabbix reads the data from the currently opened file and sets the new location.
4. The new data are parsed. If this was the rotated file, the file is closed and goes back to step 2.
5. If there was no new data, Zabbix sleeps for 1 second and goes back to step 2.

File system

Because of the trap file implementation, Zabbix needs the file system to support inodes to differentiate files (the information is acquired by a stat() call).

Setup examples using different SNMP protocol versions

This example uses snmptrapd and a Bash receiver script to pass traps to Zabbix server.

Setup:

1. Configure Zabbix to start SNMP trapper and set the trap file. Add to `zabbix_server.conf`:

```
StartSNMPTrapper=1
SNMPTrapperFile=/tmp/my_zabbix_traps.tmp
```

2. Download the Bash script to `/usr/sbin/zabbix_trap_handler.sh`:

```
curl -o /usr/sbin/zabbix_trap_handler.sh https://raw.githubusercontent.com/zabbix/zabbix-docker/6.4/Docker
```

If necessary, adjust the ZABBIX_TRAPS_FILE variable in the script. To use the default value, create the parent directory first:

```
mkdir -p /var/lib/zabbix/snmptraps
```

3. Add the following to `snmtrapd.conf` (refer to working example)

```
traphandle default /bin/bash /usr/sbin/zabbix_trap_handler.sh
```

4. Create an SNMP item TEST:

   Host SNMP interface IP: 127.0.0.1
   Key: `snmptrap["linkup"]`
   Log time format: yyyyMMdd.hhmmss

5. Next we will configure `snmptrapd` for our chosen SNMP protocol version and send test traps using the `snmptrap` utility.

SNMPv1, SNMPv2

SNMPv1 and SNMPv2 protocols rely on "community string" authentication. In the example below we will use "secret" as community string. It must be set to the same value on SNMP trap senders.

Please note that while still widely used in production environments, SNMPv2 doesn't offer any encryption and real sender authentication. The data is sent as plain text and therefore these protocol versions should only be used in secure environments such as private network and should never be used over any public or third-party network.

SNMP version 1 isn't really used these days since it doesn't support 64-bit counters and is considered a legacy protocol.

To enable accepting SNMPv1 or SNMPv2 traps you should add the following line to `snmptrapd.conf`. Replace "secret" with the SNMP community string configured on SNMP trap senders:

```
authCommunity log,execute,net secret
```

Next we can send a test trap using `snmptrap`. We will use the common "link up" OID in this example:

```
snmptrap -v 2c -c secret localhost 0 linkUp.0
```

SNMPv3

SNMPv3 addresses SNMPv1/v2 security issues and provides authentication and encryption. You can use the MD5 or multiple SHA authentication methods and DES/multiple AES as cipher.

To enable accepting SNMPv3 add the following lines to `snmptrapd.conf`:

```
createUser -e 0x8000000001020304 traptest SHA mypassword AES
authuser log,execute traptest
```

Attention:
Please note the "execute" keyword that allows to execute scripts for this user security model.

```
snmptrap -v 3 -n "" -a SHA -A mypassword -x AES -X mypassword -l authPriv -u traptest -e 0x80000000010203
```

Warning:
If you wish to use strong encryption methods such as AES192 or AES256, please use net-snmp starting with version 5.8. You might have to recompile it with `configure` option: `--enable-blumenthal-aes`. Older versions of net-snmp do not support AES192/AES256. See also: http://www.net-snmp.org/wiki/index.php/Strong_Authentication_or_Encryption

Verification

In both examples you will see similar lines in your `/var/lib/zabbix/snmptraps/snmptraps.log`:

```
20220805.102235 ZBXTRAP 127.0.0.1
UDP: [127.0.0.1]:35736->[127.0.0.1]:162
DISMAN-EVENT-MIB::sysUpTimeInstance = 0:0:00:00.00
SNMPv2-MIB::snmpTrapOID.0 = IF-MIB::linkUp.0
```

The item value in Zabbix will be:

```
2022-08-05 10:22:352022-08-05 10:22:33

20220805.102233 UDP: [127.0.0.1]:35736->[127.0.0.1]:162
DISMAN-EVENT-MIB::sysUpTimeInstance = 0:0:00:00.00
SNMPv2-MIB::snmpTrapOID.0 = IF-MIB::linkUp.0
```

See also

- Zabbix blog article on SNMP traps
- Configuring snmptrapd (official net-snmp documentation)
- Configuring snmptrapd to receive SNMPv3 notifications (official net-snmp documentation)

4 IPMI checks

Overview

You can monitor the health and availability of Intelligent Platform Management Interface (IPMI) devices in Zabbix. To perform IPMI checks Zabbix server must be initially configured with IPMI support.

IPMI is a standardized interface for remote "lights-out" or "out-of-band" management of computer systems. It allows to monitor hardware status directly from the so-called "out-of-band" management cards, independently from the operating system or whether the machine is powered on at all.

Zabbix IPMI monitoring works only for devices having IPMI support (HP iLO, DELL DRAC, IBM RSA, Sun SSP, etc).

Since Zabbix 3.4, a new IPMI manager process has been added to schedule IPMI checks by IPMI pollers. Now a host is always polled by only one IPMI poller at a time, reducing the number of open connections to BMC controllers. With those changes it's safe to increase the number of IPMI pollers without worrying about BMC controller overloading. The IPMI manager process is automatically started when at least one IPMI poller is started.

See also known issues for IPMI checks.

Configuration

Host configuration

A host must be configured to process IPMI checks. An IPMI interface must be added, with the respective IP and port numbers, and IPMI authentication parameters must be defined.

See the configuration of hosts for more details.

Server configuration

By default, the Zabbix server is not configured to start any IPMI pollers, thus any added IPMI items won't work. To change this, open the Zabbix server configuration file (zabbix_server.conf) as root and look for the following line:

```
# StartIPMIPollers=0
```

Uncomment it and set poller count to, say, 3, so that it reads:

```
StartIPMIPollers=3
```

Save the file and restart zabbix_server afterwards.

Item configuration

When configuring an item on a host level:

- Select 'IPMI agent' as the *Type*
- Enter an item key that is unique within the host (say, ipmi.fan.rpm)
- For *Host interface* select the relevant IPMI interface (IP and port). Note that an IPMI interface must exist on the host.
- Specify the *IPMI sensor* (for example 'FAN MOD 1A RPM' on Dell Poweredge) to retrieve the metric from. By default, the sensor ID should be specified. It is also possible to use prefixes before the value:
  - `id:` - to specify sensor ID;
  - `name:` - to specify sensor full name. This can be useful in situations when sensors can only be distinguished by specifying the full name.
- Select the respective type of information ('Numeric (float)' in this case; for discrete sensors - 'Numeric (unsigned)'), units (most likely 'rpm') and any other required item attributes

Supported checks

The table below describes in-built items that are supported in IPMI agent checks.

Item key			
▲ ipmi.get	Description	Return value	Comments
	IPMI-sensor related information.	JSON object	This item can be used for the discovery of IPMI sensors. Supported since Zabbix 5.0.0.

Timeout and session termination

IPMI message timeouts and retry counts are defined in OpenIPMI library. Due to the current design of OpenIPMI, it is not possible to make these values configurable in Zabbix, neither on interface nor item level.

IPMI session inactivity timeout for LAN is 60 +/-3 seconds. Currently it is not possible to implement periodic sending of Activate Session command with OpenIPMI. If there are no IPMI item checks from Zabbix to a particular BMC for more than the session timeout configured in BMC then the next IPMI check after the timeout expires will time out due to individual message timeouts, retries or receive error. After that a new session is opened and a full rescan of the BMC is initiated. If you want to avoid unnecessary rescans of the BMC it is advised to set the IPMI item polling interval below the IPMI session inactivity timeout configured in BMC.

Notes on IPMI discrete sensors

To find sensors on a host start Zabbix server with **DebugLevel=4** enabled. Wait a few minutes and find sensor discovery records in Zabbix server logfile:

```
$ grep 'Added sensor' zabbix_server.log
8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:7 id:'CATERR' reading_type:
8358:20130318:111122.170 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'CPU Therm Trip' read
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'System Event Log' re
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'PhysicalSecurity' re
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'IPMI Watchdog' readi
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'Power Unit Stat' rea
8358:20130318:111122.171 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Ctrl %' rea
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:16 id:'P1 Therm Margin' rea
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:13 id:'System Fan 2' readin
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:13 id:'System Fan 3' readin
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'P1 Mem Margin' readi
8358:20130318:111122.172 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'Front Panel Temp' re
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:15 id:'Baseboard Temp' read
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:9 id:'BB +5.0V' reading_typ
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'BB +3.3V STBY' readi
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:9 id:'BB +3.3V' reading_typ
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'BB +1.5V P1 DDR3' re
8358:20130318:111122.173 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:17 id:'BB +1.1V P1 Vccp' re
8358:20130318:111122.174 Added sensor: host:'192.168.1.12:623' id_type:0 id_sz:14 id:'BB +1.05V PCH' readi
```

To decode IPMI sensor types and states, a copy of IPMI 2.0 specifications is available (please note that no further updates to the IPMI specification are planned).

The first parameter to start with is "reading_type". Use "Table 42-1, Event/Reading Type Code Ranges" from the specifications to decode "reading_type" code. Most of the sensors in our example have "reading_type:0x1" which means "threshold" sensor. "Table 42-3, Sensor Type Codes" shows that "type:0x1" means temperature sensor, "type:0x2" - voltage sensor, "type:0x4" - Fan etc. Threshold sensors sometimes are called "analog" sensors as they measure continuous parameters like temperature, voltage, revolutions per minute.

Another example - a sensor with "reading_type:0x3". "Table 42-1, Event/Reading Type Code Ranges" says that reading type codes 02h-0Ch mean "Generic Discrete" sensor. Discrete sensors have up to 15 possible states (in other words - up to 15 meaningful bits). For example, for sensor 'CATERR' with "type:0x7" the "Table 42-3, Sensor Type Codes" shows that this type means "Processor" and the meaning of individual bits is: 00h (the least significant bit) - IERR, 01h - Thermal Trip etc.

There are few sensors with "reading_type:0x6f" in our example. For these sensors the "Table 42-1, Event/Reading Type Code Ranges" advises to use "Table 42-3, Sensor Type Codes" for decoding meanings of bits. For example, sensor 'Power Unit Stat' has type "type:0x9" which means "Power Unit". Offset 00h means "PowerOff/Power Down". In other words if the least significant bit is 1, then server is powered off. To test this bit, the **bitand** function with mask '1' can be used. The trigger expression could be like

```
bitand(last(/www.example.com/Power Unit Stat,#1),1)=1
```

to warn about a server power off.

Notes on discrete sensor names in OpenIPMI-2.0.16, 2.0.17, 2.0.18 and 2.0.19

Names of discrete sensors in OpenIPMI-2.0.16, 2.0.17 and 2.0.18 often have an additional ”0” (or some other digit or letter) appended at the end. For example, while `ipmitool` and OpenIPMI-2.0.19 display sensor names as ”`PhysicalSecurity`” or ”`CATERR`”, in OpenIPMI-2.0.16, 2.0.17 and 2.0.18 the names are ”`PhysicalSecurity0`” or ”`CATERR0`”, respectively.

When configuring an IPMI item with Zabbix server using OpenIPMI-2.0.16, 2.0.17 and 2.0.18, use these names ending with ”0” in the *IPMI sensor* field of IPMI agent items. When your Zabbix server is upgraded to a new Linux distribution, which uses OpenIPMI-2.0.19 (or later), items with these IPMI discrete sensors will become ”NOT SUPPORTED”. You have to change their *IPMI sensor* names (remove the ’0’ in the end) and wait for some time before they turn ”Enabled” again.

Notes on threshold and discrete sensor simultaneous availability

Some IPMI agents provide both a threshold sensor and a discrete sensor under the same name. In Zabbix versions prior to 2.2.8 and 2.4.3, the first provided sensor was chosen. Since versions 2.2.8 and 2.4.3, preference is always given to the threshold sensor.

Notes on connection termination

If IPMI checks are not performed (by any reason: all host IPMI items disabled/notsupported, host disabled/deleted, host in maintenance etc.) the IPMI connection will be terminated from Zabbix server or proxy in 3 to 4 hours depending on the time when Zabbix server/proxy was started.

5 Simple checks

Overview

Simple checks are normally used for remote agent-less checks of services.

Note that Zabbix agent is not needed for simple checks. Zabbix server/proxy is responsible for the processing of simple checks (making external connections, etc).

Examples of using simple checks:

```
net.tcp.service[ftp,,155]
net.tcp.service[http]
net.tcp.service.perf[http,,8080]
net.udp.service.perf[ntp]
```

Note:
User name and *Password* fields in simple check item configuration are used for VMware monitoring items; ignored otherwise.

Supported checks

The item keys are listed without optional parameters and additional information. Click on the item key to see the full details.

See also VMware monitoring item keys.

Item key	Description
icmpping	The host accessibility by ICMP ping.
icmppingloss	The percentage of lost packets.
icmppingsec	The ICMP ping response time.
net.tcp.service	Checks if a service is running and accepting TCP connections.
net.tcp.service.perf	Checks the performance of a TCP service.
net.udp.service	Checks if a service is running and responding to UDP requests.
net.udp.service.perf	Checks the performance of a UDP service.

Item key details

icmpping[<target>,<packets>,<interval>,<size>,<timeout>]

 The host accessibility by ICMP ping.
 Return value: *0* - ICMP ping fails; *1* - ICMP ping successful.

Parameters:

- **target** - the host IP or DNS name;
- **packets** - the number of packets;
- **interval** - the time between successive packets in milliseconds;
- **size** - the packet size in bytes;
- **timeout** - the timeout in milliseconds.

See also the table of default values.

Example:

```
icmpping[,4] #If at least one packet of the four is returned, the item will return 1.
```

icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>]

 The percentage of lost packets.
 Return value: *Float*.

Parameters:

- **target** - the host IP or DNS name;
- **packets** - the number of packets;
- **interval** - the time between successive packets in milliseconds;
- **size** - the packet size in bytes;
- **timeout** - the timeout in milliseconds.

See also the table of default values.

icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]

 The ICMP ping response time (in seconds).
 Return value: *Float*.

Parameters:

- **target** - the host IP or DNS name;
- **packets** - the number of packets;
- **interval** - the time between successive packets in milliseconds;
- **size** - the packet size in bytes;
- **timeout** - the timeout in milliseconds;
- **mode** - possible values: *min*, *max*, or *avg* (default).

Comments:

- Packets which are lost or timed out are not used in the calculation;
- If the host is not available (timeout reached), the item will return 0;
- If the return value is less than 0.0001 seconds, the value will be set to 0.0001 seconds;
- See also the table of default values.

net.tcp.service[service,<ip>,<port>]

 Checks if a service is running and accepting TCP connections.
 Return value: *0* - the service is down; *1* - the service is running.

Parameters:

- **service** - possible values: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details);
- **ip** - the IP address or DNS name (by default the host IP/DNS is used);
- **port** - the port number (by default the standard service port number is used).

Comments:

- Note that with *tcp* service indicating the port is mandatory;
- These checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually);
- Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use `net.tcp.service[tcp,<ip>,port]` for checks like these.

Example:

```
net.tcp.service[ftp,,45] #This item can be used to test the availability of FTP server on TCP port 45.
```

net.tcp.service.perf[service,<ip>,<port>]

 Checks the performance of a TCP service.
 Return value: *Float*: *0.000000* - the service is down; *seconds* - the number of seconds spent while connecting to the service.

Parameters:

- **service** - possible values: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details);
- **ip** - the IP address or DNS name (by default the host IP/DNS is used);
- **port** - the port number (by default the standard service port number is used).

Comments:

- Note that with *tcp* service indicating the port is mandatory;

- Checking of encrypted protocols (like IMAP on port 993 or POP on port 995) is currently not supported. As a workaround, please use `net.tcp.service[tcp,<ip>,port]` for checks like these.

Example:

```
net.tcp.service.perf[ssh] #This item can be used to test the speed of initial response from SSH server.
```

net.udp.service[service,<ip>,<port>]

 Checks if a service is running and responding to UDP requests.
 Return value: *0* - the service is down; *1* - the service is running.

Parameters:

- **service** - possible values: *ntp* (see details);
- **ip** - the IP address or DNS name (by default the host IP/DNS is used);
- **port** - the port number (by default the standard service port number is used).

Example:

```
net.udp.service[ntp,,45] #This item can be used to test the availability of NTP service on UDP port 45.
```

net.udp.service.perf[service,<ip>,<port>]

 Checks the performance of a UDP service.
 Return value: *Float*: *0.000000* - the service is down; *seconds* - the number of seconds spent waiting for response from the service.

Parameters:

- **service** - possible values: *ntp* (see details);
- **ip** - the IP address or DNS name (by default the host IP/DNS is used);
- **port** - the port number (by default the standard service port number is used).

Example:

```
net.udp.service.perf[ntp] #This item can be used to test the response time from NTP service.
```

Attention:
For SourceIP support in LDAP simple checks (e.g. `net.tcp.service[ldap]`), OpenLDAP version 2.6.1 or above is required.

Timeout processing

Zabbix will not process a simple check longer than the Timeout seconds defined in the Zabbix server/proxy configuration file.

ICMP pings

Zabbix uses an external utility **fping** to process ICMP pings (**icmpping**, **icmppingloss**, **icmppingsec**).

Installation

fping is not included with Zabbix and needs to be installed separately:

- Various Unix-based platforms have the fping package in their default repositories, but it is not pre-installed. In this case you can use the package manager to install fping.
- Zabbix provides fping packages for RHEL. Please note that these packages are provided without official support.
- fping can also be compiled from source.

Configuration

Specify fping location in the *FpingLocation* parameter of Zabbix server/proxy configuration file (or *Fping6Location* parameter for using IPv6 addresses).

fping should be executable by the user Zabbix server/proxy run as and this user should have sufficient rights.

See also: Known issues for processing simple checks with fping versions below 3.10.

Default values

Defaults, limits and description of values for ICMP check parameters:

Parameter	Unit	Description	Fping's flag	Defaults set by		Allowed limits by Zabbix	
				fping	**Zabbix**	**min**	**max**
packets	number	number of request packets sent to a target	-C		3	1	10000
interval	milliseconds	time to wait between successive packets to an individual target	-p	1000		20	unlimited
size	bytes	packet size in bytes 56 bytes on x86, 68 bytes on x86_64	-b	56 or 68		24	65507
timeout	milliseconds	**fping v3.x** - timeout to wait after last packet sent, affected by *-C* flag **fping v4.x** - individual timeout for each packet	-t	**fping v3.x** - 500 **fping v4.x** and newer - inherited from *-p* flag, but not more than 2000		50	unlimited

The defaults may differ slightly depending on the platform and version.

In addition, Zabbix uses fping options *-i interval ms* (do not mix up with the item parameter *interval* mentioned in the table above, which corresponds to fping option *-p*) and *-S source IP address* (or *-I* in older fping versions). These options are auto-detected by running checks with different option combinations. Zabbix tries to detect the minimal value in milliseconds that fping allows to use with *-i* by trying 3 values: 0, 1 and 10. The value that first succeeds is then used for subsequent ICMP checks. This process is done by each ICMP pinger process individually.

Auto-detected fping options are invalidated every hour and detected again on the next attempt to perform ICMP check. Set DebugLevel>=4 in order to view details of this process in the server or proxy log file.

Zabbix writes IP addresses to be checked by any of the three *icmpping** keys to a temporary file, which is then passed to fping. If items have different key parameters, only the ones with identical key parameters are written to a single file. All IP addresses written to the single file will be checked by fping in parallel, so Zabbix ICMP pinger process will spend fixed amount of time disregarding the number of IP addresses in the file.

1 VMware monitoring item keys

List of VMware monitoring item keys has been moved to VMware monitoring section.

6 Log file monitoring

Overview

Zabbix can be used for centralized monitoring and analysis of log files with/without log rotation support.

Notifications can be used to warn users when a log file contains certain strings or string patterns.

To monitor a log file you must have:

- Zabbix agent running on the host
- log monitoring item set up

Attention:
The size limit of a monitored log file depends on large file support.

Configuration

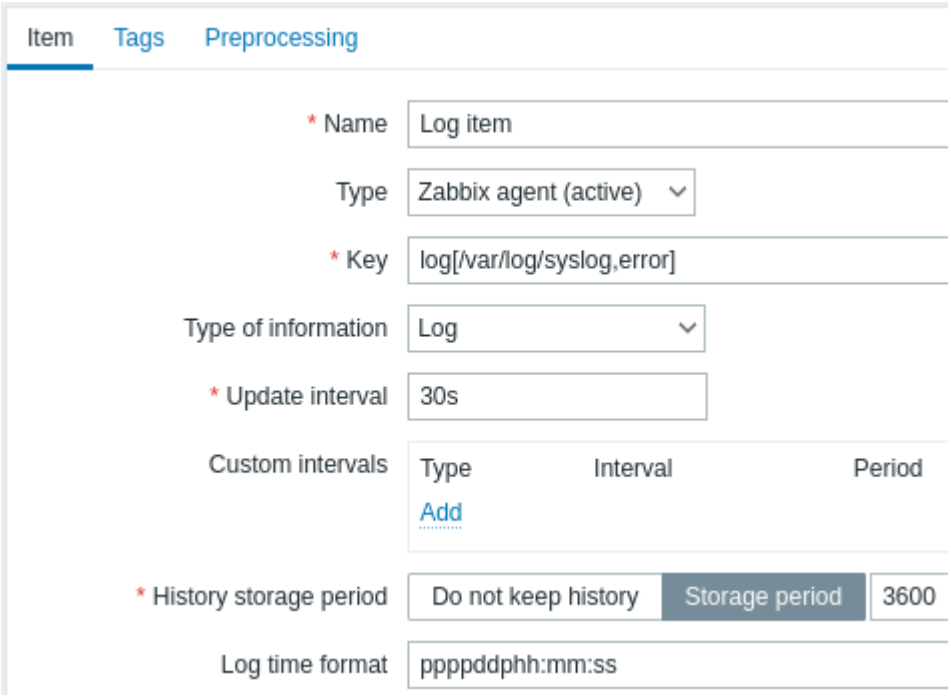
Verify agent parameters

Make sure that in the agent configuration file:

- 'Hostname' parameter matches the host name in the frontend
- Servers in the 'ServerActive' parameter are specified for the processing of active checks

Item configuration

Configure a log monitoring item.



All mandatory input fields are marked with a red asterisk.

Specifically for log monitoring items you enter:

Type	Select **Zabbix agent (active)** here.
Key	Use one of the following item keys: **log[]** or **logrt[]**: These two item keys allow to monitor logs and filter log entries by the content regexp, if present. For example: `log[/var/log/syslog,error]`. Make sure that the file has read permissions for the 'zabbix' user otherwise the item status will be set to 'unsupported'. **log.count[]** or **logrt.count[]**: These two item keys allow to return the number of matching lines only. See supported Zabbix agent item key section for details on using these item keys and their parameters.
Type of information	Prefilled automatically: For log[] or logrt[] items - Log; For log.count[] or logrt.count[] items - `Numeric (unsigned)`. If optionally using the `output` parameter, you may manually select the appropriate type of information other than Log. Note that choosing a non-Log type of information will lead to the loss of local timestamp.
Update interval (in sec)	The parameter defines how often Zabbix agent will check for any changes in the log file. Setting it to 1 second will make sure that you get new records as soon as possible.

Log time format	In this field you may optionally specify the pattern for parsing the log line timestamp. Supported placeholders: * **y**: *Year (1970-2038)* * **M**: *Month (01-12)* * **d**: *Day (01-31)* * **h**: *Hour (00-23)* * **m**: *Minute (00-59)* * **s**: *Second (00-59)* If left blank, the timestamp will be set to 0 in Unix time, representing January 1, 1970. For example, consider the following line from the Zabbix agent log file: " 23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211)." It begins with six character positions for PID, followed by date, time, and the rest of the message. The log time format for this line would be "ppppp:yyyyMMdd:hhmmss". Note that "p" and ":" characters are placeholders and can be any character except "yMdhms".

Important notes

- The server and agent keep the trace of a monitored log's size and last modification time (for logrt) in two counters. Additionally:
  - The agent also internally uses inode numbers (on UNIX/GNU/Linux), file indexes (on Microsoft Windows) and MD5 sums of the first 512 log file bytes for improving decisions when logfiles get truncated and rotated.
  - On UNIX/GNU/Linux systems it is assumed that the file systems where log files are stored report inode numbers, which can be used to track files.
  - On Microsoft Windows Zabbix agent determines the file system type the log files reside on and uses:
    - On NTFS file systems 64-bit file indexes.
    - On ReFS file systems (only from Microsoft Windows Server 2012) 128-bit file IDs.
    - On file systems where file indexes change (e.g. FAT32, exFAT) a fall-back algorithm is used to take a sensible approach in uncertain conditions when log file rotation results in multiple log files with the same last modification time.
  - The inode numbers, file indexes and MD5 sums are internally collected by Zabbix agent. They are not transmitted to Zabbix server and are lost when Zabbix agent is stopped.
  - Do not modify the last modification time of log files with 'touch' utility, do not copy a log file with later restoration of the original name (this will change the file inode number). In both cases the file will be counted as different and will be analyzed from the start, which may result in duplicated alerts.
  - If there are several matching log files for `logrt[]` item and Zabbix agent is following the most recent of them and this most recent log file is deleted, a warning message `"there are no files matching "<regexp mask>" in "<directory>"` is logged. Zabbix agent ignores log files with modification time less than the most recent modification time seen by the agent for the `logrt[]` item being checked.
- The agent starts reading the log file from the point it stopped the previous time.
- The number of bytes already analyzed (the size counter) and last modification time (the time counter) are stored in the Zabbix database and are sent to the agent to make sure the agent starts reading the log file from this point in cases when the agent is just started or has received items which were previously disabled or not supported. However, if the agent has received a non-zero size counter from server, but the logrt[] or logrt.count[] item is unable to find matching files, the size counter is reset to 0 to analyze from the start if the files appear later.
- Whenever the log file becomes smaller than the log size counter known by the agent, the counter is reset to zero and the agent starts reading the log file from the beginning taking the time counter into account.
- If there are several matching files with the same last modification time in the directory, then the agent tries to correctly analyze all log files with the same modification time and avoid skipping data or analyzing the same data twice, although it cannot be guaranteed in all situations. The agent does not assume any particular log file rotation scheme nor determines one. When presented multiple log files with the same last modification time, the agent will process them in a lexicographically descending order. Thus, for some rotation schemes the log files will be analyzed and reported in their original order. For other rotation schemes the original log file order will not be honored, which can lead to reporting matched log file records in altered order (the problem does not happen if log files have different last modification times).
- Zabbix agent processes new records of a log file once per *Update interval* seconds.
- Zabbix agent does not send more than **maxlines** of a log file per second. The limit prevents overloading of network and CPU resources and overrides the default value provided by **MaxLinesPerSecond** parameter in the agent configuration file.
- To find the required string Zabbix will process 10 times more new lines than set in MaxLinesPerSecond. Thus, for example, if a `log[]` or `logrt[]` item has *Update interval* of 1 second, by default the agent will analyze no more than 200 log file records and will send no more than 20 matching records to Zabbix server in one check. By increasing **MaxLinesPerSecond** in the agent configuration file or setting **maxlines** parameter in the item key, the limit can be increased up to 10000 analyzed log file records and 1000 matching records sent to Zabbix server in one check. If the *Update interval* is set to 2 seconds the limits for one check would be set 2 times higher than with *Update interval* of 1 second.

- Additionally, log and log.count values are always limited to 50% of the agent send buffer size, even if there are no non-log values in it. So for the **maxlines** values to be sent in one connection (and not in several connections), the agent BufferSize parameter must be at least maxlines x 2. Zabbix agent can upload data during log gathering and thus free the buffer, whereas Zabbix agent 2 will stop log gathering until the data is uploaded and the buffer is freed, which is performed asynchronously.
- In the absence of log items all agent buffer size is used for non-log values. When log values come in they replace the older non-log values as needed, up to the designated 50%.
- For log file records longer than 256kB, only the first 256kB are matched against the regular expression and the rest of the record is ignored. However, if Zabbix agent is stopped while it is dealing with a long record the agent internal state is lost and the long record may be analyzed again and differently after the agent is started again.
- Special note for "\" path separators: if file_format is "file\.log", then there should not be a "file" directory, since it is not possible to unambiguously define whether "." is escaped or is the first symbol of the file name.
- Regular expressions for `logrt` are supported in filename only, directory regular expression matching is not supported.
- On UNIX platforms a `logrt[]` item becomes NOTSUPPORTED if a directory where the log files are expected to be found does not exist.
- On Microsoft Windows, if a directory does not exist the item will not become NOTSUPPORTED (for example, if directory is misspelled in item key).
- An absence of log files for `logrt[]` item does not make it NOTSUPPORTED. Errors of reading log files for `logrt[]` item are logged as warnings into Zabbix agent log file but do not make the item NOTSUPPORTED.
- Zabbix agent log file can be helpful to find out why a `log[]` or `logrt[]` item became NOTSUPPORTED. Zabbix can monitor its agent log file, except when at DebugLevel=4 or DebugLevel=5.
- Searching for a question mark using a regular expression, e.g. `\?` may result in false positives if the text file contains NUL symbols, as those are replaced with "?" by Zabbix to continue processing the line until the newline character.

Extracting matching part of regular expression

Sometimes we may want to extract only the interesting value from a target file instead of returning the whole line when a regular expression match is found.

Since Zabbix 2.2.0, log items have the ability to extract desired values from matched lines. This is accomplished by the additional **output** parameter in `log` and `logrt` items.

Using the 'output' parameter allows to indicate the "capturing group" of the match that we may be interested in.

So, for example

```
log[/path/to/the/file,"large result buffer allocation.*Entries: ([0-9]+)",,,,\1]
```

should allow returning the entry count as found in the content of:

```
Fr Feb 07 2014 11:07:36.6690 */ Thread Id 1400 (GLEWF) large result
buffer allocation - /Length: 437136/Entries: 5948/Client Ver: >=10/RPC
ID: 41726453/User: AUser/Form: CFG:ServiceLevelAgreement
```

Only the number will be returned because **\1** refers to the first and only capturing group: **([0-9]+)**.

And, with the ability to extract and return a number, the value can be used to define triggers.

Using maxdelay parameter

The 'maxdelay' parameter in log items allows ignoring some older lines from log files in order to get the most recent lines analyzed within the 'maxdelay' seconds.

> **Warning:**
> Specifying 'maxdelay' > 0 may lead to **ignoring important log file records and missed alerts**. Use it carefully at your own risk only when necessary.

By default items for log monitoring follow all new lines appearing in the log files. However, there are applications which in some situations start writing an enormous number of messages in their log files. For example, if a database or a DNS server is unavailable, such applications flood log files with thousands of nearly identical error messages until normal operation is restored. By default, all those messages will be dutifully analyzed and matching lines sent to server as configured in `log` and `logrt` items.

Built-in protection against overload consists of a configurable 'maxlines' parameter (protects server from too many incoming matching log lines) and a 10*'maxlines' limit (protects host CPU and I/O from overloading by agent in one check). Still, there are 2 problems with the built-in protection. First, a large number of potentially not-so-informative messages are reported to server and consume space in the database. Second, due to the limited number of lines analyzed per second the agent may lag behind the newest log records for hours. Quite likely, you might prefer to be sooner informed about the current situation in the log files instead of crawling through old records for hours.

The solution to both problems is using the 'maxdelay' parameter. If 'maxdelay' > 0 is specified, during each check the number of processed bytes, the number of remaining bytes and processing time is measured. From these numbers the agent calculates an estimated delay - how many seconds it would take to analyze all remaining records in a log file.

If the delay does not exceed 'maxdelay' then the agent proceeds with analyzing the log file as usual.

If the delay is greater than 'maxdelay' then the agent **ignores a chunk of a log file by "jumping" over it** to a new estimated position so that the remaining lines could be analyzed within 'maxdelay' seconds.

Note that agent does not even read ignored lines into buffer, but calculates an approximate position to jump to in a file.

The fact of skipping log file lines is logged in the agent log file like this:

```
14287:20160602:174344.206 item:"logrt["/home/zabbix32/test[0-9].log",ERROR,,1000,,,120.0]"
logfile:"/home/zabbix32/test1.log" skipping 679858 bytes
(from byte 75653115 to byte 76332973) to meet maxdelay
```

The "to byte" number is approximate because after the "jump" the agent adjusts the position in the file to the beginning of a log line which may be further in the file or earlier.

Depending on how the speed of growing compares with the speed of analyzing the log file you may see no "jumps", rare or often "jumps", large or small "jumps", or even a small "jump" in every check. Fluctuations in the system load and network latency also affect the calculation of delay and hence, "jumping" ahead to keep up with the "maxdelay" parameter.

Setting 'maxdelay' < 'update interval' is not recommended (it may result in frequent small "jumps").

Notes on handling 'copytruncate' log file rotation

`logrt` with the `copytruncate` option assumes that different log files have different records (at least their timestamps are different), therefore MD5 sums of initial blocks (up to the first 512 bytes) will be different. Two files with the same MD5 sums of initial blocks means that one of them is the original, another - a copy.

`logrt` with the `copytruncate` option makes effort to correctly process log file copies without reporting duplicates. However, things like producing multiple log file copies with the same timestamp, log file rotation more often than logrt[] item update interval, frequent restarting of agent are not recommended. The agent tries to handle all these situations reasonably well, but good results cannot be guaranteed in all circumstances.

Notes on persistent files for log*[] items

Purpose of persistent files

When Zabbix agent is started it receives a list of active checks from Zabbix server or proxy. For log*[] metrics it receives the processed log size and the modification time for finding where to start log file monitoring from. Depending on the actual log file size and modification time reported by file system the agent decides either to continue log file monitoring from the processed log size or re-analyze the log file from the beginning.

A running agent maintains a larger set of attributes for tracking all monitored log files between checks. This in-memory state is lost when the agent is stopped.

The new optional parameter **persistent_dir** specifies a directory for storing this state of log[], log.count[], logrt[] or logrt.count[] item in a file. The state of log item is restored from the persistent file after the Zabbix agent is restarted.

The primary use-case is monitoring of log file located on a mirrored file system. Until some moment in time the log file is written to both mirrors. Then mirrors are split. On the active copy the log file is still growing, getting new records. Zabbix agent analyzes it and sends processed logs size and modification time to server. On the passive copy the log file stays the same, well behind the active copy. Later the operating system and Zabbix agent are rebooted from the passive copy. The processed log size and modification time the Zabbix agent receives from server may not be valid for situation on the passive copy. To continue log file monitoring from the place the agent left off at the moment of file system mirror split the agent restores its state from the persistent file.

Agent operation with persistent file

On startup Zabbix agent knows nothing about persistent files. Only after receiving a list of active checks from Zabbix server (proxy) the agent sees that some log items should be backed by persistent files under specified directories.

During agent operation the persistent files are opened for writing (with fopen(filename, "w")) and overwritten with the latest data. The chance of losing persistent file data if the overwriting and file system mirror split happen at the same time is very small, no special handling for it. Writing into persistent file is NOT followed by enforced synchronization to storage media (fsync() is not called).

Overwriting with the latest data is done after successful reporting of matching log file record or metadata (processed log size and modification time) to Zabbix server. That may happen as often as every item check if log file keeps changing.

No special actions during agent shutdown.

After receiving a list of active checks the agent marks obsolete persistent files for removal. A persistent file becomes obsolete if: 1) the corresponding log item is no longer monitored, 2) a log item is reconfigured with a different **persistent_dir** location than before.

Removing is done with delay 24 hours because log files in NOTSUPPORTED state are not included in the list of active checks but they may become SUPPORTED later and their persistent files will be useful.

If the agent is stopped before 24 hours expire, then the obsolete files will not be deleted as Zabbix agent is not getting info about their location from Zabbix server anymore.

Warning:
Reconfiguring a log item's **persistent_dir** back to the old **persistent_dir** location while the agent is stopped, without deleting the old persistent file by user - will cause restoring the agent state from the old persistent file resulting in missed messages or false alerts.

Naming and location of persistent files

Zabbix agent distinguishes active checks by their keys. For example, logrt[/home/zabbix/test.log] and logrt[/home/zabbix/test.log,] are different items. Modifying the item logrt[/home/zabbix/test.log„,10] in frontend to logrt[/home/zabbix/test.log„,20] will result in deleting the item logrt[/home/zabbix/test.log„,10] from the agent's list of active checks and creating logrt[/home/zabbix/test.log„,20] item (some attributes are carried across modification in frontend/server, not in agent).

The file name is composed of MD5 sum of item key with item key length appended to reduce possibility of collisions. For example, the state of logrt[/home/zabbix50/test.log„„„„„/home/zabbix50/agent_private] item will be kept in persistent file c963ade4008054813bbc0a650bb8e09266.

Multiple log items can use the same value of **persistent_dir**.

persistent_dir is specified by taking into account specific file system layouts, mount points and mount options and storage mirroring configuration - the persistent file should be on the same mirrored filesystem as the monitored log file.

If **persistent_dir** directory cannot be created or does not exist, or access rights for Zabbix agent does not allow to create/write/read/delete files the log item becomes NOTSUPPORTED.

If access rights to persistent storage files are removed during agent operation or other errors occur (e.g. disk full) then errors are logged into the agent log file but the log item does not become NOTSUPPORTED.

Load on I/O

Item's persistent file is updated after successful sending of every batch of data (containing item's data) to server. For example, default 'BufferSize' is 100. If a log item has found 70 matching records then the first 50 records will be sent in one batch, persistent file will be updated, then remaining 20 records will be sent (maybe with some delay when more data is accumulated) in the 2nd batch, and the persistent file will be updated again.

Actions if communication fails between agent and server

Each matching line from `log[]` and `logrt[]` item and a result of each `log.count[]` and `logrt.count[]` item check requires a free slot in the designated 50% area in the agent send buffer. The buffer elements are regularly sent to server (or proxy) and the buffer slots are free again.

While there are free slots in the designated log area in the agent send buffer and communication fails between agent and server (or proxy) the log monitoring results are accumulated in the send buffer. This helps to mitigate short communication failures.

During longer communication failures all log slots get occupied and the following actions are taken:

- `log[]` and `logrt[]` item checks are stopped. When communication is restored and free slots in the buffer are available the checks are resumed from the previous position. No matching lines are lost, they are just reported later.
- `log.count[]` and `logrt.count[]` checks are stopped if `maxdelay = 0` (default). Behavior is similar to `log[]` and `logrt[]` items as described above. Note that this can affect `log.count[]` and `logrt.count[]` results: for example, one check counts 100 matching lines in a log file, but as there are no free slots in the buffer the check is stopped. When communication is restored the agent counts the same 100 matching lines and also 70 new matching lines. The agent now sends count = 170 as if they were found in one check.
- `log.count[]` and `logrt.count[]` checks with `maxdelay > 0`: if there was no "jump" during the check, then behavior is similar to described above. If a "jump" over log file lines took place then the position after "jump" is kept and the counted result is discarded. So, the agent tries to keep up with a growing log file even in case of communication failure.

Handling of regular expression compilation and runtime errors

If a regular expression used in `log[]`, `logrt[]`, `log.count[]` or `logrt.count[]` item cannot be compiled by PCRE or PCRE2 library then the item goes into NOTSUPPORTED state with an error message. To continue monitoring the log item, the regular expression should be fixed.

If the regular expression compiles successfully, but fails at runtime (on some or on all log records), then the log item remains supported and monitoring continues. The runtime error is logged in the Zabbix agent log file (without the log file record).

Note that the logging of regular expression runtime errors is supported since Zabbix 6.4.6.

The logging rate is limited to one runtime error per check to allow Zabbix agent to monitor its own log file. For example, if 10 records are analyzed and 3 records fail with a regexp runtime error, one record is produced in the agent log.

Exception: if MaxLinesPerSecond=1 and update interval=1 (only 1 record is allowed to analyze per check) then regexp runtime errors are not logged.

zabbix_agentd logs the item key in case of a runtime error, zabbix_agent2 logs the item ID to help identify which log item has runtime errors. It is recommended to redesign the regular expression in case of runtime errors.

7 Calculated items

Overview

A calculated item allows to create a calculation based on the values of some existing items. For example, you may want to calculate the hourly average of some item value or to calculate the total value for a group of items. That is what a calculated item is for.

Calculations may use both:

- single values of individual items
- a complex filter to select multiple items for aggregation (see aggregate calculations for details)

Calculated items are a way of creating virtual data sources. All calculations are done by Zabbix server only. The values are periodically calculated based on the arithmetical expression used.

The resulting data is stored in the Zabbix database as for any other item; both history and trend values are stored and graphs can be generated.

Note:
If the calculation result is a float value it will be trimmed to an integer if the calculated item type of information is *Numeric (unsigned)*.
Also, if there is no recent data in the cache and there is no defined querying period in the function, Zabbix will by default go as far back in the past as one week to query the database for historical values.

Calculated items share their syntax with trigger expressions. Comparison to strings is allowed in calculated items. Calculated items may be referenced by macros or other entities same as any other item type.

To use calculated items, choose the item type **Calculated**.

Configurable fields

The **key** is a unique item identifier (per host). You can create any key name using supported symbols.

The calculation definition should be entered in the **Formula** field. There is no connection between the formula and the key. The key parameters are not used in the formula in any way.

The syntax of a simple formula is:

```
function(/host/key,<parameter1>,<parameter2>,...)
```

where:

function	One of the supported functions: last, min, max, avg, count, etc
host	Host of the item that is used for calculation. The current host can be omitted (i.e. as in `function(//key,parameter,...)`).
key	Key of the item that is used for calculation.
parameter(s)	Parameters of the function, if required.

Attention:
User macros in the formula will be expanded if used to reference a function parameter, item filter parameter, or a constant. User macros will NOT be expanded if referencing a function, host name, item key, item key parameter or operator.

A more complex formula may use a combination of functions, operators and brackets. You can use all functions and operators supported in trigger expressions. The logic and operator precedence are exactly the same.

Unlike trigger expressions, Zabbix processes calculated items according to the item update interval, not upon receiving a new value.

All items that are referenced by history functions in the calculated item formula must exist and be collecting data. Also, if you change the item key of a referenced item, you have to manually update any formulas using that key.

A calculated item may become unsupported in several cases:

- referenced item(s)
  - is not found
  - is disabled
  - belongs to a disabled host
  - is not supported (except with nodata() function and operators with unknown values)
- no data to calculate a function
- division by zero
- incorrect syntax used

Usage examples

Example 1

Calculating percentage of free disk space on '/'.

Use of function **last**:

```
100*last(//vfs.fs.size[/,free])/last(//vfs.fs.size[/,total])
```

Zabbix will take the latest values for free and total disk spaces and calculate percentage according to the given formula.

Example 2

Calculating a 10-minute average of the number of values processed by Zabbix.

Use of function **avg**:

```
avg(/Zabbix Server/zabbix[wcache,values],10m)
```

Note that extensive use of calculated items with long time periods may affect performance of Zabbix server.

Example 3

Calculating total bandwidth on eth0.

Sum of two functions:

```
last(//net.if.in[eth0,bytes])+last(//net.if.out[eth0,bytes])
```

Example 4

Calculating percentage of incoming traffic.

More complex expression:

```
100*last(//net.if.in[eth0,bytes])/(last(//net.if.in[eth0,bytes])+last(//net.if.out[eth0,bytes]))
```

See also: Examples of aggregate calculations

1 Aggregate calculations

Overview

Aggregate calculations are a calculated item type allowing to collect information from several items by Zabbix server and then calculate an aggregate, depending on the aggregate function used.

Only unsigned integer and float values (type of information) are supported for aggregate calculation items.

Aggregate calculations do not require any agent running on the host being monitored.

Syntax

To retrieve aggregates, you may:

- list several items for aggregation:

```
aggregate_function(function(/host/key,parameter),function(/host2/key2,parameter),...)
```

Note that `function` here must be a history/trend function.

- use the **foreach** function, as the only parameter, and its item filter to select the required items:

```
aggregate_function(foreach_function(/host/key?[group="host group"],timeperiod))
```

Aggregate function is one of the supported aggregate functions: avg, max, min, sum, etc.

A **foreach** function (e.g. *avg_foreach*, *count_foreach*, etc.) returns one aggregate value for each selected item. Items are selected by using the item filter (`/host/key?[group="host group"]`), from item history.

If some of the items have no data for the requested period, they are ignored in the calculation. If no items have data, the function will return an error.

For more details, see foreach functions.

> **Note:**
> If the aggregate results in a float value it will be trimmed to an integer if the aggregated item type of information is *Numeric (unsigned)*.

An aggregate calculation may become unsupported if:

- none of the referenced items is found (which may happen if the item key is incorrect, none of the items exists or all included groups are incorrect)
- no data to calculate a function

Usage examples

Examples of keys for aggregate calculations.

Example 1

Total disk space of host group 'MySQL Servers'.

```
sum(last_foreach(/*/vfs.fs.size[/,total]?[group="MySQL Servers"]))
```

Example 2

Sum of latest values of all items matching net.if.in[*] on the host.

```
sum(last_foreach(/host/net.if.in[*]))
```

Example 3

Average processor load of host group 'MySQL Servers'.

```
avg(last_foreach(/*/system.cpu.load[,avg1]?[group="MySQL Servers"]))
```

Example 4

5-minute average of the number of queries per second for host group 'MySQL Servers'.

```
avg(avg_foreach(/*/mysql.qps?[group="MySQL Servers"],5m))
```

Example 5

Average CPU load on all hosts in multiple host groups that have the specific tags.

```
avg(last_foreach(/*/system.cpu.load?[(group="Servers A" or group="Servers B" or group="Servers C") and (ta
```

Example 6

Calculation used on the latest item value sums of a whole host group.

```
sum(last_foreach(/*/net.if.out[eth0,bytes]?[group="video"])) / sum(last_foreach(/*/nginx_stat.sh[active]?[
```

Example 7

The total number of unsupported items in host group 'Zabbix servers'.

```
sum(last_foreach(/*/zabbix[host,,items_unsupported]?[group="Zabbix servers"]))
```

Examples of correct/incorrect syntax

Expressions (including function calls) cannot be used as history, trend, or foreach function parameters. However, those functions themselves can be used in other (non-historical) function parameters.

Expression	Example
Valid	`avg(last(/host/key1),last(/host/key2)*10,last(/host/key1)*100)` `max(avg(avg_foreach(/*/system.cpu.load?[group="Servers A"],5m)),avg(avg_foreach(/*/system.cpu.load?[group="Servers B"],5m)),avg(avg_foreach(/*/system.cpu.load?[group="Servers C"],5m)))`

Invalid	`sum(/host/key,10+2)`
	`sum(/host/key, avg(10,2))`
	`sum(/host/key,last(/host/key2))`

Note that in an expression like:

`sum(sum_foreach(//resptime[*],5m))/sum(count_foreach(//resptime[*],5m))`

it cannot be guaranteed that both parts of the equation will always have the same set of values. While one part of the expression is evaluated, a new value for the requested period may arrive and then the other part of the expression will have a different set of values.

8 Internal checks

Overview

Internal checks allow to monitor the internal processes of Zabbix. In other words, you can monitor what goes on with Zabbix server or Zabbix proxy.

Internal checks are calculated:

- on Zabbix server - if the host is monitored by server
- on Zabbix proxy - if the host is monitored by proxy

Internal checks are processed by server or proxy regardless of the host maintenance status.

To use this item, choose the **Zabbix internal** item type.

Note:
Internal checks are processed by Zabbix pollers.

Performance

Using some internal items may negatively affect performance. These items are:

- `zabbix[host,,items]`
- `zabbix[host,,items_unsupported]`
- `zabbix[hosts]`
- `zabbix[items]`
- `zabbix[items_unsupported]`
- `zabbix[queue,,]`
- `zabbix[requiredperformance]`
- `zabbix[stats,,,queue,,]`
- `zabbix[triggers]`

The System information and Queue frontend sections are also affected.

Supported checks

The item keys are listed without customizable parameters and additional information. Click on the item key to see the full details.

Item key	Description
zabbix[boottime]	The startup time of Zabbix server or Zabbix proxy process in seconds.
zabbix[cluster,discovery,nodes]	Discovers the high availability cluster nodes.
zabbix[connector_queue]	The count of values enqueued in the connector queue.
zabbix[host,,items]	The number of enabled items (supported and not supported) on the host.
zabbix[host,,items_unsupported]	The number of enabled unsupported items on the host.
zabbix[host,maintenance]	The current maintenance status of the host.
zabbix[host,active_agent,available]	The availability of active agent checks on the host.
zabbix[host,discovery,interfaces]	The details of all configured interfaces of the host in Zabbix frontend.
zabbix[host,available]	The availability of the main interface of a particular type of checks on the host.

Item key	Description
zabbix[hosts]	The number of monitored hosts.
zabbix[items]	The number of enabled items (supported and not supported).
zabbix[items_unsupported]	The number of unsupported items.
zabbix[java„]	The information about Zabbix Java gateway.
zabbix[lld_queue]	The count of values enqueued in the low-level discovery processing queue.
zabbix[preprocessing_queue]	The count of values enqueued in the preprocessing queue.
zabbix[process„,]	The percentage of time a particular Zabbix process or a group of processes (identified by <type> and <mode>) spent in <state>.
zabbix[proxy„]	The information about Zabbix proxy.
zabbix[proxy,discovery]	The list of Zabbix proxies.
zabbix[proxy_history]	The number of values in the proxy history table waiting to be sent to the server.
zabbix[queue„]	The number of monitored items in the queue which are delayed at least by <from> seconds, but less than <to> seconds.
zabbix[rcache„]	The availability statistics of the Zabbix configuration cache.
zabbix[requiredperformance]	The required performance of Zabbix server or Zabbix proxy, in new values per second expected.
zabbix[stats„]	The internal metrics of a remote Zabbix server or proxy.
zabbix[stats„,queue„]	The internal queue metrics of a remote Zabbix server or proxy.
zabbix[tcache„]	The effectiveness statistics of the Zabbix trend function cache.
zabbix[triggers]	The number of enabled triggers in Zabbix database, with all items enabled on enabled hosts.
zabbix[uptime]	The uptime of the Zabbix server or proxy process in seconds.
zabbix[vcache,buffer,]	The availability statistics of the Zabbix value cache.
zabbix[vcache,cache,]	The effectiveness statistics of the Zabbix value cache.
zabbix[version]	The version of Zabbix server or proxy.
zabbix[vmware,buffer,]	The availability statistics of the Zabbix vmware cache.
zabbix[wcache„]	The statistics and availability of the Zabbix write cache.

Item key details

- Parameters without angle brackets are mandatory and must be used *as is* (for example, ”host” and ”available” in `zabbix[host,<type>,available]`).
- Parameters with angle brackets **< >** must be replaced with a valid value. If a parameter has a default value, it can be omitted.
- Values for items and item parameters labeled ”not supported on proxy” can only be retrieved if the host is monitored by server. Conversely, values ”not supported on server” can only be retrieved if the host is monitored by proxy.

zabbix[boottime]

 The startup time of Zabbix server or Zabbix proxy process in seconds.
 Return value: *Integer*.

zabbix[cluster,discovery,nodes]

 Discovers the high availability cluster nodes.
 Return value: *JSON object*.

Comments:

- This item can be used in low-level discovery.

zabbix[connector_queue]

 The count of values enqueued in the connector queue.
 Return value: *Integer*.

Comments:

- This item is supported since Zabbix 6.4.0.

zabbix[host„items]

 The number of enabled items (supported and not supported) on the host.
 Return value: *Integer*.

zabbix[host„items_unsupported]

 The number of enabled unsupported items on the host.
 Return value: *Integer*.

zabbix[host„maintenance]

 The current maintenance status of the host.
 Return values: *0* - normal state; *1* - maintenance with data collection; *2* - maintenance without data collection.

Comments:

- This item is always processed by Zabbix server regardless of the host location (on server or proxy). The proxy will not receive this item with configuration data.
- The second parameter must be empty and is reserved for future use.

zabbix[host,active_agent,available]

 The availability of active agent checks on the host.
 Return values: *0* - unknown; *1* - available; *2* - not available.

zabbix[host,discovery,interfaces]

 The details of all configured interfaces of the host in Zabbix frontend.
 Return value: *JSON object*.

Comments:

- This item can be used in low-level discovery.
- This item is not supported on Zabbix proxy.

zabbix[host,<type>,available]

 The availability of the main interface of a particular type of checks on the host.
 Return values: *0* - not available; *1* - available; *2* - unknown.

Parameters:

- **type** - *agent*, *snmp*, *ipmi*, or *jmx*.

Comments:

- The item value is calculated according to the configuration parameters regarding host unreachability/unavailability.

zabbixhosts

 The number of monitored hosts.
 Return value: *Integer*.

zabbixitems

 The number of enabled items (supported and not supported).
 Return value: *Integer*.

zabbix[items_unsupported]

 The number of unsupported items.
 Return value: *Integer*.

zabbix[java,,<param>]

 The information about Zabbix Java gateway.
 Return values: *1* - if <param> is *ping*; *Java gateway version* - if <param> is *version* (for example: ”2.0.0”).

Parameters:

- **param** - *ping* or *version*.

Comments:

- This item can be used to check Java gateway availability using the `nodata()` trigger function.
- The second parameter must be empty and is reserved for future use.

zabbix[lld_queue]

 The count of values enqueued in the low-level discovery processing queue.
 Return value: *Integer*.

Comments:

- This item can be used to monitor the low-level discovery processing queue length.

zabbix[preprocessing_queue]

 The count of values enqueued in the preprocessing queue.
 Return value: *Integer*.

Comments:

- This item can be used to monitor the preprocessing queue length.

zabbix[process,<type>,<mode>,<state>]

 The percentage of time a particular Zabbix process or a group of processes (identified by <type> and <mode>) spent in <state>. It is calculated for the last minute only.
 Return value: *Float*.

Parameters:

- **type** - for server processes: *alert manager*, *alert syncer*, *alerter*, *availability manager*, *configuration syncer*, *connector manager*, *connector worker*, *discoverer*, *escalator*, *ha manager*, *history poller*, *history syncer*, *housekeeper*, *http poller*, *icmp pinger*, *ipmi manager*, *ipmi poller*, *java poller*, *lld manager*, *lld worker*, *odbc poller*, *poller*, *preprocessing manager*, *preprocessing worker*, *proxy poller*, *self-monitoring*, *service manager*, *snmp trapper*, *task manager*, *timer*, *trapper*, *trigger housekeeper*, *unreachable poller*, *vmware collector*;
for proxy processes: *availability manager*, *configuration syncer*, *data sender*, *discoverer*, *history syncer*, *housekeeper*, *http poller*, *icmp pinger*, *ipmi manager*, *ipmi poller*, *java poller*, *odbc poller*, *poller*, *preprocessing manager*, *preprocessing worker*, *self-monitoring*, *snmp trapper*, *task manager*, *trapper*, *unreachable poller*, *vmware collector*;
- **mode** - *avg* - average value for all processes of a given type (default);
count - returns number of forks for a given process type, <state> should not be specified;
max - maximum value;
min - minimum value;
<*process number*> - process number (between 1 and the number of pre-forked instances; for example, if 4 trappers are running, the value is between 1 and 4);
- **state** - *busy* - process is in busy state, for example, the processing request (default);
idle - process is in idle state doing nothing.

Comments:

- If <mode> is a Zabbix process number that is not running (for example, with 5 pollers running the <mode> is specified to be 6), such an item will turn unsupported.
- Minimum and maximum refers to the usage percentage for a single process. So if in a group of 3 pollers usage percentages per process were 2, 18 and 66, min would return 2 and max would return 66.
- Processes report what they are doing in shared memory and the self-monitoring process summarizes that data each second. State changes (busy/idle) are registered upon change - thus a process that becomes busy registers as such and doesn't change or update the state until it becomes idle. This ensures that even fully hung processes will be correctly registered as 100% busy.
- Currently, "busy" means "not sleeping", but in the future additional states might be introduced - waiting for locks, performing database queries, etc.
- On Linux and most other systems, resolution is 1/100 of a second.

Examples:

```
zabbix[process,poller,avg,busy] #the average time of poller processes spent doing something during the las
zabbix[process,"icmp pinger",max,busy] #the maximum time spent doing something by any ICMP pinger process
zabbix[process,"history syncer",2,busy] #the time spent doing something by history syncer number 2 during
zabbix[process,trapper,count] #the amount of currently running trapper processes
```

zabbix[proxy,<name>,<param>]

 The information about Zabbix proxy.
 Return value: *Integer*.

Parameters:

- **name** - the proxy name;
- **param** - *lastaccess* - the timestamp of the last heartbeat message received from proxy;
delay - how long the collected values are unsent; calculated as "proxy delay" + ("current server time" - "proxy lastaccess"), where "proxy delay" is the difference between the current proxy time and the timestamp of the oldest unsent value on proxy.

Comments:

- This item is always processed by Zabbix server regardless of host location (on server or proxy).
- The `fuzzytime()` function can be used to check the availability of proxy.

Example:

```
zabbix[proxy,"Germany",lastaccess] #the timestamp of the last heartbeat message received from "Germany" pr
```

zabbix[proxy,discovery]

 The list of Zabbix proxies with name, mode, encryption, compression, version, last seen, host count, item count, required values per second (vps) and version status (current/outdated/unsupported).
 Return value: *JSON object*.

zabbix[proxy_history]

 The number of values in the proxy history table waiting to be sent to the server.
 Return values: *Integer*.

Comments:

- This item is not supported on Zabbix server.

zabbix[queue,<from>,<to>]

 The number of monitored items in the queue which are delayed at least by <from> seconds, but less than <to> seconds.
 Return value: *Integer*.

Parameters:

- **from** - delayed by at least (default is 6 seconds);
- **to** - delayed by at most (default is infinity).

Comments:

- Time-unit symbols (s,m,h,d,w) are supported in the parameters.

zabbix[rcache,<cache>,<mode>]

 The availability statistics of the Zabbix configuration cache.
 Return values: *Integer* (for size); *Float* (for percentage).

Parameters:

- **cache** - *buffer*;
- **mode** - *total* - the total size of buffer;
free - the size of free buffer;
pfree - the percentage of free buffer;
used - the size of used buffer;
pused - the percentage of used buffer.

zabbix[requiredperformance]

 The required performance of Zabbix server or Zabbix proxy, in new values per second expected.
 Return value: *Float*.

Comments:

- Approximately correlates with "Required server performance, new values per second" in *Reports* → *System information*.

zabbix[stats,<ip>,<port>]

 The internal metrics of a remote Zabbix server or proxy.
 Return values: *JSON object*.

Parameters:

- **ip** - the IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1);
- **port** - the port of server/proxy to be remotely queried (default is 10051).

Comments:

- The stats request will only be accepted from the addresses listed in the 'StatsAllowedIP' server/proxy parameter on the target instance.
- A selected set of internal metrics is returned by this item. For details, see Remote monitoring of Zabbix stats.

zabbix[stats,<ip>,<port>,queue,<from>,<to>]

 The internal queue metrics (see `zabbix[queue,<from>,<to>]`) of a remote Zabbix server or proxy.
 Return values: *JSON object*.

Parameters:

- **ip** - the IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1);
- **port** - the port of server/proxy to be remotely queried (default is 10051);
- **from** - delayed by at least (default is 6 seconds);
- **to** - delayed by at most (default is infinity).

Comments:

- The stats request will only be accepted from the addresses listed in the 'StatsAllowedIP' server/proxy parameter on the target instance.
- A selected set of internal metrics is returned by this item. For details, see Remote monitoring of Zabbix stats.

zabbix[tcache,<cache>,<parameter>]

 The effectiveness statistics of the Zabbix trend function cache.
 Return values: *Integer* (for size); *Float* (for percentage).

Parameters:

- **cache** - *buffer*;
- **mode** - *all* - total cache requests (default);
hits - cache hits;
phits - percentage of cache hits;
misses - cache misses;
pmisses - percentage of cache misses;
items - the number of cached items;
requests - the number of cached requests;
pitems - percentage of cached items from cached items + requests. Low percentage most likely means that the cache size can be reduced.

Comments:

- This item is not supported on Zabbix proxy.

zabbixtriggers

 The number of enabled triggers in Zabbix database, with all items enabled on enabled hosts.
 Return value: *Integer*.

Comments:

- This item is not supported on Zabbix proxy.

zabbix[uptime]

 The uptime of the Zabbix server or proxy process in seconds.
 Return value: *Integer*.

zabbix[vcache,buffer,<mode>]

 The availability statistics of the Zabbix value cache.
 Return values: *Integer* (for size); *Float* (for percentage).

Parameters:

- **mode** - *total* - the total size of buffer;
free - the size of free buffer;
pfree - the percentage of free buffer;
used - the size of used buffer;
pused - the percentage of used buffer.

Comments:

- This item is not supported on Zabbix proxy.

zabbix[vcache,cache,<parameter>]

 The effectiveness statistics of the Zabbix value cache.
 Return values: *Integer*. With the *mode* parameter returns: 0 - normal mode; 1 - low memory mode.

Parameters:

- **parameter** - *requests* - the total number of requests;
hits - the number of cache hits (history values taken from the cache);
misses - the number of cache misses (history values taken from the database);
mode - the value cache operating mode.

Comments:

- Once the low-memory mode has been switched on, the value cache will remain in this state for 24 hours, even if the problem that triggered this mode is resolved sooner.
- You may use this key with the *Change per second* preprocessing step in order to get values-per-second statistics.
- This item is not supported on Zabbix proxy.

zabbixversion

 The version of Zabbix server or proxy.
 Return value: *String*. For example: `6.0.0beta1`.

zabbix[vmware,buffer,<mode>]

 The availability statistics of the Zabbix vmware cache.
 Return values: *Integer* (for size); *Float* (for percentage).

Parameters:

- **mode** - *total* - the total size of buffer;
free - the size of free buffer;
pfree - the percentage of free buffer;
used - the size of used buffer;
pused - the percentage of used buffer.

zabbix[wcache,<cache>,<mode>]

 The statistics and availability of the Zabbix write cache.
 Return values: *Integer* (for number/size); *Float* (for percentage).

Parameters:

- **cache** - *values*, *history*, *index*, or *trend*;
- **mode** - (with *values*) *all* (default) - the total number of values processed by Zabbix server/proxy, except unsupported items (counter);
float - the number of processed float values (counter);
uint - the number of processed unsigned integer values (counter);
str - the number of processed character/string values (counter);
log - the number of processed log values (counter);
text - the number of processed text values (counter);
not supported - the number of times item processing resulted in item becoming unsupported or keeping that state (counter);
(with *history*, *index*, *trend* cache) *pfree* (default) - the percentage of free buffer;
total - the total size of buffer;
free - the size of free buffer;
used - the size of used buffer;
pused - the percentage of used buffer.

Comments:

- Specifying <cache> is mandatory. The `trend` cache parameter is not supported with Zabbix proxy.
- The history cache is used to store item values. A low number indicates performance problems on the database side.
- The history index cache is used to index the values stored in the history cache.

- After the history cache is filled and then cleared, the history index cache will still keep some data. This behavior is expected and helps the system run more efficiently by avoiding the extra processing required to constantly resize the memory.
- The trend cache stores the aggregate for the current hour for all items that receive data.
- You may use the zabbix[wcache,values] key with the *Change per second* preprocessing step in order to get values-per-second statistics.

9 SSH checks

Overview

SSH checks are performed as agent-less monitoring. Zabbix agent is not needed for SSH checks.

To perform SSH checks Zabbix server must be initially configured with SSH2 support (libssh or libssh2). See also: Requirements.

> **Attention:**
> Starting with RHEL 8, only libssh is supported. For other distributions, libssh is suggested over libssh2.

Configuration

Passphrase authentication

SSH checks provide two authentication methods - a user/password pair and key-file based.

If you do not intend to use keys, no additional configuration is required, besides linking libssh or libssh2 to Zabbix, if you're building from source.

Key file authentication

To use key based authentication for SSH items, certain changes to the server configuration are required.

Open the Zabbix server configuration file (*zabbix_server.conf*) as `root` and look for the following line:

```
##### SSHKeyLocation=
```

Uncomment it and set the full path to the folder where the public and private keys will be located:

```
SSHKeyLocation=/home/zabbix/.ssh
```

Save the file and restart Zabbix server afterwards.

The path */home/zabbix* here is the home directory for the *zabbix* user account, and *.ssh* is a directory where by default public and private keys will be generated by an ssh-keygen command inside the home directory.

Usually installation packages of Zabbix server from different OS distributions create the *zabbix* user account with a home directory elsewhere, for example, */var/lib/zabbix* (as for system accounts).

Before generating the keys, you could reallocate the home directory to */home/zabbix*, so that it corresponds with the `SSHKeyLocation` Zabbix server configuration parameter mentioned above.

> **Note:**
> The following steps can be skipped if *zabbix* account has been added manually according to the installation section. In such a case the home directory for the *zabbix* account is most likely already */home/zabbix*.

To change the home directory of the *zabbix* user account, all working processes which are using it have to be stopped:

```
systemctl stop zabbix-agent
systemctl stop zabbix-server
```

To change the home directory location with an attempt to move it (if it exists) the following command should be executed:

```
usermod -m -d /home/zabbix zabbix
```

It is also possible that a home directory did not exist in the old location, so it should be created at the new location. A safe attempt to do that is:

```
test -d /home/zabbix || mkdir /home/zabbix
```

To be sure that all is secure, additional commands could be executed to set permissions to the home directory:

```
chown zabbix:zabbix /home/zabbix
chmod 700 /home/zabbix
```

Previously stopped processes can now be started again:

```
systemctl start zabbix-agent
systemctl start zabbix-server
```

Now, the steps to generate the public and private keys can be performed with the following commands (for better readability, command prompts are commented out):

```
sudo -u zabbix ssh-keygen -t rsa
##### Generating public/private rsa key pair.
##### Enter file in which to save the key (/home/zabbix/.ssh/id_rsa):
/home/zabbix/.ssh/id_rsa
##### Enter passphrase (empty for no passphrase):
<Leave empty>
##### Enter same passphrase again:
<Leave empty>
##### Your identification has been saved in /home/zabbix/.ssh/id_rsa.
##### Your public key has been saved in /home/zabbix/.ssh/id_rsa.pub.
##### The key fingerprint is:
##### 90:af:e4:c7:e3:f0:2e:5a:8d:ab:48:a2:0c:92:30:b9 zabbix@it0
##### The key's randomart image is:
##### +--[ RSA 2048]----+
##### |                 |
##### |       .         |
##### |      o          |
##### | .     o         |
##### |+     . S        |
##### |.+   o =         |
##### |E .   * =        |
##### |=o . ..* .       |
##### |... oo.o+        |
##### +-----------------+
```

Note:
The public and private keys (*id_rsa.pub* and *id_rsa*) have been generated by default in the */home/zabbix/.ssh* directory, which corresponds to the Zabbix server `SSHKeyLocation` configuration parameter.

Attention:
Key types other than "rsa" may be supported by the ssh-keygen tool and SSH servers but they may not be supported by libssh2 used by Zabbix.

Shell configuration form

This step should be performed only once for every host that will be monitored by SSH checks.

By using the following commands, the **public** key file can be installed on a remote host *10.10.10.10*, so that the SSH checks can be performed with a *root* account (for better readability, command prompts are commented out):

```
sudo -u zabbix ssh-copy-id root@10.10.10.10
##### The authenticity of host '10.10.10.10 (10.10.10.10)' can't be established.
##### RSA key fingerprint is 38:ba:f2:a4:b5:d9:8f:52:00:09:f7:1f:75:cc:0b:46.
##### Are you sure you want to continue connecting (yes/no)?
yes
##### Warning: Permanently added '10.10.10.10' (RSA) to the list of known hosts.
##### root@10.10.10.10's password:
<Enter root password>
##### Now try logging into the machine, with "ssh 'root@10.10.10.10'",
##### and check to make sure that only the key(s) you wanted were added.
```

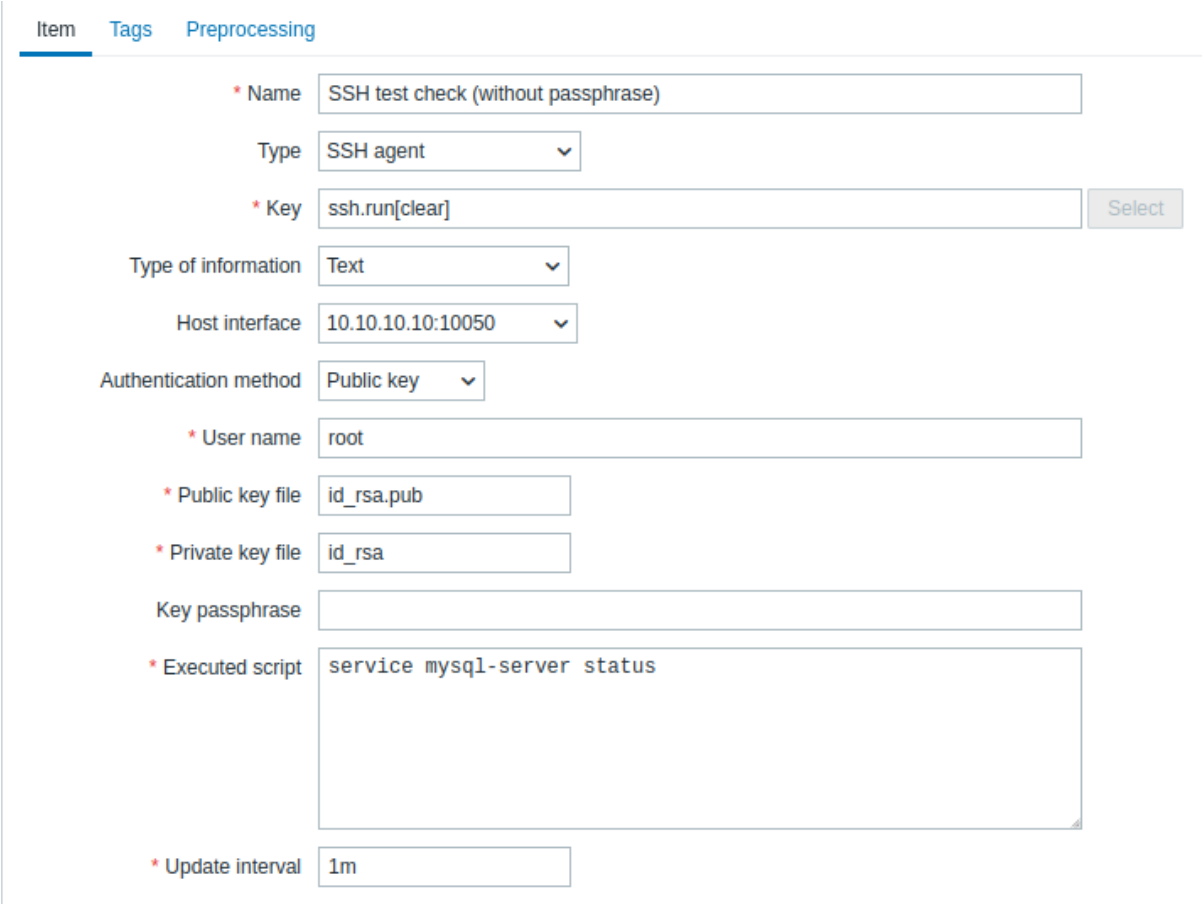
Now it is possible to check the SSH login using the default private key (*/home/zabbix/.ssh/id_rsa*) for the *zabbix* user account:

```
sudo -u zabbix ssh root@10.10.10.10
```

If the login is successful, then the configuration part in the shell is finished and the remote SSH session can be closed.

Item configuration

Actual command(s) to be executed must be placed in the *Executed script* field in the item configuration. Multiple commands can be executed one after another by placing them on a new line. In this case returned values will also be formatted as multilined.



All mandatory input fields are marked with a red asterisk.

The fields that require specific information for SSH items are:

Parameter	Description	Comments
Type	Select **SSH agent** here.	

Parameter	Description	Comments

Key	Unique (per host) item key in the format **ssh.run[unique short description,<ip>,<port>,<encoding>,<ssh options>]**	**unique short description** is required and should be unique for each SSH item per host.
		
		Default port is 22, not the port specified in the interface to which this item is assigned.
		
		ssh options (supported since version 6.4.0) allow to pass additional SSH options in the format *key1=value1;key2=value2,value3*. Multiple values for one key can be passed separated by comma (in this case, the parameter must be quoted); multiple option keys can be passed separated by semicolon.
		
		The following option keys are supported: `KexAlgorithms`, `HostkeyAlgorithms`, `Ciphers`, `MACs`. Option key and value support depends on the SSH library; if an option is not supported, an error will be returned, and the item will become unsupported.
		
		Note that "+" sign for appending cipher settings and "!" for disabling specific cipher settings (as in GnuTLS and OpenSSL) are not supported.
		
		Examples:
		=>
		`ssh.run[KexAlgorithms,127.0.0.1,,,Ciphers=aes128-`
		=>
		`ssh.run[KexAlgorithms,,,,"KexAlgorithms=diffie-he`

Authentication method	One of the "Password" or "Public key".	
User name	User name to authenticate on remote host. Required.	
Public key file	File name of public key if *Authentication method* is "Public key". Required.	Example: *id_rsa.pub* - default public key file name generated by a command ssh-keygen.
Private key file	File name of private key if *Authentication method* is "Public key". Required.	Example: *id_rsa* - default private key file name.
Password or *Key passphrase*	Password to authenticate or Passphrase **if** it was used for the private key.	Leave the *Key passphrase* field empty if passphrase was not used. See also known issues regarding passphrase usage.
Executed script	Executed shell command(s) using SSH remote session.	The return value of the executed shell command(s) is limited to 16MB (including trailing whitespace that is truncated); database limits also apply. Note that the libssh2 library may truncate executable scripts to ~32kB. Examples: *date +%s* *systemctl status mysql-server* *ps auxww | grep httpd | wc -l*

10 Telnet checks

Overview

Telnet checks are performed as agent-less monitoring. Zabbix agent is not needed for Telnet checks.

Configurable fields

Actual command(s) to be executed must be placed in the **Executed script** field in the item configuration.
Multiple commands can be executed one after another by placing them on a new line. In this case returned value also will be formatted as multilined.

Supported characters that the shell prompt can end with:

- $
- #
- >
- %

Note:
A telnet prompt line which ended with one of these characters will be removed from the returned value, but only for the first command in the commands list, i.e. only at a start of the telnet session.

Key	Description
telnet.run[<unique short description>,<ip>,<port>,<encoding>]	Run a command on a remote device using telnet connection

Attention:
If a telnet check returns a value with non-ASCII characters and in non-UTF8 encoding then the *<encoding>* parameter of the key should be properly specified. See encoding of returned values page for more details.

11 External checks

Overview

External check is a check executed by Zabbix server by running a shell script or a binary. However, when hosts are monitored by a Zabbix proxy, the external checks are executed by the proxy.

External checks do not require any agent running on a host being monitored.

The syntax of the item key is:

`script[<parameter1>,<parameter2>,...]`

Where:

ARGUMENT	DEFINITION
script	Name of a shell script or a binary.
parameter(s)	Optional command line parameters.

If you don't want to pass any parameters to the script you may use:

```
script[] or
script
```

Zabbix server or proxy will search the directory specified for external scripts and execute the command (see `ExternalScripts` parameter in Zabbix server/proxy configuration file). The command will be executed under the same user as Zabbix server/proxy, so any access permissions or environment variables should be handled in a wrapper script, if necessary. Permissions on the command should also allow that user to execute it. Only commands in the specified directory are available for execution.

Warning:
Do not overuse external checks, as each script requires starting a fork process by Zabbix server/proxy, and running many scripts can significantly decrease Zabbix performance.

Usage example

Executing the script **check_oracle.sh** with the first parameters '-h'. The second parameter will be replaced by IP address or DNS name, depending on the selection in the host properties.

```
check_oracle.sh["-h","{HOST.CONN}"]
```

Assuming host is configured to use IP address, Zabbix server/proxy will execute:

```
check_oracle.sh '-h' '192.168.1.4'
```

External check result

The return value of an external check is a standard output together with a standard error produced by the check.

Attention:
An item that returns text (character, log, or text type of information) will not become unsupported in case of a standard error output.

The return value is limited to 16MB (including trailing whitespace that is truncated); database limits also apply.

If the requested script is not found or Zabbix server/proxy has no permissions to execute it, the item will become unsupported and a corresponding error message will be displayed.

In case of a timeout, the item will become unsupported, a corresponding error message will be displayed, and the process forked for the script will be terminated.

12 Trapper items

Overview

Trapper items accept incoming data instead of querying for it.

It is useful for any data you might want to ”push” into Zabbix.

To use a trapper item you must:

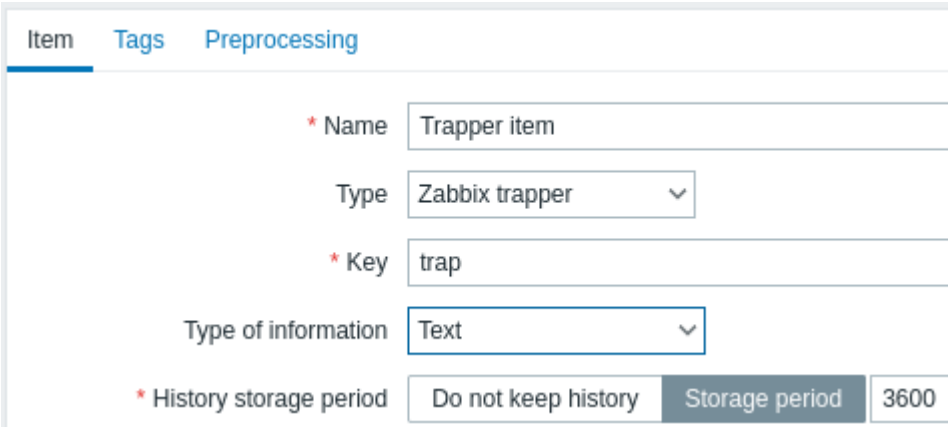
- have a trapper item set up in Zabbix
- send in the data into Zabbix

Configuration

Item configuration

To configure a trapper item:

- Go to: *Data collection → Hosts*
- Click on *Items* in the row of the host
- Click on *Create item*
- Enter parameters of the item in the form



All mandatory input fields are marked with a red asterisk.

The fields that require specific information for trapper items are:

Type	Select **Zabbix trapper** here.
Key	Enter a key that will be used to recognize the item when sending in data.
Type of information	Select the type of information that will correspond the format of data that will be sent in.

Allowed hosts	List of comma delimited IP addresses, optionally in CIDR notation, or DNS names. If specified, incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. Note that "IPv4-compatible IPv6 addresses" (0000::/96 prefix) are supported but deprecated by RFC4291. Example: 127.0.0.1, 192.168.1.0/24, 192.168.3.1-255, 192.168.1-10.1-255, ::1,2001:db8::/32, mysqlserver1, zabbix.example.com, {HOST.HOST} Spaces and user macros are allowed in this field since Zabbix 2.2.0. Host macros {HOST.HOST}, {HOST.NAME}, {HOST.IP}, {HOST.DNS}, {HOST.CONN} are allowed in this field since Zabbix 4.0.2.

Note:
You may have to wait up to 60 seconds after saving the item until the server picks up the changes from a configuration cache update, before you can send in values.

Sending in data

In the simplest of cases, we may use zabbix_sender utility to send in some 'test value':

```
zabbix_sender -z <server IP address> -p 10051 -s "New host" -k trap -o "test value"
```

To send in the value we use these keys:

-z - to specify Zabbix server IP address

-p - to specify Zabbix server port number (10051 by default)

-s - to specify the host (make sure to use the 'technical' host name here, instead of the 'visible' name)

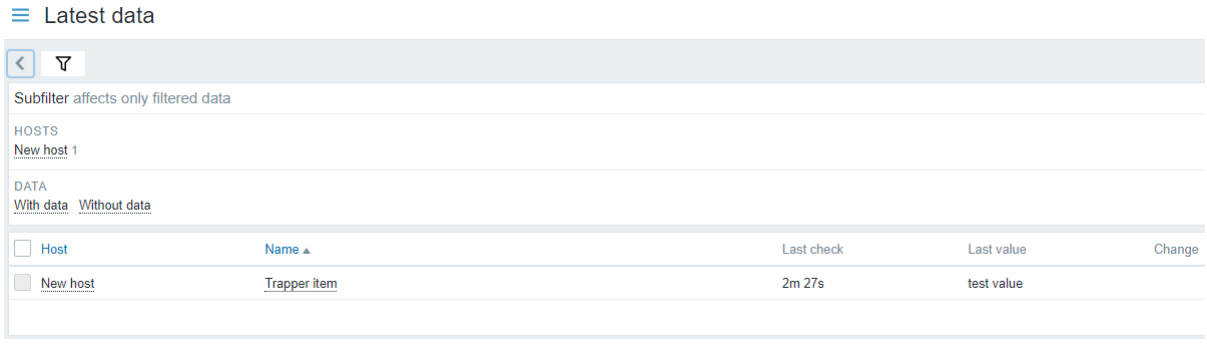
-k - to specify the key of the item we just defined

-o - to specify the actual value to send

Attention:
Zabbix trapper process does not expand macros used in the item key in attempt to check corresponding item key existence for targeted host.

Display

This is the result in *Monitoring* → *Latest data*:



Note that if a single numeric value is sent in, the data graph will show a horizontal line to the left and to the right of the time point of the value.

13 JMX monitoring

Overview

JMX monitoring can be used to monitor JMX counters of a Java application.

JMX monitoring has native support in Zabbix in the form of a Zabbix daemon called "Zabbix Java gateway", introduced since Zabbix 2.0.

To retrieve the value of a particular JMX counter on a host, Zabbix server queries the Zabbix **Java gateway**, which in turn uses the JMX management API to query the application of interest remotely.

For more details and setup see the Zabbix Java gateway section.

> **Warning:**
> Communication between Java gateway and the monitored JMX application should not be firewalled.

Enabling remote JMX monitoring for Java application

A Java application does not need any additional software installed, but it needs to be started with the command-line options specified below to have support for remote JMX monitoring.

As a bare minimum, if you just wish to get started by monitoring a simple Java application on a local host with no security enforced, start it with these options:

```
java \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=false \
-Dcom.sun.management.jmxremote.ssl=false \
-Dcom.sun.management.jmxremote.registry.ssl=false \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

This makes Java listen for incoming JMX connections on port 12345, from local host only, and tells it not to require authentication or SSL.

If you want to allow connections on another interface, set the -Djava.rmi.server.hostname parameter to the IP of that interface.

If you wish to be more stringent about security, there are many other Java options available to you. For instance, the next example starts the application with a more versatile set of options and opens it to a wider network, not just local host.

```
java \
-Djava.rmi.server.hostname=192.168.3.14 \
-Dcom.sun.management.jmxremote \
-Dcom.sun.management.jmxremote.port=12345 \
-Dcom.sun.management.jmxremote.authenticate=true \
-Dcom.sun.management.jmxremote.password.file=/etc/java-6-openjdk/management/jmxremote.password \
-Dcom.sun.management.jmxremote.access.file=/etc/java-6-openjdk/management/jmxremote.access \
-Dcom.sun.management.jmxremote.ssl=true \
-Dcom.sun.management.jmxremote.registry.ssl=true \
-Djavax.net.ssl.keyStore=$YOUR_KEY_STORE \
-Djavax.net.ssl.keyStorePassword=$YOUR_KEY_STORE_PASSWORD \
-Djavax.net.ssl.trustStore=$YOUR_TRUST_STORE \
-Djavax.net.ssl.trustStorePassword=$YOUR_TRUST_STORE_PASSWORD \
-Dcom.sun.management.jmxremote.ssl.need.client.auth=true \
-jar /usr/share/doc/openjdk-6-jre-headless/demo/jfc/Notepad/Notepad.jar
```

Most (if not all) of these settings can be specified in `/etc/java-6-openjdk/management/management.properties` (or wherever that file is on your system).

Note that if you wish to use SSL, you have to modify startup.sh script by adding `-Djavax.net.ssl.*` options to Java gateway, so that it knows where to find key and trust stores.

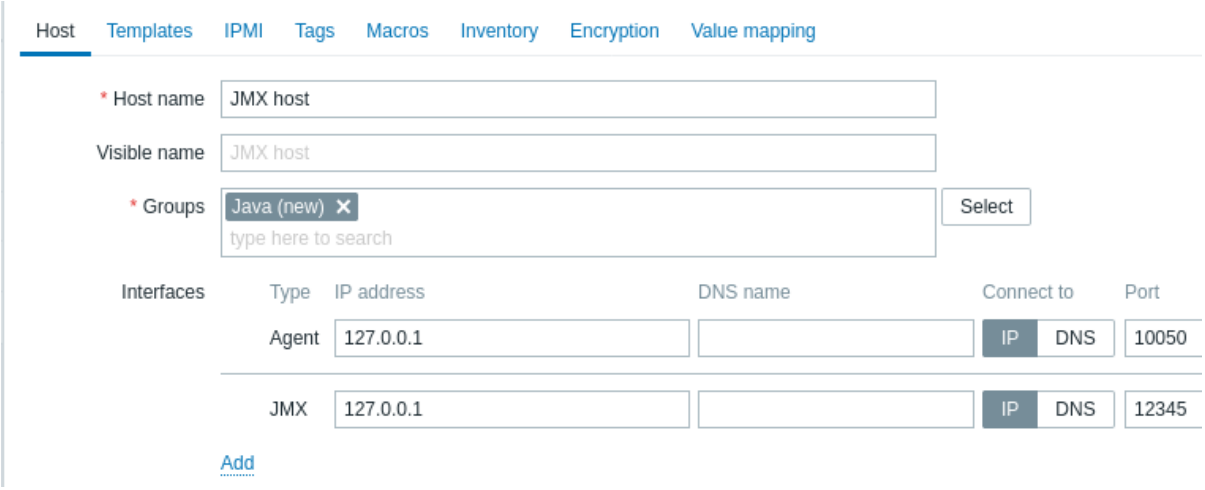
See Monitoring and Management Using JMX for a detailed description.

Configuring JMX interfaces and items in Zabbix frontend

With Java gateway running, server knowing where to find it and a Java application started with support for remote JMX monitoring, it is time to configure the interfaces and items in Zabbix GUI.

Configuring JMX interface

You begin by creating a JMX-type interface on the host of interest.

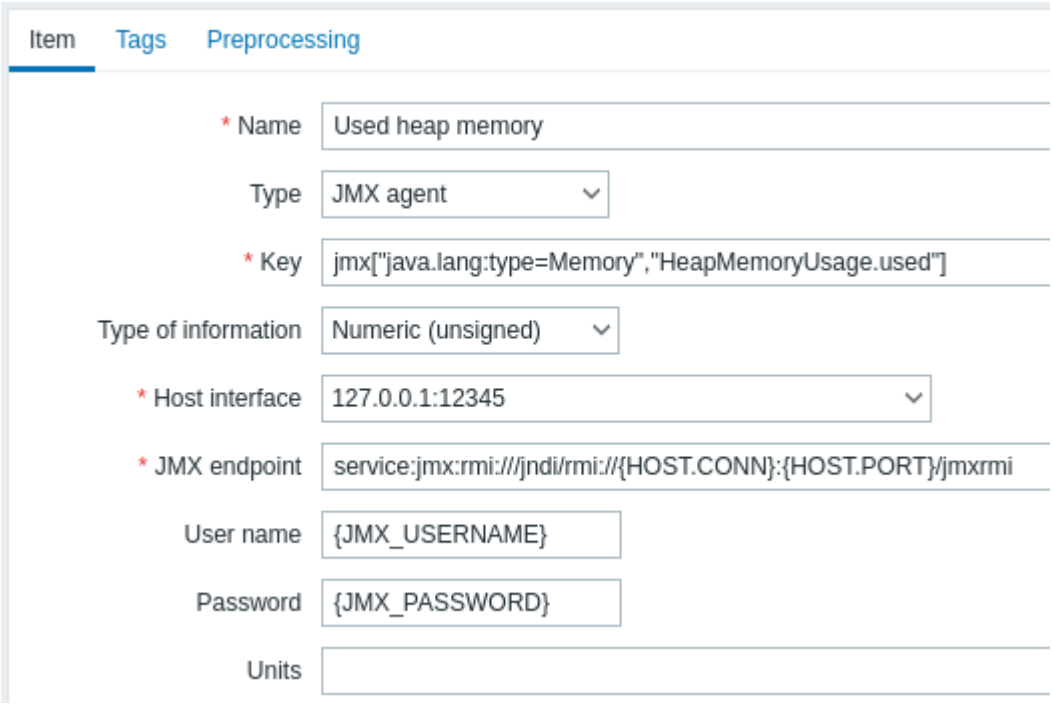


All mandatory input fields are marked with a red asterisk.

Adding JMX agent item

For each JMX counter you are interested in you add **JMX agent** item attached to that interface.

The key in the screenshot below says `jmx["java.lang:type=Memory","HeapMemoryUsage.used"]`.



All mandatory input fields are marked with a red asterisk.

The fields that require specific information for JMX items are:

Type	Set **JMX agent** here.
Key	The `jmx[]` item key contains three parameters: **object name** - the object name of an MBean **attribute name** - an MBean attribute name with optional composite data field names separated by dots **unique short description** - a unique description that allows multiple JMX items with the same object name and attribute name on the host (optional) See below for more detail on JMX item keys. Since Zabbix 3.4, you may discover MBeans and MBean attributes using a `jmx.discovery[]` low-level discovery item.
JMX endpoint	You may specify a custom JMX endpoint. Make sure that JMX endpoint connection parameters match the JMX interface. This can be achieved by using {HOST.*} macros as done in the default JMX endpoint. This field is supported since 3.4.0. {HOST.*} macros and user macros are supported.

User name	Specify the user name, if you have configured authentication on your Java application. User macros are supported.
Password	Specify the password, if you have configured authentication on your Java application. User macros are supported.

If you wish to monitor a Boolean counter that is either "true" or "false", then you specify type of information as "Numeric (unsigned)" and select "Boolean to decimal" preprocessing step in the Preprocessing tab. Server will store Boolean values as 1 or 0, respectively.

JMX item keys in more detail

Simple attributes

An MBean object name is nothing but a string which you define in your Java application. An attribute name, on the other hand, can be more complex. In case an attribute returns primitive data type (an integer, a string etc.) there is nothing to worry about, the key will look like this:

`jmx[com.example:Type=Hello,weight]`

In this example the object name is "com.example:Type=Hello", the attribute name is "weight", and the returned value type should probably be "Numeric (float)".

Attributes returning composite data

It becomes more complicated when your attribute returns composite data. For example: your attribute name is "apple" and it returns a hash representing its parameters, like "weight", "color" etc. Your key may look like this:

`jmx[com.example:Type=Hello,apple.weight]`

This is how an attribute name and a hash key are separated, by using a dot symbol. Same way, if an attribute returns nested composite data the parts are separated by a dot:

`jmx[com.example:Type=Hello,fruits.apple.weight]`

Attributes returning tabular data

Tabular data attributes consist of one or multiple composite attributes. If such an attribute is specified in the attribute name parameter then this item value will return the complete structure of the attribute in JSON format. The individual element values inside the tabular data attribute can be retrieved using preprocessing.

Tabular data attribute example:

`jmx[com.example:type=Hello,foodinfo]`

Item value:

```
[
  {
    "a": "apple",
    "b": "banana",
    "c": "cherry"
  },
  {
    "a": "potato",
    "b": "lettuce",
    "c": "onion"
  }
]
```

Problem with dots

So far so good. But what if an attribute name or a hash key contains dot symbol? Here is an example:

`jmx[com.example:Type=Hello,all.fruits.apple.weight]`

That's a problem. How to tell Zabbix that attribute name is "all.fruits", not just "all"? How to distinguish a dot that is part of the name from the dot that separates an attribute name and hash keys?

Before **2.0.4** Zabbix Java gateway was unable to handle such situations and users were left with UNSUPPORTED items. Since 2.0.4 this is possible, all you need to do is to escape the dots that are part of the name with a backslash:

`jmx[com.example:Type=Hello,all\.fruits.apple.weight]`

Same way, if your hash key contains a dot you escape it:

```
jmx[com.example:Type=Hello,all\.fruits.apple.total\.weight]
```

Other issues

A backslash character in an attribute name should be escaped:

```
jmx[com.example:type=Hello,c:\\documents]
```

For handling any other special characters in JMX item key, please see the item key format section.

This is actually all there is to it. Happy JMX monitoring!

Non-primitive data types

Since Zabbix 4.0.0 it is possible to work with custom MBeans returning non-primitive data types, which override the **toString()** method.

Using custom endpoint with JBoss EAP 6.4

Custom endpoints allow working with different transport protocols other than the default RMI.

To illustrate this possibility, let's try to configure JBoss EAP 6.4 monitoring as an example. First, let's make some assumptions:

- You have already installed Zabbix Java gateway. If not, then you can do it in accordance with the documentation.
- Zabbix server and Java gateway are installed with the prefix `/usr/local/`
- JBoss is already installed in `/opt/jboss-eap-6.4/` and is running in standalone mode
- We shall assume that all these components work on the same host
- Firewall and SELinux are disabled (or configured accordingly)

Let's make some simple settings in zabbix_server.conf:

```
JavaGateway=127.0.0.1
StartJavaPollers=5
```

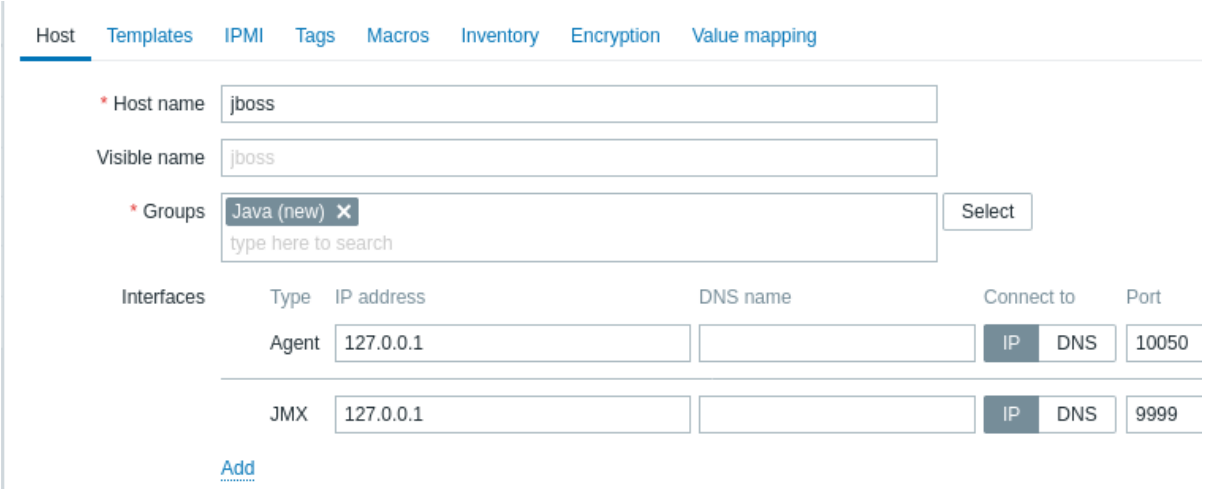
And in the `zabbix_java/settings.sh` configuration file (or `zabbix_java_gateway.conf`):

```
START_POLLERS=5
```

Check that JBoss listens to its standard management port:

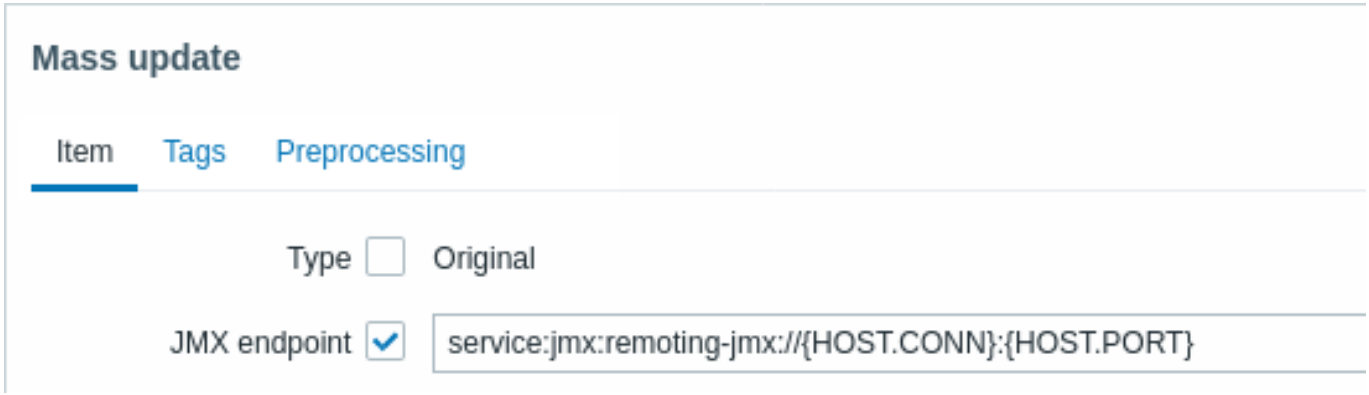
```
$ netstat -natp | grep 9999
tcp        0      0 127.0.0.1:9999          0.0.0.0:*               LISTEN      10148/java
```

Now let's create a host with JMX interface 127.0.0.1:9999 in Zabbix.



As we know that this version of JBoss uses the JBoss Remoting protocol instead of RMI, we may mass update the JMX endpoint parameter for items in our JMX template accordingly:

```
service:jmx:remoting-jmx://{HOST.CONN}:{HOST.PORT}
```



Let's update the configuration cache:

```
$ /usr/local/sbin/zabbix_server -R config_cache_reload
```

Note that you may encounter an error first.

```
3. mc [root@centos7-dev]:/home/vagrant/zabbix-3.2.6/src/zabbix_java (ssh)
com.zabbix.gateway.ZabbixException: java.net.MalformedURLException: Unsupported protocol: remoting-jmx
        at com.zabbix.gateway.JMXItemChecker.getValues(JMXItemChecker.java:97) ~[zabbix-java-gateway-3.4.2.jar:na]
        at com.zabbix.gateway.SocketProcessor.run(SocketProcessor.java:63) ~[zabbix-java-gateway-3.4.2.jar:na]
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) [na:1.8.0_144]
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) [na:1.8.0_144]
        at java.lang.Thread.run(Thread.java:748) [na:1.8.0_144]
Caused by: java.net.MalformedURLException: Unsupported protocol: remoting-jmx
        at javax.management.remote.JMXConnectorFactory.newJMXConnector(JMXConnectorFactory.java:359) ~[na:1.8.0_144]
        at javax.management.remote.JMXConnectorFactory.connect(JMXConnectorFactory.java:269) ~[na:1.8.0_144]
        at com.zabbix.gateway.ZabbixJMXConnectorFactory$1.run(ZabbixJMXConnectorFactory.java:76) ~[zabbix-java-gatewa
-3.4.2.jar:na]
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511) ~[na:1.8.0_144]
        at java.util.concurrent.FutureTask.run(FutureTask.java:266) ~[na:1.8.0_144]
        ... 3 common frames omitted
2017-11-07 13:52:12.644 [pool-1-thread-1] WARN  com.zabbix.gateway.SocketProcessor - error processing request
com.zabbix.gateway.ZabbixException: java.net.MalformedURLException: Unsupported protocol: remoting-jmx
        at com.zabbix.gateway.JMXItemChecker.getValues(JMXItemChecker.java:97) ~[zabbix-java-gateway-3.4.2.jar:na]
        at com.zabbix.gateway.SocketProcessor.run(SocketProcessor.java:63) ~[zabbix-java-gateway-3.4.2.jar:na]
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) [na:1.8.0_144]
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) [na:1.8.0_144]
        at java.lang.Thread.run(Thread.java:748) [na:1.8.0_144]
Caused by: java.net.MalformedURLException: Unsupported protocol: remoting-jmx
        at javax.management.remote.JMXConnectorFactory.newJMXConnector(JMXConnectorFactory.java:359) ~[na:1.8.0_144]
        at javax.management.remote.JMXConnectorFactory.connect(JMXConnectorFactory.java:269) ~[na:1.8.0_144]
        at com.zabbix.gateway.ZabbixJMXConnectorFactory$1.run(ZabbixJMXConnectorFactory.java:76) ~[zabbix-java-gatewa
-3.4.2.jar:na]
        at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:511) ~[na:1.8.0_144]
        at java.util.concurrent.FutureTask.run(FutureTask.java:266) ~[na:1.8.0_144]
        ... 3 common frames omitted
2017-11-07 13:52:14.889 [Thread-0] INFO  com.zabbix.gateway.JavaGateway - Zabbix Java Gateway 3.4.2 (revision 72885)
as stopped
2017-11-07 13:52:26.167 [main] INFO  com.zabbix.gateway.JavaGateway - Zabbix Java Gateway 3.4.2 (revision 72885) has
tarted
```

"Unsupported protocol: remoting-jmx" means that Java gateway does not know how to work with the specified protocol. That can be fixed by creating a `~/needed_modules.txt` file with the following content:

```
jboss-as-remoting
jboss-logging
jboss-logmanager
jboss-marshalling
jboss-remoting
jboss-sasl
jcl-over-slf4j
jul-to-slf4j-stub
log4j-jboss-logmanager
remoting-jmx
slf4j-api
xnio-api
xnio-nio
```

and then executing the command:

```
$ for i in $(cat ~/needed_modules.txt); do find /opt/jboss-eap-6.4 -iname "${i}*.jar" -exec cp '{}' /usr/l
```

Thus, Java gateway will have all the necessary modules for working with jmx-remoting. What's left is to restart the Java gateway, wait a bit and if you did everything right, see that JMX monitoring data begin to arrive in Zabbix (see also: Latest data).

14 ODBC monitoring

Overview

ODBC monitoring corresponds to the *Database monitor* item type in the Zabbix frontend.

ODBC is a C programming language middle-ware API for accessing database management systems (DBMS). The ODBC concept was developed by Microsoft and later ported to other platforms.

Zabbix may query any database, which is supported by ODBC. To do that, Zabbix does not directly connect to the databases, but uses the ODBC interface and drivers set up in ODBC. This function allows for more efficient monitoring of different databases for multiple purposes - for example, checking specific database queues, usage statistics and so on. Zabbix supports unixODBC, which is one of the most commonly used open source ODBC API implementations.

Attention:
See also the known issues for ODBC checks.

Installing unixODBC

The suggested way of installing unixODBC is to use the Linux operating system default package repositories. In the most popular Linux distributions unixODBC is included in the package repository by default. If it's not available, it can be obtained at the unixODBC homepage: http://www.unixodbc.org/download.html.

Installing unixODBC on Ubuntu/Debian systems using the *apt* package manager:

```
apt install unixodbc unixodbc-dev
```

Installing unixODBC on RedHat/Fedora-based systems using the *dnf* package manager:

```
dnf install unixODBC unixODBC-devel
```

Installing unixODBC on SUSE-based systems using the *zypper* package manager:

```
zypper in unixODBC-devel
```

Note:
The unixodbc-dev or unixODBC-devel package is needed to compile Zabbix with unixODBC support.

Installing unixODBC drivers

A unixODBC database driver should be installed for the database, which will be monitored. unixODBC has a list of supported databases and drivers: http://www.unixodbc.org/drivers.html. In some Linux distributions database drivers are included in package repositories.

Installing MySQL database driver on Ubuntu/Debian systems using the *apt* package manager:

```
apt install odbc-mariadb
```

Installing MySQL database driver on RedHat/Fedora-based systems using the *dnf* package manager:

```
dnf install mariadb-connector-odbc
```

Installing MySQL database driver on SUSE-based systems using the *zypper* package manager:

```
zypper in mariadb-connector-odbc
```

Configuring unixODBC

ODBC configuration is done by editing the **odbcinst.ini** and **odbc.ini** files. To verify the configuration file location, type:

```
odbcinst -j
```

odbcinst.ini is used to list the installed ODBC database drivers:

```
[mysql]
Description = ODBC for MySQL
Driver      = /usr/lib/libmyodbc5.so
```

Parameter details:

Attribute	Description
mysql	Database driver name.
Description	Database driver description.

Attribute	Description
Driver	Database driver library location.

odbc.ini is used to define data sources:

```
[test]
Description = MySQL test database
Driver      = mysql
Server      = 127.0.0.1
User        = root
Password    =
Port        = 3306
Database    = zabbix
```

Parameter details:

Attribute	Description
test	Data source name (DSN).
Description	Data source description.
Driver	Database driver name - as specified in odbcinst.ini
Server	Database server IP/DNS.
User	Database user for connection.
Password	Database user password.
Port	Database connection port.
Database	Database name.

To verify if ODBC connection is working successfully, a connection to database should be tested. That can be done with the **isql** utility (included in the unixODBC package):

```
isql test
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
```

Compiling Zabbix with ODBC support

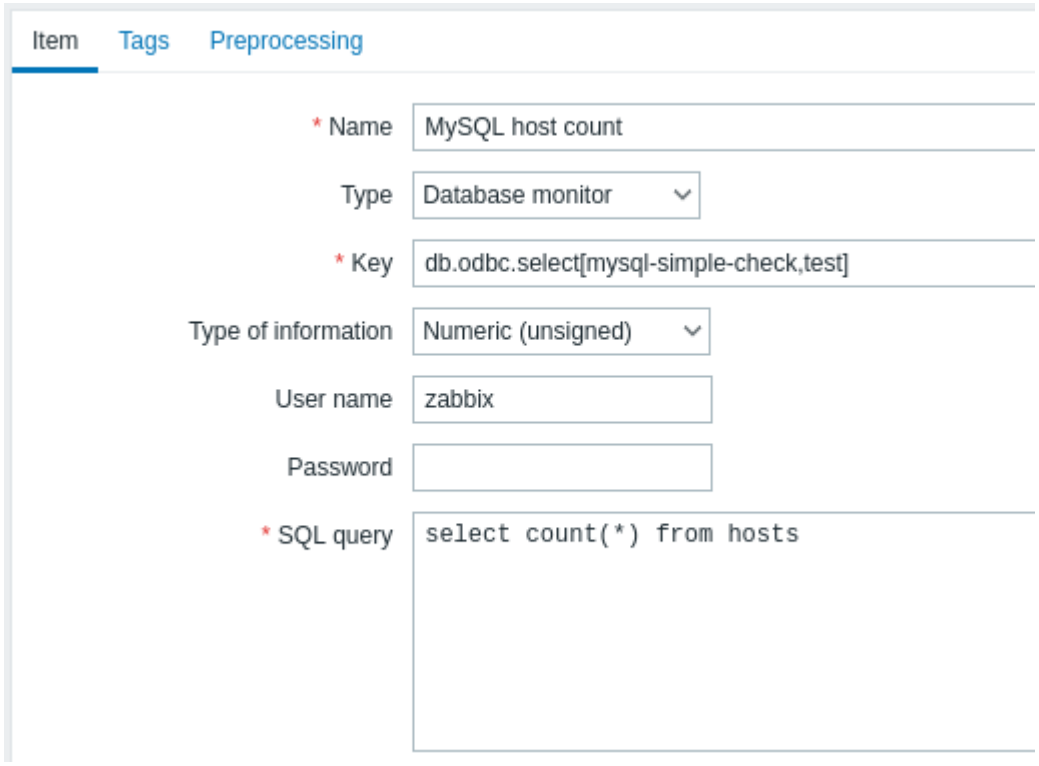
To enable ODBC support, Zabbix should be compiled with the following flag:

```
--with-unixodbc[=ARG] # Use ODBC driver against unixODBC package.
```

Note:
See more about Zabbix installation from the source code.

Item configuration in Zabbix frontend

Configure a database monitoring item.



All mandatory input fields are marked with a red asterisk.

Specifically for database monitoring items you must enter:

Type	Select *Database monitor* here.
Key	Enter one of the two supported item keys: **db.odbc.select**[<unique short description>,<dsn>,<connection string>] - this item is designed to return one value, i.e. the first column of the first row of the SQL query result. If a query returns more than one column, only the first column is read. If a query returns more than one line, only the first line is read. **db.odbc.get**[<unique short description>,<dsn>,<connection string>] - this item is capable of returning multiple rows/columns in JSON format. Thus it may be used as a master item that collects all data in one system call, while JSONPath preprocessing may be used in dependent items to extract individual values. For more information, see an example of the returned format, used in low-level discovery. This item is supported since Zabbix 4.4. The unique description will serve to identify the item in triggers, etc. Although `dsn` and `connection string` are optional parameters, at least one of them should be present. If both data source name (DSN) and connection string are defined, the DSN will be ignored. The data source name, if used, must be set as specified in odbc.ini. The connection string may contain driver-specific arguments. Example (connection for MySQL ODBC driver 5): => db.odbc.get[MySQL example,"Driver=/usr/local/lib/libmyodbc5a.so;Database=master;Server=127.0.0.1;Port=3306"]
User name	Enter the database user name This parameter is optional if user is specified in odbc.ini. If connection string is used, and *User name* field is not empty, it is appended to the connection string as `UID=<user>`

Password	Enter the database user password This parameter is optional if password is specified in odbc.ini. If connection string is used, and *Password* field is not empty, it is appended to the connection string as `PWD=<password>`. Since Zabbix 6.4.19, special characters are supported in this field. Before Zabbix 6.4.19, if the password contains a semicolon, it should be wrapped in curly brackets, for example, `{P?;)*word}`. After 6.4.19, wrapping the password in this case is still supported, but not required. The password will be appended to connection string after the username as `UID=<username>;PWD={P?;)*word}`. To test the resulting string, you can run the following command: `isql -v -k 'Driver=libmaodbc.so;Database=zabbix;UID=zabbix;PWD={P?;)*word}'`
SQL query	Enter the SQL query. Note that with the `db.odbc.select[]` item the query must return one value only.
Type of information	It is important to know what type of information will be returned by the query, so that it is selected correctly here. With an incorrect *type of information* the item will turn unsupported.

Important notes

- Database monitoring items will become unsupported if no *odbc poller* processes are started in the server or proxy configuration. To activate ODBC pollers, set *StartODBCPollers* parameter in Zabbix server configuration file or, for checks performed by proxy, in Zabbix proxy configuration file.
- Zabbix does not limit the query execution time. It is up to the user to choose queries that can be executed in a reasonable amount of time.
- The Timeout parameter value from Zabbix server is used as the ODBC login timeout (note that depending on ODBC drivers the login timeout setting might be ignored).
- The SQL command must return a result set like any query with `select ...`. The query syntax will depend on the RDBMS which will process them. The syntax of request to a storage procedure must be started with `call` keyword.

Error messages

ODBC error messages are structured into fields to provide detailed information. For example, an error message might look like this:

```
Cannot execute ODBC query: [SQL_ERROR]:[42601][7][ERROR: syntax error at or near ";"; Error while executin
```

- "`Cannot execute ODBC query`" - Zabbix message
- "`[SQL_ERROR]`" - ODBC return code
- "`[42601]`" - SQLState
- "`[7]`" - Native error code
- "`[ERROR: syntax error at or near ";"; Error while executing the query]`" - Native error message

Note that the error message length is limited to 2048 bytes, so the message can be truncated. If there is more than one ODBC diagnostic record Zabbix tries to concatenate them (separated with `|`) as far as the length limit allows.

1 Recommended UnixODBC settings for MySQL

Installation

- **Red Hat Enterprise Linux**:

```
dnf install mariadb-connector-odbc
```

- **Debian/Ubuntu**:

Please refer to MySQL documentation (for `mysql-connector-odbc`), or MariaDB documentation (for `mariadb-connector-odbc`) to download necessary database driver for the corresponding platform.

For some additional information please refer to: installing unixODBC.

Configuration

ODBC configuration is done by editing **odbcinst.ini** and **odbc.ini** files. These configuration files can be found in */etc* folder. The file **odbcinst.ini** may be missing and in this case it is necessary to create it manually.

odbcinst.ini

```
[mysql]
Description = General ODBC for MySQL
Driver      = /usr/lib64/libmyodbc5.so
Setup       = /usr/lib64/libodbcmyS.so
FileUsage   = 1
```

Please consider the following examples of **odbc.ini** configuration parameters.

- An example with a connection through an IP:

```
[TEST_MYSQL]
Description = MySQL database 1
Driver  = mysql
Port = 3306
Server = 127.0.0.1
```

- An example with a connection through an IP and with the use of credentials. A Zabbix database is used by default:

```
[TEST_MYSQL_FILLED_CRED]
Description = MySQL database 2
Driver  = mysql
User = root
Port = 3306
Password = zabbix
Database = zabbix
Server = 127.0.0.1
```

- An example with a connection through a socket and with the use of credentials. A Zabbix database is used by default:

```
[TEST_MYSQL_FILLED_CRED_SOCK]
Description = MySQL database 3
Driver  = mysql
User = root
Password = zabbix
Socket = /var/run/mysqld/mysqld.sock
Database = zabbix
```

All other possible configuration parameter options can be found in MySQL official documentation web page.

2 Recommended UnixODBC settings for PostgreSQL

Installation

- **Red Hat Enterprise Linux**:

```
dnf install postgresql-odbc
```

- **Debian/Ubuntu**:

Please refer to PostgreSQL documentation to download necessary database driver for the corresponding platform.

For some additional information please refer to: installing unixODBC.

Configuration

ODBC configuration is done by editing the **odbcinst.ini** and **odbc.ini** files. These configuration files can be found in */etc* folder. The file **odbcinst.ini** may be missing and in this case it is necessary to create it manually.

Please consider the following examples:

odbcinst.ini

```
[postgresql]
Description = General ODBC for PostgreSQL
Driver      = /usr/lib64/libodbcpsql.so
Setup       = /usr/lib64/libodbcpsqlS.so
FileUsage   = 1
# Since 1.6 if the driver manager was built with thread support you may add another entry to each driver e
# This entry alters the default thread serialization level.
Threading   = 2
```

odbc.ini

```
[TEST_PSQL]
Description = PostgreSQL database 1
Driver  = postgresql
#CommLog = /tmp/sql.log
Username = zbx_test
Password = zabbix
# Name of Server. IP or DNS
Servername = 127.0.0.1
# Database name
Database = zabbix
# Postmaster listening port
Port = 5432
# Database is read only
# Whether the datasource will allow updates.
ReadOnly = No
# PostgreSQL backend protocol
# Note that when using SSL connections this setting is ignored.
# 7.4+: Use the 7.4(V3) protocol. This is only compatible with 7.4 and higher backends.
Protocol = 7.4+
# Includes the OID in SQLColumns
ShowOidColumn = No
# Fakes a unique index on OID
FakeOidIndex  = No
# Row Versioning
# Allows applications to detect whether data has been modified by other users
# while you are attempting to update a row.
# It also speeds the update process since every single column does not need to be specified in the where c
RowVersioning = No
# Show SystemTables
# The driver will treat system tables as regular tables in SQLTables. This is good for Access so you can s
ShowSystemTables = No
# If true, the driver automatically uses declare cursor/fetch to handle SELECT statements and keeps 100 ro
Fetch = Yes
# Bools as Char
# Bools are mapped to SQL_CHAR, otherwise to SQL_BIT.
BoolsAsChar = Yes
# SSL mode
SSLmode = Require
# Send to backend on connection
ConnSettings =
```

3 Recommended UnixODBC settings for Oracle

Installation

Please refer to Oracle documentation for all the necessary instructions.

For some additional information please refer to: Installing unixODBC.

4 Recommended UnixODBC settings for MSSQL

Installation

- **Red Hat Enterprise Linux** (EPEL packages):

```
dnf install epel-release
dnf install freetds
```

- **Debian/Ubuntu**:

Please refer to FreeTDS user guide to download necessary database driver for the corresponding platform.

For some additional information please refer to: installing unixODBC.

Configuration

ODBC configuration is done by editing the **odbcinst.ini** and **odbc.ini** files. These configuration files can be found in */etc* folder. The file **odbcinst.ini** may be missing and in this case it is necessary to create it manually.

Please consider the following examples:

odbcinst.ini

```
$ vi /etc/odbcinst.ini
[FreeTDS]
Driver = /usr/lib64/libtdsodbc.so.0
```

odbc.ini

```
$ vi /etc/odbc.ini
[sql1]
Driver = FreeTDS
Server = <SQL server 1 IP>
PORT = 1433
TDS_Version = 8.0
```

15 Dependent items

Overview

There are situations when one item gathers multiple metrics at a time or it even makes more sense to collect related metrics simultaneously, for example:

- CPU utilization of individual cores
- Incoming/outgoing/total network traffic

To allow for bulk metric collection and simultaneous use in several related items, Zabbix supports dependent items. Dependent items depend on the master item that collects their data simultaneously, in one query. A new value for the master item automatically populates the values of the dependent items. Dependent items cannot have a different update interval than the master item.

Zabbix preprocessing options can be used to extract the part that is needed for the dependent item from the master item data.

Preprocessing is managed by a `preprocessing manager` process, which has been added in Zabbix 3.4, along with workers that perform the preprocessing steps. All values (with or without preprocessing) from different data gatherers pass through the preprocessing manager before being added to the history cache. Socket-based IPC communication is used between data gatherers (pollers, trappers, etc) and the preprocessing process.

Zabbix server or Zabbix proxy (if host is monitored by proxy) are performing preprocessing steps and processing dependent items.

Item of any type, even dependent item, can be set as master item. Additional levels of dependent items can be used to extract smaller parts from the value of an existing dependent item.

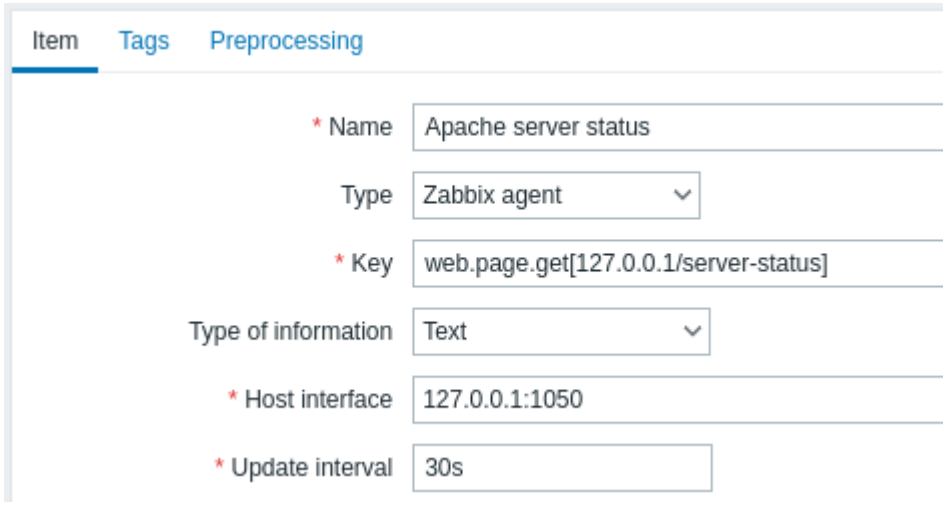
Limitations

- Only same host (template) dependencies are allowed
- An item prototype can depend on another item prototype or regular item from the same host
- Maximum count of dependent items for one master item is limited to 29999 (regardless of the number of dependency levels)
- Maximum 3 dependency levels allowed
- Dependent item on a host with master item from template will not be exported to XML

Item configuration

A dependent item depends on its master item for data. That is why the **master item** must be configured (or exist) first:

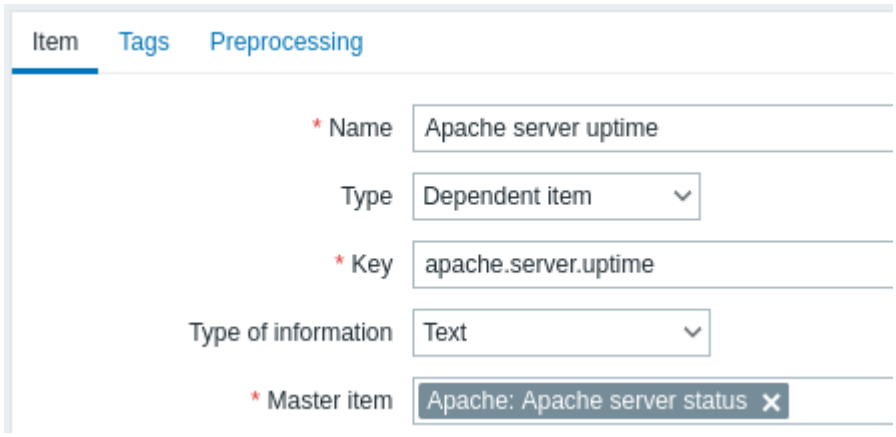
- Go to: *Data collection → Hosts*
- Click on *Items* in the row of the host
- Click on *Create item*
- Enter parameters of the item in the form



All mandatory input fields are marked with a red asterisk.

Click on *Add* to save the master item.

Then you can configure a **dependent item**.

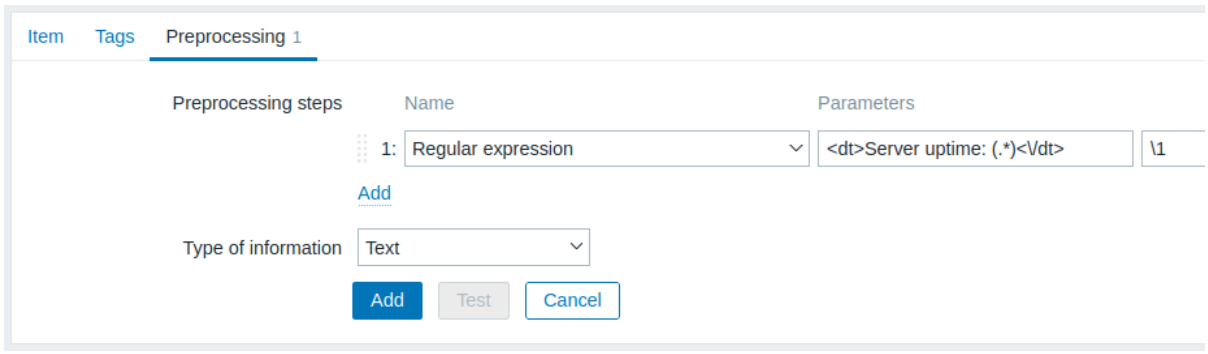


All mandatory input fields are marked with a red asterisk.

The fields that require specific information for dependent items are:

Type	Select **Dependent item** here.
Key	Enter a key that will be used to recognize the item.
Master item	Select the master item. Master item value will be used to populate dependent item value.
Type of information	Select the type of information that will correspond the format of data that will be stored.

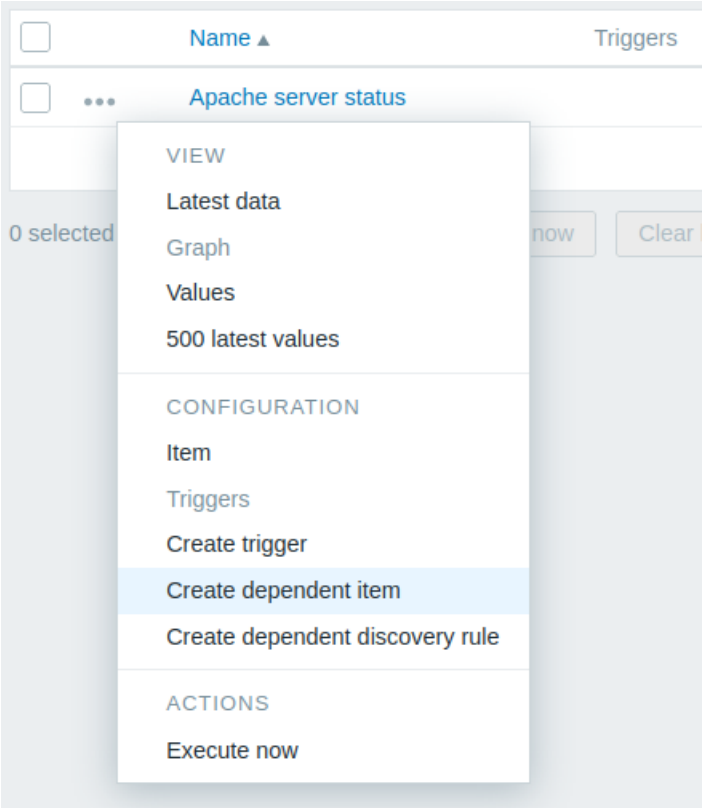
You may use item value preprocessing to extract the required part of the master item value.



Without preprocessing, the dependent item value will be exactly the same as the master item value.

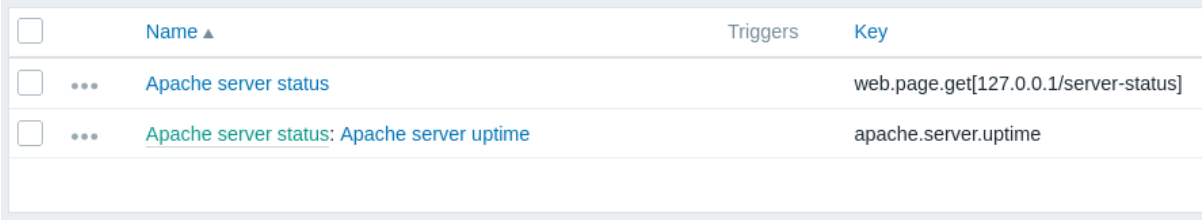
Click on *Add* to save the dependent item.

A shortcut to creating a dependent item quicker can be accessed by clicking on the ••• button in the item list and selecting *Create dependent item*.



Display

In the item list dependent items are displayed with their master item name as prefix.



If a master item is deleted, so are all its dependent items.

16 HTTP agent

Overview

This item type allows data polling using the HTTP/HTTPS protocol. Trapping is also possible using Zabbix sender or Zabbix sender protocol.

HTTP item check is executed by Zabbix server. However, when hosts are monitored by a Zabbix proxy, HTTP item checks are executed by the proxy.

HTTP item checks do not require any agent running on a host being monitored.

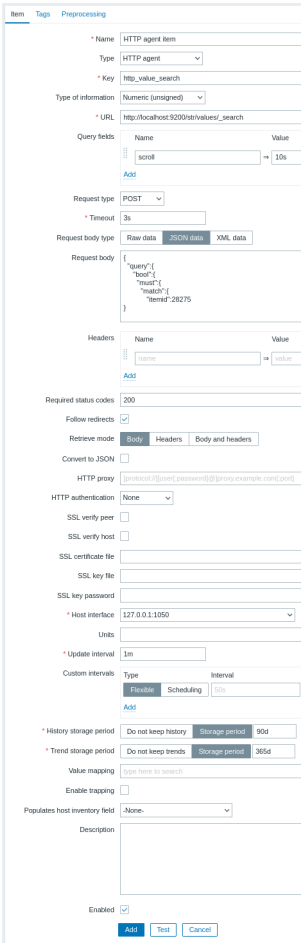
HTTP agent supports both HTTP and HTTPS. Zabbix will optionally follow redirects (see the *Follow redirects* option below). Maximum number of redirects is hard-coded to 10 (using cURL option CURLOPT_MAXREDIRS).

Attention:
Zabbix server/proxy must be initially configured with cURL (libcurl) support.

Configuration

To configure an HTTP item:

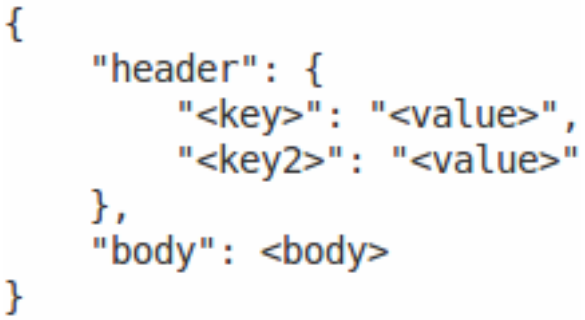
- Go to: *Data collection → Hosts*
- Click on *Items* in the row of the host
- Click on *Create item*
- Enter parameters of the item in the form



All mandatory input fields are marked with a red asterisk.

The fields that require specific information for HTTP items are:

Parameter	Description
Type	Select **HTTP agent** here.
Key	Enter a unique item key.
URL	URL to connect to and retrieve data. For example: https://www.example.com http://www.example.com/download Domain names can be specified in Unicode characters. They are automatically punycode-converted to ASCII when executing the HTTP check. The *Parse* button can be used to separate optional query fields (like ?name=Admin&password=mypassword) from the URL, moving the attributes and values into *Query fields* for automatic URL-encoding. Limited to 2048 characters. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros. This sets the CURLOPT_URL cURL option.
Query fields	Variables for the URL (see above). Specified as attribute and value pairs. Values are URL-encoded automatically. Values from macros are resolved and then URL-encoded automatically. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros. This sets the CURLOPT_URL cURL option.
Request type	Select request method type: *GET*, *POST*, *PUT* or *HEAD*

Parameter	Description
Timeout	Zabbix will not spend more than the set amount of time on processing the URL (1-60 seconds). Actually this parameter defines the maximum time for making a connection to the URL and maximum time for performing an HTTP request. Therefore, Zabbix will not spend more than 2 x Timeout seconds on one check. Time suffixes are supported, e.g. 30s, 1m. Supported macros: user macros, low-level discovery macros. This sets the CURLOPT_TIMEOUT cURL option.
Request body type	Select the request body type: **Raw data** - custom HTTP request body, macros are substituted but no encoding is performed **JSON data** - HTTP request body in JSON format. Macros can be used as string, number, true and false; macros used as strings must be enclosed in double quotes. Values from macros are resolved and then escaped automatically. If ”Content-Type” is not specified in headers then it will default to ”Content-Type: application/json” **XML data** - HTTP request body in XML format. Macros can be used as a text node, attribute or CDATA section. Values from macros are resolved and then escaped automatically in a text node and attribute. If ”Content-Type” is not specified in headers then it will default to ”Content-Type: application/xml” *Note* that selecting *XML data* requires libxml2.
Request body	Enter the request body. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros.
Headers	Custom HTTP headers that will be sent when performing a request. Specified as attribute and value pairs. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros. This sets the CURLOPT_HTTPHEADER cURL option.
Required status codes	List of expected HTTP status codes. If Zabbix gets a code which is not in the list, the item will become unsupported. If empty, no check is performed. For example: 200,201,210-299 Supported macros in the list: user macros, low-level discovery macros. This uses the CURLINFO_RESPONSE_CODE cURL option.
Follow redirects	Mark the checkbox to follow HTTP redirects. This sets the CURLOPT_FOLLOWLOCATION cURL option.
Retrieve mode	Select the part of response that must be retrieved: **Body** - body only **Headers** - headers only **Body and headers** - body and headers
Convert to JSON	Headers are saved as attribute and value pairs under the ”header” key. If ’Content-Type: application/json’ is encountered then body is saved as an object, otherwise it is stored as string, for example: 

Parameter	Description
HTTP proxy	You can specify an HTTP proxy to use, using the format `[protocol://][username[:password]@]proxy.example.com[:port]`. The optional `protocol://` prefix may be used to specify alternative proxy protocols (e.g. https, socks4, socks5; see documentation; the protocol prefix support was added in cURL 7.21.7). With no protocol specified, the proxy will be treated as an HTTP proxy. If you specify the wrong protocol, the connection will fail and the item will become unsupported. By default, 1080 port will be used. If specified, the proxy will overwrite proxy related environment variables like http_proxy, HTTPS_PROXY. If not specified, the proxy will not overwrite proxy-related environment variables. The entered value is passed on "as is", no sanity checking takes place. *Note* that only simple authentication is supported with HTTP proxy. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros. This sets the CURLOPT_PROXY cURL option.
HTTP authentication	Authentication type: **None** - no authentication used. **Basic** - basic authentication is used. **NTLM** - NTLM (Windows NT LAN Manager) authentication is used. **Kerberos** - Kerberos authentication is used. See also: Configuring Kerberos with Zabbix. **Digest** - Digest authentication is used. Selecting an authentication method will provide two additional fields for entering a user name and password, where user macros and low-level discovery macros are supported. This sets the CURLOPT_HTTPAUTH cURL option.
SSL verify peer	Mark the checkbox to verify the SSL certificate of the web server. The server certificate will be automatically taken from system-wide certificate authority (CA) location. You can override the location of CA files using Zabbix server or proxy configuration parameter SSLCALocation. This sets the CURLOPT_SSL_VERIFYPEER cURL option.
SSL verify host	Mark the checkbox to verify that the Common Name field or the Subject Alternate Name field of the web server certificate matches. This sets the CURLOPT_SSL_VERIFYHOST cURL option.
SSL certificate file	Name of the SSL certificate file used for client authentication. The certificate file must be in PEM[1] format. If the certificate file contains also the private key, leave the SSL key file field empty. If the key is encrypted, specify the password in SSL key password field. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLCertLocation. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros. This sets the CURLOPT_SSLCERT cURL option.
SSL key file	Name of the SSL private key file used for client authentication. The private key file must be in PEM[1] format. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLKeyLocation. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, {ITEM.KEY.ORIG}, user macros, low-level discovery macros. This sets the CURLOPT_SSLKEY cURL option.
SSL key password	SSL private key file password. Supported macros: user macros, low-level discovery macros. This sets the CURLOPT_KEYPASSWD cURL option.
Enable trapping	With this checkbox marked, the item will also function as trapper item and will accept data sent to this item by Zabbix sender or using Zabbix sender protocol.
Allowed hosts	Visible only if *Enable trapping* checkbox is marked. List of comma delimited IP addresses, optionally in CIDR notation, or DNS names. If specified, incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. Note that "IPv4-compatible IPv6 addresses" (0000::/96 prefix) are supported but deprecated by RFC4291. Example: 127.0.0.1, 192.168.1.0/24, 192.168.3.1-255, 192.168.1-10.1-255, ::1,2001:db8::/32, mysqlserver1, zabbix.example.com, {HOST.HOST} Spaces and user macros are allowed in this field. Host macros: {HOST.HOST}, {HOST.NAME}, {HOST.IP}, {HOST.DNS}, {HOST.CONN} are allowed in this field.

Note:
If the *HTTP proxy* field is left empty, another way for using an HTTP proxy is to set proxy-related environment variables. For HTTP - set the `http_proxy` environment variable for the Zabbix server user. For example: `http_proxy=http://proxy_ip:proxy_port`.
For HTTPS - set the `HTTPS_PROXY` environment variable. For example: `HTTPS_PROXY=http://proxy_ip:proxy_port`. More details are available by running a shell command: *# man curl*.

Attention:
[1] Zabbix supports certificate and private key files in PEM format only. In case you have your certificate and private key data in PKCS #12 format file (usually with extension *.p12 or *.pfx) you may generate the PEM file from it using the following commands:

```
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes  -out ssl-cert.key
```

Examples

Example 1

Send simple GET requests to retrieve data from services such as Elasticsearch:

- Create a GET item with URL: `localhost:9200/?pretty`
- Notice the response:

```
{
  "name" : "YQ2VAY-",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "kH4CYqh5QfqgeTsjh2F9zg",
  "version" : {
    "number" : "6.1.3",
    "build_hash" : "af51318",
    "build_date" : "2018-01-26T18:22:55.523Z",
    "build_snapshot" : false,
    "lucene_version" : "7.1.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You know, for search"
}
```

- Now extract the version number using a JSONPath preprocessing step: `$.version.number`

Example 2

Send simple POST requests to retrieve data from services such as Elasticsearch:

- Create a POST item with URL: `http://localhost:9200/str/values/_search?scroll=10s`
- Configure the following POST body to obtain the processor load (1 min average per core)

```
{
    "query": {
        "bool": {
            "must": [{
                "match": {
                    "itemid": 28275
                }
            }],
            "filter": [{
                "range": {
                    "clock": {
                        "gt": 1517565836,
                        "lte": 1517566137
                    }
                }
            }]
        }
    }
```

```
}
```

- Received:

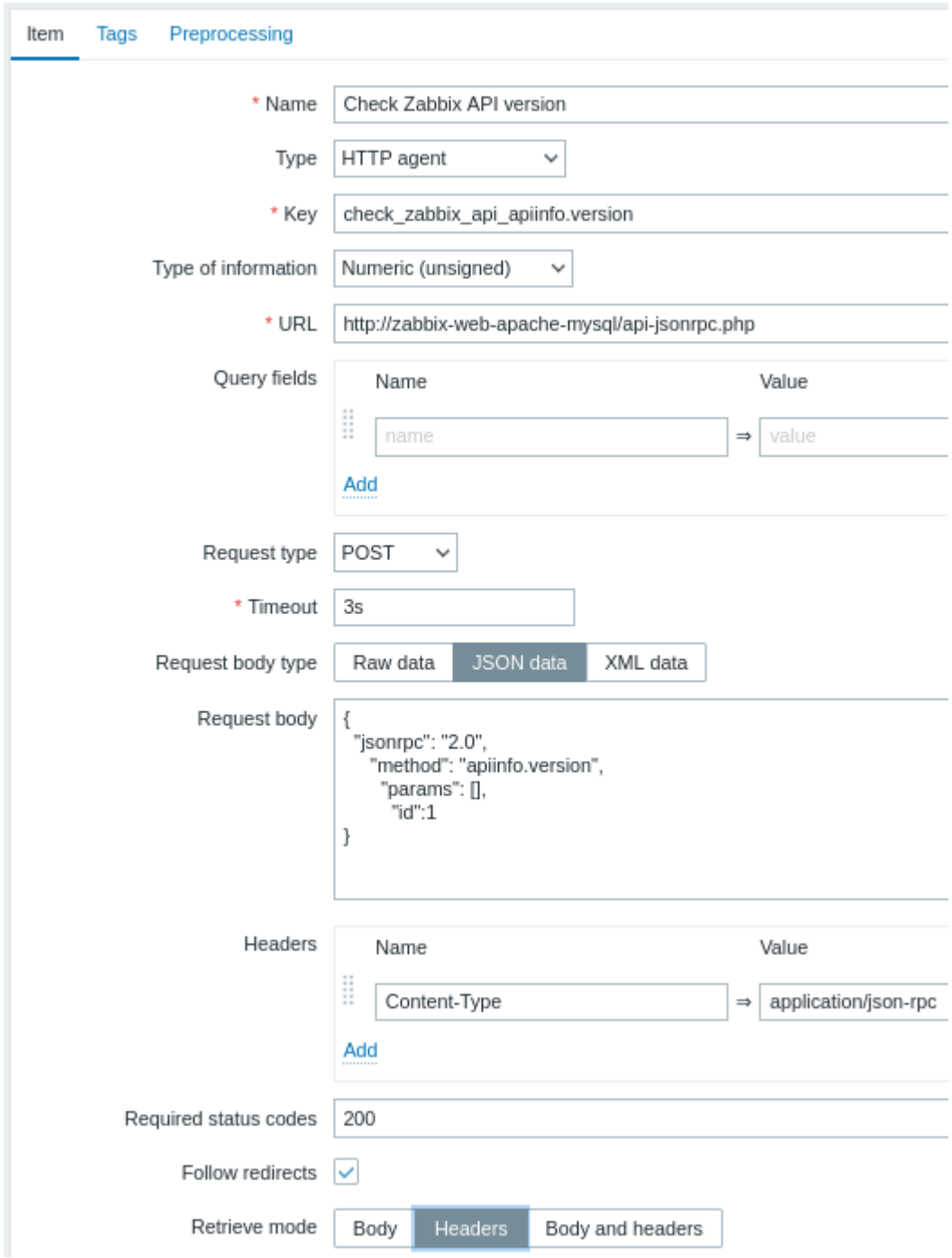

```
{
    "_scroll_id": "DnF1ZXJ5VGhlbkZldGNoBQAAAAAAAAAkFllRMlZBWS1UU1pxTmdEeGVwQjRBTFEAAAAAAAAAAJRZZUTJWQVktVFN
    "took": 18,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 1,
        "max_score": 1.0,
        "hits": [{
            "_index": "dbl",
            "_type": "values",
            "_id": "dqX9VWEBV6sEKSMyk6sw",
            "_score": 1.0,
            "_source": {
                "itemid": 28275,
                "value": "0.138750",
                "clock": 1517566136,
                "ns": 25388713,
                "ttl": 604800
            }
        }]
    }
}
```


- Now use a JSONPath preprocessing step to get the item value: `$.hits.hits[0]._source.value`

Example 3

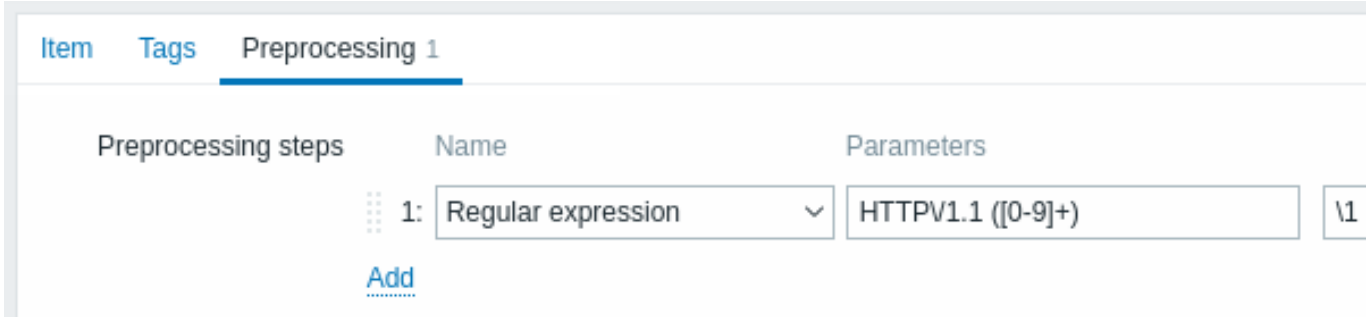
Checking if Zabbix API is alive, using apiinfo.version.

- Item configuration:

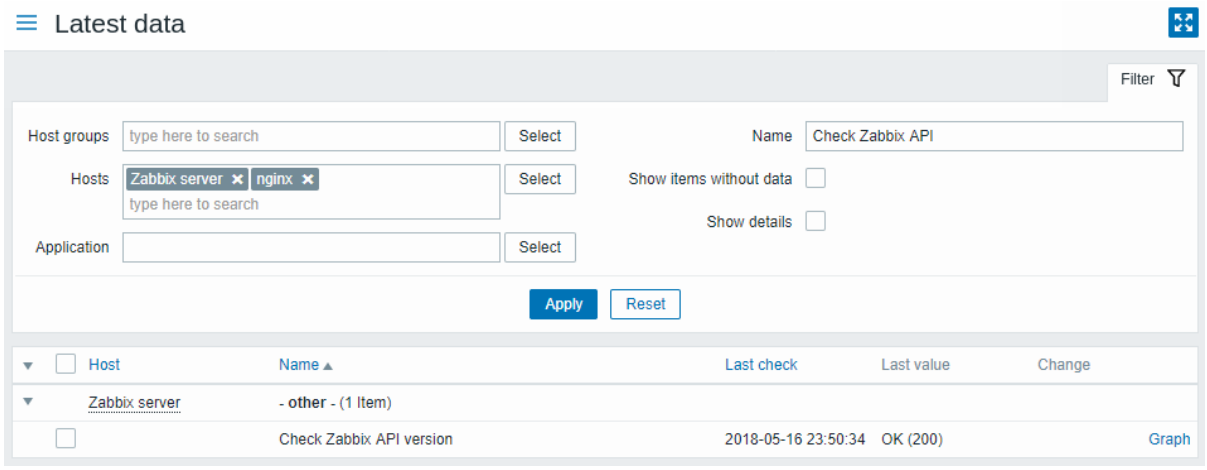


Note the use of the POST method with JSON data, setting request headers and asking to return headers only:

- Item value preprocessing with regular expression to get HTTP code:



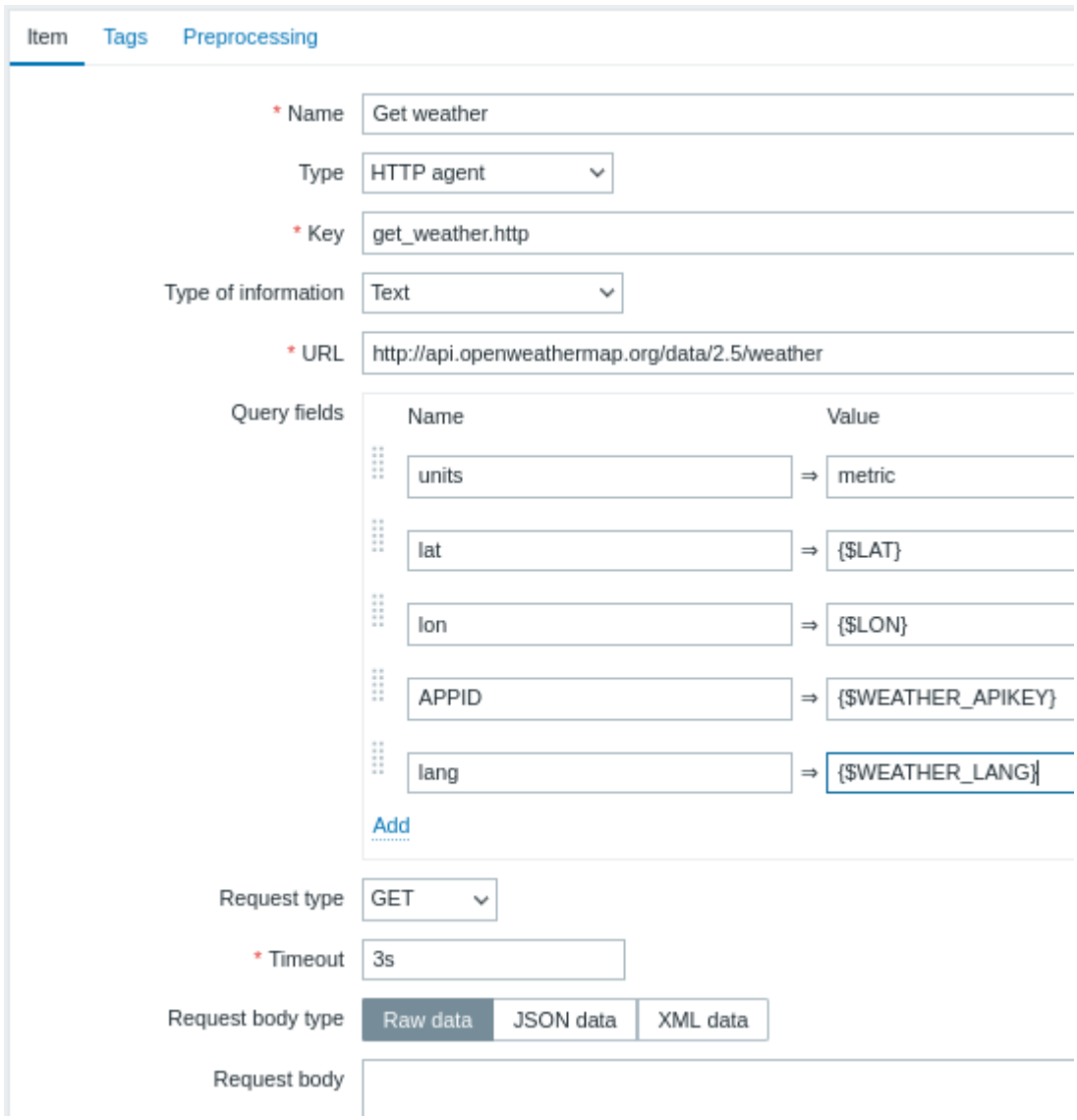
- Checking the result in *Latest data*:



Example 4

Retrieving weather information by connecting to the Openweathermap public service.

- Configure a master item for bulk data collection in a single JSON:



Note the usage of macros in query fields. Refer to the Openweathermap API for how to fill them.

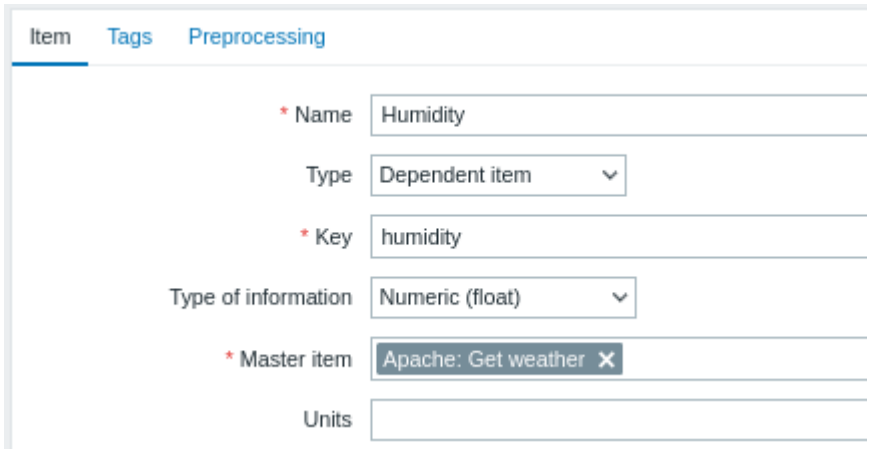
Sample JSON returned in response to HTTP agent:

```
{
    "body": {
        "coord": {
            "lon": 40.01,
```

```
            "lat": 56.11
        },
        "weather": [{
            "id": 801,
            "main": "Clouds",
            "description": "few clouds",
            "icon": "02n"
        }],
        "base": "stations",
        "main": {
            "temp": 15.14,
            "pressure": 1012.6,
            "humidity": 66,
            "temp_min": 15.14,
            "temp_max": 15.14,
            "sea_level": 1030.91,
            "grnd_level": 1012.6
        },
        "wind": {
            "speed": 1.86,
            "deg": 246.001
        },
        "clouds": {
            "all": 20
        },
        "dt": 1526509427,
        "sys": {
            "message": 0.0035,
            "country": "RU",
            "sunrise": 1526432608,
            "sunset": 1526491828
        },
        "id": 487837,
        "name": "Stavrovo",
        "cod": 200
    }
}
```

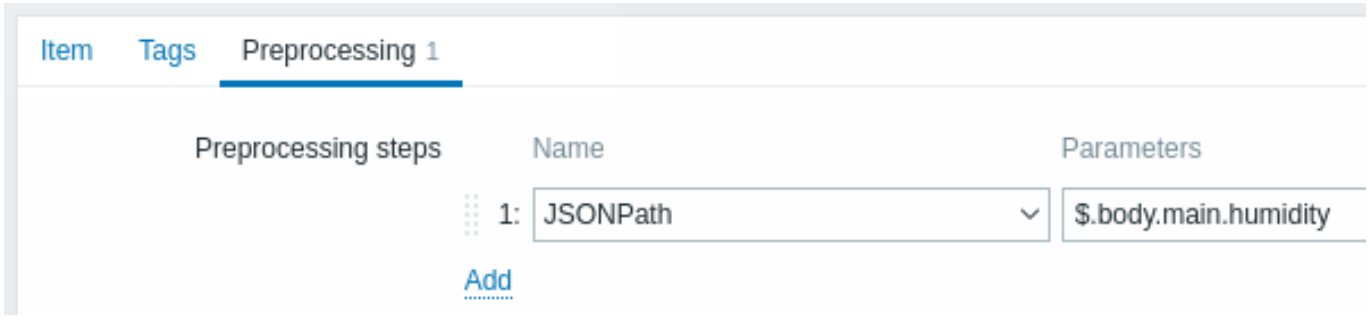

The next task is to configure dependent items that extract data from the JSON.

- Configure a sample dependent item for humidity:

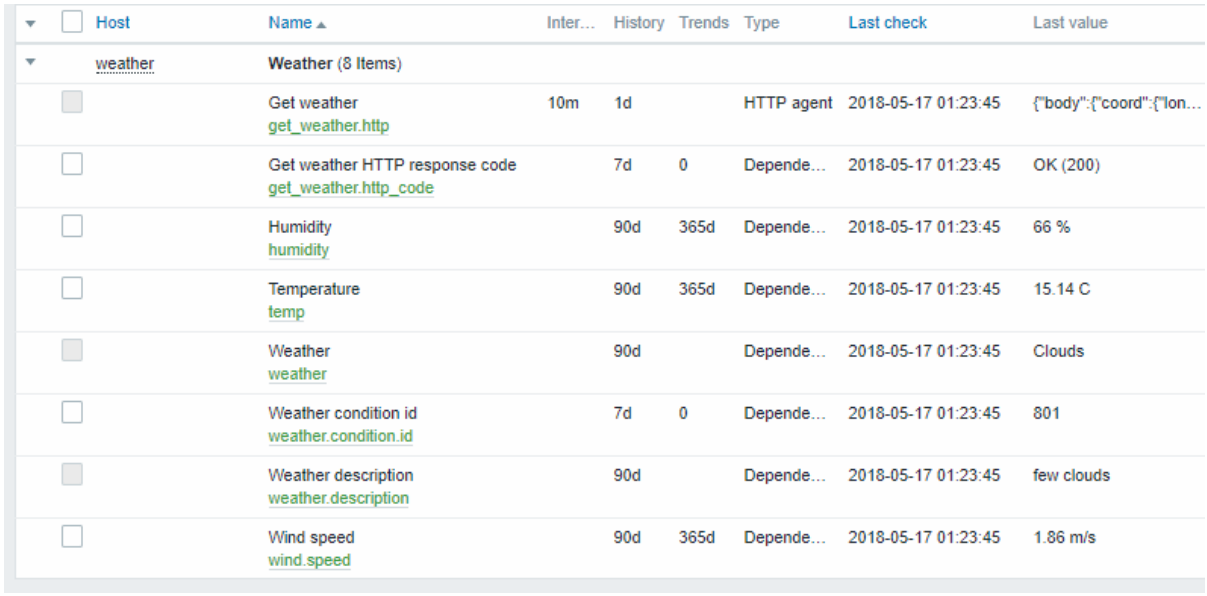


Other weather metrics such as 'Temperature' are added in the same manner.

- Sample dependent item value preprocessing with JSONPath:



- Check the result of weather data in *Latest data*:

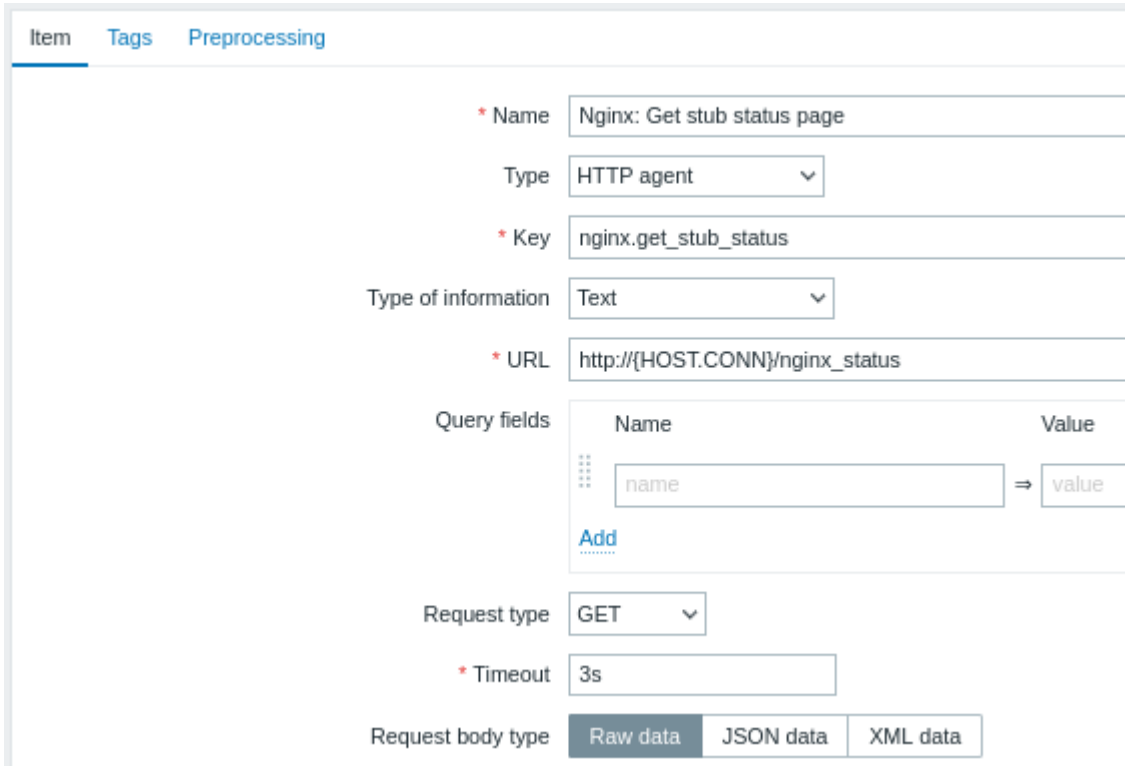


Host	Name ▲	Inter...	History	Trends	Type	Last check	Last value
weather	**Weather** (8 Items)						
	Get weather get_weather.http	10m	1d		HTTP agent	2018-05-17 01:23:45	{"body":{"coord":{"lon...
	Get weather HTTP response code get_weather.http_code		7d	0	Depende...	2018-05-17 01:23:45	OK (200)
	Humidity humidity		90d	365d	Depende...	2018-05-17 01:23:45	66 %
	Temperature temp		90d	365d	Depende...	2018-05-17 01:23:45	15.14 C
	Weather weather		90d		Depende...	2018-05-17 01:23:45	Clouds
	Weather condition id weather.condition.id		7d	0	Depende...	2018-05-17 01:23:45	801
	Weather description weather.description		90d		Depende...	2018-05-17 01:23:45	few clouds
	Wind speed wind.speed		90d	365d	Depende...	2018-05-17 01:23:45	1.86 m/s

Example 5

Connecting to Nginx status page and getting its metrics in bulk.

- Configure Nginx following the official guide.
- Configure a master item for bulk data collection:

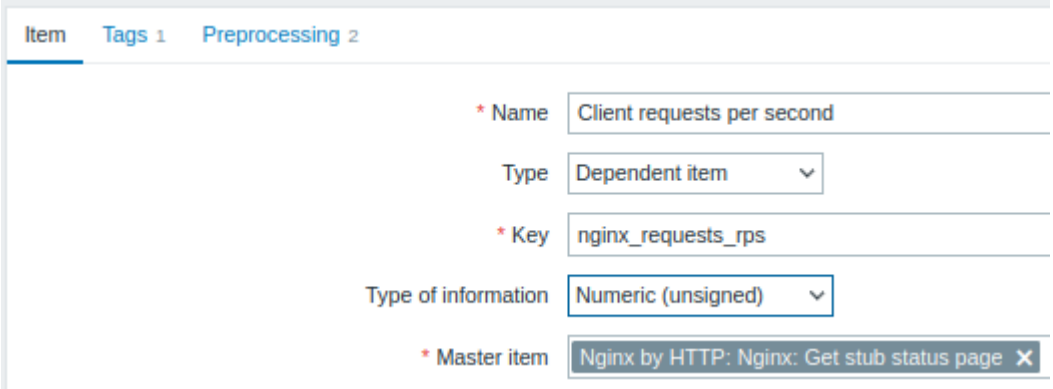


Sample Nginx stub status output:

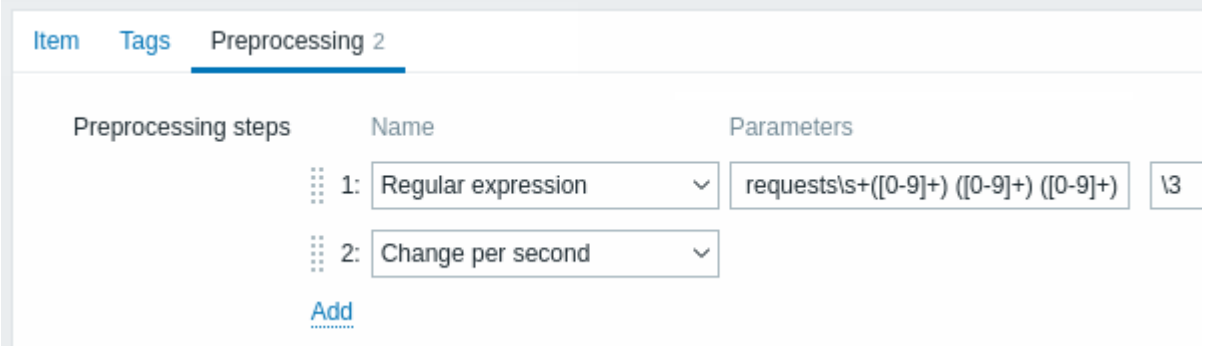
```
Active connections: 1 Active connections:
server accepts handled requests
 52 52 52
Reading: 0 Writing: 1 Waiting: 0
```

The next task is to configure dependent items that extract data.

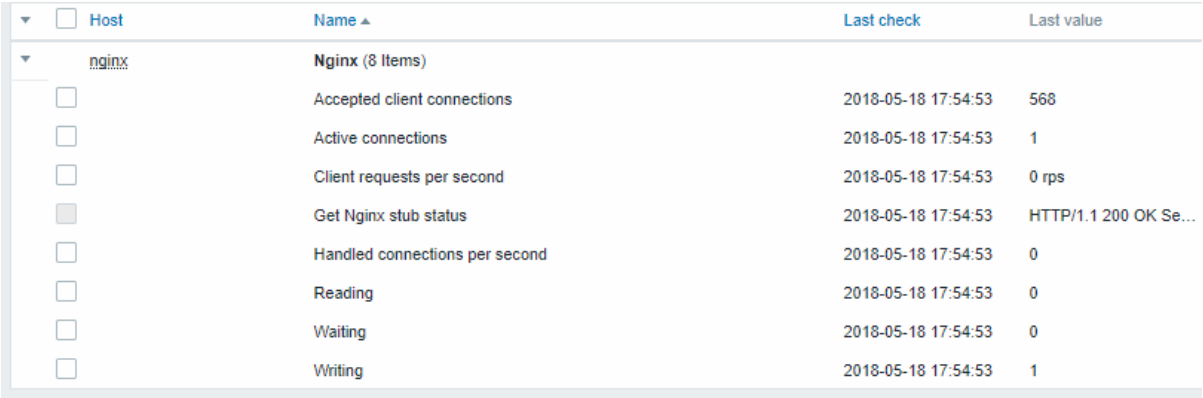
- Configure a sample dependent item for requests per second:



- Sample dependent item value preprocessing with regular expression `server accepts handled requests\s+([0-9]+) ([0-9]+) ([0-9]+)`:



- Check the complete result from stub module in *Latest data*:



17 Prometheus checks

Overview

Zabbix can query metrics exposed in the Prometheus line format.

Two steps are required to start gathering Prometheus data:

- an HTTP master item pointing to the appropriate data endpoint, e.g. `https://<prometheus host>/metrics`
- dependent items using a Prometheus preprocessing option to query required data from the metrics gathered by the master item

There are two Prometheus data preprocessing options:

- *Prometheus pattern* - used in normal items to query Prometheus data

- *Prometheus to JSON* - used in normal items and for low-level discovery. In this case queried Prometheus data are returned in a JSON format.

Bulk processing

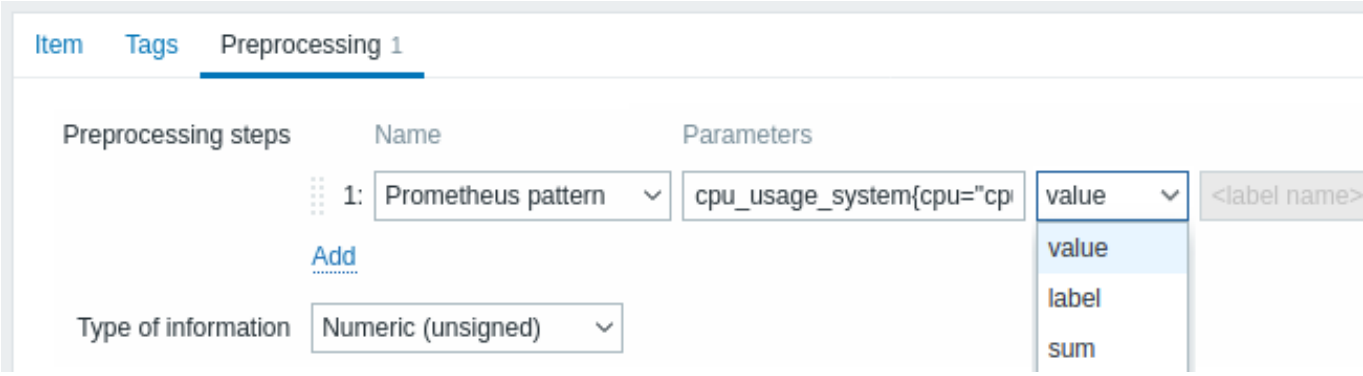
Bulk processing is supported for dependent items. To enable caching and indexing, the *Prometheus pattern* preprocessing must be the **first** preprocessing step. When *Prometheus pattern* is first preprocessing step then the parsed Prometheus data is cached and indexed by the first `<label>==<value>` condition in the *Prometheus pattern* preprocessing step. This cache is reused when processing other dependent items in this batch. For optimal performance, the first label should be the one with most different values.

If there is other preprocessing to be done before the first step, it should be moved either to the master item or to a new dependent item which would be used as the master item for the dependent items.

Configuration

Providing you have the HTTP master item configured, you need to create a dependent item that uses a Prometheus preprocessing step:

- Enter general dependent item parameters in the configuration form
- Go to the Preprocessing tab
- Select a Prometheus preprocessing option (*Prometheus pattern* or *Prometheus to JSON*)



The following parameters are specific to the *Prometheus pattern* preprocessing option:

Parameter	Description	Examples
Pattern	To define the required data pattern you may use a query language that is similar to Prometheus query language (see comparison table), e.g.: <metric name> - select by metric name {__name__="<metric name>"} - select by metric name {__name__=~"<regex>"} - select by metric name matching a regular expression {<label name>="<label value>",...} - select by label name {<label name>=~"<regex>",...} - select by label name matching a regular expression {__name__=~".*"}==<value> - select by metric value Or a combination of the above: <metric name>{<label1 name>="<label1 value>",<label2 name>=~"<regex>",...}==<value> Label value can be any sequence of UTF-8 characters, but the backslash, double-quote and line feed characters have to be escaped as \\, \" and \n respectively; other characters shall not be escaped.	*wmi_os_physical_memory_free_bytes* *cpu_usage_system{cpu="cpu-total"}* *cpu_usage_system{cpu=~".*"}* *cpu_usage_system{cpu="cpu-total",host=~".*"}* *wmi_service_state{name="dhcp"}==1* *wmi_os_timezone{timezone=~".*"}==1*

Parameter	Description	Examples
Result processing	Specify whether to return the value, the label or apply the appropriate function (if the pattern matches several lines and the result needs to be aggregated): **value** - return metric value (error if multiple lines matched) **label** - return value of the label specified in the *Label* field (error if multiple metrics are matched) **sum** - return the sum of values **min** - return the minimum value **max** - return the maximum value **avg** - return the average value **count** - return the count of values This field is only available for the *Prometheus pattern* option.	See also examples of using parameters below.
Output	Define label name (optional). In this case the value corresponding to the label name is returned. This field is only available for the *Prometheus pattern* option, if 'Label' is selected in the *Result processing* field.	

Examples of using parameters

1. The most common use case is to return the **value**. To return the value of `/var/db` from:

   `node_disk_usage_bytes{path="/var/cache"} 2.1766144e+09`
`node_disk_usage_bytes{path="/var/db"} 20480`
`node_disk_usage_bytes{path="/var/dpkg"} 8192`
`node_disk_usage_bytes{path="/var/empty"} 4096`

use the following parameters:

- *Pattern* - `node_disk_usage_bytes{path="/var/db"}`
- *Result processing* - select 'value'

2. You may also be interested in the **average** value of all `node_disk_usage_bytes` parameters:

- *Pattern* - `node_disk_usage_bytes`
- *Result processing* - select 'avg'

3. While Prometheus supports only numerical data, it is popular to use a workaround that allows to return the relevant textual description as well. This can be accomplished with a filter and specifying the label. So, to return the value of the 'color' label from

   `elasticsearch_cluster_health_status{cluster="elasticsearch",color="green"} 1`
`elasticsearch_cluster 0`
`elasticsearch_cluster_health_status{cluster="elasticsearch",color="yellow"} 0`

use the following parameters:

- *Pattern* - `elasticsearch_cluster_health_status {cluster="elasticsearch"} == 1`
- *Result processing* - select 'label'
- *Label* - specify 'color'

The filter (based on the numeric value '1') will match the appropriate row, while the label will return the health status description (currently 'green'; but potentially also 'red' or 'yellow').

Prometheus to JSON

Data from Prometheus can be used for low-level discovery. In this case data in JSON format are needed and the *Prometheus to JSON* preprocessing option will return exactly that.

For more details, see Discovery using Prometheus data.

Query language comparison

The following table lists differences and similarities between PromQL and Zabbix Prometheus preprocessing query language.

PromQL instant vector selector	Zabbix Prometheus preprocessing
Differences	

PromQL instant vector selector		Zabbix Prometheus preprocessing
Query target	Prometheus server	Plain text in Prometheus exposition format
Returns	Instant vector	Metric or label value (Prometheus pattern) Array of metrics for single value in JSON (Prometheus to JSON)
Label matching operators	**=**, **!=**, **=~**, **!~**	**=**, **!=**, **=~**, **!~**
Regular expression used in label or metric name matching	RE2	PCRE
Comparison operators	See list	Only **==** (equal) is supported for value filtering
Similarities		
Selecting by metric name that equals string	<metric name> or {__name__=”<metric name>”}	<metric name> or {__name__=”<metric name>”}
Selecting by metric name that matches regular expression	{__name__=~”<regex>”}	{__name__=~”<regex>”}
Selecting by <label name> value that equals string	{<label name>=”<label value>”,...}	{<label name>=”<label value>”,...}
Selecting by <label name> value that matches regular expression	{<label name>=~”<regex>”,...}	{<label name>=~”<regex>”,...}
Selecting by value that equals string	{__name__=~”.*”} == <value>	{__name__=~”.*”} == <value>

18 Script items

Overview

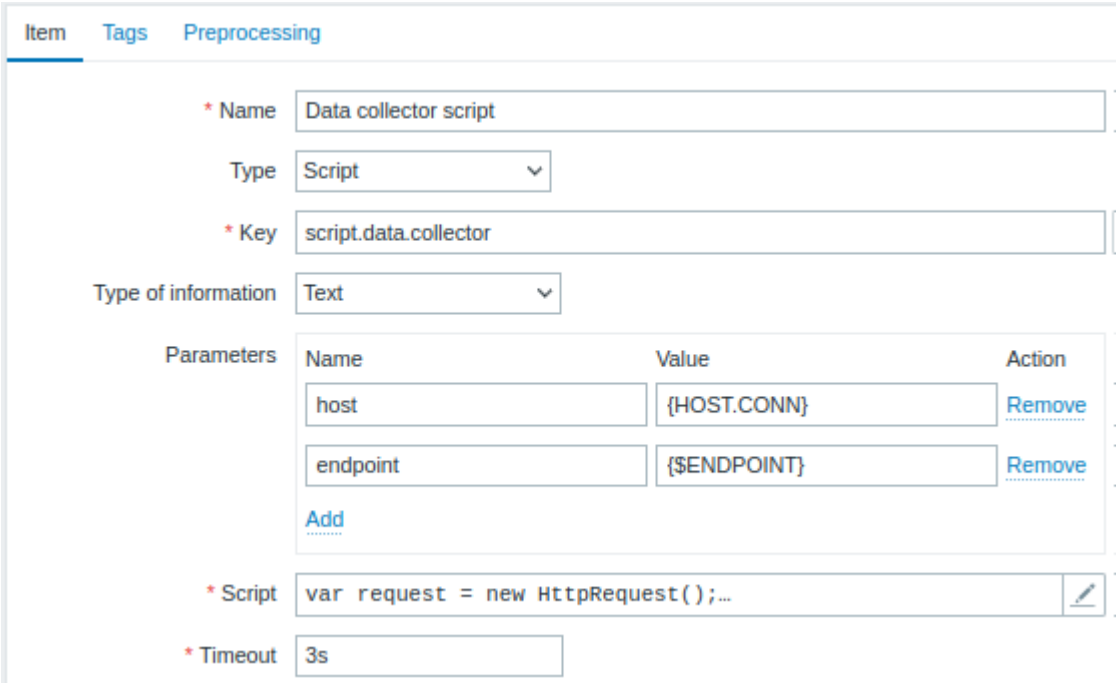
Script items can be used to collect data by executing a user-defined JavaScript code with the ability to retrieve data over HTTP/HTTPS. In addition to the script, an optional list of parameters (pairs of name and value) and timeout can be specified.

This item type may be useful in data collection scenarios that require multiple steps or complex logic. As an example, a Script item can be configured to make an HTTP call, then process the data received in the first step in some way, and pass transformed value to the second HTTP call.

Script items are processed by Zabbix server or proxy pollers.

Configuration

In the *Type* field of item configuration form select Script then fill out required fields.



All mandatory input fields are marked with a red asterisk.

The fields that require specific information for Script items are:

Field	Description
Key	Enter a unique key that will be used to identify the item.
Parameters	Specify the variables to be passed to the script as the attribute and value pairs. User macros are supported. To see which built-in macros are supported, do a search for "Script-type item" in the supported macro table.
Script	Enter JavaScript code in the block that appears when clicking in the parameter field (or on the view/edit button next to it). This code must provide the logic for returning the metric value. The code has access to all parameters, it may perform HTTP GET, POST, PUT and DELETE requests and has control over HTTP headers and request body. See also: Additional JavaScript objects, JavaScript Guide.
Timeout	JavaScript execution timeout (1-60s, default 3s); exceeding it will return error. Time suffixes are supported, e.g. 30s, 1m. Depending on the script it might take longer for the timeout to trigger.

Examples

Simple data collection

Collect the content of *https://www.example.com/release_notes*:

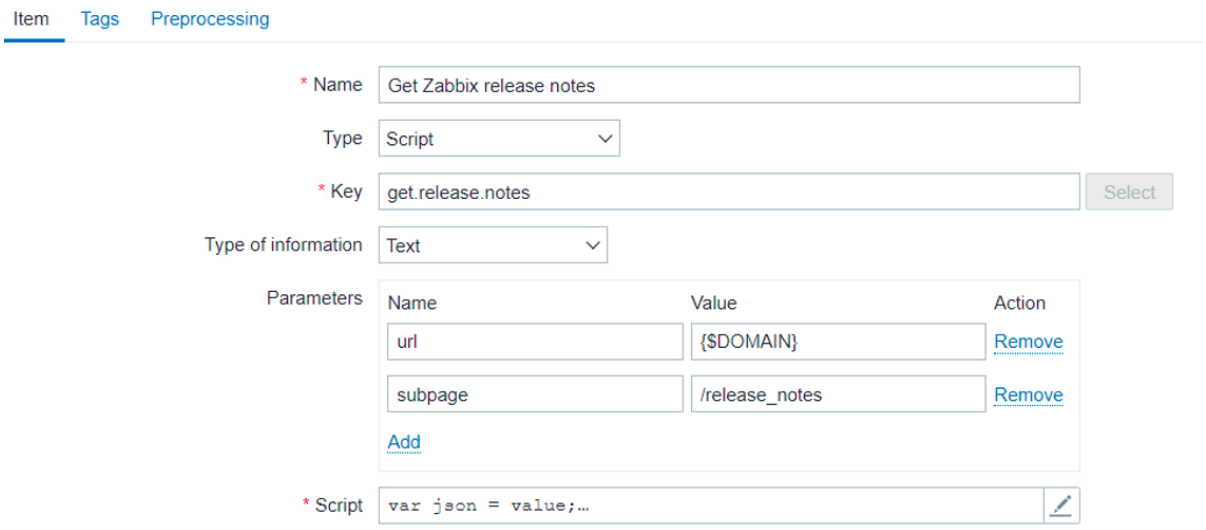
- Create an item with type "Script".
- In the *Script* field, enter:

```
var request = new HttpRequest();
return request.get("https://www.example.com/release_notes");
```

Data collection with parameters

Collect the content of a specific page and make use of parameters:

- Create an item with type "Script" and two parameters:
  - **url : {$DOMAIN}** (the user macro {$DOMAIN} should be defined, preferably on the host level)
  - **subpage : /release_notes**



- In the *Script* field, enter:

```
var obj = JSON.parse(value);
var url = obj.url;
var subpage = obj.subpage;
var request = new HttpRequest();
return request.get(url + subpage);
```

Multiple HTTP requests

Collect the content of both *https://www.example.com* and *https://www.example.com/release_notes*:

- Create an item with type "Script".
- In the *Script* field, enter:

```
var request = new HttpRequest();
return request.get("https://www.example.com") + request.get("https://www.example.com/release_notes");
```

Logging

Add the "Log test" entry to the Zabbix server log and receive the item value "1" in return:

- Create an item with type "Script".
- In the *Script* field, enter:

```
Zabbix.log(3, 'Log test');
return 1;
```

4 History and trends

Overview

History and trends are the two ways of storing collected data in Zabbix.

Whereas history keeps each collected value, trends keep averaged information on hourly basis and therefore are less resource-hungry.

Keeping history

You can set for how many days history will be kept:

- in the item properties form
- when mass-updating items
- when setting up housekeeper tasks

Any older data will be removed by the housekeeper.

The general strong advice is to keep history for the smallest possible number of days and that way not to overload the database with lots of historical values.

Instead of keeping a long history, you can keep longer data of trends. For example, you could keep history for 14 days and trends for 5 years.

You can get a good idea of how much space is required by history versus trends data by referring to the database sizing page.

While keeping shorter history, you will still be able to review older data in graphs, as graphs will use trend values for displaying older data.

Attention:
If history is set to '0', the item will update only dependent items and inventory. No trigger functions will be evaluated because trigger evaluation is based on history data only.

Note:
As an alternative way to preserve history consider to use history export functionality of loadable modules.

Keeping trends

Trends is a built-in historical data reduction mechanism which stores minimum, maximum, average and the total number of values per every hour for numeric data types.

You can set for how many days trends will be kept:

- in the item properties form
- when mass-updating items
- when setting up Housekeeper tasks

Trends usually can be kept for much longer than history. Any older data will be removed by the housekeeper.

Zabbix server accumulates trend data in runtime in the trend cache, as the data flows in. Server flushes **previous hour** trends of every item into the database (where frontend can find them) in these situations:

- server receives the first current hour value of the item
- 5 or less minutes of the current hour left and still no current hour values of the item
- server stops

To see trends on a graph you need to wait at least to the beginning of the next hour (if item is updated frequently) and at most to the end of the next hour (if item is updated rarely), which is 2 hours maximum.

When server flushes trend cache and there are already trends in the database for this hour (for example, server has been restarted mid-hour), server needs to use update statements instead of simple inserts. Therefore on a bigger installation if restart is needed it is desirable to stop server in the end of one hour and start in the beginning of the next hour to avoid trend data overlap.

History tables do not participate in trend generation in any way.

Attention:
If trends are set to '0', Zabbix server does not calculate or store trends at all.

Note:
The trends are calculated and stored with the same data type as the original values. As a result the average value calculations of unsigned data type values are rounded and the less the value interval is the less precise the result will be. For example if item has values 0 and 1, the average value will be 0, not 0.5.
Also restarting server might result in the precision loss of unsigned data type average value calculations for the current hour.

5 User parameters

Overview

Sometimes you may want to run an agent check that does not come predefined with Zabbix. This is where user parameters come to help.

You may write a command that retrieves the data you need and include it in the user parameter in the agent configuration file ('UserParameter' configuration parameter).

A user parameter has the following syntax:

```
UserParameter=<key>,<command>
```

As you can see, a user parameter also contains a key. The key will be necessary when configuring an item. Enter a key of your choice that will be easy to reference (it must be unique within a host).

Restart the agent or use the agent runtime control option to pick up the new parameter, e. g.:

```
zabbix_agentd -R userparameter_reload
```

Then, when configuring an item, enter the key to reference the command from the user parameter you want executed.

User parameters are commands executed by Zabbix agent. Note that up to 16MB of data can be returned before item value preprocessing steps.

/bin/sh is used as a command line interpreter under UNIX operating systems. User parameters obey the agent check timeout; if timeout is reached the forked user parameter process is terminated.

See also:

- Step-by-step tutorial on making use of user parameters
- Command execution

Examples of simple user parameters

A simple command:

```
UserParameter=ping,echo 1
```

The agent will always return '1' for an item with 'ping' key.

A more complex example:

```
UserParameter=mysql.ping,mysqladmin -uroot ping | grep -c alive
```

The agent will return '1', if MySQL server is alive, '0' - otherwise.

Flexible user parameters

Flexible user parameters accept parameters with the key. This way a flexible user parameter can be the basis for creating several items.

Flexible user parameters have the following syntax:

```
UserParameter=key[*],command
```

Parameter	Description
Key	Unique item key. The [*] defines that this key accepts parameters within the brackets. Parameters are given when configuring the item.
Command	Command to be executed to evaluate value of the key. *For flexible user parameters only*: You may use positional references $1...$9 in the command to refer to the respective parameter in the item key. Zabbix parses the parameters enclosed in [ ] of the item key and substitutes $1,...,$9 in the command accordingly. $0 will be substituted by the original command (prior to expansion of $0,...,$9) to be run. Positional references are interpreted regardless of whether they are enclosed between double (") or single (') quotes. To use positional references unaltered, specify a double dollar sign - for example, awk '{print $$2}'. In this case $$2 will actually turn into $2 when executing the command.

Attention:
Positional references with the $ sign are searched for and replaced by Zabbix agent only for flexible user parameters. For simple user parameters, such reference processing is skipped and, therefore, any $ sign quoting is not necessary.

Attention:
Certain symbols are not allowed in user parameters by default. See UnsafeUserParameters documentation for a full list.

Example 1

Something very simple:

```
UserParameter=ping[*],echo $1
```

We may define unlimited number of items for monitoring all having format ping[something].

- ping[0] - will always return '0'
- ping[aaa] - will always return 'aaa'

Example 2

Let's add more sense!

```
UserParameter=mysql.ping[*],mysqladmin -u$1 -p$2 ping | grep -c alive
```

This parameter can be used for monitoring availability of MySQL database. We can pass user name and password:

```
mysql.ping[zabbix,our_password]
```

Example 3

How many lines matching a regular expression in a file?

```
UserParameter=wc[*],grep -c "$2" $1
```

This parameter can be used to calculate number of lines in a file.

```
wc[/etc/passwd,root]
wc[/etc/services,zabbix]
```

Command result

The return value of the command is a standard output together with a standard error produced by the command.

Attention:
An item that returns text (character, log, or text type of information) will not become unsupported in case of a standard error output.

The return value is limited to 16MB (including trailing whitespace that is truncated); database limits also apply.

User parameters that return text (character, log, or text type of information) can also return a whitespace. In case of an invalid result, the item will become unsupported.

1 Extending Zabbix agents

This tutorial provides step-by-step instructions on how to extend the functionality of Zabbix agent with the use of a user parameter.

Step 1

Write a script or command line to retrieve required parameter.

For example, we may write the following command in order to get total number of queries executed by a MySQL server:

```
mysqladmin -uroot status | cut -f4 -d":" | cut -f1 -d"S"
```

When executed, the command returns total number of SQL queries.

Step 2

Add the command to zabbix_agentd.conf:

```
UserParameter=mysql.questions,mysqladmin -uroot status | cut -f4 -d":" | cut -f1 -d"S"
```

mysql.questions is a unique identifier. It can be any valid key identifier, for example, *queries*.

Test this parameter by using Zabbix agent with "-t" flag (if running under root, however, note that the agent may have different permissions when launched as a daemon):

```
zabbix_agentd -t mysql.questions
```

Step 3

Reload user parameters from the configuration file by running:

```
zabbix_agentd -R userparameter_reload
```

You may also restart the agent instead of the runtime control command.

Test the parameter by using zabbix_get utility.

Step 4

Add new item with Key=mysql.questions to the monitored host. Type of the item must be either Zabbix Agent or Zabbix Agent (active).

Be aware that type of returned values must be set correctly on Zabbix server. Otherwise Zabbix won't accept them.

6 Windows performance counters

Overview

You can effectively monitor Windows performance counters using the *perf_counter[]* key.

For example:

```
perf_counter["\Processor(0)\Interrupts/sec"]
```

or

```
perf_counter["\Processor(0)\Interrupts/sec", 10]
```

For more information on using this key or its English-only equivalent `perf_counter_en`, see Windows-specific item keys.

In order to get a full list of performance counters available for monitoring, you may run:

```
typeperf -qx
```

You may also use low-level discovery to discover multiple object instances of Windows performance counters and automate the creation of perf_counter items for multiple instance objects.

Numeric representation

Windows maintains numeric representations (indexes) for object and performance counter names. Zabbix supports these numeric representations as parameters to the `perf_counter`, `perf_counter_en` item keys and in `PerfCounter`, `PerfCounterEn` configuration parameters.

However, it's not recommended to use them unless you can guarantee your numeric indexes map to correct strings on specific hosts. If you need to create portable items that work across different hosts with various localized Windows versions, you can use the `perf_counter_en` key or `PerfCounterEn` configuration parameter which allow to use English names regardless of system locale.

To find out the numeric equivalents, run **regedit**, then find *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\00*

The registry entry contains information like this:

```
1
1847
2
System
4
Memory
6
% Processor Time
10
File Read Operations/sec
12
File Write Operations/sec
14
File Control Operations/sec
16
File Read Bytes/sec
18
File Write Bytes/sec
....
```

Here you can find the corresponding numbers for each string part of the performance counter, like in '\System\% Processor Time':

```
System → 2
% Processor Time → 6
```

Then you can use these numbers to represent the path in numbers:

```
\2\6
```

Performance counter parameters

You can deploy some PerfCounter parameters for the monitoring of Windows performance counters.

For example, you can add these to the Zabbix agent configuration file:

```
PerfCounter=UserPerfCounter1,"\Memory\Page Reads/sec",30
or
PerfCounter=UserPerfCounter2,"\4\24",30
```

With such parameters in place, you can then simply use *UserPerfCounter1* or *UserPerfCounter2* as the keys for creating the respective items.

Remember to restart Zabbix agent after making changes to the configuration file.

7 Mass update

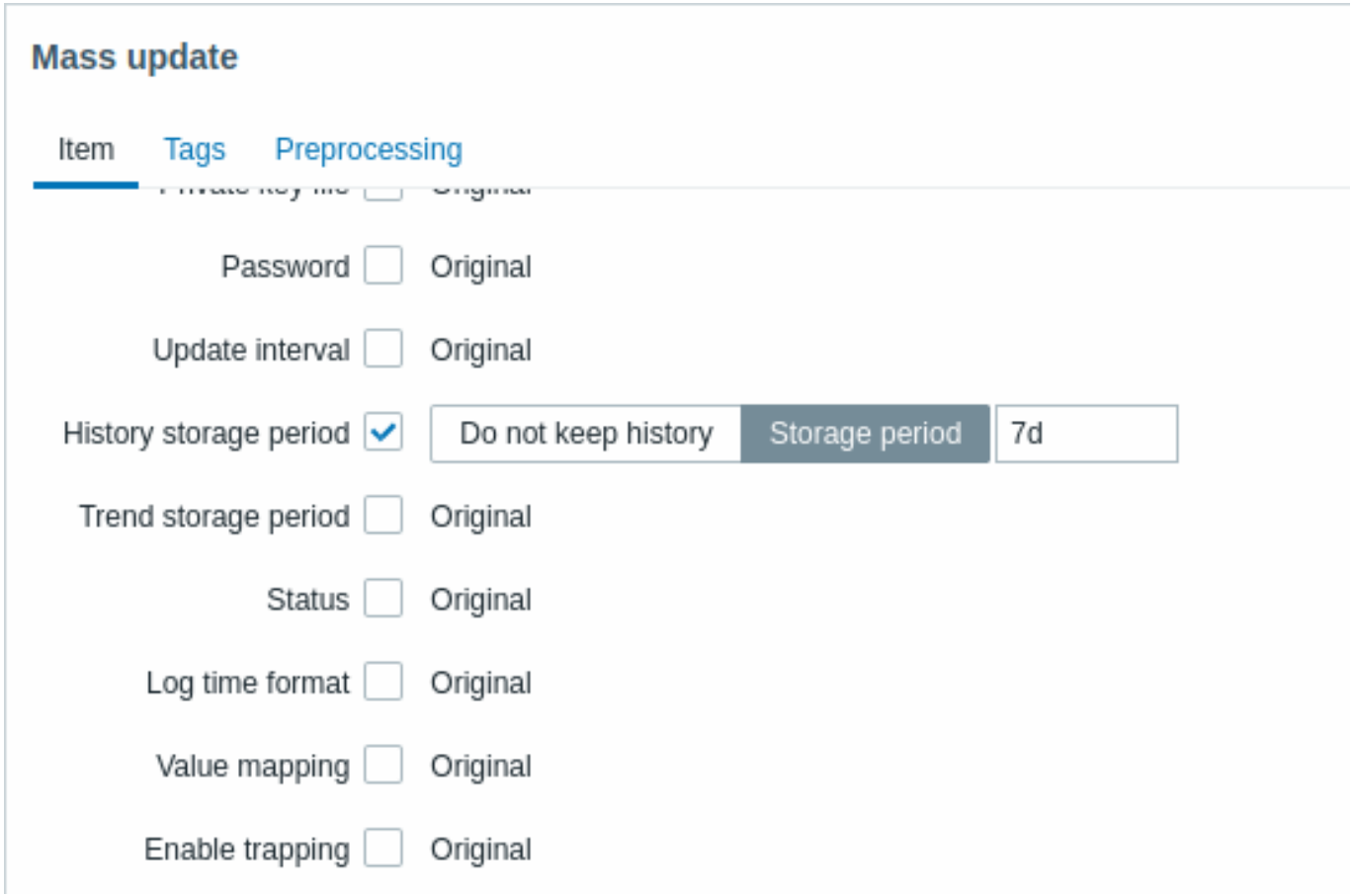
Overview

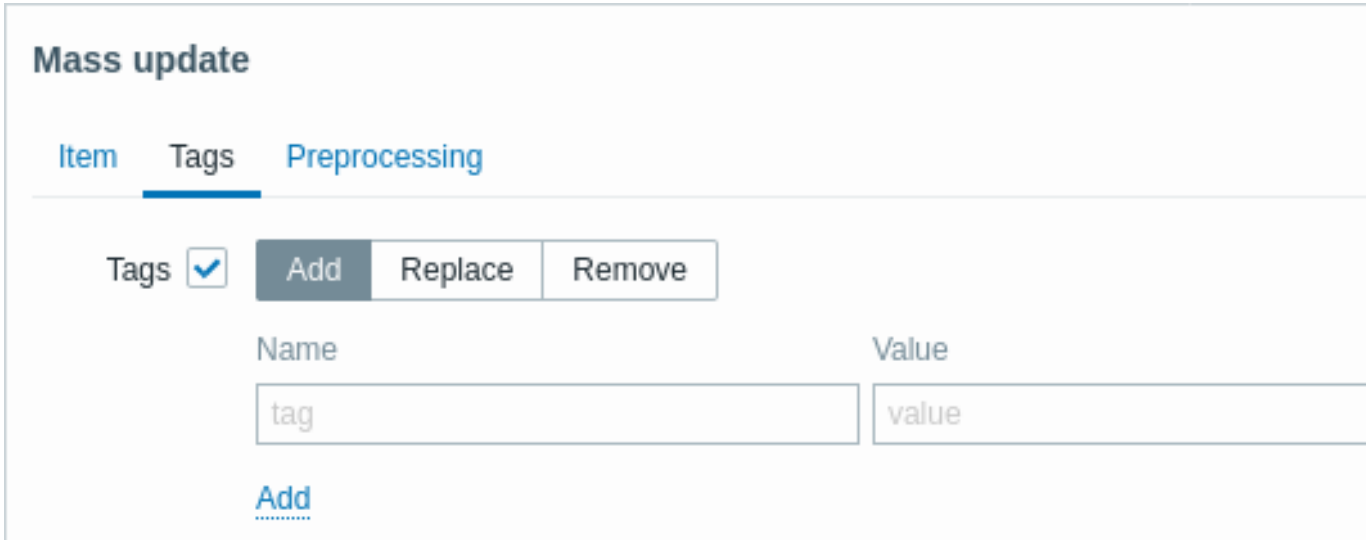
Sometimes you may want to change some attribute for a number of items at once. Instead of opening each individual item for editing, you may use the mass update function for that.

Using mass update

To mass-update some items, do the following:

- Mark the checkboxes of the items to update in the list
- Click on *Mass update* below the list
- Navigate to the tab with required attributes (*Item*, *Tags* or *Preprocessing*)
- Mark the checkboxes of the attributes to update
- Enter new values for the attributes

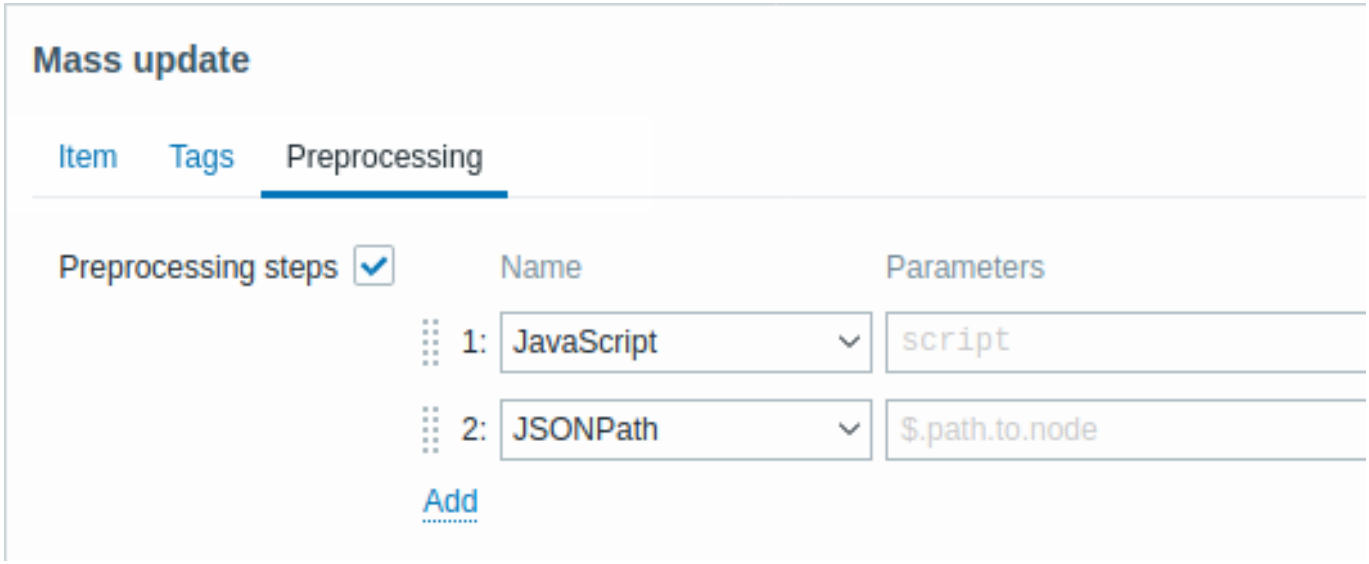




The *Tags* option allows to:

- *Add* - add specified tags to the items (tags with the same name, but different values are not considered 'duplicates' and can be added to the same item).
- *Replace* - remove the specified tags and add tags with new values
- *Remove* - remove specified tags from the items

User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags.



The *Preprocessing* option allows to **replace** the existing preprocessing steps on the items with the steps that are specified here.

When done, click on *Update*.

8 Value mapping

Overview

Value mapping allows configuring a more user-friendly representation of received values using mappings between numeric/string values and string representations.

For example, when an item's value is "0" or "1," value mappings can be used to represent these values in a more user-friendly manner:

- 0 → Not Available
- 1 → Available

Value mappings for data backup types could be configured as follows:

- F → Full
- D → Differential
- I → Incremental

Value ranges for voltage could be configured as follows:

- <=209 → Low
- 210-230 → OK
- >=231 → High

Value mapping is used in Zabbix frontend and notifications sent by media types.

Attention:
Substitution of the received value with the configured representation is performed both in Zabbix frontend and server; however, the server handles substitution only in the following cases:

- when populating host inventory;
- when expanding supported macros {ITEM.VALUE}, {ITEM.LASTVALUE}, {EVENT.OPDATA}, and {EVENT.CAUSE.OPDATA}.

Value mappings are set up on templates or hosts. Once configured, they are available for all items within the respective template or host. When configuring items, specify the name of a previously configured value mapping in the *Value mapping* parameter.

Note:
There is no value map inheritance - hosts and templates do not inherit value mappings from linked templates. Template items on a host will continue to use the value mappings configured on the template.

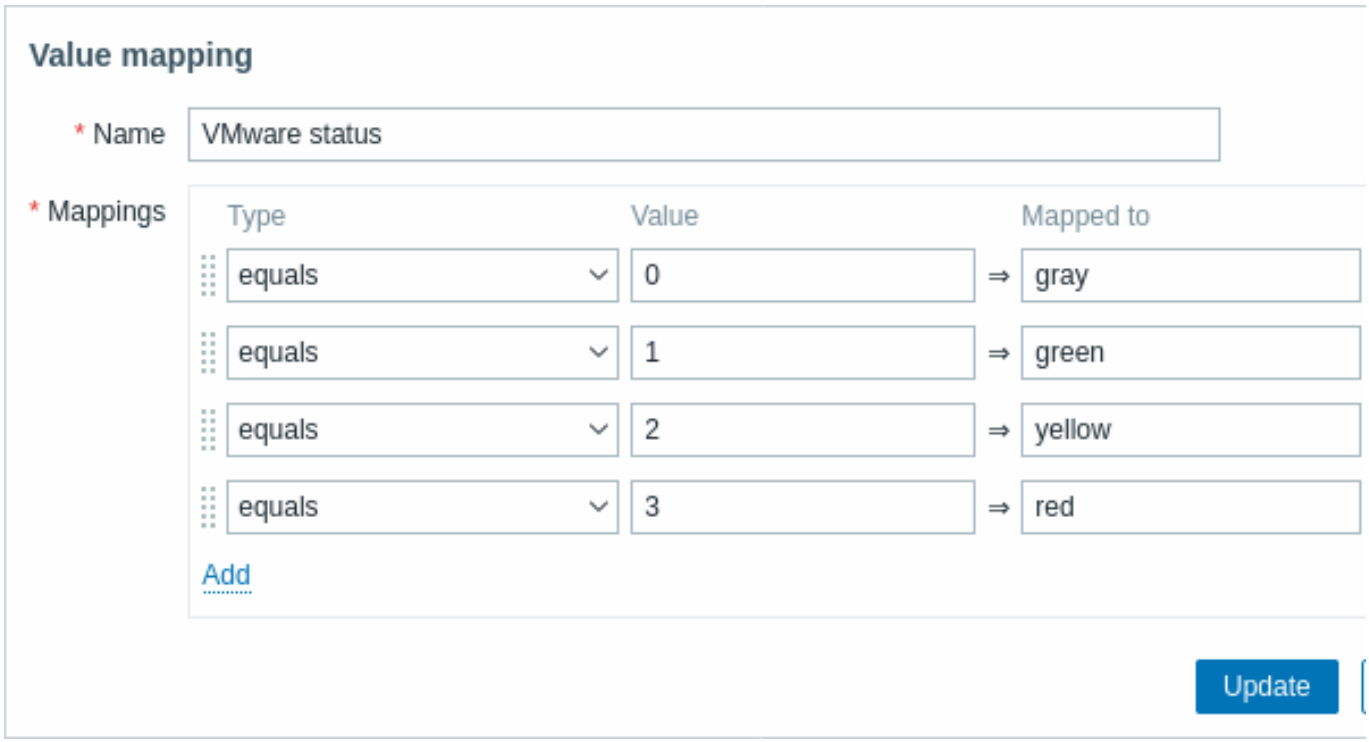
Note:
Value mappings can be used with items having *Numeric (unsigned)*, *Numeric (float)*, and *Character* types of information.

Value mappings are exported/imported with the respective template or host. They can also be mass-updated using the host and template mass update forms.

Configuration

To configure a value mapping, follow these steps:

1. Open the host or template configuration form.
2. In the *Value mapping* tab, click *Add* to add a new value mapping, or click on the name of an existing mapping to edit it.

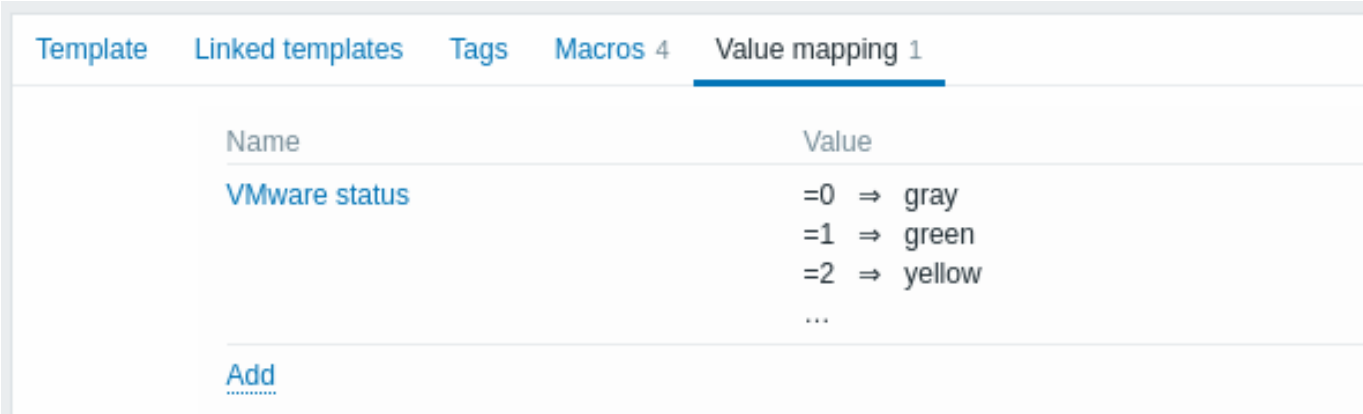


Parameters of a value mapping:

Parameter	Description
Name	Unique name for the set of value mappings.
Mappings	Individual rules for mapping numeric/string values to string representations. Mapping is applied in the order of the rules that can be reordered by dragging.
Type	Mapping type: **equals** - equal values will be mapped; **is greater than or equals** - equal or greater values will be mapped; **is less than or equals** - equal or smaller values will be mapped; **in range** - values in range will be mapped; the range is expressed as <number1>-<number2> or <number>; multiple ranges are supported (for example, 1-10,101-110,201); **regexp** - values corresponding to the regular expression will be mapped (global regular expressions are not supported); **default** - all outstanding values will be mapped, other than those with specific mappings. For mapping ranges, only numeric value types (*is greater than or equals*, *is less than or equals*, *in range*) are supported.
Value	Incoming value (may contain a range or regular expression, depending on the mapping type).
Mapped to	String representation (up to 64 characters) for the incoming value.

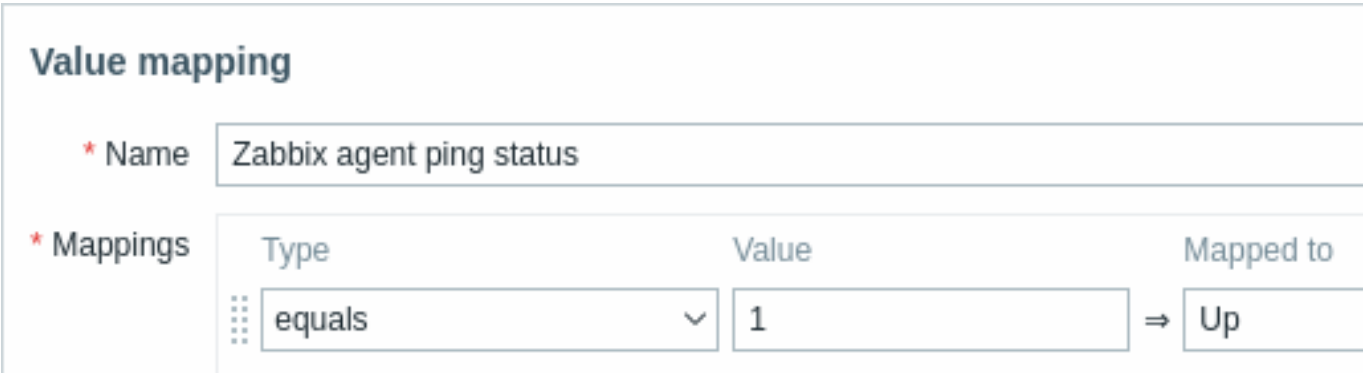
All mandatory input fields are marked with a red asterisk.

When viewing the value mapping in the list, only the first three mappings are visible, with three dots indicating that more mappings exist.



Value mapping example

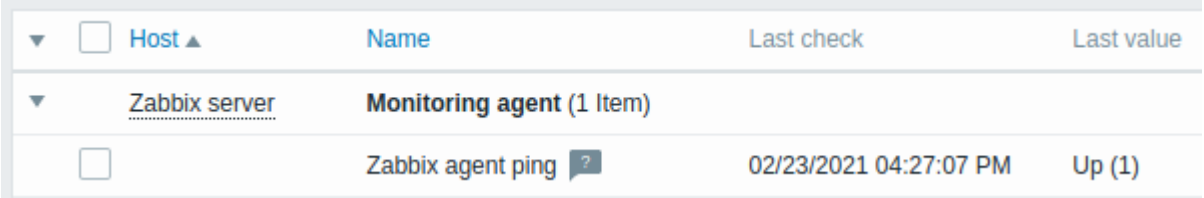
One of the predefined agent items *Zabbix agent ping* uses a template-level value mapping "Zabbix agent ping status" to display its values.



In the item configuration form, you can find a reference to this value mapping in the *Value mapping* field:

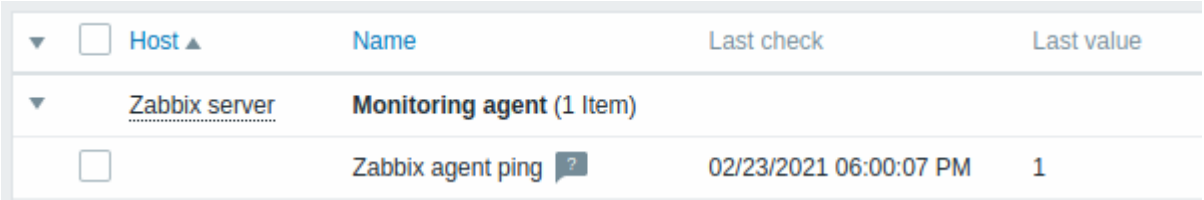


This mapping is used in the *Monitoring* → *Latest data* section to display "Up" (with the raw value in parentheses).



Note:
In the *Latest data* section, displayed values are shortened to 20 symbols. If value mapping is used, this shortening is not applied to the mapped value but only to the raw value (displayed in parentheses).

Without a predefined value mapping, you would only see "1", which might be challenging to understand.



9 Queue

Overview

The queue displays items that are waiting for a refresh. The queue is just a **logical** representation of data. There is no IPC queue or any other queue mechanism in Zabbix.

Items monitored by proxies are also included in the queue - they will be counted as queued for the proxy history data update period.

Only items with scheduled refresh times are displayed in the queue. This means that the following item types are excluded from the queue:

- log, logrt and event log active Zabbix agent items
- SNMP trap items
- trapper items
- web monitoring items
- dependent items

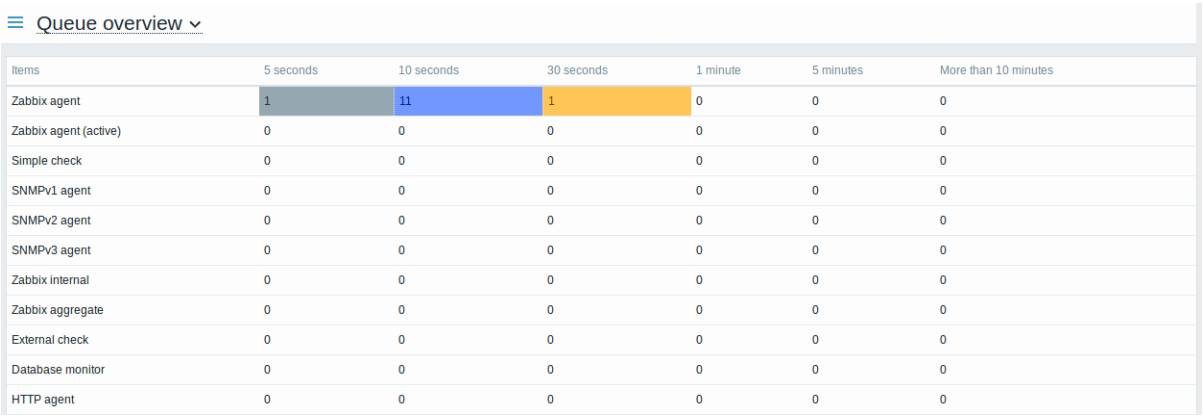
Statistics shown by the queue is a good indicator of the performance of Zabbix server.

The queue is retrieved directly from Zabbix server using JSON protocol. The information is available only if Zabbix server is running.

Attention:
Items are not counted in the queue if the item interface becomes unavailable due to connection problems or agent not working properly.

Reading the queue

To read the queue, go to *Administration* → *Queue*.

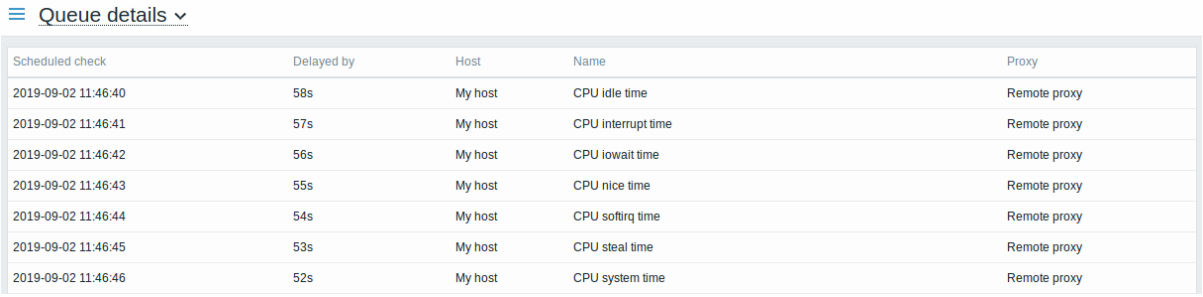

Queue overview

Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	1	11	1	0	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
HTTP agent	0	0	0	0	0	0

The picture here is generally ”ok” so we may assume that the server is doing fine.

The queue shows some items waiting up to 30 seconds. It would be great to know what items these are.

To do just that, select *Queue details* in the title dropdown. Now you can see a list of those delayed items.

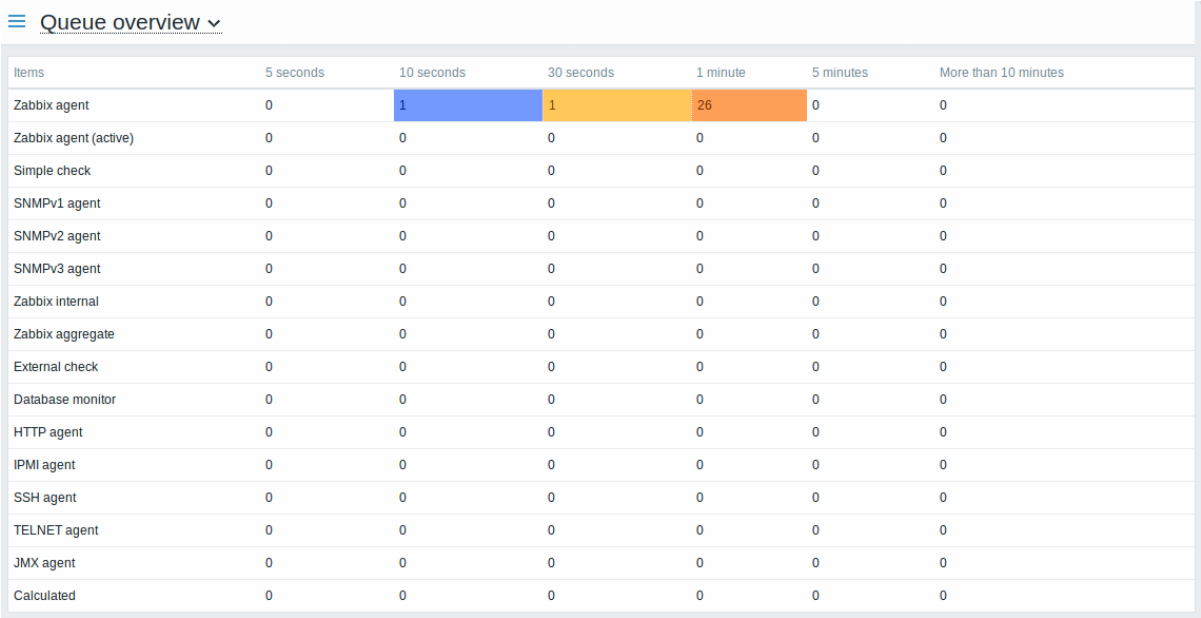

Queue details

Scheduled check	Delayed by	Host	Name	Proxy
2019-09-02 11:46:40	58s	My host	CPU idle time	Remote proxy
2019-09-02 11:46:41	57s	My host	CPU interrupt time	Remote proxy
2019-09-02 11:46:42	56s	My host	CPU iowait time	Remote proxy
2019-09-02 11:46:43	55s	My host	CPU nice time	Remote proxy
2019-09-02 11:46:44	54s	My host	CPU softirq time	Remote proxy
2019-09-02 11:46:45	53s	My host	CPU steal time	Remote proxy
2019-09-02 11:46:46	52s	My host	CPU system time	Remote proxy

With these details provided it may be possible to find out why these items might be delayed.

With one or two delayed items there perhaps is no cause for alarm. They might get updated in a second. However, if you see a bunch of items getting delayed for too long, there might be a more serious problem.

See also: Aligning time zones when using scheduling intervals.



Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	0	1	1	26	0	0
Zabbix agent (active)	0	0	0	0	0	0
Simple check	0	0	0	0	0	0
SNMPv1 agent	0	0	0	0	0	0
SNMPv2 agent	0	0	0	0	0	0
SNMPv3 agent	0	0	0	0	0	0
Zabbix internal	0	0	0	0	0	0
Zabbix aggregate	0	0	0	0	0	0
External check	0	0	0	0	0	0
Database monitor	0	0	0	0	0	0
HTTP agent	0	0	0	0	0	0
IPMI agent	0	0	0	0	0	0
SSH agent	0	0	0	0	0	0
TELNET agent	0	0	0	0	0	0
JMX agent	0	0	0	0	0	0
Calculated	0	0	0	0	0	0

Queue item

A special internal item **zabbix[queue,<from>,<to>]** can be used to monitor the health of the queue in Zabbix. It will return the number of items delayed by the set amount of time. For more information see Internal items.

10 Value cache

Overview

To make the calculation of trigger expressions, calculated items and some macros much faster, a value cache option is supported by the Zabbix server.

This in-memory cache can be used for accessing historical data, instead of making direct SQL calls to the database. If historical values are not present in the cache, the missing values are requested from the database and the cache updated accordingly.

Item values remain in value cache either until:

- the item is deleted (cached values are deleted after the next configuration sync);
- the item value is outside the time or count range specified in the trigger/calculated item expression (cached value is removed when a new value is received);
- the time or count range specified in the trigger/calculated item expression is changed so that less data is required for calculation (unnecessary cached values are removed after 24 hours).

Note:
Value cache status can be observed by using the server runtime control option `diaginfo` (or `diaginfo=valuecache`) and inspecting the section for value cache diagnostic information. This can be useful for determining misconfigured triggers or calculated items.

To enable the value cache functionality, an optional **ValueCacheSize** parameter is supported by the Zabbix server configuration file.

Two internal items are supported for monitoring the value cache: **zabbix[vcache,buffer,<mode>]** and **zabbix[vcache,cache,<parameter**
See more details with internal items.

11 Execute now

Overview

Checking for a new item value in Zabbix is a cyclic process that is based on configured update intervals. While for many items the update intervals are quite short, there are others (including low-level discovery rules) for which the update intervals are quite long, so in real-life situations there may be a need to check for a new value quicker - to pick up changes in discoverable resources, for example. To accommodate such a necessity, it is possible to reschedule a passive check and retrieve a new value immediately.

This functionality is supported for **passive** checks only. The following item types are supported:

- Zabbix agent (passive)

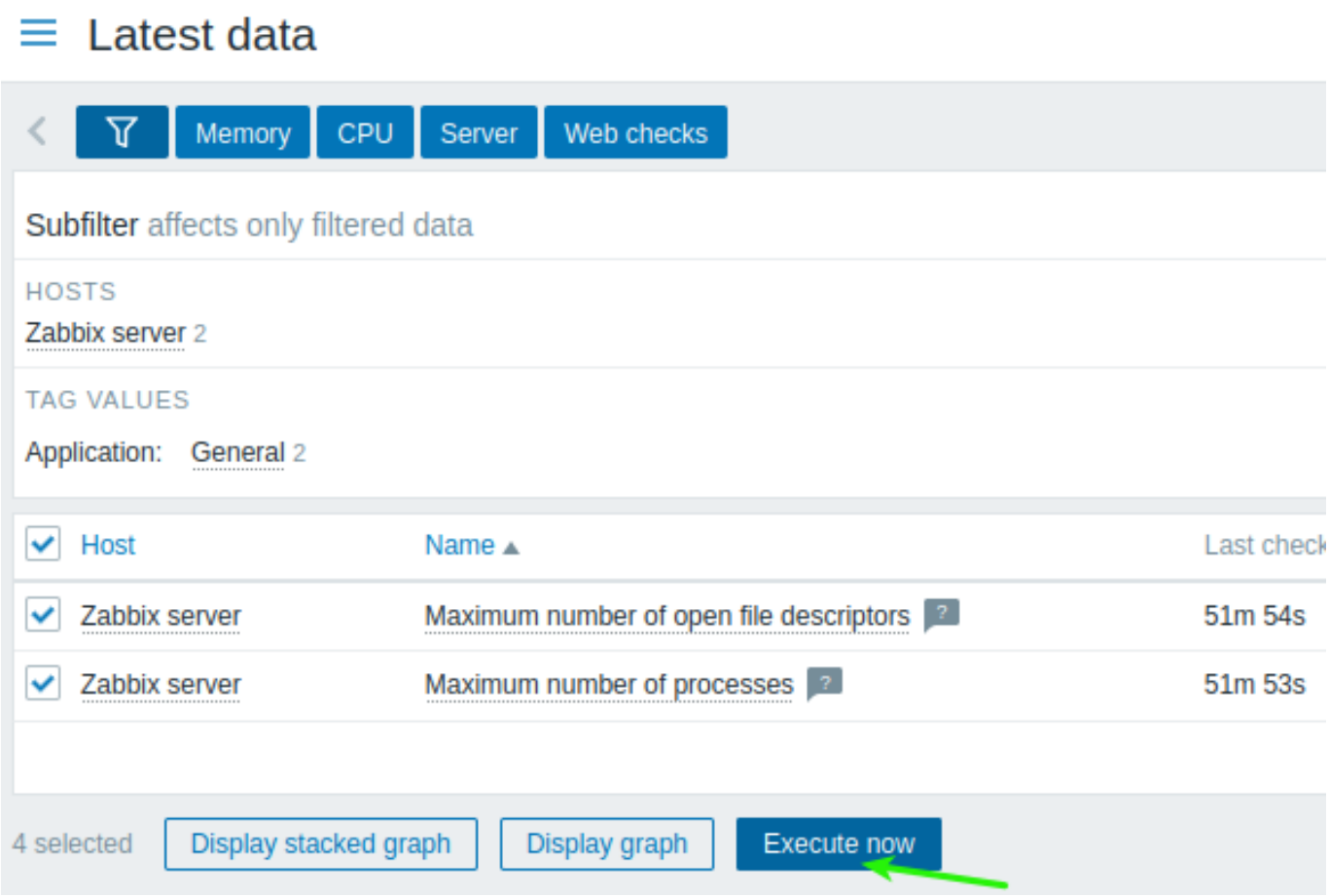
- SNMPv1/v2/v3 agent
- IPMI agent
- Simple check
- Zabbix internal
- External check
- Database monitor
- JMX agent
- SSH agent
- Telnet
- Calculated
- HTTP agent
- Dependent item
- Script

Attention:
The check must be present in the configuration cache in order to get executed; for more information see CacheUpdateFrequency. Before executing the check, the configuration cache is **not** updated, thus very recent changes to item/discovery rule configuration will not be picked up. Therefore, it is also not possible to check for a new value for an item/rule that is being created or has been created just now; use the *Test* option while configuring an item for that.

Configuration

To execute a passive check immediately:

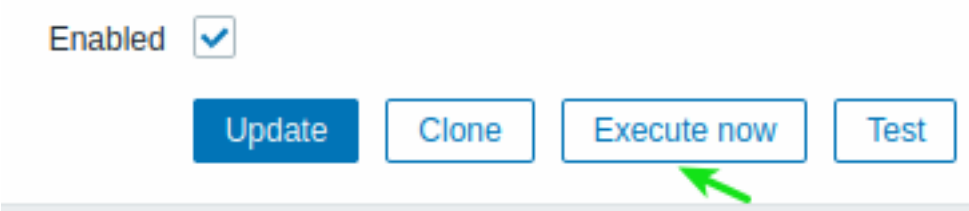
- click on *Execute now* for selected items in the list of latest data:



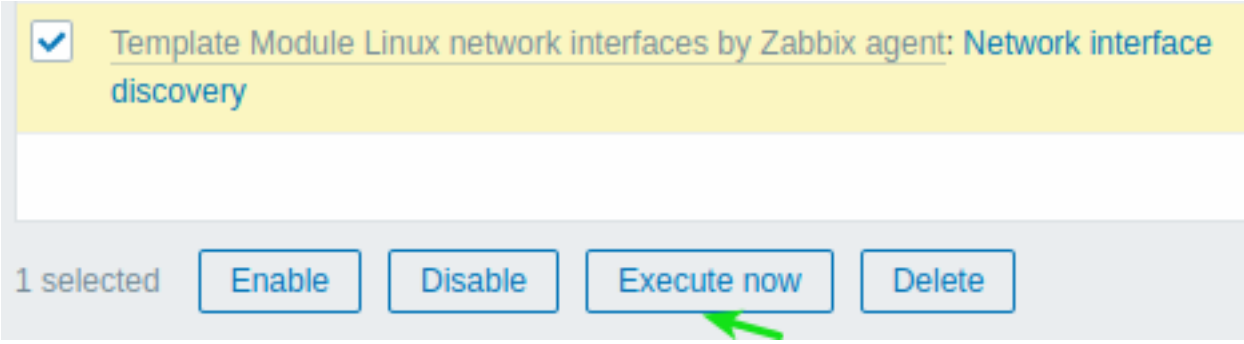
Several items can be selected and "executed now" at once.

In latest data this option is available only for hosts with read-write access. Accessing this option for hosts with read-only permissions depends on the user role option called *Invoke "Execute now" on read-only hosts*.

- click on *Execute now* in an existing item (or discovery rule) configuration form:



- click on *Execute now* for selected items/rules in the list of items/discovery rules:



Several items/rules can be selected and ”executed now” at once.

12 Restricting agent checks

Overview

It is possible to restrict checks on the agent side by creating an item blacklist, a whitelist, or a combination of whitelist/blacklist.

To do that use a combination of two agent configuration parameters:

- `AllowKey=<pattern>` - which checks are allowed; <pattern> is specified using a wildcard (*) expression
- `DenyKey=<pattern>` - which checks are denied; <pattern> is specified using a wildcard (*) expression

Note that:

- All system.run[*] items (remote commands, scripts) are disabled by default, even when no deny keys are specified, it should be assumed that DenyKey=system.run[*] is implicitly appended.
- Since Zabbix 5.0.2 the EnableRemoteCommands agent parameter is:
  - deprecated by Zabbix agent
  - unsupported by Zabbix agent2

Therefore, to allow remote commands, specify an AllowKey=system.run[<command>,*] for each allowed command, * stands for wait and nowait mode. It is also possible to specify AllowKey=system.run[*] parameter to allow all commands with wait and nowait modes. To disallow specific remote commands, add DenyKey parameters with system.run[] commands before the AllowKey=system.run[*] parameter.

Important rules

- A whitelist without a deny rule is only allowed for system.run[*] items. For all other items, AllowKey parameters are not allowed without a DenyKey parameter; in this case Zabbix agent **will not start** with only AllowKey parameters.
- The order matters. The specified parameters are checked one by one according to their appearance order in the configuration file:
  - As soon as an item key matches an allow/deny rule, the item is either allowed or denied; and rule checking stops. So if an item matches both an allow rule and a deny rule, the result will depend on which rule comes first.
  - The order affects also EnableRemoteCommands parameter (if used).
- Unlimited numbers of AllowKey/DenyKey parameters is supported.
- AllowKey, DenyKey rules do not affect HostnameItem, HostMetadataItem, HostInterfaceItem configuration parameters.
- Key pattern is a wildcard expression where the wildcard (*) character matches any number of any characters in certain position. It might be used in both the key name and parameters.
- If a specific item key is disallowed in the agent configuration, the item will be reported as unsupported (no hint is given as to the reason);
- Zabbix agent with --print (-p) command line option will not show keys that are not allowed by configuration;
- Zabbix agent with --test (-t) command line option will return ”Unsupported item key.” status for keys that are not allowed by configuration;
- Denied remote commands will not be logged in the agent log (if LogRemoteCommands=1).

Use cases

Deny specific check

- Blacklist a specific check with DenyKey parameter. Matching keys will be disallowed. All non-matching keys will be allowed, except system.run[] items.

For example:

```
# Deny secure data access
DenyKey=vfs.file.contents[/etc/passwd,*]
```

Attention:
A blacklist may not be a good choice, because a new Zabbix version may have new keys that are not explicitly restricted by the existing configuration. This could cause a security flaw.

Deny specific command, allow others

- Blacklist a specific command with DenyKey parameter. Whitelist all other commands, with the AllowKey parameter.

```
# Disallow specific command
DenyKey=system.run[ls -l /]

# Allow other scripts
AllowKey=system.run[*]
```

Allow specific check, deny others

- Whitelist specific checks with AllowKey parameters, deny others with `DenyKey=*`

For example:

```
# Allow reading logs:
AllowKey=vfs.file.*[/var/log/*]

# Allow localtime checks
AllowKey=system.localtime[*]

# Deny all other keys
DenyKey=*
```

Pattern examples

Pattern	Description	Matches	No match
*	Matches all possible keys with or without parameters.	Any	None
vfs.file.contents	Matches `vfs.file.contents` without parameters.	vfs.file.contents	vfs.file.contents[/etc/passwd]
vfs.file.contents[]	Matches `vfs.file.contents` with empty parameters.	vfs.file.contents[]	vfs.file.contents
vfs.file.contents[*]	Matches `vfs.file.contents` with any parameters; will not match `vfs.file.contents` without square brackets.	vfs.file.contents[] vfs.file.contents[/path/to/file]	vfs.file.contents
vfs.file.contents[/etc/passwd,]*	Matches `vfs.file.contents` with first parameters matching /etc/passwd and all other parameters having any value (also empty).	vfs.file.contents[/etc/passwd,] vfs.file.contents[/etc/passwd,utf8]	vfs.file.contents[/etc/passwd] vfs.file.contents[/var/log/zabbix_s vfs.file.contents[]
*vfs.file.contents[*passwd*]*	Matches `vfs.file.contents` with first parameter matching *passwd* and no other parameters.	vfs.file.contents[/etc/passwd]	vfs.file.contents[/etc/passwd,] vfs.file.contents[/etc/passwd, utf8]
*vfs.file.contents[*passwd*,*]*	Matches `vfs.file.contents` with only first parameter matching *passwd* and all following parameters having any value (also empty).	vfs.file.contents[/etc/passwd,] vfs.file.contents[/etc/passwd, utf8]	vfs.file.contents[/etc/passwd] vfs.file.contents[/tmp/test]
vfs.file.contents[/var/log/zabbix_server.log,,abc]*	Matches `vfs.file.contents` with first parameter matching /var/log/zabbix_server.log, third parameter matching 'abc' and any (also empty) second parameter.	vfs.file.contents[/var/log/zabbix_server.log,,abc] vfs.file.contents[/var/log/zabbix_server.log,utf8,abc]	vfs.file.contents[/var/log/zabbix_s vfs.file.contents[/var/log/zabbix_s
vfs.file.contents[/etc/passwd,utf8]	Matches `vfs.file.contents` with first parameter matching /etc/passwd, second parameter matching 'utf8' and no other arguments.	vfs.file.contents[/etc/passwd,utf8]	vfs.file.contents[/etc/passwd,] vfs.file.contents[/etc/passwd,utf
*vfs.file.**	Matches any keys starting with `vfs.file.` without any parameters.	vfs.file.contents vfs.file.size	vfs.file.contents[] vfs.file.size[/var/log/zabbix_serve

Pattern	Description	Matches	No match
vfs.file.[*]*	Matches any keys starting with `vfs.file.` with any parameters.	vfs.file.size.bytes[] vfs.file.size[/var/log/zabbix_server.log, utf8]	vfs.file.size.bytes
vfs..contents*	Matches any key starting with `vfs.` and ending with `.contents` without any parameters.	vfs.mount.point.file.contents vfs..contents	vfs.contents

system.run and AllowKey

A hypothetical script like 'myscript.sh' may be executed on a host via Zabbix agent in several ways:

1. As an item key in a passive or active check, for example:

- system.run[myscript.sh]
- system.run[myscript.sh,wait]
- system.run[myscript.sh.nowait]

Here the user may add "wait", "nowait" or omit the 2nd argument to use its default value in system.run[].

2. As a global script (initiated by user in frontend or API).

A user configures this script in *Alerts* → *Scripts*, sets "Execute on: Zabbix agent" and puts "myscript.sh" into the script's "Commands" input field. When invoked from frontend or API the Zabbix server sends to agent:

- system.run[myscript.sh,wait] - up to Zabbix 5.0.4
- system.run[myscript.sh] - since 5.0.5

Here the user does not control the "wait"/"nowait" parameters.

3. As a remote command from an action. The Zabbix server sends to agent:

- system.run[myscript.sh,nowait]

Here again the user does not control the "wait"/"nowait" parameters.

What that means is if we set AllowKey like:

```
AllowKey=system.run[myscript.sh]
```

then

- system.run[myscript.sh] - will be allowed
- system.run[myscript.sh,wait], system.run[myscript.sh,nowait] will not be allowed - the script will not be run if invoked as a step of action

To allow all described variants you may add:

```
AllowKey=system.run[myscript.sh,*]
DenyKey=system.run[*]
```

to the agent/agent2 parameters.

3 Triggers

Overview

Triggers are logical expressions that "evaluate" data gathered by items and represent the current system state.

While items are used to gather system data, it is highly impractical to follow these data all the time waiting for a condition that is alarming or deserves attention. The job of "evaluating" data can be left to trigger expressions.

Trigger expressions allow to define a threshold of what state of data is "acceptable". Therefore, should the incoming data surpass the acceptable state, a trigger is "fired" - or changes its status to PROBLEM.

A trigger may have the following status:

Status	Description
OK	This is a normal trigger status.
Problem	Something has happened. For example, the processor load is too high.

Status	Description
Unknown	The trigger value cannot be calculated. See Unknown status.

In a simple trigger we may want to set a threshold for a five-minute average of some data, for example, the CPU load. This is accomplished by defining a trigger expression where:

- the 'avg' function is applied to the value received in the item key
- a five minute period for evaluation is used
- a threshold of '2' is set

```
avg(/host/key,5m)>2
```

This trigger will "fire" (become PROBLEM) if the five-minute average is *over* 2.

In a more complex trigger, the expression may include a **combination** of multiple functions and multiple thresholds. See also: Trigger expression.

> **Note:**
> After enabling a trigger (changing its configuration status from *Disabled* to *Enabled*), the trigger expression is evaluated as soon as an item in it receives a value or the time to handle a time-based function comes.

Most trigger functions are evaluated based on item value history data, while some trigger functions for long-term analytics, e.g. **trendavg()**, **trendcount()**, etc, use trend data.

Calculation time

A trigger is recalculated every time Zabbix server receives a new value that is part of the expression. When a new value is received, each function that is included in the expression is recalculated (not just the one that received the new value).

Additionally, a trigger is recalculated each time when a new value is received **and** every 30 seconds if time-based functions are used in the expression.

Time-based functions are **nodata()**, **date()**, **dayofmonth()**, **dayofweek()**, **time()**, **now()**; they are recalculated every 30 seconds by the Zabbix history syncer process.

Triggers that reference trend functions **only** are evaluated once per the smallest time period in the expression. See also trend functions.

Evaluation period

An evaluation period is used in functions referencing the item history. It allows to specify the interval we are interested in. It can be specified as time period (30s, 10m, 1h) or as a value range (#5 - for five latest values).

The evaluation period is measured up to "now" - where "now" is the latest recalculation time of the trigger (see Calculation time above); "now" is not the "now" time of the server.

The evaluation period specifies either:

- To consider all values between "now-time period" and "now" (or, with time shift, between "now-time shift-time period" and "now-time_shift")
- To consider no more than the num count of values from the past, up to "now"
  - If there are 0 available values for the time period or num count specified - then the trigger or calculated item that uses this function becomes unsupported

Note that:

- If only a single function (referencing data history) is used in the trigger, "now" is always the latest received value. For example, if the last value was received an hour ago, the evaluation period will be regarded as up to the latest value an hour ago.
- A new trigger is calculated as soon as the first value is received (history functions); it will be calculated within 30 seconds for time-based functions. Thus the trigger will be calculated even though perhaps the set evaluation period (for example, one hour) has not yet passed since the trigger was created. The trigger will also be calculated after the first value, even though the evaluation range was set, for example, to ten latest values.

Unknown status

It is possible that an unknown operand appears in a trigger expression if:

- an unsupported item is used
- the function evaluation for a supported item results in an error

In this case a trigger generally evaluates to ”unknown” (although there are some exceptions). For more details, see Expressions with unknown operands.

It is possible to get notified on unknown triggers.

1 Configuring a trigger

Overview

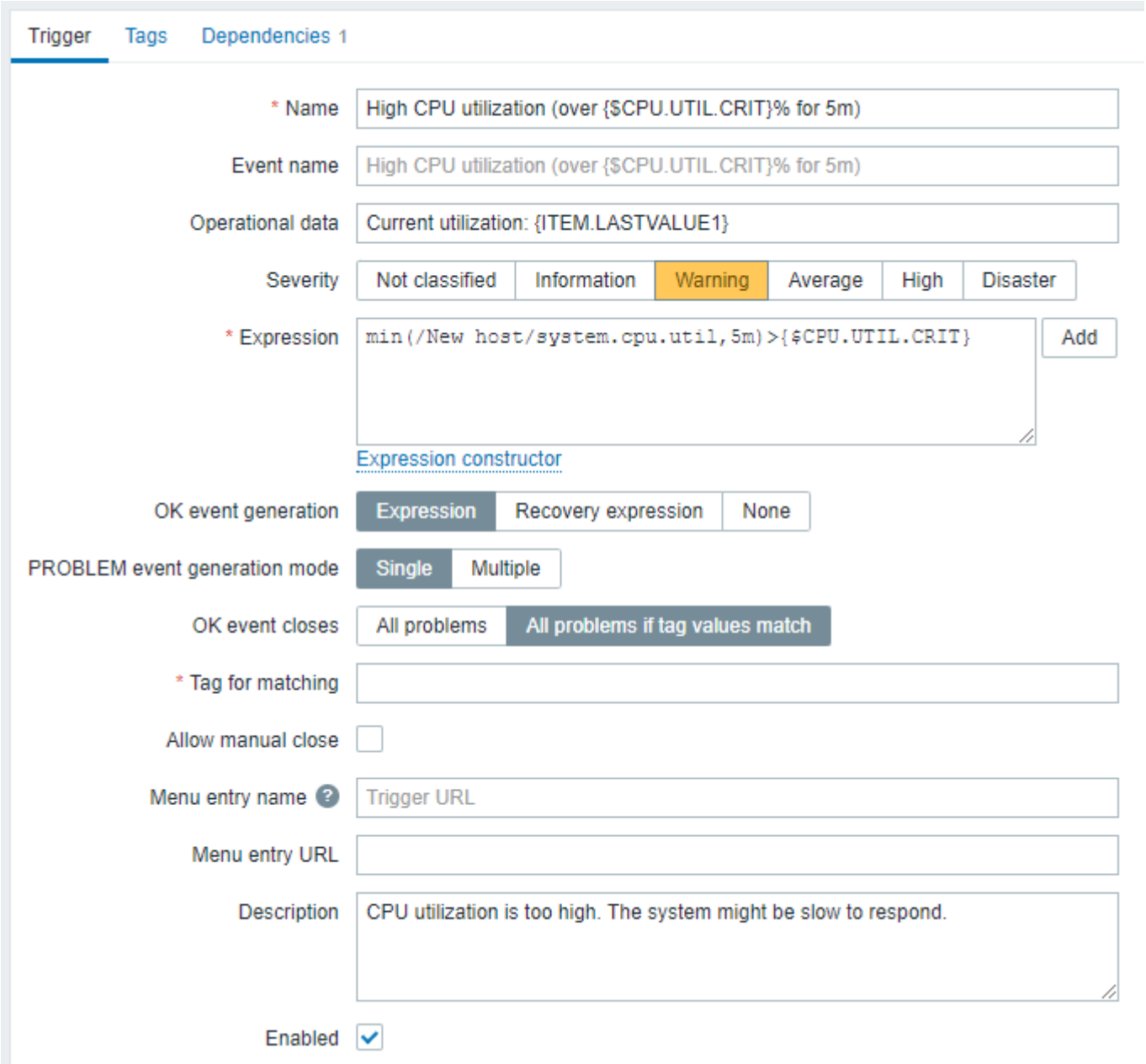
To configure a trigger, do the following:

- Go to: *Data collection → Hosts*
- Click on *Triggers* in the row of the host
- Click on *Create trigger* to the right (or on the trigger name to edit an existing trigger)
- Enter parameters of the trigger in the form

See also general information on triggers and their calculation times.

Configuration

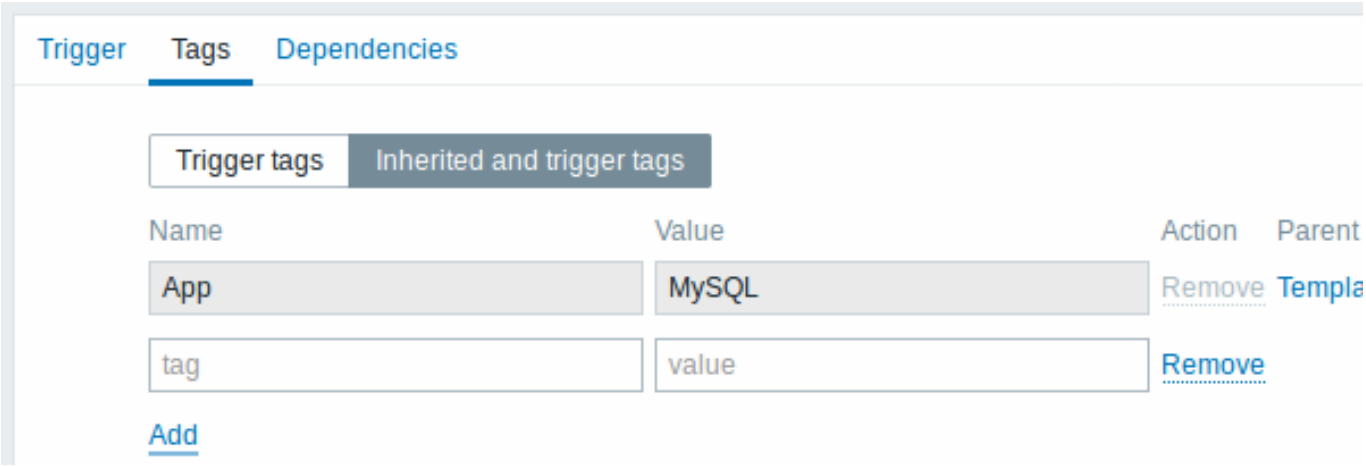
The **Trigger** tab contains all the essential trigger attributes.



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Trigger name. Supported macros are: {HOST.HOST}, {HOST.NAME}, {HOST.PORT}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {ITEM.VALUE}, {ITEM.LASTVALUE}, {ITEM.LOG.*} and {$MACRO} user macros. **$1, $2...$9** macros can be used to refer to the first, second...ninth constant of the expression. *Note*: $1-$9 macros will resolve correctly if referring to constants in relatively simple, straightforward expressions. For example, the name "Processor load above $1 on {HOST.NAME}" will automatically change to "Processor load above 5 on New host" if the expression is last(/New host/system.cpu.load[percpu,avg1])>5
Event name	If defined, this name will be used to create the problem event name, instead of the trigger name. The event name may be used to build meaningful alerts containing problem data (see example). The same set of macros is supported as in the trigger name, plus {TIME} and {?EXPRESSION} expression macros. Supported since Zabbix 5.2.0.
Operational data	Operational data allow to define arbitrary strings along with macros. The macros will resolve dynamically to real time data in *Monitoring* → *Problems*. While macros in the trigger name (see above) will resolve to their values at the moment of a problem happening and will become the basis of a static problem name, the macros in the operational data maintain the ability to display the very latest information dynamically. The same set of macros is supported as in the trigger name.
Severity	Set the required trigger severity by clicking the buttons.
Expression	Logical expression used to define the conditions of a problem. A problem is created after all the conditions included in the expression are met, i.e. the expression evaluates to TRUE. The problem will be resolved as soon as the expression evaluates to FALSE, unless additional recovery conditions are specified in *Recovery expression*.
OK event generation	OK event generation options: **Expression** - OK events are generated based on the same expression as problem events; **Recovery expression** - OK events are generated if the problem expression evaluates to FALSE and the recovery expression evaluates to TRUE; **None** - in this case the trigger will never return to an OK state on its own.
Recovery expression	Logical expression (optional) defining additional conditions that have to be met before the problem is resolved, after the original problem expression has already been evaluated as FALSE. Recovery expression is useful for trigger hysteresis. It is **not** possible to resolve a problem by recovery expression alone if the problem expression is still TRUE. This field is only available if 'Recovery expression' is selected for *OK event generation*.
PROBLEM event generation mode	Mode for generating problem events: **Single** - a single event is generated when a trigger goes into the 'Problem' state for the first time; **Multiple** - an event is generated upon *every* 'Problem' evaluation of the trigger.
OK event closes	Select if OK event closes: **All problems** - all problems of this trigger **All problems if tag values match** - only those trigger problems with matching event tag values
Tag for matching	Enter event tag name to use for event correlation. This field is displayed if 'All problems if tag values match' is selected for the *OK event closes* property and is mandatory in this case.
Allow manual close	Check to allow manual closing of problem events generated by this trigger. Manual closing is possible when acknowledging problem events.
Menu entry name	If not empty, the name entered here (up to 64 characters) is used in several frontend locations as a label for the trigger URL specified in the *Menu entry URL* parameter. If empty, the default name *Trigger URL* is used. The same set of macros is supported as in the trigger URL.
Menu entry URL	If not empty, the URL entered here (up to 2048 characters) is available as a link in the event menu in several frontend locations, for example, when clicking on the problem name in *Monitoring* → *Problems* or *Problems* dashboard widget. The same set of macros is supported as in the trigger name, plus {EVENT.ID}, {HOST.ID} and {TRIGGER.ID}. Note: user macros with secret values will not be resolved in the URL.
Description	Text field used to provide more information about this trigger. May contain instructions for fixing specific problem, contact detail of responsible staff, etc. The same set of macros is supported as in the trigger name.
Enabled	Unchecking this box will disable the trigger if required. Problems of a disabled trigger are no longer displayed in the frontend, but are not deleted.

The **Tags** tab allows you to define trigger-level tags. All problems of this trigger will be tagged with the values entered here.



In addition the *Inherited and trigger tags* option allows to view tags defined on template level, if the trigger comes from that template. If there are multiple templates with the same tag, these tags are displayed once and template names are separated with commas. A trigger does not "inherit" and display host-level tags.

Parameter	Description
Name/Value	Set custom tags to mark trigger events. Tags are a pair of tag name and value. You can use only the name or pair it with a value. A trigger may have several tags with the same name, but different values. User macros, user macros with context, low-level discovery macros and macro functions with `{{ITEM.VALUE}}`, `{{ITEM.LASTVALUE}}` and low-level discovery macros are supported in event tags. Low-level discovery macros can be used inside macro context. {TRIGGER.ID} macro is supported in trigger tag values. It may be useful for identifying triggers created from trigger prototypes and, for example, suppressing problems from these triggers during maintenance. If the total length of expanded value exceeds 255, it will be cut to 255 characters. See all macros supported for event tags. Event tags can be used for event correlation, in action conditions and will also be seen in *Monitoring* → *Problems* or the *Problems* widget.

The **Dependencies** tab contains all the dependencies of the trigger.

Click on *Add* to add a new dependency.

Note:
You can also configure a trigger by opening an existing one, pressing the *Clone* button and then saving under a different name.

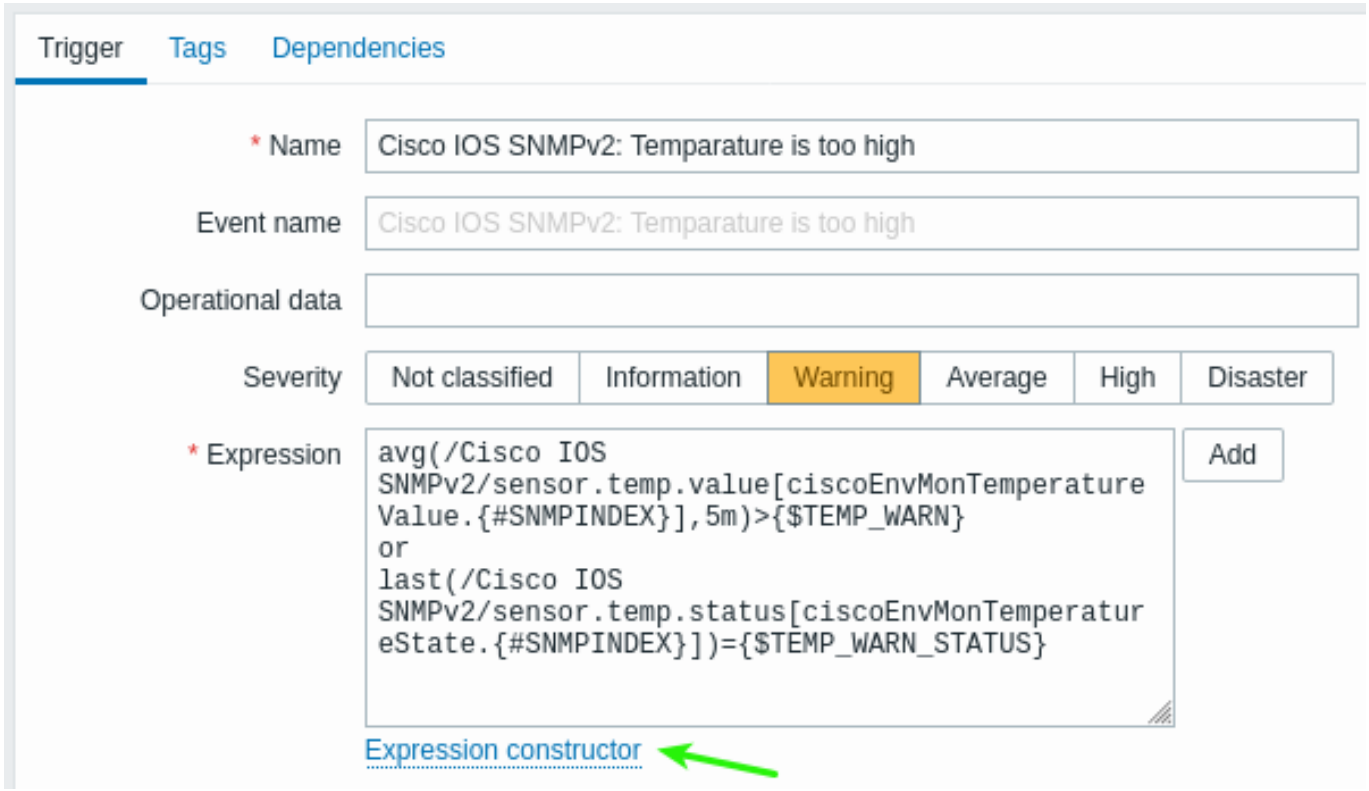
Testing expressions

It is possible to test the configured trigger expression as to what the expression result would be depending on the received value.

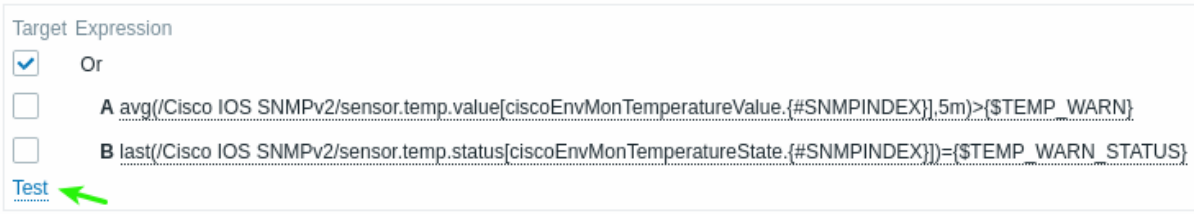
The following expression from an official template is taken as an example:

```
avg(/Cisco IOS SNMPv2/sensor.temp.value[ciscoEnvMonTemperatureValue.{#SNMPINDEX}],5m)>{$TEMP_WARN}
or
last(/Cisco IOS SNMPv2/sensor.temp.status[ciscoEnvMonTemperatureState.{#SNMPINDEX}])={$TEMP_WARN_STATUS}
```

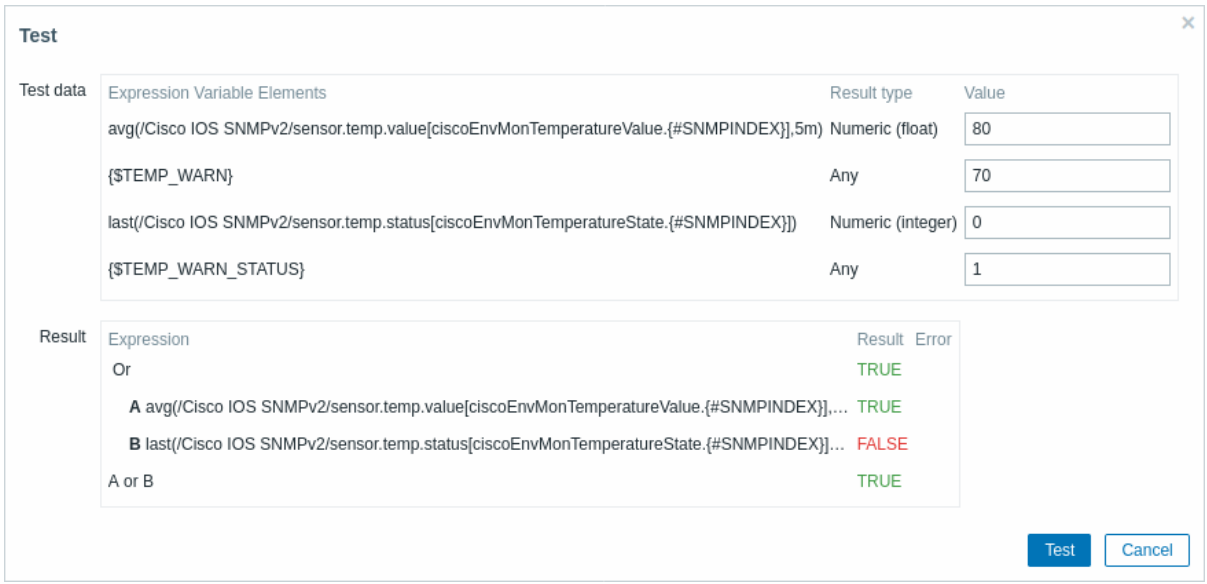
To test the expression, click on *Expression constructor* under the expression field.



In the Expression constructor, all individual expressions are listed. To open the testing window, click on *Test* below the expression list.



In the testing window you can enter sample values (’80’, ’70’, ’0’, ’1’ in this example) and then see the expression result, by clicking on the *Test* button.



The result of the individual expressions as well as the whole expression can be seen.

”TRUE” means that the specified expression is correct. In this particular case A, ”80” is greater than the {$TEMP_WARN} specified value, ”70” in this example. As expected, a ”TRUE” result appears.

”FALSE” means that the specified expression is incorrect. In this particular case B, {$TEMP_WARN_STATUS} ”1” needs to be equal with specified value, ”0” in this example. As expected, a ”FALSE” result appears.

The chosen expression type is "OR". If at least one of the specified conditions (A or B in this case) is TRUE, the overall result will be TRUE as well. Meaning that the current value exceeds the warning value and a problem has occurred.

2 Trigger expression

Overview

The expressions used in triggers are very flexible. You can use them to create complex logical tests regarding monitored statistics.

A simple expression uses a **function** that is applied to the item with some parameters. The function returns a result that is compared to the threshold, using an operator and a constant.

The syntax of a simple useful expression is `function(/host/key,parameter)<operator><constant>`.

For example:

```
min(/Zabbix server/net.if.in[eth0,bytes],5m)>100K
```

will trigger if the number of received bytes during the last five minutes was always over 100 kilobytes.

While the syntax is exactly the same, from the functional point of view there are two types of trigger expressions:

- problem expression - defines the conditions of the problem
- recovery expression (optional) - defines additional conditions of the problem resolution

When defining a problem expression alone, this expression will be used both as the problem threshold and the problem recovery threshold. As soon as the problem expression evaluates to TRUE, there is a problem. As soon as the problem expression evaluates to FALSE, the problem is resolved.

When defining both problem expression and the supplemental recovery expression, problem resolution becomes more complex: not only the problem expression has to be FALSE, but also the recovery expression has to be TRUE. This is useful to create hysteresis and avoid trigger flapping.

Functions

Functions allow to calculate the collected values (average, minimum, maximum, sum), find strings, reference current time and other factors.

A complete list of supported functions is available.

Typically functions return numeric values for comparison. When returning strings, comparison is possible with the **=** and **<>** operators (see example).

Function parameters

Function parameters allow to specify:

- host and item key (functions referencing the host item history only)
- function-specific parameters
- other expressions (not available for functions referencing the host item history, see other expressions for examples)

The host and item key can be specified as `/host/key`. The referenced item must be in a supported state (except for **nodata()** function, which is calculated for unsupported items as well).

While other trigger expressions as function parameters are limited to non-history functions in triggers, this limitation does not apply in calculated items.

Function-specific parameters

Function-specific parameters are placed after the item key and are separated from the item key by a comma. See the supported functions for a complete list of these parameters.

Most of numeric functions accept time as a parameter. You may use seconds or time suffixes to indicate time. Preceded by a hash mark, the parameter has a different meaning:

Expression	Description
sum(/host/key,**10m)**	Sum of values in the last 10 minutes.
sum(/host/key,**#10)**	Sum of the last ten values.

Parameters with a hash mark have a different meaning with the function **last** - they denote the Nth previous value, so given the values 3, 7, 2, 6, 5 (from the most recent to the least recent):

- `last(/host/key,#2)` would return '7'
- `last(/host/key,#5)` would return '5'

Time shift

An optional time shift is supported with time or value count as the function parameter. This parameter allows to reference data from a period of time in the past.

Time shift starts with `now` - specifying the current time, and is followed by `+N<time unit>` or `-N<time unit>` - to add or subtract N time units.

For example, `avg(/host/key,1h:now-1d)` will return the average value for an hour one day ago.

Attention:
Time shift specified in months (M) and years (y) is only supported for trend functions. Other functions support seconds (s), minutes (m), hours (h), days (d), and weeks (w).

Time shift with absolute time periods

Absolute time periods are supported in the time shift parameter, for example, midnight to midnight for a day, Monday-Sunday for a week, first day-last day of the month for a month.

Time shift for absolute time periods starts with `now` - specifying the current time, and is followed by any number of time operations: `/<time unit>` - defines the beginning and end of the time unit, for example, midnight to midnight for a day, `+N<time unit>` or `-N<time unit>` - to add or subtract N time units.

Please note that the value of time shift can be greater or equal to 0, while the time period minimum value is 1.

Parameter	Description
1d:now/d	Yesterday
1d:now/d+1d	Today
2d:now/d+1d	Last 2 days
1w:now/w	Last week
1w:now/w+1w	This week

Other expressions

Function parameters may contain other expressions, as in the following syntax:

`min(min(/host/key,1h),min(/host2/key2,1h)*10)`

Note that other expressions may not be used, if the function references item history. For example, the following syntax is not allowed:

~~`min(/host/key,#5*10)`~~

Operators

The following operators are supported for triggers **(in descending priority of execution)**:

Priority	Operator	Definition	**Notes for unknown values**	Force cast operand to float [1]
1	**-**	Unary minus	**-**Unknown → Unknown	Yes
2	**not**	Logical NOT	**not** Unknown → Unknown	Yes
3	*	Multiplication	0 * Unknown → Unknown (yes, Unknown, not 0 - to not lose Unknown in arithmetic operations) 1.2 * Unknown → Unknown	Yes
	/	Division	Unknown / 0 → error Unknown / 1.2 → Unknown 0.0 / Unknown → Unknown	Yes
4	+	Arithmetical plus	1.2 + Unknown → Unknown	Yes
	-	Arithmetical minus	1.2 - Unknown → Unknown	Yes

Priority	Operator	Definition	**Notes for unknown values**	Force cast operand to float [1]
5	**<**	Less than. The operator is defined as: A<B ⇔ (A<B-0.000001)	1.2 **<** Unknown → Unknown	Yes
	<=	Less than or equal to. The operator is defined as: A<=B ⇔ (A≤B+0.000001)	Unknown **<=** Unknown → Unknown	Yes
	>	More than. The operator is defined as: A>B ⇔ (A>B+0.000001)		Yes
	>=	More than or equal to. The operator is defined as: A>=B ⇔ (A≥B-0.000001)		Yes
6	**=**	Is equal. The operator is defined as: A=B ⇔ (A≥B-0.000001) and (A≤B+0.000001)		No [1]
	<>	Not equal. The operator is defined as: A<>B ⇔ (A<B-0.000001) or (A>B+0.000001)		No [1]
7	**and**	Logical AND	0 **and** Unknown → 0 1 **and** Unknown → Unknown Unknown **and** Unknown → Unknown	Yes
8	**or**	Logical OR	1 **or** Unknown → 1 0 **or** Unknown → Unknown Unknown **or** Unknown → Unknown	Yes

[1] String operand is still cast to numeric if:

- another operand is numeric
- operator other than **=** or **<>** is used on an operand

(If the cast fails - numeric operand is cast to a string operand and both operands get compared as strings.)

not, **and** and **or** operators are case-sensitive and must be in lowercase. They also must be surrounded by spaces or parentheses.

All operators, except unary **-** and **not**, have left-to-right associativity. Unary **-** and **not** are non-associative (meaning **-(-1)** and **not (not 1)** should be used instead of **--1** and **not not 1**).

Evaluation result:

- **<**, **<=**, **>**, **>=**, **=**, **<>** operators shall yield '1' in the trigger expression if the specified relation is true and '0' if it is false. If at least one operand is Unknown the result is Unknown;
- **and** for known operands shall yield '1' if both of its operands compare unequal to '0'; otherwise, it yields '0'; for unknown operands **and** yields '0' only if one operand compares equal to '0'; otherwise, it yields 'Unknown';
- **or** for known operands shall yield '1' if either of its operands compare unequal to '0'; otherwise, it yields '0'; for unknown operands **or** yields '1' only if one operand compares unequal to '0'; otherwise, it yields 'Unknown';
- The result of the logical negation operator **not** for a known operand is '0' if the value of its operand compares unequal to '0'; '1' if the value of its operand compares equal to '0'. For unknown operand **not** yields 'Unknown'.

Value caching

Values required for trigger evaluation are cached by Zabbix server. Because of this trigger evaluation causes a higher database load for some time after the server restarts. The value cache is not cleared when item history values are removed (either manually or by housekeeper), so the server will use the cached values until they are older than the time periods defined in trigger functions or server is restarted.

Note:
If there is no recent data in the cache and there is no defined querying period in the function, Zabbix will by default go as far back in the past as one week to query the database for historical values.

Examples of triggers

Example 1

The processor load is too high on Zabbix server.

```
last(/Zabbix server/system.cpu.load[all,avg1])>5
```

By using the function 'last()', we are referencing the most recent value. `/Zabbix server/system.cpu.load[all,avg1]` gives a short name of the monitored parameter. It specifies that the host is 'Zabbix server' and the key being monitored is 'system.cpu.load[all,avg1]'. Finally, >5 means that the trigger is in the PROBLEM state whenever the most recent processor load measurement from Zabbix server is greater than 5.

Example 2

www.example.com is overloaded.

```
last(/www.example.com/system.cpu.load[all,avg1])>5 or min(/www.example.com/system.cpu.load[all,avg1],10m)>
```

The expression is true when either the current processor load is more than 5 or the processor load was more than 2 during last 10 minutes.

Example 3

/etc/passwd has been changed.

```
last(/www.example.com/vfs.file.cksum[/etc/passwd],#1)<>last(/www.example.com/vfs.file.cksum[/etc/passwd],#
```

The expression is true when the previous value of /etc/passwd checksum differs from the most recent one.

Similar expressions could be useful to monitor changes in important files, such as /etc/passwd, /etc/inetd.conf, /kernel, etc.

Example 4

Someone is downloading a large file from the Internet.

Use of function min:

```
min(/www.example.com/net.if.in[eth0,bytes],5m)>100K
```

The expression is true when number of received bytes on eth0 is more than 100 KB within last 5 minutes.

Example 5

Both nodes of clustered SMTP server are down.

Note use of two different hosts in one expression:

```
last(/smtp1.example.com/net.tcp.service[smtp])=0 and last(/smtp2.example.com/net.tcp.service[smtp])=0
```

The expression is true when both SMTP servers are down on both smtp1.example.com and smtp2.example.com.

Example 6

Zabbix agent needs to be upgraded.

Use of function find():

```
find(/example.example.com/agent.version,,"like","beta8")=1
```

The expression is true if Zabbix agent has version beta8.

Example 7

Server is unreachable.

```
count(/example.example.com/icmpping,30m,,"0")>5
```

The expression is true if host ”example.example.com” is unreachable more than 5 times in the last 30 minutes.

Example 8

No heartbeats within last 3 minutes.

Use of function nodata():

```
nodata(/example.example.com/tick,3m)=1
```

To make use of this trigger, 'tick' must be defined as a Zabbix trapper item. The host should periodically send data for this item using zabbix_sender. If no data is received within 180 seconds, the trigger value becomes PROBLEM.

Note that 'nodata' can be used for any item type.

Example 9

CPU activity at night time.

Use of function time():

```
min(/Zabbix server/system.cpu.load[all,avg1],5m)>2 and time()<060000
```

The trigger may change its state to problem only at night time (00:00 - 06:00).

Example 10

CPU activity at any time with exception.

Use of function time() and **not** operator:

```
min(/zabbix/system.cpu.load[all,avg1],5m)>2
and not (dayofweek()=7 and time()>230000)
and not (dayofweek()=1 and time()<010000)
```

The trigger may change its state to problem at any time, except for 2 hours on a week change (Sunday, 23:00 - Monday, 01:00).

Example 11

Check if client local time is in sync with Zabbix server time.

Use of function fuzzytime():

```
fuzzytime(/MySQL_DB/system.localtime,10s)=0
```

The trigger will change to the problem state in case when local time on server MySQL_DB and Zabbix server differs by more than 10 seconds. Note that 'system.localtime' must be configured as a passive check.

Example 12

Comparing average load today with average load of the same time yesterday (using time shift as `now-1d`).

```
avg(/server/system.cpu.load,1h)/avg(/server/system.cpu.load,1h:now-1d)>2
```

This trigger will fire if the average load of the last hour tops the average load of the same hour yesterday more than two times.

Example 13

Using the value of another item to get a trigger threshold:

```
last(/Template PfSense/hrStorageFree[{#SNMPVALUE}])<last(/Template PfSense/hrStorageSize[{#SNMPVALUE}])*0.
```

The trigger will fire if the free storage drops below 10 percent.

Example 14

Using evaluation result to get the number of triggers over a threshold:

```
(last(/server1/system.cpu.load[all,avg1])>5) + (last(/server2/system.cpu.load[all,avg1])>5) + (last(/serve
```

The trigger will fire if at least two of the triggers in the expression are in a problem state.

Example 15

Comparing string values of two items - operands here are functions that return strings.

Problem: create an alert if Ubuntu version is different on different hosts

```
last(/NY Zabbix server/vfs.file.contents[/etc/os-release])<>last(/LA Zabbix server/vfs.file.contents[/etc/
```

Example 16

Comparing two string values - operands are:

- a function that returns a string
- a combination of macros and strings

Problem: detect changes in the DNS query

The item key is:

```
net.dns.record[8.8.8.8,{$WEBSITE_NAME},{$DNS_RESOURCE_RECORD_TYPE},2,1]
```

with macros defined as

```
{$WEBSITE_NAME} = example.com
{$DNS_RESOURCE_RECORD_TYPE} = MX
```

and normally returns:

```
example.com             MX       0 mail.example.com
```

So our trigger expression to detect if the DNS query result deviated from the expected result is:

```
last(/Zabbix server/net.dns.record[8.8.8.8,{$WEBSITE_NAME},{$DNS_RESOURCE_RECORD_TYPE},2,1])<>"{$WEBSITE_N
```

Notice the quotes around the second operand.

Example 17

Comparing two string values - operands are:

- a function that returns a string
- a string constant with special characters \ and "

Problem: detect if the `/tmp/hello` file content is equal to:

```
\" //hello ?\"
```

Option 1) write the string directly

```
last(/Zabbix server/vfs.file.contents[/tmp/hello])="\\\" //hello ?\\\""
```

Notice how \ and " characters are escaped when the string gets compared directly.

Option 2) use a macro

```
{$HELLO_MACRO} = \" //hello ?\"
```

in the expression:

```
last(/Zabbix server/vfs.file.contents[/tmp/hello])={$HELLO_MACRO}
```

Example 18

Comparing long-term periods.

Problem: Load of Exchange server increased by more than 10% last month

```
trendavg(/Exchange/system.cpu.load,1M:now/M)>1.1*trendavg(/Exchange/system.cpu.load,1M:now/M-1M)
```

You may also use the Event name field in trigger configuration to build a meaningful alert message, for example to receive something like

```
"Load of Exchange server increased by 24% in July (0.69) comparing to June (0.56)"
```

the event name must be defined as:

```
Load of {HOST.HOST} server increased by {{?100*trendavg(//system.cpu.load,1M:now/M)/trendavg(//system.cpu.
```

It is also useful to allow manual closing in trigger configuration for this kind of problem.

Hysteresis

Sometimes an interval is needed between problem and recovery states, rather than a simple threshold. For example, if we want to define a trigger that reports a problem when server room temperature goes above 20°C and we want it to stay in the problem state until the temperature drops below 15°C, a simple trigger threshold at 20°C will not be enough.

Instead, we need to define a trigger expression for the problem event first (temperature above 20°C). Then we need to define an additional recovery condition (temperature below 15°C). This is done by defining an additional *Recovery expression* parameter when defining a trigger.

In this case, problem recovery will take place in two steps:

- First, the problem expression (temperature above 20°C) will have to evaluate to FALSE
- Second, the recovery expression (temperature below 15°C) will have to evaluate to TRUE

The recovery expression will be evaluated only when the problem event is resolved first.

Warning:
The recovery expression being TRUE alone does not resolve a problem if the problem expression is still TRUE!

Example 1

Temperature in server room is too high.

Problem expression:

`last(/server/temp)>20`

Recovery expression:

`last(/server/temp)<=15`

Example 2

Free disk space is too low.

Problem expression: it is less than 10GB for last 5 minutes

`max(/server/vfs.fs.size[/,free],5m)<10G`

Recovery expression: it is more than 40GB for last 10 minutes

`min(/server/vfs.fs.size[/,free],10m)>40G`

Expressions with unknown operands

Generally an unknown operand (such as an unsupported item) in the expression will immediately render the trigger value to `Unknown`.

However, in some cases unknown operands (unsupported items, function errors) are admitted into expression evaluation:

- The `nodata()` function is evaluated regardless of whether the referenced item is supported or not.
- Logical expressions with OR and AND can be evaluated to known values in two cases regardless of unknown operands:
  - **Case 1**: "`1 or some_function(unsupported_item1) or some_function(unsupported_item2) or ...`" can be evaluated to known result ('1' or "Problem"),
  - **Case 2**: "`0 and some_function(unsupported_item1) and some_function(unsupported_item2) and ...`" can be evaluated to known result ('0' or "OK").
    Zabbix tries to evaluate such logical expressions by taking unsupported items as unknown operands. In the two cases above a known value will be produced ("Problem" or "OK", respectively); in **all other** cases the trigger will evaluate to `Unknown`.
- If the function evaluation for a supported item results in error, the function value becomes `Unknown` and it takes part as unknown operand in further expression evaluation.

Note that unknown operands may "disappear" only in logical expressions as described above. In arithmetic expressions unknown operands always lead to the result `Unknown` (except division by 0).

Attention:
An expression that results in `Unknown` does not change the trigger state ("Problem/OK"). So, if it was "Problem" (see Case 1), it stays in the same problem state even if the known part is resolved ('1' becomes '0'), because the expression is now evaluated to `Unknown` and that does not change the trigger state.

If a trigger expression with several unsupported items evaluates to `Unknown` the error message in the frontend refers to the last unsupported item evaluated.

3 Trigger dependencies

Overview

Sometimes the availability of one host depends on another. A server that is behind a router will become unreachable if the router goes down. With triggers configured for both, you might get notifications about two hosts down - while only the router was the guilty party.

This is where some dependency between hosts might be useful. With dependency set, notifications of the dependents could be withheld and only the notification on the root problem sent.

While Zabbix does not support dependencies between hosts directly, they may be defined with another, more flexible method - trigger dependencies. A trigger may have one or more triggers it depends on.

So in our simple example we open the server trigger configuration form and set that it depends on the respective trigger of the router. With such dependency, the server trigger will not change its state as long as the trigger it depends on is in the 'PROBLEM' state - and thus no dependent actions will be taken and no notifications sent.

If both the server and the router are down and dependency is there, Zabbix will not execute actions for the dependent trigger.

While the parent trigger is in the PROBLEM state, its dependents may report values that cannot be trusted. Therefore dependent triggers will not be re-evaluated until the parent trigger (the router in the example above):

- goes back from 'PROBLEM' to 'OK' state;
- changes its state from 'PROBLEM' to 'UNKNOWN';
- is closed manually, by correlation or with the help of time-based functions;
- is resolved by a value of an item not involved in the dependent trigger;
- is disabled, has a disabled item or a disabled item host

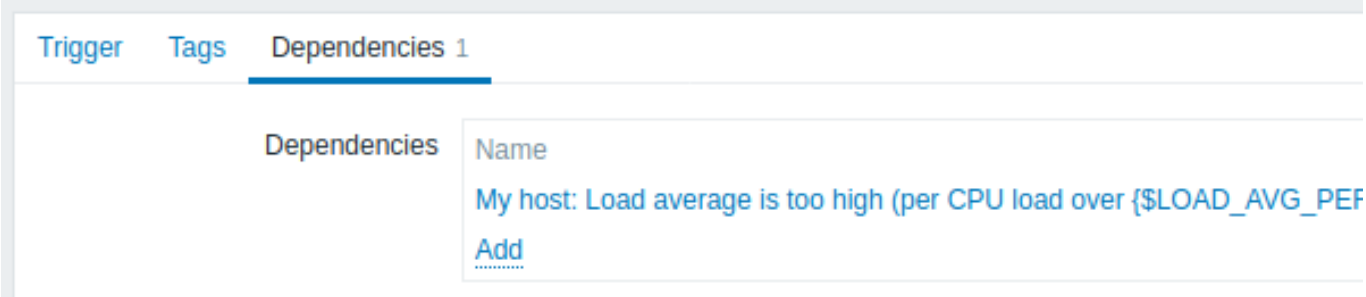
In all of the cases mentioned above, the dependent trigger (server) will be re-evaluated only when a new metric for it is received. This means that the dependent trigger may not be updated immediately.

Also:

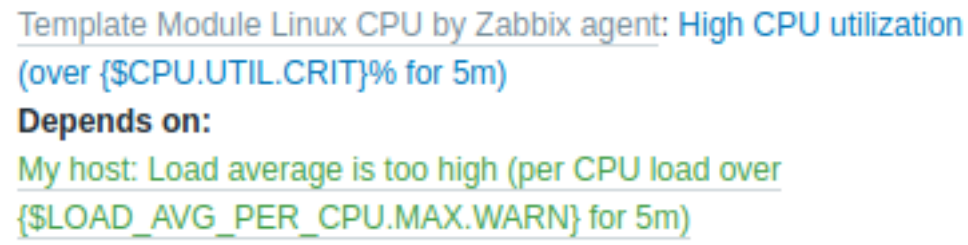
- Trigger dependency may be added from any host trigger to any other host trigger, as long as it doesn't result in a circular dependency.
- Trigger dependency may be added from one template to another. If some trigger from template A depends on some trigger from template B, template A may only be linked to a host (or another template) together with template B, but template B may be linked to a host (or another template) alone.
- Trigger dependency may be added from a template trigger to a host trigger. In this case, linking such a template to a host will create a host trigger that depends on the same trigger template that the trigger was depending on. This allows to, for example, have a template where some triggers depend on the router (host) triggers. All hosts linked to this template will depend on that specific router.
- Trigger dependency may not be added from a host trigger to a template trigger.
- Trigger dependency may be added from a trigger prototype to another trigger prototype (within the same low-level discovery rule) or a real trigger. A trigger prototype may not depend on a trigger prototype from a different LLD rule or on a trigger created from trigger prototype. A host trigger prototype cannot depend on a trigger from a template.

Configuration

To define a dependency, open the Dependencies tab in the trigger configuration form. Click on *Add* in the 'Dependencies' block and select one or more triggers that the trigger will depend on.



Click *Update*. Now the trigger has the indication of its dependency in the list.



Example of several dependencies

For example, the Host is behind the Router2 and the Router2 is behind the Router1.

```
Zabbix - Router1 - Router2 - Host
```

If the Router1 is down, then obviously the Host and the Router2 are also unreachable, yet receiving three notifications about the Host, the Router1 and the Router2 all being down is excessive.

So in this case we define two dependencies:

```
the 'Host is down' trigger depends on the 'Router2 is down' trigger
the 'Router2 is down' trigger depends on the 'Router1 is down' trigger
```

Before changing the status of the 'Host is down' trigger, Zabbix will check for the corresponding trigger dependencies. If such are found and one of those triggers is in the 'Problem' state, then the trigger status will not be changed, the actions will not be executed and no notifications will be sent.

Zabbix performs this check recursively. If the Router1 or the Router2 is unreachable, the Host trigger won't be updated.

4 Trigger severity

Trigger severity represents the level of importance of a trigger.



Zabbix supports the following default trigger severities.

Severity	Color	Description
Not classified	Gray	Can be used where the severity level of an event is unknown, has not been determined, is not part of the regular monitoring scope, etc., for example, during initial configuration, as a placeholder for future assessment, or as part of an integration process.
Information	Light blue	Can be used for informational events that do not require immediate attention, but can still provide valuable insights.
Warning	Yellow	Can be used to indicate a potential issue that might require investigation or action, but that is not critical.
Average	Orange	Can be used to indicate a significant issue that should be addressed relatively soon to prevent further problems.
High	Light red	Can be used to indicate critical issues that need immediate attention to avoid significant disruptions.
Disaster	Red	Can be used to indicate a severe incident that requires immediate action to prevent, for example, system outages or data loss.

Note:
Trigger severity names and colors can be customized.

Trigger severities are used for:

- visual representation of triggers - different colors for different severities;
- audio in global alarms - different audio for different severities;
- user media - different media (notification channel) for different severities (for example, SMS for triggers of *High* and *Disaster* severity, and Email for triggers of other severities);
- limiting actions by conditions against trigger severities.

5 Customizing trigger severities

Trigger severity names and colors for severity related GUI elements can be configured in *Administration* → *General* → *Trigger displaying options*. Colors are shared among all GUI themes.

Translating customized severity names

> **Attention:**
> If Zabbix frontend translations are used, custom severity names will override translated names by default.

Default trigger severity names are available for translation in all locales. If a severity name is changed, a custom name is used in all locales and additional manual translation is needed.

Custom severity name translation procedure:

- set required custom severity name, for example, 'Important'
- edit <frontend_dir>/locale/<required_locale>/LC_MESSAGES/frontend.po
- add 2 lines:

```
msgid "Important"
msgstr "<translation string>"
```

and save file.

- create .mo files as described in <frontend_dir>/locale/README

Here **msgid** should match the new custom severity name and **msgstr** should be the translation for it in the specific language.

This procedure should be performed after each severity name change.

6 Mass update

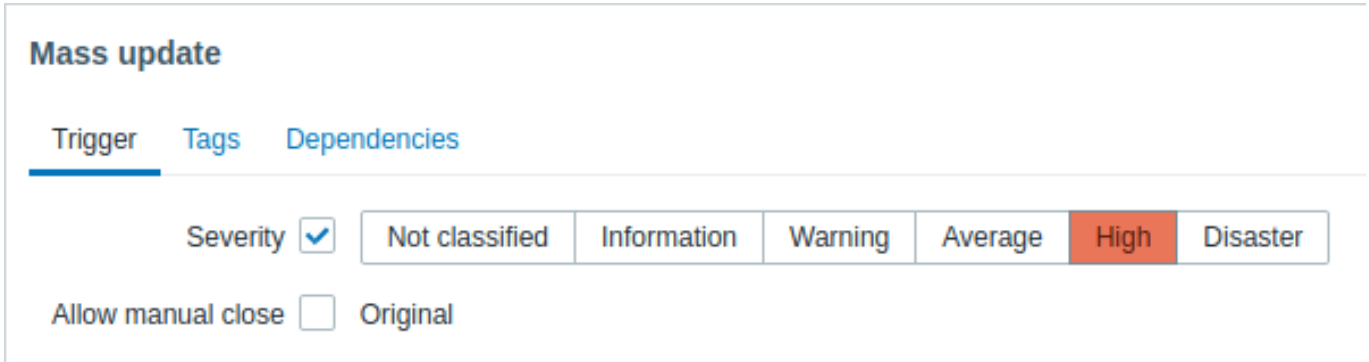
Overview

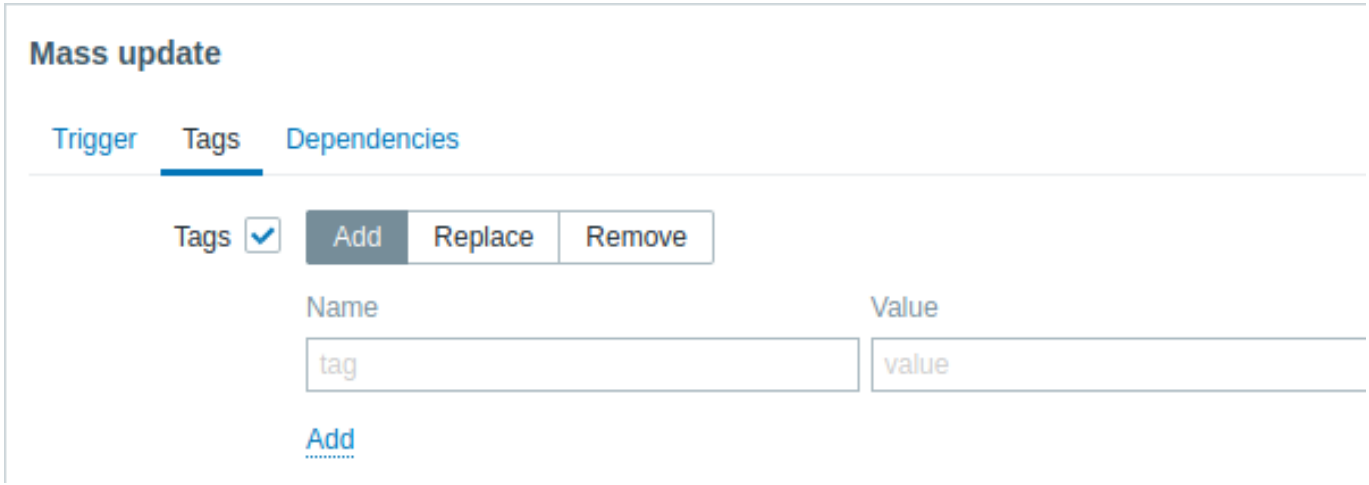
With mass update you may change some attribute for a number of triggers at once, saving you the need to open each individual trigger for editing.

Using mass update

To mass-update some triggers, do the following:

- Mark the checkboxes of the triggers you want to update in the list
- Click on *Mass update* below the list
- Navigate to the tab with required attributes (*Trigger*, *Tags* or *Dependencies*)
- Mark the checkboxes of any attribute to update

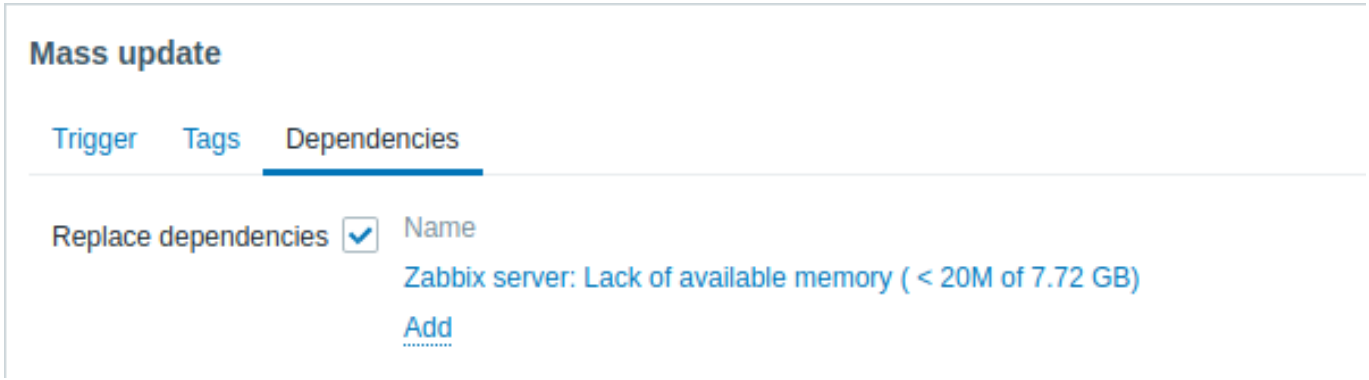




The following options are available when selecting the respective button for tag update:

- *Add* - allows to add new tags for the triggers;
- *Replace* - will remove any existing tags from the trigger and replace them with the one(s) specified below;
- *Remove* - will remove specified tags from triggers.

Note that tags with the same name but different values are not considered 'duplicates' and can be added to the same trigger.



Replace dependencies - will remove any existing dependencies from the trigger and replace them with the one(s) specified.

Click on *Update* to apply the changes.

7 Predictive trigger functions

Overview

Sometimes there are signs of the upcoming problem. These signs can be spotted so that actions may be taken in advance to prevent or at least minimize the impact of the problem.

Zabbix has tools to predict the future behavior of the monitored system based on historic data. These tools are realized through predictive trigger functions.

Functions

Before setting a trigger, it is necessary to define what a problem state is and how much time is needed to take action. Then there are two ways to set up a trigger signaling about a potential unwanted situation. First: the trigger must fire when the system is expected to be in a problem state after the "time to act". Second: the trigger must fire when the system is going to reach the problem state in less than "time to act". Corresponding trigger functions to use are **forecast** and **timeleft**. Note that underlying statistical analysis is basically identical for both functions. You may set up a trigger whichever way you prefer with similar results.

Parameters

Both functions use almost the same set of parameters. Use the list of supported functions for reference.

Time interval

First of all, you should specify the historic period Zabbix should analyze to come up with the prediction. You do it in a familiar way by means of the `time period` parameter and optional time shift like you do it with **avg**, **count**, **delta**, **max**, **min** and **sum** functions.

Forecasting horizon

(**forecast** only)
Parameter `time` specifies how far in the future Zabbix should extrapolate dependencies it finds in historic data. No matter if you use `time_shift` or not, `time` is always counted starting from the current moment.

Threshold to reach

(**timeleft** only)
Parameter `threshold` specifies a value the analyzed item has to reach, no difference if from above or from below. Once we have determined f(t) (see below), we should solve equation f(t) = `threshold` and return the root which is closer to now and to the right from now or 999999999999.9999 if there is no such root.

Note:
When item values approach the threshold and then cross it, **timeleft** assumes that intersection is already in the past and therefore switches to the next intersection with `threshold` level, if any. Best practice should be to use predictions as a complement to ordinary problem diagnostics, not as a substitution.[1]

Fit functions

Default `fit` is the *linear* function. But if your monitored system is more complicated you have more options to choose from.

`fit`	x = f(t)
linear	$x = a + b*t$
polynomialN[2]	$x = a_0 + a_1*t + a_2*t^2 + ... + a_n*t^n$
exponential	$x = a*exp(b*t)$
logarithmic	$x = a + b*log(t)$
power	$x = a*t^b$

Modes

(**forecast** only)
Every time a trigger function is evaluated, it gets data from the specified history period and fits a specified function to the data. So, if the data is slightly different, the fitted function will be slightly different. If we simply calculate the value of the fitted function at a specified time in the future, you will know nothing about how the analyzed item is expected to behave between now and that moment in the future. For some `fit` options (like *polynomial*) a simple value from the future may be misleading.

`mode`	**forecast** result
value	f(now + `time`)
max	$max_{now <= t <= now + time}$ f(t)
min	$min_{now <= t <= now + time}$ f(t)
delta	*max* - *min*
avg	average of f(t) (now <= t <= now + `time`) according to definition

Details

To avoid calculations with huge numbers, we consider the timestamp of the first value in specified period plus 1 ns as a new zero-time (current epoch time is of order 10^9, epoch squared is 10^{18}, double precision is about 10^{-16}). 1 ns is added to provide all positive time values for *logarithmic* and *power* fits which involve calculating log(t). Time shift does not affect *linear*, *polynomial*, *exponential* (apart from easier and more precise calculations) but changes the shape of *logarithmic* and *power* functions.

Potential errors

Functions return -1 in such situations:

- specified evaluation period contains no data;
- result of mathematical operation is not defined[^3];
- numerical complications (unfortunately, for some sets of input data range and precision of double-precision floating-point format become insufficient)[4].

Note:
No warnings or errors are flagged if chosen fit poorly describes provided data or there is just too few data for accurate prediction.

Examples and dealing with errors

To get a warning when you are about to run out of free disk space on your host, you may use a trigger expression like this:

`timeleft(/host/vfs.fs.size[/,free],1h,0)}<1h`

However, error code -1 may come into play and put your trigger in a problem state. Generally it's good because you get a warning that your predictions don't work correctly and you should look at them more thoroughly to find out why. But sometimes it's bad because -1 can simply mean that there was no data about the host free disk space obtained in the last hour. If you are getting too many false positive alerts, consider using more complicated trigger expression [5]:

`timeleft(/host/vfs.fs.size[/,free],1h,0)<1h and timeleft(/host/vfs.fs.size[/,free],1h,0)<>-1`

The situation is a bit more difficult with **forecast**. First of all, -1 may or may not put the trigger in a problem state depending on whether you have expression like `forecast(/host/item,(...))<...` or like `forecast(/host/item,(...))>...`

Furthermore, -1 may be a valid forecast if it's normal for the item value to be negative. But probability of this situation in the real world situation is negligible (see how the operator **=** works). So add `... or forecast(/host/item,(...))=-1` or `... and forecast(/host/item,(...))<>-1` if you want or don't want to treat -1 as a problem respectively.

Footnotes

[1] For example, a simple trigger like `timeleft(/host/item,1h,X) < 1h` may go into problem state when the item value approaches X and then suddenly recover once value X is reached. If the problem is item value being below X, use: `last(/host/item) < X or timeleft(/host/item,1h,X) < 1h` If the problem is item value being above X use: `last(/host/item) > X or timeleft(/host/item,1h,X) < 1h`

[2] Polynomial degree can be from 1 to 6, *polynomial1* is equivalent to *linear*. However, use higher degree polynomials with caution. If the evaluation period contains less points than needed to determine polynomial coefficients, polynomial degree will be lowered (e.g., *polynomial5* is requested, but there are only 4 points, therefore *polynomial3* will be fitted).

[3] For example, fitting *exponential* or *power* functions involves calculating log() of item values. If data contains zeros or negative numbers, you will get an error since log() is defined for positive values only.

[4] For *linear*, *exponential*, *logarithmic* and *power* fits all necessary calculations can be written explicitly. For *polynomial* only *value* can be calculated without any additional steps. Calculating *avg* involves computing polynomial antiderivative (analytically). Computing *max*, *min* and *delta* involves computing polynomial derivative (analytically) and finding its roots (numerically). Solving f(t) = 0 involves finding polynomial roots (numerically).

[5] But in this case -1 can cause your trigger to recover from the problem state. To be fully protected use: `timeleft(/host/vfs.fs.size[/,f and ({TRIGGER.VALUE}=0 and timeleft(/host/vfs.fs.size[/,free],1h,0)<>-1 or {TRIGGER.VALUE}=1)`

4 Events

Overview

There are several types of events generated in Zabbix:

- trigger events - whenever a trigger changes its status (*OK→PROBLEM→OK*)
- service events - whenever a service changes its status (*OK→PROBLEM→OK*)
- discovery events - when hosts or services are detected
- autoregistration events - when active agents are auto-registered by server
- internal events - when an item/low-level discovery rule becomes unsupported or a trigger goes into an unknown state

Note:
Internal events are supported starting with Zabbix 2.2 version.

Events are time-stamped and can be the basis of actions such as sending notification email etc.

To view details of events in the frontend, go to *Monitoring → Problems*. There you can click on the event date and time to view details of an event.

More information is available on:

- trigger events
- other event sources

1 Trigger event generation

Overview

Change of trigger status is the most frequent and most important source of events. Each time the trigger changes its state, an event is generated. The event contains details of the trigger state's change - when it happened and what the new state is.

Two types of events are created by triggers - Problem and OK.

Problem events

A problem event is created:

- when a trigger expression evaluates to TRUE if the trigger is in OK state;
- each time a trigger expression evaluates to TRUE if multiple problem event generation is enabled for the trigger.

OK events

An OK event closes the related problem event(s) and may be created by 3 components:

- triggers - based on 'OK event generation' and 'OK event closes' settings;
- event correlation
- task manager – when an event is manually closed

Triggers

Triggers have an 'OK event generation' setting that controls how OK events are generated:

- *Expression* - an OK event is generated for a trigger in problem state when its expression evaluates to FALSE. This is the simplest setting, enabled by default.
- *Recovery expression* - an OK event is generated for a trigger in problem state when its expression evaluates to FALSE and the recovery expression evaluates to TRUE. This can be used if trigger recovery criteria is different from problem criteria.
- *None* - an OK event is never generated. This can be used in conjunction with multiple problem event generation to simply send a notification when something happens.

Additionally triggers have an 'OK event closes' setting that controls which problem events are closed:

- *All problems* - an OK event will close all open problems created by the trigger
- *All problems if tag values match* - an OK event will close open problems created by the trigger and having at least one matching tag value. The tag is defined by 'Tag for matching' trigger setting. If there are no problem events to close then OK event is not generated. This is often called trigger level event correlation.

Event correlation

Event correlation (also called global event correlation) is a way to set up custom event closing (resulting in OK event generation) rules.

The rules define how the new problem events are paired with existing problem events and allow to close the new event or the matched events by generating corresponding OK events.

However, event correlation must be configured very carefully, as it can negatively affect event processing performance or, if misconfigured, close more events than intended (in the worst case even all problem events could be closed). A few configuration tips:

1. always reduce the correlation scope by setting a unique tag for the control event (the event that is paired with old events) and use the 'new event tag' correlation condition
2. don't forget to add a condition based on the old event when using 'close old event' operation, or all existing problems could be closed
3. avoid using common tag names used by different correlation configurations

Task manager

If the 'Allow manual close' setting is enabled for trigger, then it's possible to manually close problem events generated by the trigger. This is done in the frontend when updating a problem. The event is not closed directly – instead a 'close event' task is created, which is handled by the task manager shortly. The task manager will generate a corresponding OK event and the problem event will be closed.

2 Other event sources

Service events

Service events are generated only if service actions for these events are enabled. In this case, each service status change creates a new event:

- Problem event - when service status is changed from OK to PROBLEM
- OK event - when service status is changed from PROBLEM to OK

The event contains details of the service state change - when it happened and what the new state is.

Discovery events

Zabbix periodically scans the IP ranges defined in network discovery rules. Frequency of the check is configurable for each rule individually. Once a host or a service is discovered, a discovery event (or several events) are generated.

Zabbix generates the following events:

Event	When generated
Service Up	Every time Zabbix detects active service.
Service Down	Every time Zabbix cannot detect service.
Host Up	If at least one of the services is UP for the IP.
Host Down	If all services are not responding.
Service Discovered	If the service is back after downtime or discovered for the first time.
Service Lost	If the service is lost after being up.
Host Discovered	If host is back after downtime or discovered for the first time.
Host Lost	If host is lost after being up.

Active agent autoregistration events

Active agent autoregistration creates events in Zabbix.

If configured, active agent autoregistration event is created when a previously unknown active agent asks for checks or if the host metadata has changed. The server adds a new auto-registered host, using the received IP address and port of the agent.

For more information, see the active agent autoregistration page.

Internal events

Internal events happen when:

- an item changes state from 'normal' to 'unsupported'
- an item changes state from 'unsupported' to 'normal'
- a low-level discovery rule changes state from 'normal' to 'unsupported'
- a low-level discovery rule changes state from 'unsupported' to 'normal'
- a trigger changes state from 'normal' to 'unknown'
- a trigger changes state from 'unknown' to 'normal'

Internal events are supported since Zabbix 2.2. The aim of introducing internal events is to allow users to be notified when any internal event takes place, for example, an item becomes unsupported and stops gathering data.

Internal events are only created when internal actions for these events are enabled. To stop generation of internal events (for example, for items becoming unsupported), disable all actions for internal events in Alerts → Actions → Internal actions.

Note:
If internal actions are disabled, while an object is in the 'unsupported' state, recovery event for this object will still be created.

If internal actions are enabled, while an object is in the 'unsupported' state, recovery event for this object will be created, even though 'problem event' has not been created for the object.

See also: Receiving notification on unsupported items

3 Manual closing of problems

Overview

While generally problem events are resolved automatically when trigger status goes from 'Problem' to 'OK', there may be cases when it is difficult to determine if a problem has been resolved by means of a trigger expression. In such cases, the problem needs to be resolved manually.

For example, *syslog* may report that some kernel parameters need to be tuned for optimal performance. In this case the issue is reported to Linux administrators, they fix it and then close the problem manually.

Problems can be closed manually only for triggers with the *Allow manual close* option enabled.

When a problem is "manually closed", Zabbix generates a new internal task for Zabbix server. Then the *task manager* process executes this task and generates an OK event, therefore closing problem event.

A manually closed problem does not mean that the underlying trigger will never go into a 'Problem' state again. The trigger expression is re-evaluated and may result in a problem:

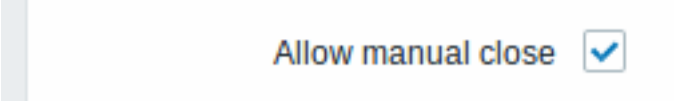
- When new data arrive for any item included in the trigger expression (note that the values discarded by a throttling preprocessing step are not considered as received and will not cause trigger expression to be re-evaluated);
- When time-based functions are used in the expression. Complete time-based function list can be found on Triggers page.

Configuration

Two steps are required to close a problem manually.

Trigger configuration

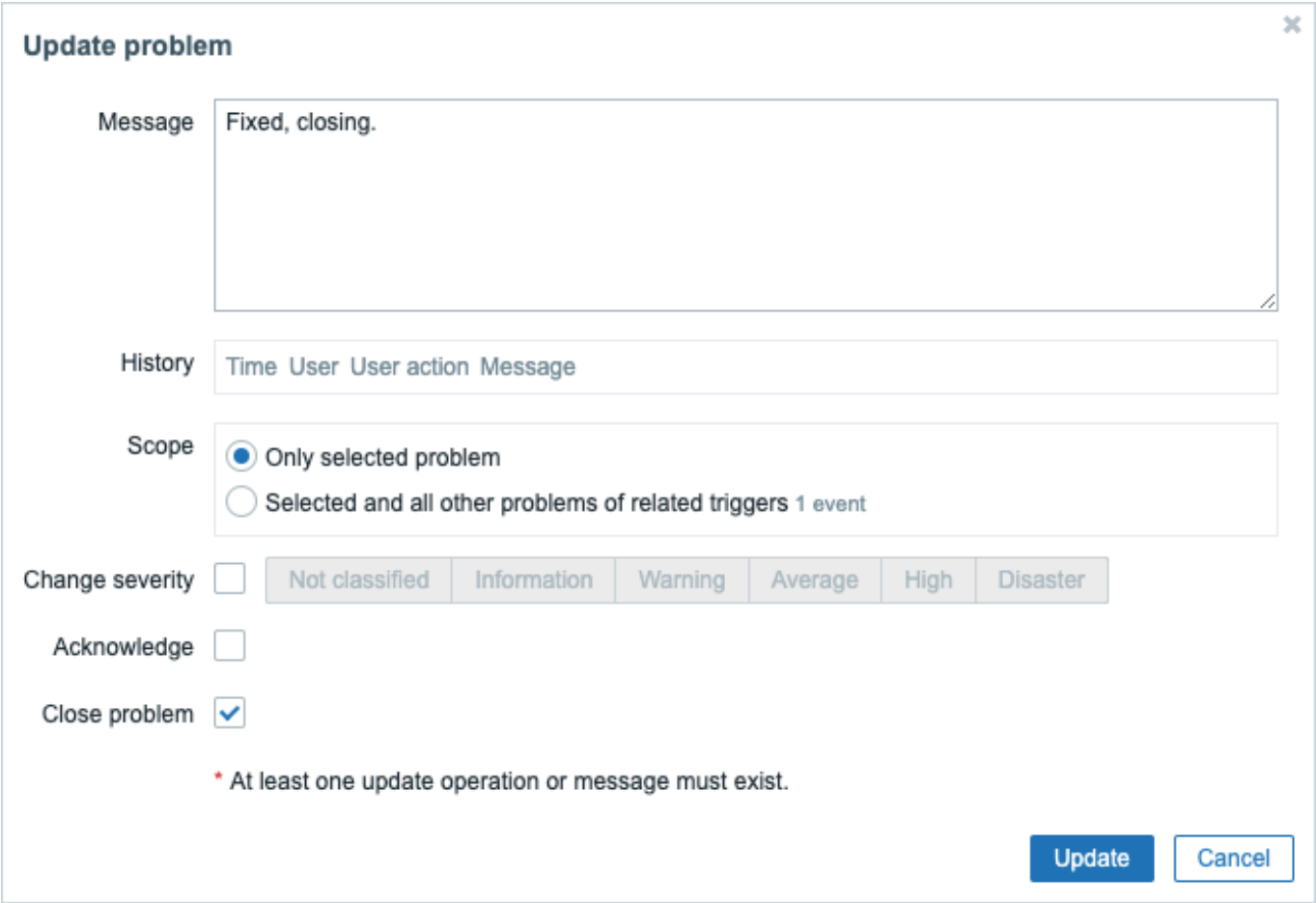
In trigger configuration, enable the *Allow manual close* option.



Problem update window

If a problem arises for a trigger with the *Manual close* flag, you can open the problem update popup window of that problem and close the problem manually.

To close the problem, check the *Close problem* option in the form and click on *Update*.



All mandatory input fields are marked with a red asterisk.

The request is processed by Zabbix server. Normally it will take a few seconds to close the problem. During that process *CLOSING* is displayed in *Monitoring* → *Problems* as the status of the problem.

Verification

It can be verified that a problem has been closed manually:

- in event details, available through *Monitoring* → *Problems*;
- by using the {EVENT.UPDATE.HISTORY} macro in notification messages that will provide this information.

5 Event correlation

Overview

Event correlation allows to correlate problem events to their resolution in a manner that is very precise and flexible.

Event correlation can be defined:

- on trigger level - one trigger may be used to relate separate problems to their solution
- globally - problems can be correlated to their solution from a different trigger/polling method using global correlation rules

1 Trigger-based event correlation

Overview

Trigger-based event correlation allows to correlate separate problems reported by one trigger.

While generally an OK event can close all problem events created by one trigger, there are cases when a more detailed approach is needed. For example, when monitoring log files you may want to discover certain problems in a log file and close them individually rather than all together.

This is the case with triggers that have *PROBLEM event generation mode* parameter set to *Multiple*. Such triggers are normally used for log monitoring, trap processing, etc.

It is possible in Zabbix to relate problem events based on tagging. Tags are used to extract values and create identification for problem events. Taking advantage of that, problems can also be closed individually based on matching tag.

In other words, the same trigger can create separate events identified by the event tag. Therefore problem events can be identified one-by-one and closed separately based on the identification by the event tag.

How it works

In log monitoring you may encounter lines similar to these:

```
Line1: Application 1 stopped
Line2: Application 2 stopped
Line3: Application 1 was restarted
Line4: Application 2 was restarted
```

The idea of event correlation is to be able to match the problem event from Line1 to the resolution from Line3 and the problem event from Line2 to the resolution from Line4, and close these problems one by one:

```
Line1: Application 1 stopped
Line3: Application 1 was restarted #problem from Line 1 closed

Line2: Application 2 stopped
Line4: Application 2 was restarted #problem from Line 2 closed
```

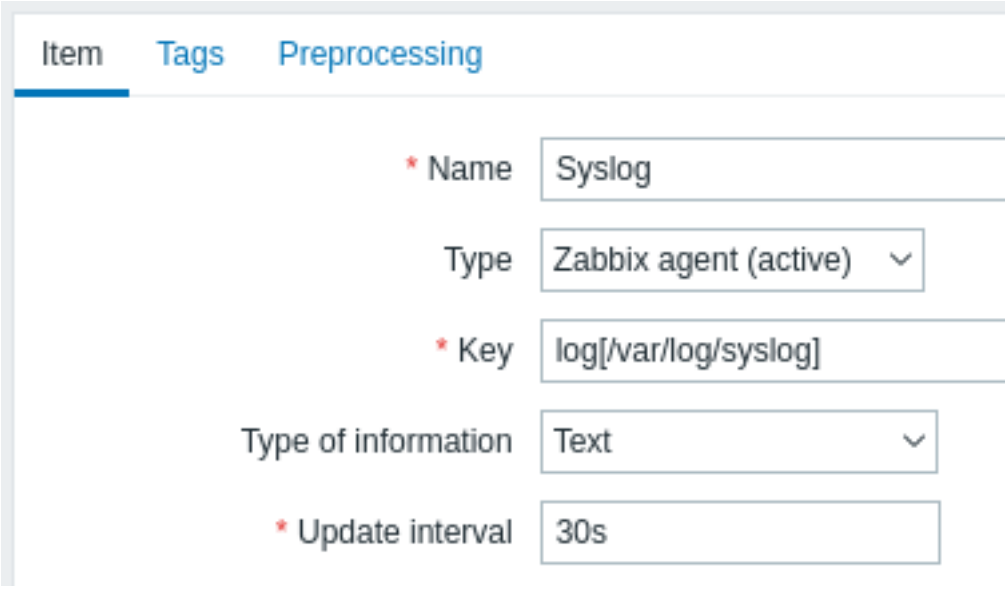
To do this you need to tag these related events as, for example, ”Application 1” and ”Application 2”. That can be done by applying a regular expression to the log line to extract the tag value. Then, when events are created, they are tagged ”Application 1” and ”Application 2” respectively and problem can be matched to the resolution.

Configuration

Item

To begin with, you may want to set up an item that monitors a log file, for example:

```
log[/var/log/syslog]
```



With the item set up, wait a minute for the configuration changes to be picked up and then go to Latest data to make sure that the item has started collecting data.

Trigger

With the item working you need to configure the trigger. It's important to decide what entries in the log file are worth paying attention to. For example, the following trigger expression will search for a string like 'Stopping' to signal potential problems:

```
find(/My host/log[/var/log/syslog],,"regexp","Stopping")=1
```

Attention:
To make sure that each line containing the string "Stopping" is considered a problem also set the *Problem event generation mode* in trigger configuration to 'Multiple'.

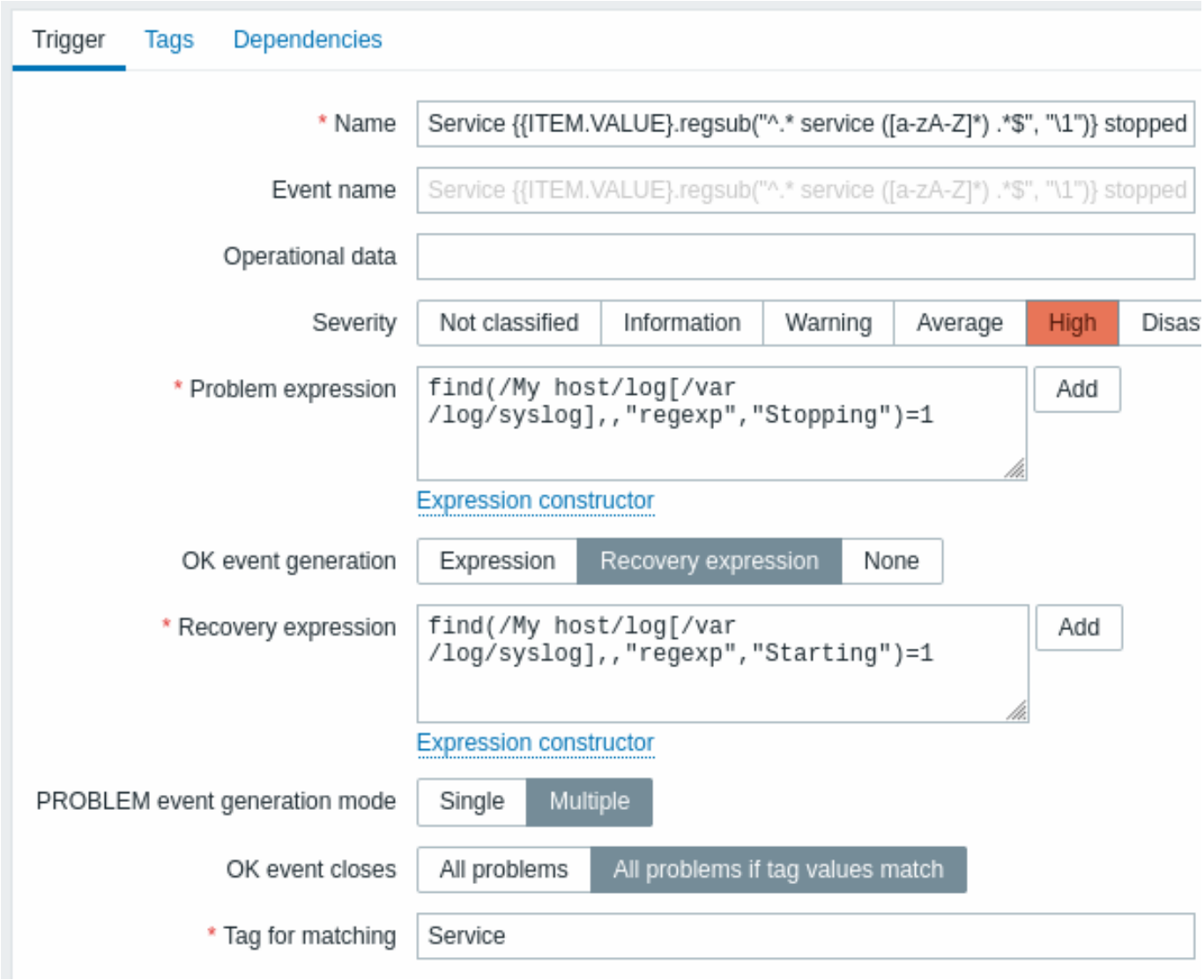
Then define a recovery expression. The following recovery expression will resolve all problems if a log line is found containing the string "Starting":

```
find(/My host/log[/var/log/syslog],,"regexp","Starting")=1
```

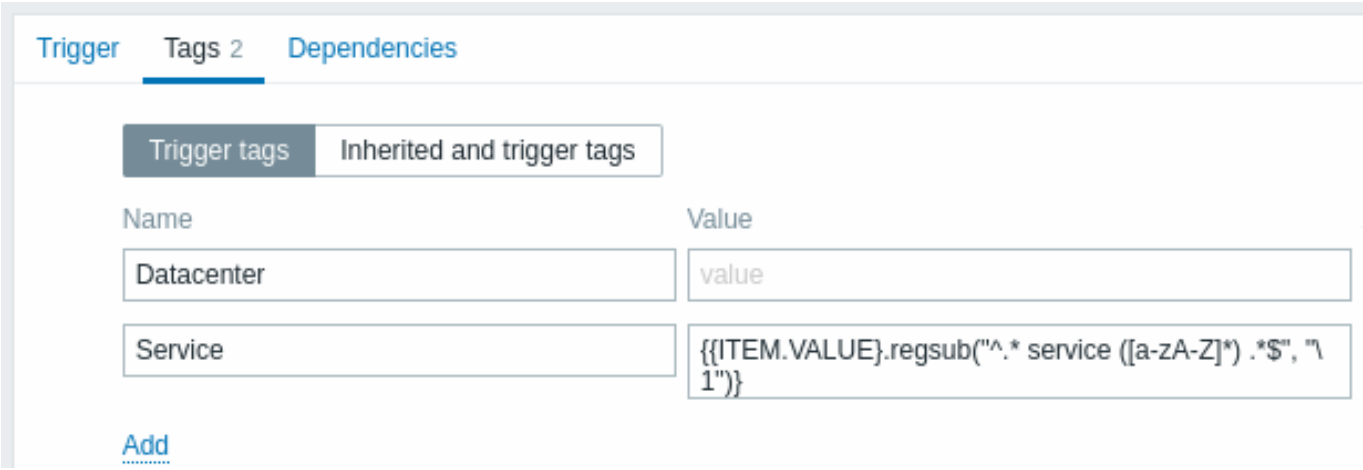
Since we do not want that it's important to make sure somehow that the corresponding root problems are closed, not just all problems. That's where tagging can help.

Problems and resolutions can be matched by specifying a tag in the trigger configuration. The following settings have to be made:

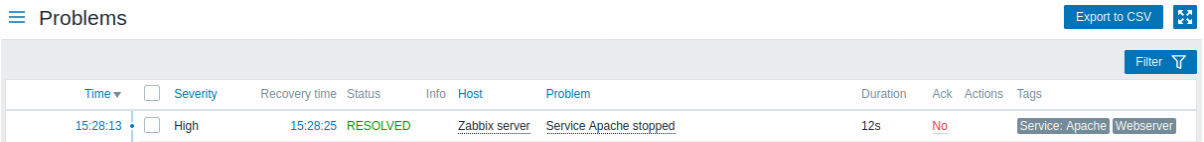
- *Problem event generation mode*: Multiple
- *OK event closes*: All problems if tag values match
- Enter the name of the tag for event matching



- configure the tags to extract tag values from log lines



If configured successfully you will be able to see problem events tagged by application and matched to their resolution in *Monitoring → Problems*.



Warning:
Because misconfiguration is possible, when similar event tags may be created for **unrelated** problems, please review the cases outlined below!

- With two applications writing error and recovery messages to the same log file a user may decide to use two *Application* tags in the same trigger with different tag values by using separate regular expressions in the tag values to extract the names of, say, application A and application B from the {ITEM.VALUE} macro (e.g. when the message formats differ). However, this may not work as planned if there is no match to the regular expressions. Non-matching regexps will yield empty tag values and a single empty tag value in both problem and OK events is enough to correlate them. So a recovery message from application A may accidentally close an error message from application B.
- Actual tags and tag values only become visible when a trigger fires. If the regular expression used is invalid, it is silently replaced with an *UNKNOWN* string. If the initial problem event with an *UNKNOWN* tag value is missed, there may appear subsequent OK events with the same *UNKNOWN* tag value that may close problem events which they shouldn't have closed.
- If a user uses the {ITEM.VALUE} macro without macro functions as the tag value, the 255-character limitation applies. When log messages are long and the first 255 characters are non-specific, this may also result in similar event tags for unrelated problems.

2 Global event correlation

Overview

Global event correlation allows to reach out over all metrics monitored by Zabbix and create correlations.

It is possible to correlate events created by completely different triggers and apply the same operations to them all. By creating intelligent correlation rules it is actually possible to save yourself from thousands of repetitive notifications and focus on root causes of a problem!

Global event correlation is a powerful mechanism, which allows you to untie yourself from one-trigger based problem and resolution logic. So far, a single problem event was created by one trigger and we were dependent on that same trigger for the problem resolution. We could not resolve a problem created by one trigger with another trigger. But with event correlation based on event tagging, we can.

For example, a log trigger may report application problems, while a polling trigger may report the application to be up and running. Taking advantage of event tags you can tag the log trigger as *Status: Down* while tag the polling trigger as *Status: Up*. Then, in a global correlation rule you can relate these triggers and assign an appropriate operation to this correlation such as closing the old events.

In another use, global correlation can identify similar triggers and apply the same operation to them. What if we could get only one problem report per network port problem? No need to report them all. That is also possible with global event correlation.

Global event correlation is configured in **correlation rules**. A correlation rule defines how the new problem events are paired with existing problem events and what to do in case of a match (close the new event, close matched old events by generating corresponding OK events). If a problem is closed by global correlation, it is reported in the *Info* column of *Monitoring → Problems*.

Configuring global correlation rules is available to Super Admin level users only.

Attention:
Event correlation must be configured very carefully, as it can negatively affect event processing performance or, if misconfigured, close more events than was intended (in the worst case even all problem events could be closed).

To configure global correlation **safely**, observe the following important tips:

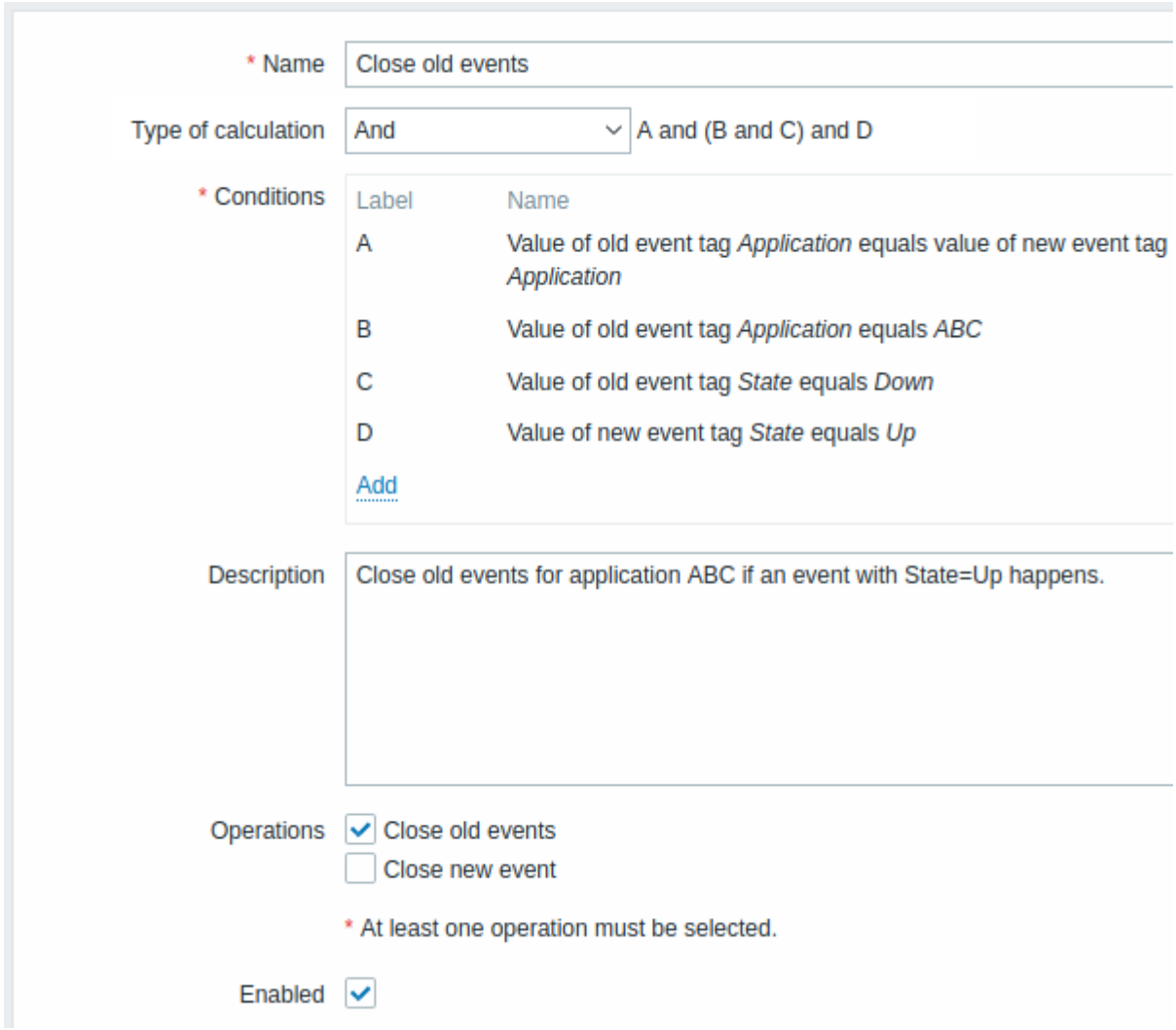
- Reduce the correlation scope. Always set a unique tag for the new event that is paired with old events and use the *New event tag* correlation condition;
- Add a condition based on the old event when using the *Close old event* operation (or else all existing problems could be closed);
- Avoid using common tag names that may end up being used by different correlation configurations;
- Keep the number of correlation rules limited to the ones you really need.

See also: known issues.

Configuration

To configure event correlation rules globally:

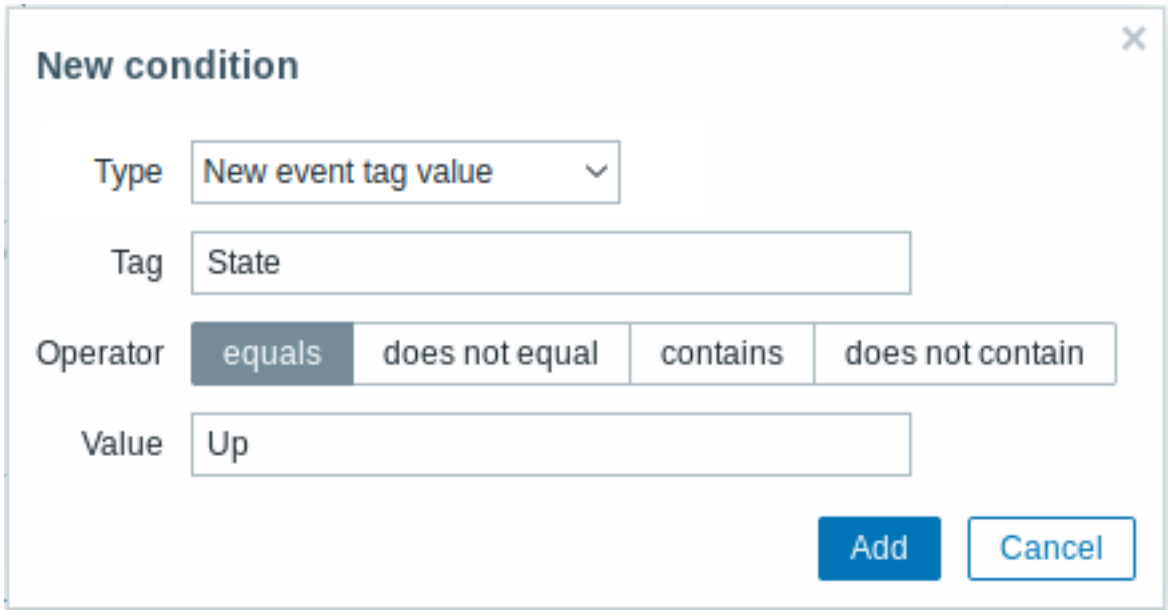
- Go to *Data collection → Event correlation*
- Click on *Create correlation* to the right (or on the correlation name to edit an existing rule)
- Enter parameters of the correlation rule in the form



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Unique correlation rule name.
Type of calculation	The following options of calculating conditions are available: **And** - all conditions must be met **Or** - enough if one condition is met **And/Or** - AND with different condition types and OR with the same condition type **Custom expression** - a user-defined calculation formula for evaluating action conditions. It must include all conditions (represented as uppercase letters A, B, C, ...) and may include spaces, tabs, brackets ( ), **and** (case sensitive), **or** (case sensitive), **not** (case sensitive).
Conditions	List of conditions. See below for details on configuring a condition.
Description	Correlation rule description.
Operations	Mark the checkbox of the operation to perform when event is correlated. The following operations are available: **Close old events** - close old events when a new event happens. Always add a condition based on the old event when using the *Close old events* operation or all existing problems could be closed. **Close new event** - close the new event when it happens
Enabled	If you mark this checkbox, the correlation rule will be enabled.

To configure details of a new condition, click on Add in the Conditions block. A popup window will open where you can edit the condition details.



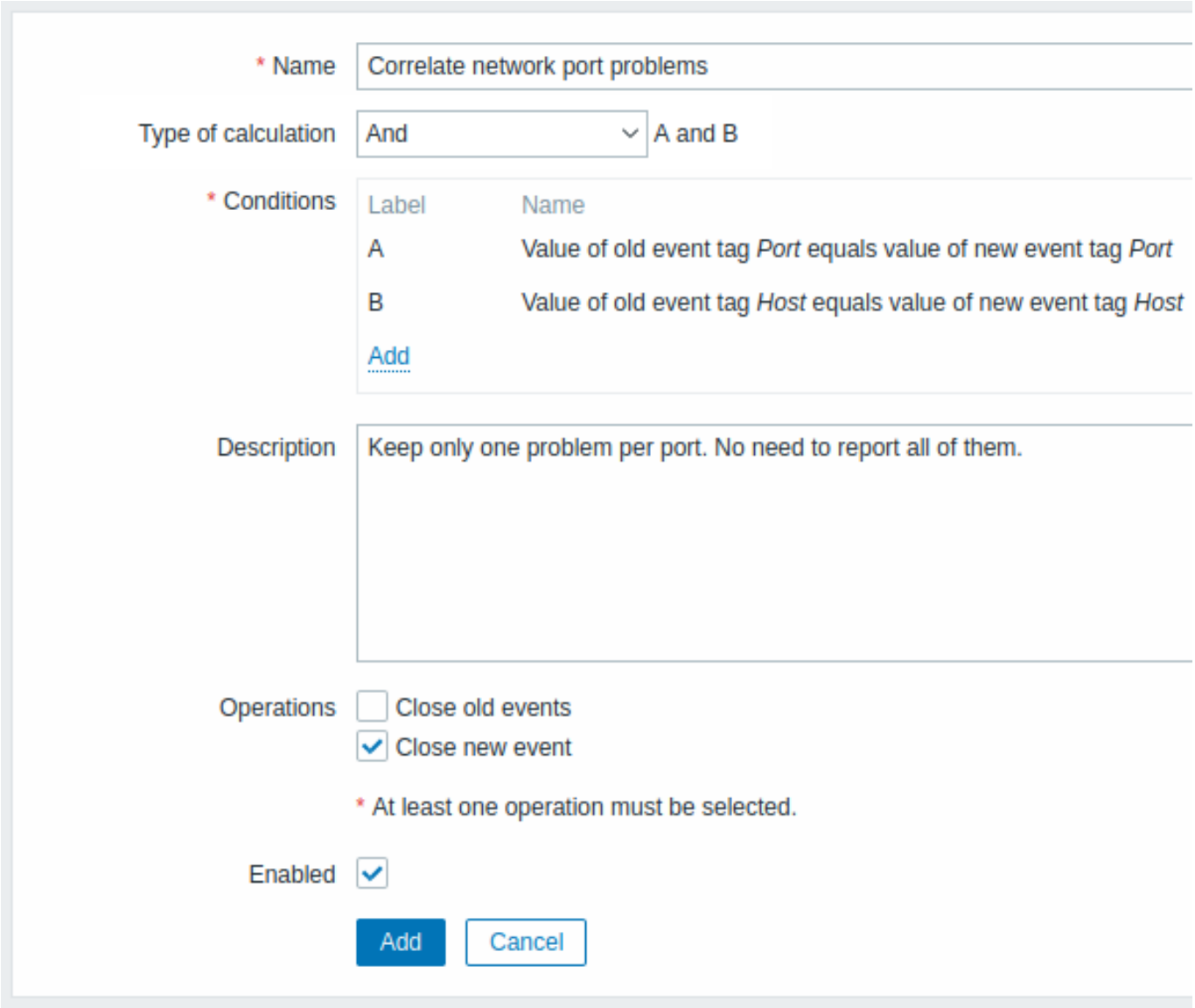
Parameter	Description
New condition	Select a condition for correlating events. *Note* that if no old event condition is specified, all old events may be matched and closed. Similarly if no new event condition is specified, all new events may be matched and closed. The following conditions are available: **Old event tag** - specify the old event tag for matching. **New event tag** - specify the new event tag for matching. **New event host group** - specify the new event host group for matching. **Event tag pair** - specify new event tag and old event tag for matching. In this case there will be a match if the **values** of the tags in both events match. Tag *names* need not match. This option is useful for matching runtime values, which may not be known at the time of configuration (see also Example 1). **Old event tag value** - specify the old event tag name and value for matching, using the following operators: *equals* - has the old event tag value *does not equal* - does not have the old event tag value *contains* - has the string in the old event tag value *does not contain* - does not have the string in the old event tag value **New event tag value** - specify the new event tag name and value for matching, using the following operators: *equals* - has the new event tag value *does not equal* - does not have the new event tag value *contains* - has the string in the new event tag value *does not contain* - does not have the string in the new event tag value

Warning:
Because misconfiguration is possible, when similar event tags may be created for **unrelated** problems, please review the cases outlined below!

- Actual tags and tag values only become visible when a trigger fires. If the regular expression used is invalid, it is silently replaced with an *UNKNOWN* string. If the initial problem event with an *UNKNOWN* tag value is missed, there may appear subsequent OK events with the same *UNKNOWN* tag value that may close problem events which they shouldn't have closed.
- If a user uses the {ITEM.VALUE} macro without macro functions as the tag value, the 255-character limitation applies. When log messages are long and the first 255 characters are non-specific, this may also result in similar event tags for unrelated problems.

Example

Stop repetitive problem events from the same network port.



This global correlation rule will correlate problems if *Host* and *Port* tag values exist on the trigger and they are the same in the original event and the new one.

The operation will close new problem events on the same network port, keeping only the original problem open.

6 Tagging

Overview

There is an option to tag various entities in Zabbix. Tags can be defined for:

- templates
- hosts
- items
- web scenarios
- triggers
- services
- template items and triggers
- host, item and trigger prototypes

Tags have several uses, most notably, to mark events. If entities are tagged, the corresponding new events get marked accordingly:

- with tagged templates - any host problems created by relevant entities (items, triggers, etc) from this template will be marked
- with tagged hosts - any problem of the host will be marked
- with tagged items, web scenarios - any data/problem of this item or web scenario will be marked
- with tagged triggers - any problem of this trigger will be marked

A problem event inherits all tags from the whole chain of templates, hosts, items, web scenarios, triggers. Completely identical `tag:value` combinations (after resolved macros) are merged into one rather than being duplicated, when marking the event.

Having custom event tags allows for more flexibility. Importantly, events can be correlated based on event tags. In other uses, actions can be defined based on tagged events. Item problems can be grouped based on tags. Problem tags can also be used to map problems to services.

Tagging is realized as a pair of *tag name* and *value*. You can use only the name or pair it with a value:

`MySQL, Service:MySQL, Services, Services:Customer, Applications, Application:Java, Priority:High`

An entity (template, host, item, web scenario, trigger or event) may be tagged with the same name, but different values - these tags will not be considered 'duplicates'. Similarly, a tag without value and the same tag with value can be used simultaneously.

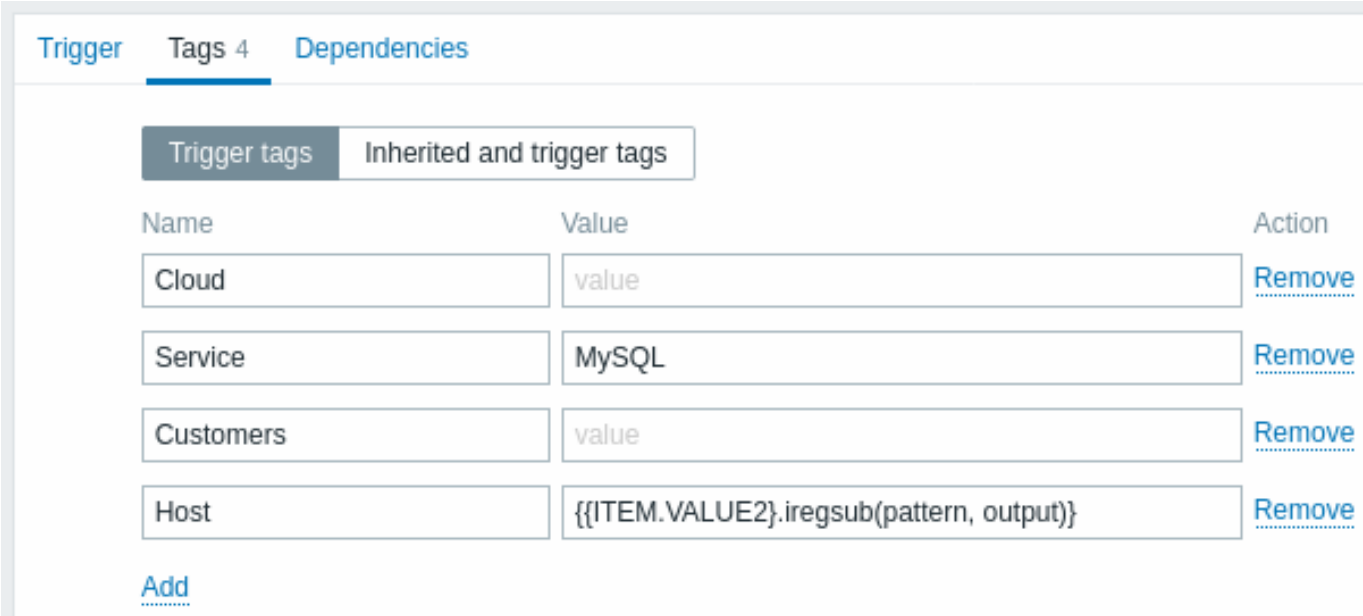
Use cases

Some use cases for this functionality are as follows:

1. Mark trigger events in the frontend:
    - Define a tag at the trigger level, for example `scope:performance`;
    - All problems created by this trigger will be marked with this tag.
2. Mark all template-inherited problems:
    - Define a tag at the template level, for example `target:MySQL`;
    - All host problems created by triggers from this template will be marked with this tag.
3. Mark all host problems:
    - Define a tag at the host level, for example `service:Jira`;
    - All problems of the host triggers will be marked with this tag.
4. Group related items:
    - Define a tag at the item level, for example `component:cpu`;
    - In the *Latest data* section, use the tag filter to view all items tagged as `component:cpu`.
5. Identify problems in a log file and close them separately:
    - Define tags in the log trigger that will identify events using value extraction by the `{{ITEM.VALUE<N>}.regsub()}` macro;
    - In the trigger configuration, set multiple problem event generation mode;
    - In the trigger configuration, use event correlation: select the option that OK event closes only matching events and choose the tag for matching;
    - See problem events created with a tag and closed individually.
6. Use it to filter notifications:
    - Define tags at the trigger level to mark events by different tags;
    - Use tag filtering in action conditions to receive notifications only on the events that match tag data.
7. Use information extracted from item value as tag value:
    - Use an `{{ITEM.VALUE<N>}.regsub()}` macro in the tag value;
    - See tag values in *Monitoring* → *Problems* as extracted data from the item value.
8. Identify problems better in notifications:
    - Define tags at the trigger level;
    - Use an {EVENT.TAGS} macro in the problem notification;
    - Easier identify which application/service the notification belongs to.
9. Simplify configuration tasks by using tags at the template level:
    - Define tags at the template trigger level;
    - See these tags on all triggers created from template triggers.
10. Create triggers with tags from low-level discovery (LLD):
    - Define tags on trigger prototypes;
    - Use LLD macros in the tag name or value;
    - See these tags on all triggers created from trigger prototypes.
11. Match services using service tags:
    - Define service actions for services with matching tags;
    - Use service tags to map a service to an SLA for SLA calculations.
12. Map services to problems using problem tags:
    - In the service configuration, specify problem tag, for example `target:MySQL`;
    - Problems with the matching tag will be automatically correlated to the service;
    - Service status will change to the status of the problem with the highest severity.
13. Suppress problems when a host is in maintenance mode:
    - Define tags in *Maintenance periods* to suppress only problems with matching tags.
14. Grant access to user groups:
    - Specify tags in the user group configuration to allow viewing only problems with matching tags.

Configuration

Tags can be entered in a dedicated tab, for example, in trigger configuration:



Macro support

Built-in and user macros in tags are resolved at the time of the event. Until the event has occurred these macros will be shown in Zabbix frontend unresolved. Low-level discovery macros are resolved during discovery process.

The following macros may be used in trigger tags:

- {ITEM.VALUE}, {ITEM.LASTVALUE}, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros can be used to populate the tag name or tag value.
- {INVENTORY.*} macros can be used to reference host inventory values from one or several hosts in a trigger expression.
- User macros and user macros with context are supported for the tag name/value; context may include low-level discovery macros.
- Low-level discovery macros can be used for the tag name/value in trigger prototypes.

The following macros may be used in trigger-based notifications:

- {EVENT.TAGS} and {EVENT.RECOVERY.TAGS} macros will resolve to a comma separated list of event tags or recovery event tags
- {EVENT.TAGSJSON} and {EVENT.RECOVERY.TAGSJSON} macros will resolve to a JSON array containing event tag objects or recovery event tag objects

The following macros may be used in template, host, item and web scenario tags:

- {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros
- {INVENTORY.*} macros
- User macros
- Low-level discovery macros can be used in item prototype tags

The following macros may be used in host prototype tags:

- {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros
- {INVENTORY.*} macros
- User macros
- Low-level discovery macros will be resolved during discovery process and then added to the discovered host

Substring extraction in trigger tags

Substring extraction is supported for populating the tag name or tag value, using a macro function - applying a regular expression to the value obtained by the {ITEM.VALUE}, {ITEM.LASTVALUE} macro or a low-level discovery macro. For example:

```
{{ITEM.VALUE}.regsub(pattern, output)}
{{ITEM.VALUE}.iregsub(pattern, output)}

{{#LLDMACRO}.regsub(pattern, output)}
{{#LLDMACRO}.iregsub(pattern, output)}
```

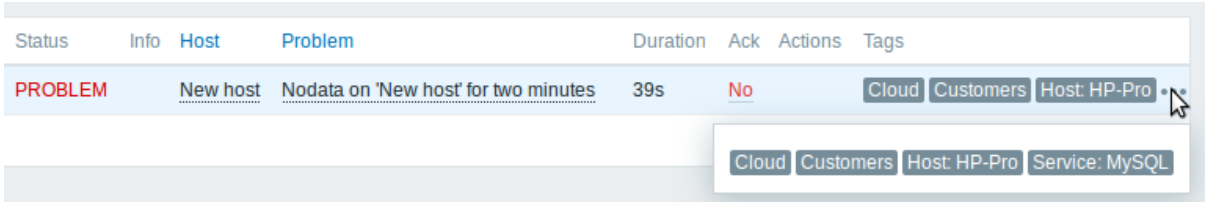
Tag name and value will be cut to 255 characters if their length exceeds 255 characters after macro resolution.

See also: Using macro functions in low-level discovery macros for event tagging.

Viewing event tags

Tagging, if defined, can be seen with new events in:

- *Monitoring* → *Problems*
- *Monitoring* → *Problems* → *Event details*
- *Dashboards* → *Problems* widget



Only the first three tag entries can be displayed. If there are more than three tag entries, it is indicated by three dots. If you roll your mouse over these three dots, all tag entries are displayed in a pop-up window.

Note that the order in which tags are displayed is affected by tag filtering and the *Tag display priority* option in the filter of *Monitoring* → *Problems* or the *Problems* dashboard widget.

7 Visualization

1 Graphs

Overview

With lots of data flowing into Zabbix, it becomes much easier for the users if they can look at a visual representation of what is going on rather than only numbers.

This is where graphs come in. Graphs allow to grasp the data flow at a glance, correlate problems, discover when something started or make a presentation of when something might turn into a problem.

Zabbix provides users with:

- built-in simple graphs of one item data
- the possibility to create more complex customized graphs
- access to a comparison of several items quickly in ad-hoc graphs
- modern customizable vector graphs

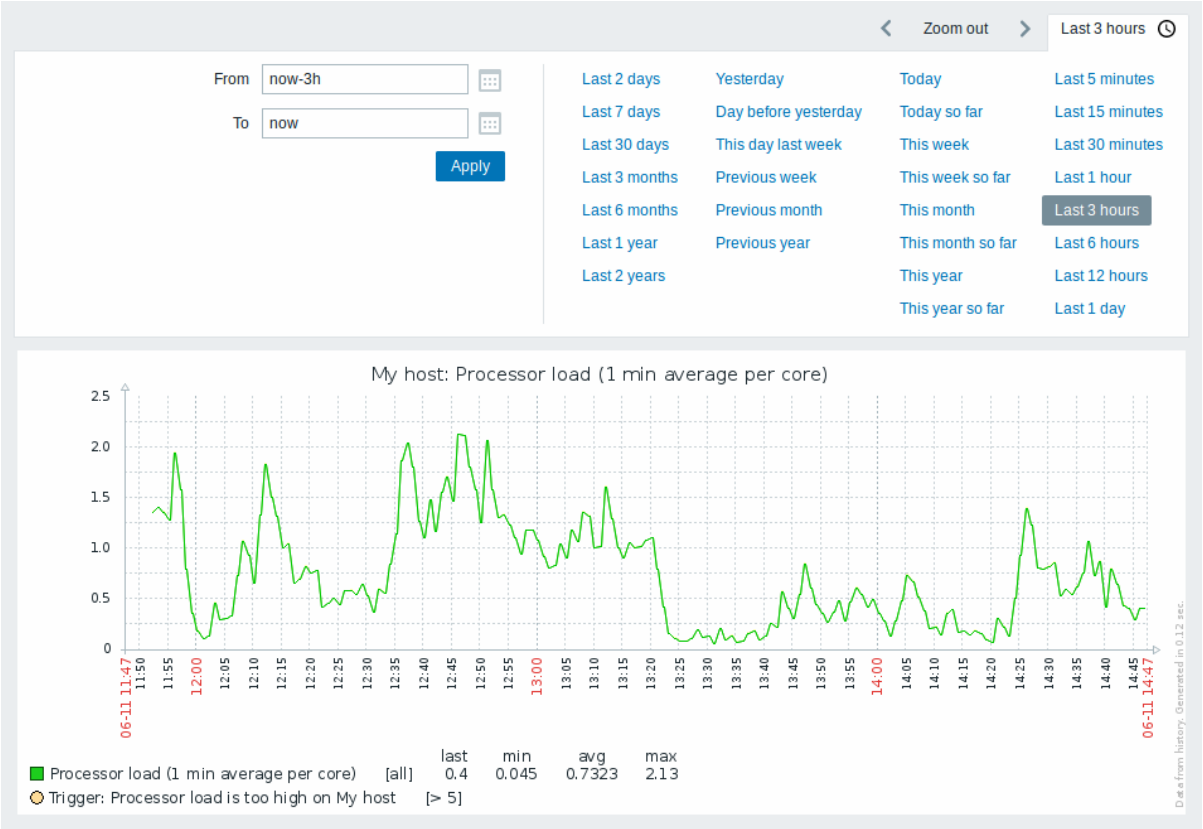
1 Simple graphs

Overview

Simple graphs are provided for the visualization of data gathered by items.

No configuration effort is required on the user part to view simple graphs. They are freely made available by Zabbix.

Just go to *Monitoring* → *Latest data* and click on the Graph link for the respective item and a graph will be displayed.



Note:
Simple graphs are provided for all numeric items. For textual items, a link to History is available in *Monitoring* → *Latest data*.

Time period selector

Take note of the time period selector above the graph. It allows to select often required periods with one mouse click.

Note that such options as *Today*, *This week*, *This month*, *This year* display the whole period, including the hours/days in the future. *Today so far*, in contrast, only displays the hours passed.

Once a period is selected, it can be moved back and forth in time by clicking on the arrow buttons. The *Zoom out* button allows to zoom out the period two times or by 50% in each direction. Zoom out is also possible by double-clicking in the graphs. The whole time period selector can be collapsed by clicking on the tab label containing the selected period string.

The *From/To* fields display the selected period in either:

- absolute time syntax in format `Y-m-d H:i:s`
- relative time syntax, e.g.: `now-1d`

A date in **relative** format can contain one or several mathematical operations (- or +), e.g. `now-1d` or `now-1d-2h+5m`. For relative time the following abbreviations are supported:

- now
- s (seconds)
- m (minutes)
- h (hours)
- d (days)
- w (weeks)
- M (months)
- y (years)

Precision is supported in the time filter (e. g., an expression like `now-1d/M`). Details of precision:

Precision	From	To
m	Y-m-d H:m:00	Y-m-d H:m:59
h	Y-m-d H:00:00	Y-m-d H:59:59
d	Y-m-d 00:00:00	Y-m-d 23:59:59
w	Monday of the week 00:00:00	Sunday of the week 23:59:59

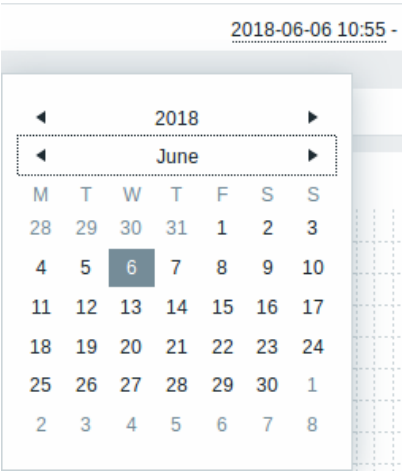
Precision	From	To
M	First day of the month 00:00:00	Last day of the month 23:59:59
y	1st of January of the year 00:00:00	31st of December of the year 23:59:59

For example:

From	To	Selected period
`now/d`	`now/d`	00:00 - 23:59 today
`now/d`	`now/d+1d`	00:00 today - 23:59 tomorrow
`now/w`	`now/w`	Monday 00:00:00 - Sunday 23:59:59 this week
`now-1y/w`	`now-1y/w`	The week of Monday 00:00:00 - Sunday 23:59:59 one year ago

Date picker

It is possible to pick a specific start/end date by clicking on the calendar icon next to the *From/To* fields. In this case, the date picker pop up will open.



Within the date picker, it is possible to navigate between the blocks of year/month/date using Tab and Shift+Tab. Keyboard arrows or arrow buttons allow to select the desired value. Pressing Enter (or clicking on the desired value) activates the choice.

Another way of controlling the displayed time is to highlight an area in the graph with the left mouse button. The graph will zoom into the highlighted area once you release the left mouse button.

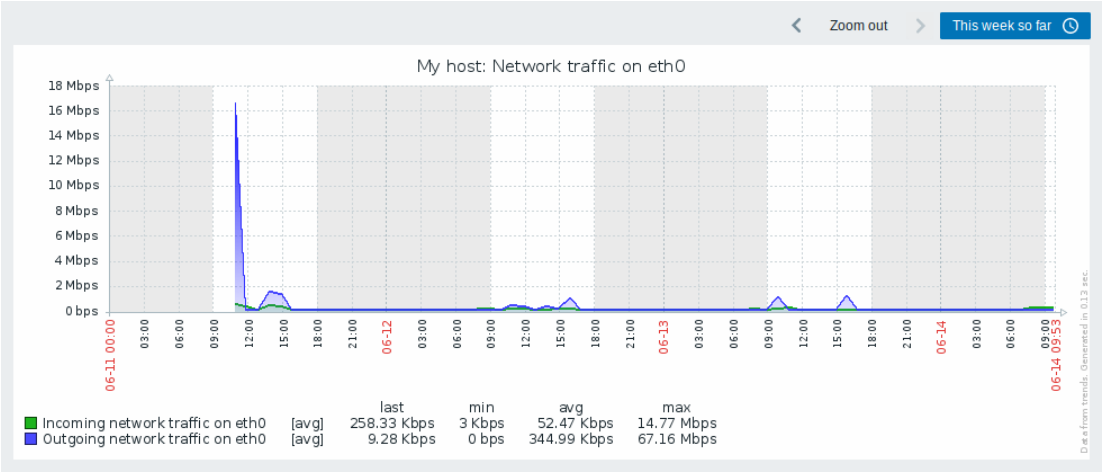
In case no time value is specified or field is left blank, time value will be set to ”00:00:00”. This doesn’t apply to today’s date selection: in that case time will be set to current value.

Recent data vs longer periods

For very recent data a **single** line is drawn connecting each received value. The single line is drawn as long as there is at least one horizontal pixel available for one value.

For data that show a longer period **three lines** are drawn - a dark green one shows the average, while a light pink and a light green line shows the maximum and minimum values at that point in time. The space between the highs and the lows is filled with yellow background.

Working time (working days) is displayed in graphs as a white background, while non-working time is displayed in gray (with the *Original blue* default frontend theme).



Working time is always displayed in simple graphs, whereas displaying it in custom graphs is a user preference.

Working time is not displayed if the graph shows more than 3 months.

Trigger lines

Simple triggers are displayed as lines with black dashes over trigger severity color -- take note of the blue line on the graph and the trigger information displayed in the legend. Up to 3 trigger lines can be displayed on the graph; if there are more triggers then the triggers with lower severity are prioritized. Triggers are always displayed in simple graphs, whereas displaying them in custom graphs is a user preference.



Generating from history/trends

Graphs can be drawn based on either item history or trends.

For the users who have frontend debug mode activated, a gray, vertical caption is displayed at the bottom right of a graph indicating where the data come from.

Several factors influence whether history of trends is used:

- longevity of item history. For example, item history can be kept for 14 days. In that case, any data older than the fourteen days will be coming from trends.
- data congestion in the graph. If the amount of seconds to display in a horizontal graph pixel exceeds 3600/16, trend data are displayed (even if item history is still available for the same period).

- if trends are disabled, item history is used for graph building - if available for that period. This is supported starting with Zabbix 2.2.1 (before, disabled trends would mean an empty graph for the period even if item history was available).

Absence of data

For items with a regular update interval, nothing is displayed in the graph if item data are not collected.

However, for trapper items and items with a scheduled update interval (and regular update interval set to 0), a straight line is drawn leading up to the first collected value and from the last collected value to the end of graph; the line is on the level of the first/last value respectively.

Switching to raw values

A dropdown on the upper right allows to switch from the simple graph to the *Values/500 latest values* listings. This can be useful for viewing the numeric values making up the graph.

The values represented here are raw, i.e. no units or postprocessing of values is used. Value mapping, however, is applied.

Known issues

See known issues for graphs.

2 Custom graphs

Overview

Custom graphs, as the name suggests, offer customization capabilities.

While simple graphs are good for viewing data of a single item, they do not offer configuration capabilities.

Thus, if you want to change graph style or the way lines are displayed or compare several items, for example, incoming and outgoing traffic in a single graph, you need a custom graph.

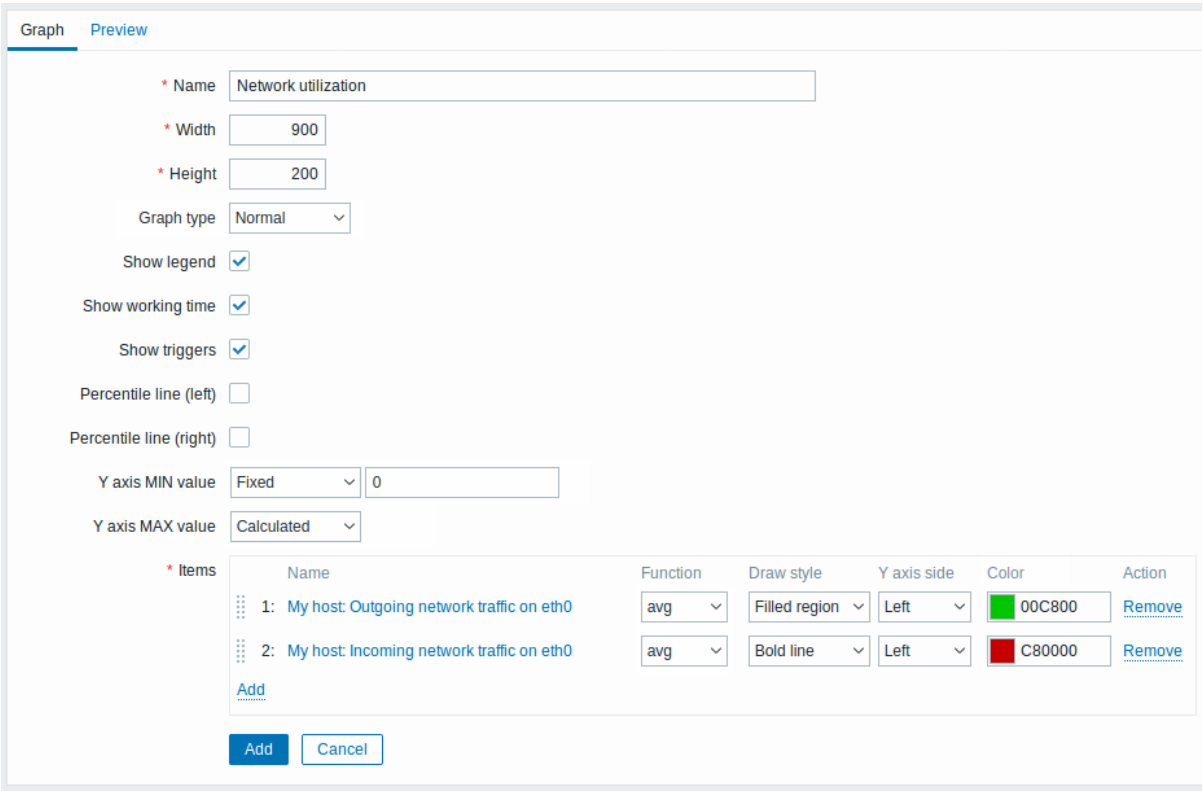
Custom graphs are configured manually.

They can be created for a host or several hosts or for a single template.

Configuring custom graphs

To create a custom graph, do the following:

- Go to *Data collection → Hosts (or Templates)*
- Click on *Graphs* in the row next to the desired host or template
- In the Graphs screen click on *Create graph*
- Edit graph attributes



All mandatory input fields are marked with a red asterisk.

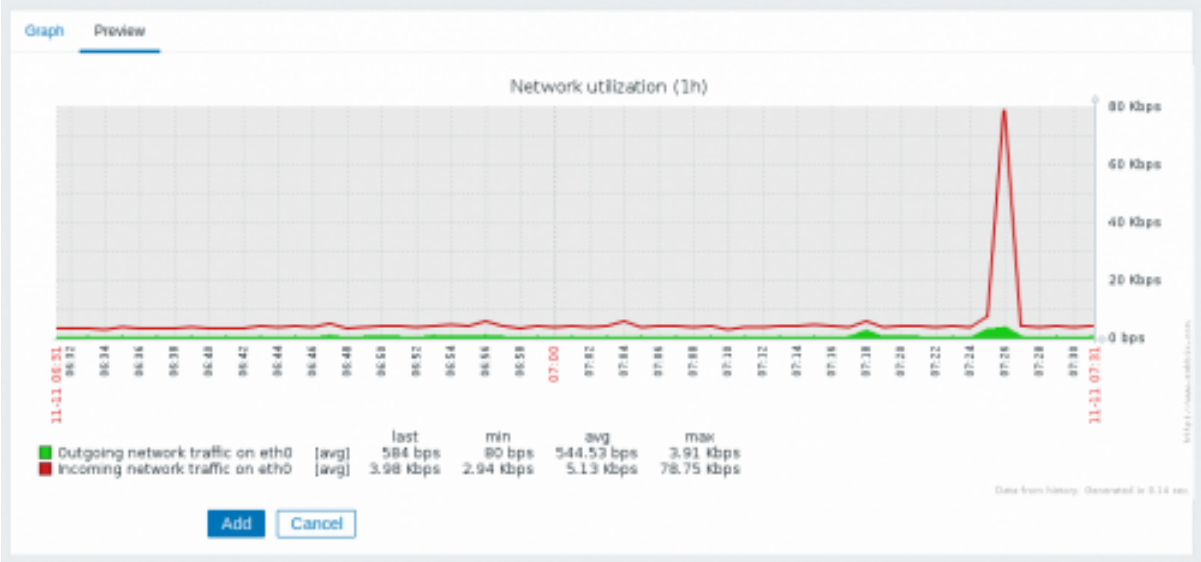
Graph attributes:

Parameter	Description
Name	Unique graph name. Expression macros are supported in this field, but only with `avg`, `last`, `min` and `max` functions, with time as parameter (for example, `{?avg(/host/key,1h)}`). {HOST.HOST<1-9>} macros are supported for the use within this macro, referencing the first, second, third, etc. host in the graph, for example `{?avg(/{HOST.HOST2}/key,1h)}`. Note that referencing the first host with this macro is redundant, as the first host can be referenced implicitly, for example `{?avg(//key,1h)}`.
Width	Graph width in pixels (for preview and pie/exploded graphs only).
Height	Graph height in pixels.
Graph type	Graph type: **Normal** - normal graph, values displayed as lines **Stacked** - stacked graph, filled areas displayed **Pie** - pie graph **Exploded** - "exploded" pie graph, portions displayed as "cut out" of the pie
Show legend	Checking this box will set to display the graph legend.
Show working time	If selected, non-working hours will be shown with a gray background. This parameter is not available for pie and exploded pie graphs.
Show triggers	If selected, simple triggers will be displayed as lines with black dashes over trigger severity color. This parameter is not available for pie and exploded pie graphs.
Percentile line (left)	Display percentile for left Y-axis. If, for example, 95% percentile is set, then the percentile line will be at the level where 95 percent of the values fall under. Displayed as a bright green line. Only available for normal graphs.
Percentile line (right)	Display percentile for right Y-axis. If, for example, 95% percentile is set, then the percentile line will be at the level where 95 percent of the values fall under. Displayed as a bright red line. Only available for normal graphs.
Y axis MIN value	Minimum value of Y-axis: **Calculated** - Y axis minimum value will be automatically calculated. **Fixed** - fixed minimum value for Y-axis. **Item** - last value of the selected item will be the minimum value. This parameter is not available for pie and exploded pie graphs.
Y axis MAX value	Maximum value of Y-axis: **Calculated** - Y axis maximum value will be automatically calculated. **Fixed** - fixed maximum value for Y-axis. **Item** - last value of the selected item will be the maximum value This parameter is not available for pie and exploded pie graphs.
3D view	Enable 3D style. For pie and exploded pie graphs only.
Items	Items, data of which are to be displayed in this graph. Click on *Add* to select items. You can also select various displaying options (function, draw style, left/right axis display, color).
Sort order (0→100)	Draw order. 0 will be processed first. Can be used to draw lines or regions behind (or in front of) another. You can drag and drop items using the icon at the beginning of a line to set the sort order or which item is displayed in front of the other.
Name	Name of the selected item is displayed as a link. Clicking on the link opens the list of other available items.
Type	Type (only available for pie and exploded pie graphs): **Simple** - the value of the item is represented proportionally on the pie **Graph sum** - the value of the item represents the whole pie Note that coloring of the "graph sum" item will only be visible to the extent that it is not taken up by "proportional" items.

Parameter	Description
Function	Select what values will be displayed when more than one value exists per vertical graph pixel for an item: **all** - display all possible values (minimum, maximum, average) in the graph. Note that for shorter periods this setting has no effect; only for longer periods, when data congestion in a vertical graph pixel increases, 'all' starts displaying minimum, maximum, and average values. This function is only available for *Normal* graph type. See also: Generating graphs from history/trends. **avg** - display the average values **last** - display the latest values. This function is only available if either *Pie/Exploded pie* is selected as graph type. **max** - display the maximum values **min** - display the minimum values
Draw style	Select the draw style (only available for normal graphs; for stacked graphs filled region is always used) to apply to the item data - *Line*, *Bold line*, *Filled region*, *Dot*, *Dashed line*, *Gradient line*.
Y axis side	Select the Y axis side to show the item data - *Left*, *Right*.
Color	Select the color to apply to the item data.

Graph preview

In the *Preview* tab, a preview of the graph is displayed so you can immediately see what you are creating.



Note that the preview will not show any data for template items.



In this example, pay attention to the dashed bold line displaying the trigger level and the trigger information displayed in the

legend.

Note:
No more than 3 trigger lines can be displayed. If there are more triggers then the triggers with lower severity are prioritized for display.

If graph height is set as less than 120 pixels, no trigger will be displayed in the legend.

3 Ad-hoc graphs

Overview

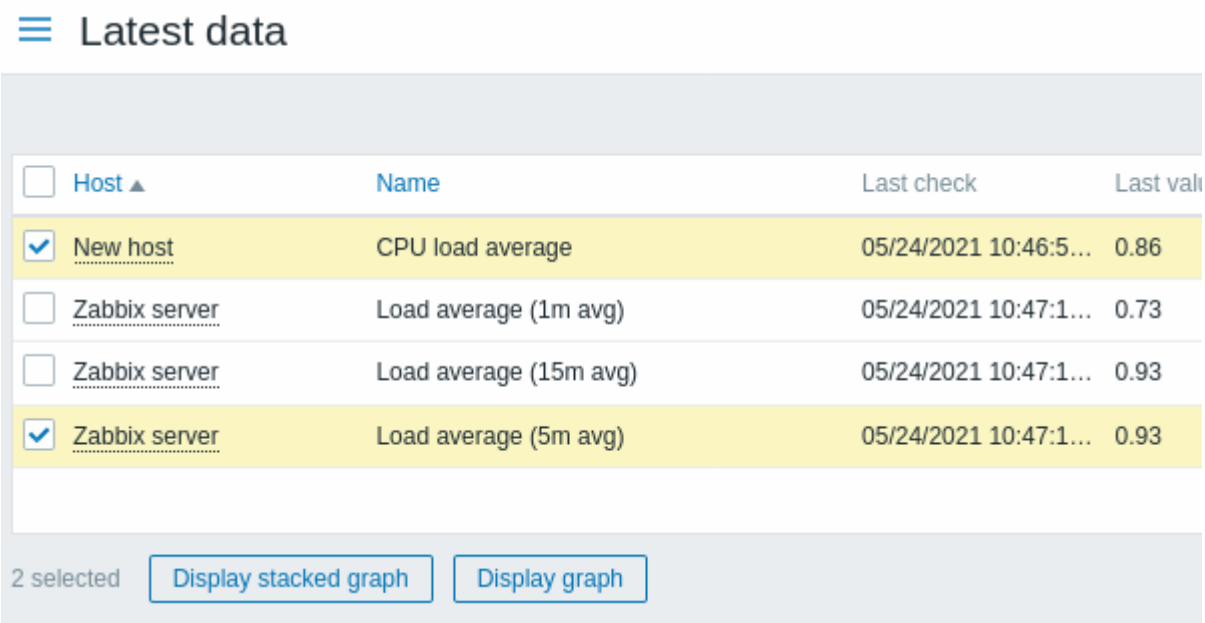
While a simple graph is great for accessing data of one item and custom graphs offer customization options, none of the two allow to quickly create a comparison graph for multiple items with little effort and no maintenance.

To address this issue, since Zabbix 2.4 it is possible to create ad-hoc graphs for several items in a very quick way.

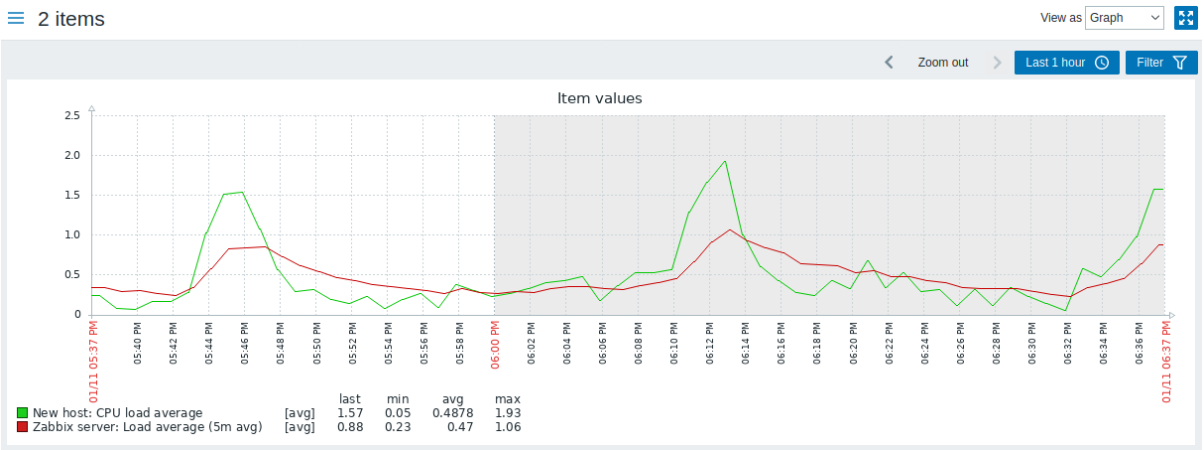
Configuration

To create an ad-hoc graph, do the following:

- Go to *Monitoring* → *Latest data*
- Use filter to display items that you want
- Mark checkboxes of the items you want to graph
- Click on *Display stacked graph* or *Display graph* buttons

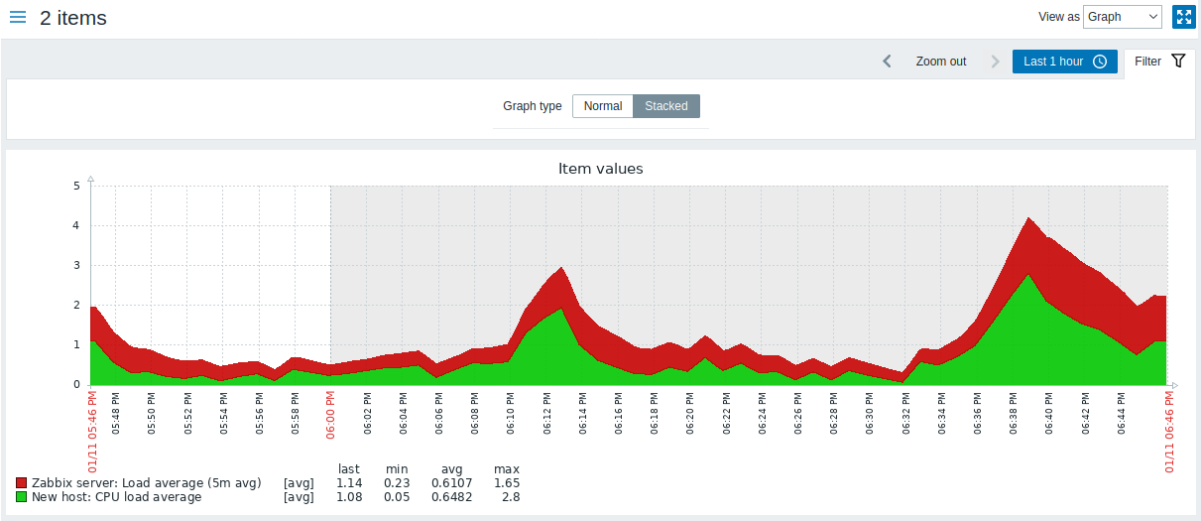


Your graph is created instantly:



Note that to avoid displaying too many lines in the graph, only the average value for each item is displayed (min/max value lines are not displayed). Triggers and trigger information is not displayed in the graph.

In the created graph window you have the time period selector available and the possibility to switch from the ”normal” line graph to a stacked one (and back).



4 Aggregation in graphs

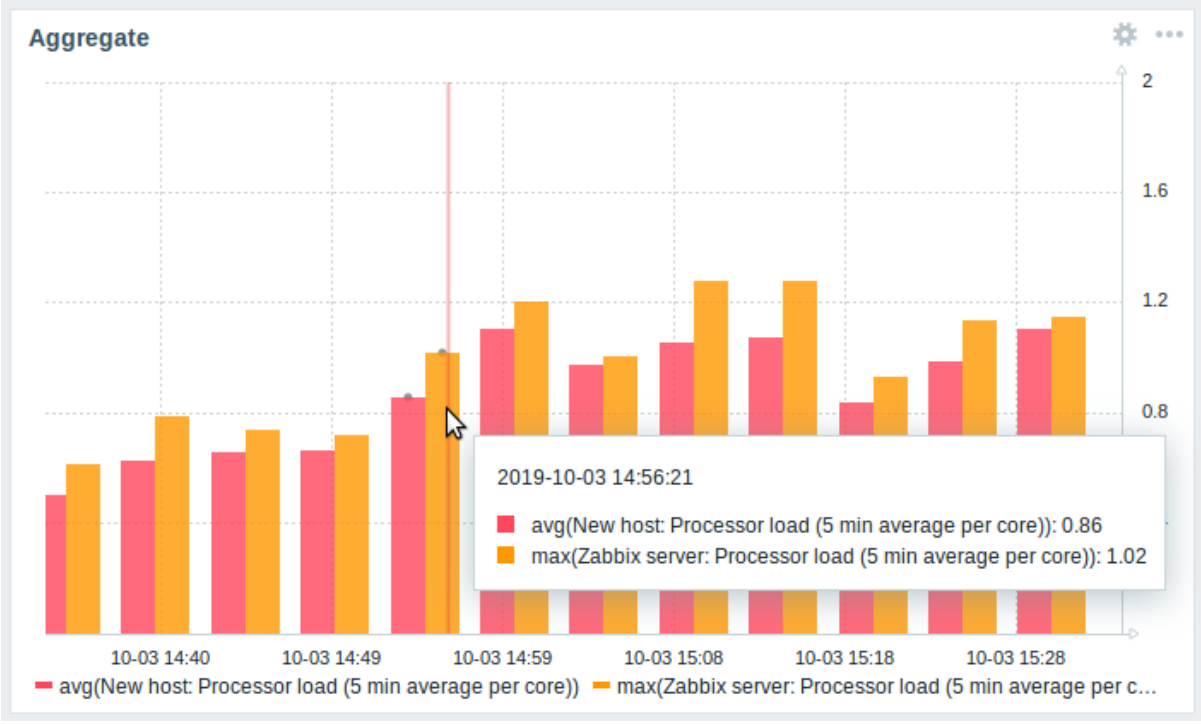
Overview

The aggregation functions, available in the graph widget of the dashboard, allow displaying an aggregated value for the chosen interval (5 minutes, an hour, a day), instead of all values.

The aggregation options are as follows:

- min
- max
- avg
- count
- sum
- first (first value displayed)
- last (last value displayed)

The most exciting use of data aggregation is the possibility to create nice side-by-side comparisons of data for some period:

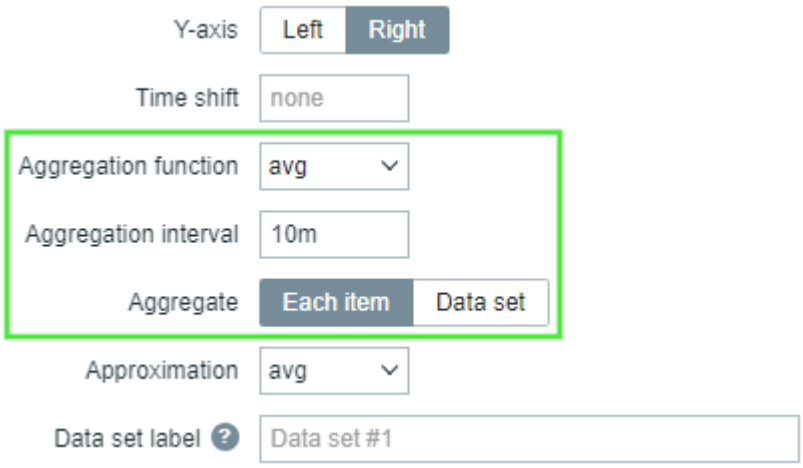


When hovering over a point in time in the graph, date and time is displayed in addition to items and their aggregated values. Items are displayed in parentheses, prefixed by the aggregation function used. If the graph widget has a *Data set label* configured, the

label is displayed in parentheses, prefixed by the aggregation function used. Note that this is the date and time of the point in the graph, not of the actual values.

Configuration

The options for aggregation are available in data set settings when configuring a graph widget.



You may pick the aggregation function and the time interval. As the data set may comprise several items, there is also another option allowing to show aggregated data for each item separately or for all data set items as one aggregated value.

Use cases

Average request count to Nginx server

View the average request count per second per day to the Nginx server:

- add the request count per second item to the data set
- select the aggregate function `avg` and specify interval `1d`
- a bar graph is displayed, where each bar represents the average number of requests per second per day

Minimum weekly disk space among clusters

View the lowest disk space among clusters over a week.

- add to the data set: hosts `cluster*`, key "Free disk space on /data"
- select the aggregate function `min` and specify interval `1w`
- a bar graph is displayed, where each bar represents the minimum disk space per week for each /data volume of the cluster

2 Network maps

Overview

If you have a network to look after, you may want to have an overview of your infrastructure somewhere. For that purpose, you can create maps in Zabbix - of networks and of anything you like.

All users can create network maps. The maps can be public (available to all users) or private (available to selected users).

Proceed to configuring a network map.

1 Configuring a network map

Overview

Configuring a map in Zabbix requires that you first create a map by defining its general parameters and then you start filling the actual map with elements and their links.

You can populate the map with elements that are a host, a host group, a trigger, an image, or another map.

Icons are used to represent map elements. You can define the information that will be displayed with the icons and set that recent problems are displayed in a special way. You can link the icons and define information to be displayed on the links.

You can add custom URLs to be accessible by clicking on the icons. Thus you may link a host icon to host properties or a map icon to another map.

The problem count in maps is displayed for cause problems only.

Maps are managed in *Monitoring* → *Maps*, where they can be configured, managed and viewed. In the monitoring view, you can click on the icons and take advantage of the links to some scripts and URLs.

Network maps are based on vector graphics (SVG) since Zabbix 3.4.

Public and private maps

All users in Zabbix (including non-admin users) can create network maps. Maps have an owner - the user who created them. Maps can be made public or private.

- *Public* maps are visible to all users, although to see it the user must have read access to at least one map element. Public maps can be edited in case a user/ user group has read-write permissions for this map and at least read permissions to all elements of the corresponding map including triggers in the links.
- *Private* maps are visible only to their owner and the users/user groups the map is shared with by the owner. Regular (non-Super admin) users can only share with the groups and users they are members of. Admin level users can see private maps regardless of being the owner or belonging to the shared user list. Private maps can be edited by the owner of the map and in case a user/ user group has read-write permissions for this map and at least read permissions to all elements of the corresponding map including triggers in the links.

Map elements that the user does not have read permission to are displayed with a grayed-out icon and all textual information on the element is hidden. However, the trigger label is visible even if the user has no permission to the trigger.

To add an element to the map the user must also have at least read permission to the element.

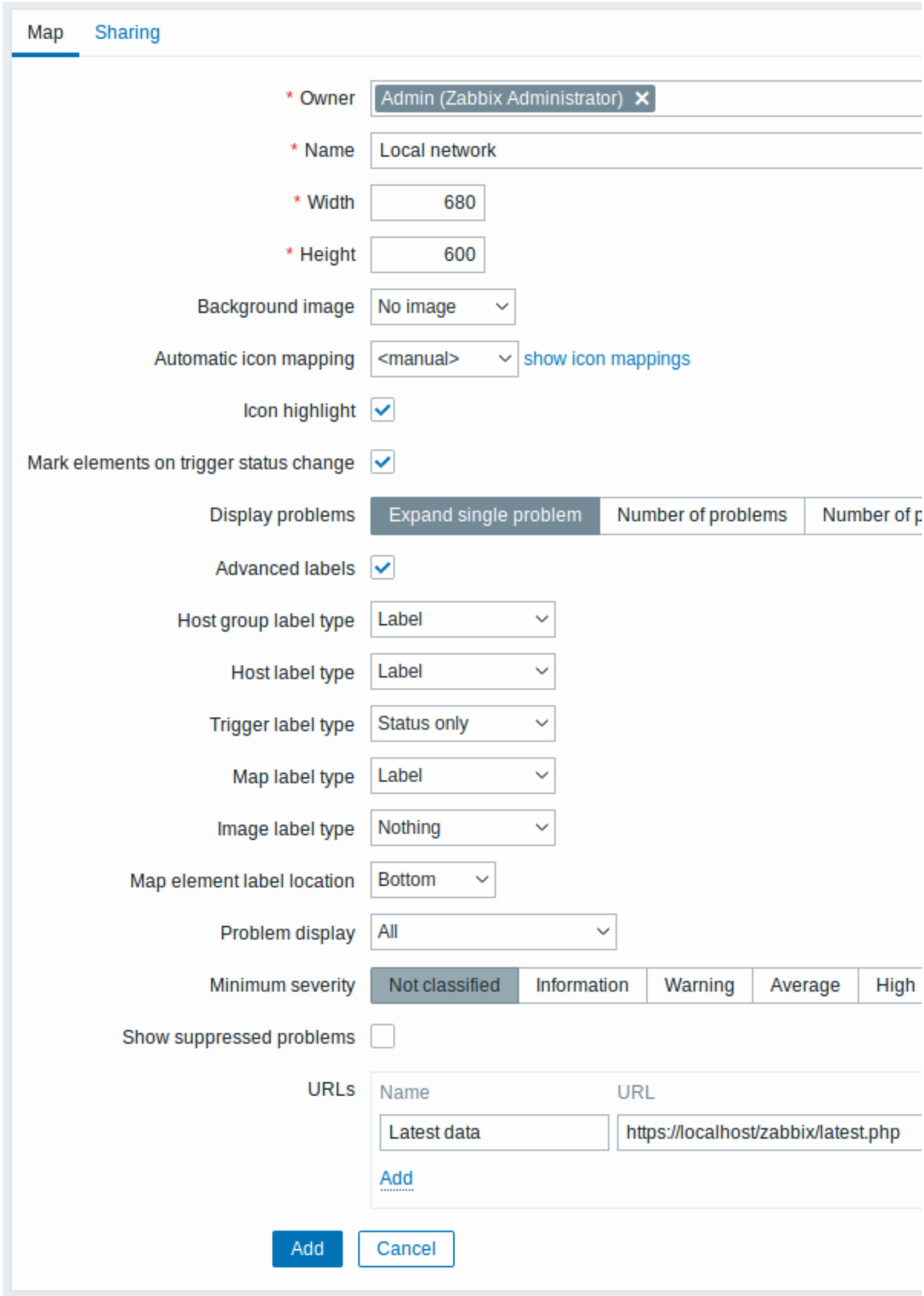
Creating a map

To create a map, do the following:

- Go to *Monitoring* → *Maps*
- Go to the view with all maps
- Click on *Create map*

You can also use the *Clone* and *Full clone* buttons in the configuration form of an existing map to create a new map. Clicking on *Clone* will retain general layout attributes of the original map, but no elements. *Full clone* will retain both the general layout attributes and all elements of the original map.

The **Map** tab contains general map attributes:



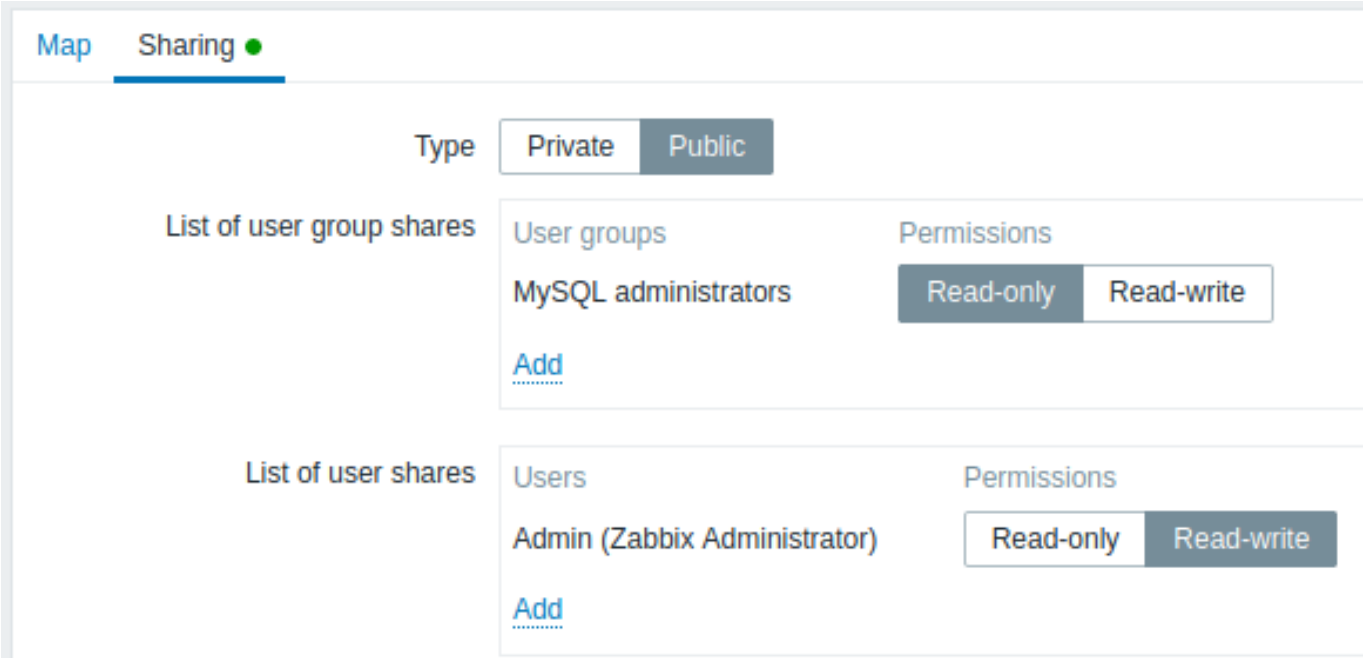
All mandatory input fields are marked with a red asterisk.

General map attributes:

Parameter	Description
Owner	Name of map owner.
Name	Unique map name.
Width	Map width in pixels.
Height	Map height in pixels.
Background image	Use background image: **No image** - no background image (white background) **Image** - selected image to be used as a background image. No scaling is performed. You may use a geographical map or any other image to enhance your map.
Automatic icon mapping	You can set to use an automatic icon mapping, configured in *Administration → General → Icon mapping*. Icon mapping allows mapping certain icons against certain host inventory fields.
Icon highlighting	If you check this box, map elements will receive highlighting. Elements with an active trigger will receive a round background, in the same color as the highest severity trigger. Moreover, a thick green line will be displayed around the circle, if all problems are acknowledged. Elements with ”disabled” or ”in maintenance” status will get a square background, gray and orange respectively. See also: Viewing maps
Mark elements on trigger status change	A recent change of trigger status (recent problem or resolution) will be highlighted with markers (inward-pointing red triangles) on the three sides of the element icon that are free of the label. Markers are displayed for 30 minutes.
Display problems	Select how problems are displayed with a map element: **Expand single problem** - if there is only one problem, the problem name is displayed. Otherwise, the total number of problems is displayed. **Number of problems** - the total number of problems is displayed **Number of problems and expand most critical one** - the name of the most critical problem and the total number of problems is displayed. ’Most critical’ is determined based on problem severity and, if equal, problem event ID (higher ID or later problem displayed first). For a *trigger map element* it is based on problem severity and if equal, trigger position in the trigger list. In case of multiple problems of the same trigger, the most recent one will be displayed.
Advanced labels	If you check this box you will be able to define separate label types for separate element types.
Map element label type	Label type used for map elements: **Label** - map element label **IP address** - IP address **Element name** - element name (for example, host name) **Status only** - status only (OK or PROBLEM) **Nothing** - no labels are displayed
Map element label location	Label location in relation to the map element: **Bottom** - beneath the map element **Left** - to the left **Right** - to the right **Top** - above the map element
Problem display	Display problem count as: **All** - full problem count will be displayed **Separated** - unacknowledged problem count will be displayed separated as a number of the total problem count **Unacknowledged only** - only the unacknowledged problem count will be displayed
Minimum trigger severity	Problems below the selected minimum severity level will not be displayed on the map. For example, with *Warning* selected, changes with *Information* and *Not classified* level triggers will not be reflected in the map. This parameter is supported starting with Zabbix 2.2.
Show suppressed problems	Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance.
URLs	URLs for each element type can be defined (with a label). These will be displayed as links when a user clicks on the element in the map viewing mode. Macros can be used in map URL names and values. For a full list, see supported macros and search for ’map URL names and values’.

Sharing

The **Sharing** tab contains the map type as well as sharing options (user groups, users) for private maps:



Parameter	Description
Type	Select map type: **Private** - map is visible only to selected user groups and users **Public** - map is visible to all
List of user group shares	Select user groups that the map is accessible to. You may allow read-only or read-write access.
List of user shares	Select users that the map is accessible to. You may allow read-only or read-write access.

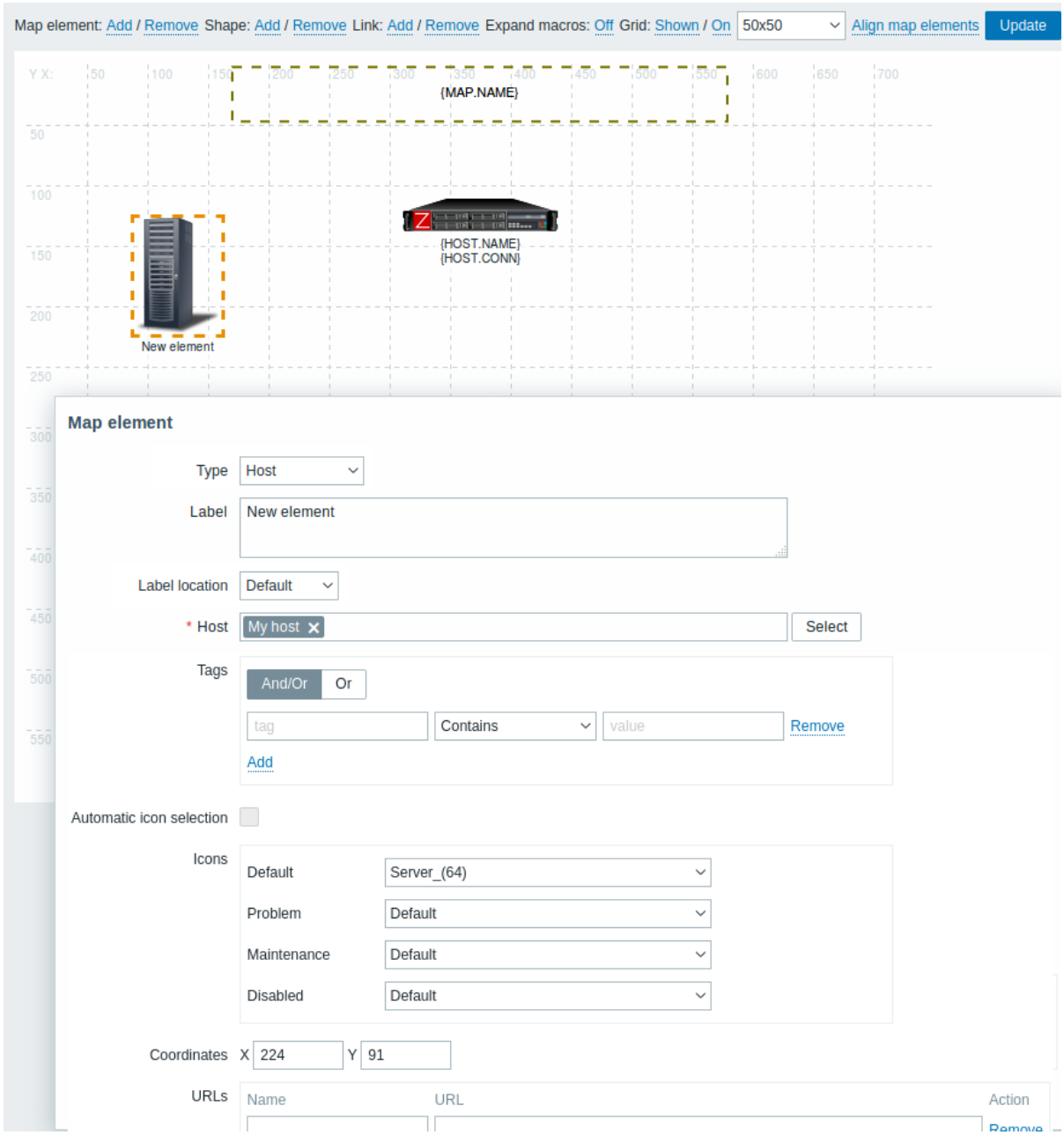
When you click on *Add* to save this map, you have created an empty map with a name, dimensions, and certain preferences. Now you need to add some elements. For that, click on *Constructor* in the map list to open the editable area.

Adding elements

To add an element, click on *Add* next to `Map element`. The new element will appear at the top left corner of the map. Drag and drop it wherever you like.

Note that with the Grid option "On", elements will always align to the grid (you can pick various grid sizes from the dropdown, also hide/show the grid). If you want to put elements anywhere without alignment, turn the option to "Off". (Random elements can later again be aligned to the grid by clicking on *Align map elements*.)

Now that you have some elements in place, you may want to start differentiating them by giving names, etc. By clicking on the element, a form is displayed and you can set the element type, give a name, choose a different icon, etc.



Map element attributes:

Parameter	Description
Type	Type of the element: **Host** - icon representing status of all triggers of the selected host **Map** - icon representing status of all elements of a map **Trigger** - icon representing status of one or more triggers **Host group** - icon representing status of all triggers of all hosts belonging to the selected group **Image** - an icon, not linked to any resource
Label	Icon label, any string. Macros and multiline strings can be used. Expression macros are supported in this field, but only with `avg`, `last`, `min` and `max` functions, with time as parameter (for example, `{?avg(/host/key,1h)}`). For a full list of supported macros, see supported macros and search for 'map element labels'.
Label location	Label location in relation to the icon: **Default** - map's default label location **Bottom** - beneath the icon **Left** - to the left **Right** - to the right **Top** - above the icon

Parameter	Description
Host	Enter the host if the element type is 'Host'. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. Scroll down to select. Click on 'x' to remove the selected.
Map	Select the map if the element type is 'Map'. This field is auto-complete so starting to type the name of a map will offer a dropdown of matching maps. Scroll down to select. Click on 'x' to remove the selected.
Triggers	If the element type is 'Trigger', select one or more triggers in the *New triggers* field below and click on *Add*. The order of selected triggers can be changed, but only within the same severity of triggers. Multiple trigger selection also affects {HOST.*} macro resolution both in the construction and view modes. // 1 In construction mode// the first displayed {HOST.*} macros will be resolved depending on the first trigger in the list (based on trigger severity). // 2 View mode// depends on the Display problems parameter in General map attributes. * If *Expand single problem* mode is chosen the first displayed {HOST.*} macros will be resolved depending on the latest detected problem trigger (not mattering the severity) or the first trigger in the list (in case no problem detected); * If *Number of problems and expand most critical one* mode is chosen the first displayed {HOST.*} macros will be resolved depending on the trigger severity.
Host group	Enter the host group if the element type is 'Host group'. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Scroll down to select. Click on 'x' to remove the selected.
Tags	Specify tags to limit the number of problems displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met This field is available for host and host group element types.
Automatic icon selection	In this case an icon mapping will be used to determine which icon to display.
Icons	You can choose to display different icons for the element in these cases: default, problem, maintenance, disabled.
Coordinate X	X coordinate of the map element.
Coordinate Y	Y coordinate of the map element.
URLs	Element-specific URLs can be set for the element. These will be displayed as links when a user clicks on the element in the map viewing mode. If the element has its own URLs and there are map level URLs for its type defined, they will be combined in the same menu. Macros can be used in map element names and values. For a full list, see supported macros and search for 'map URL names and values'.

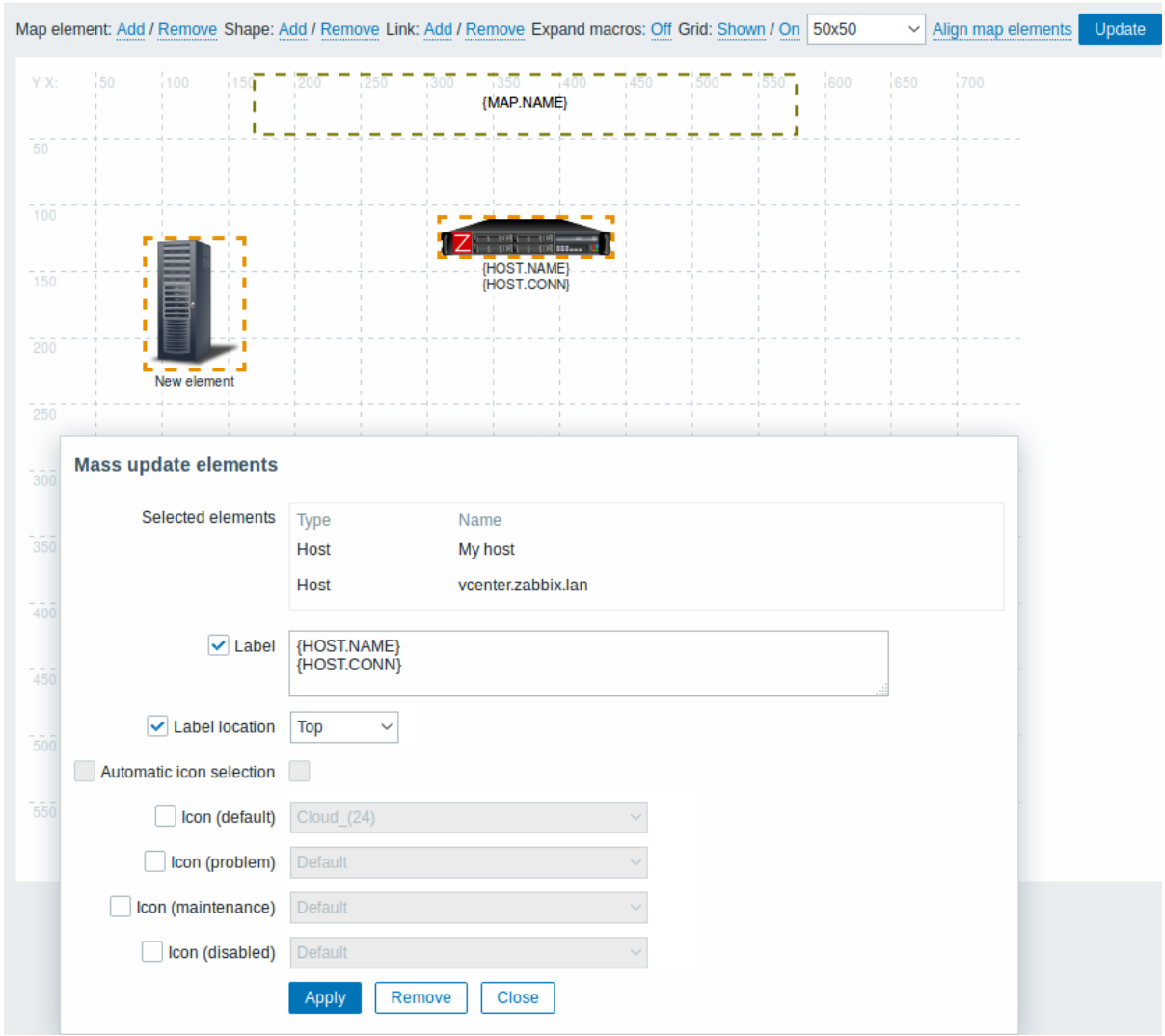
Attention:
Added elements are not automatically saved. If you navigate away from the page, all changes may be lost.
Therefore it is a good idea to click on the **Update** button in the top right corner. Once clicked, the changes are saved regardless of what you choose in the following popup.
Selected grid options are also saved with each map.

Selecting elements

To select elements, select one and then hold down *Ctrl* to select the others.

You can also select multiple elements by dragging a rectangle in the editable area and selecting all elements in it.

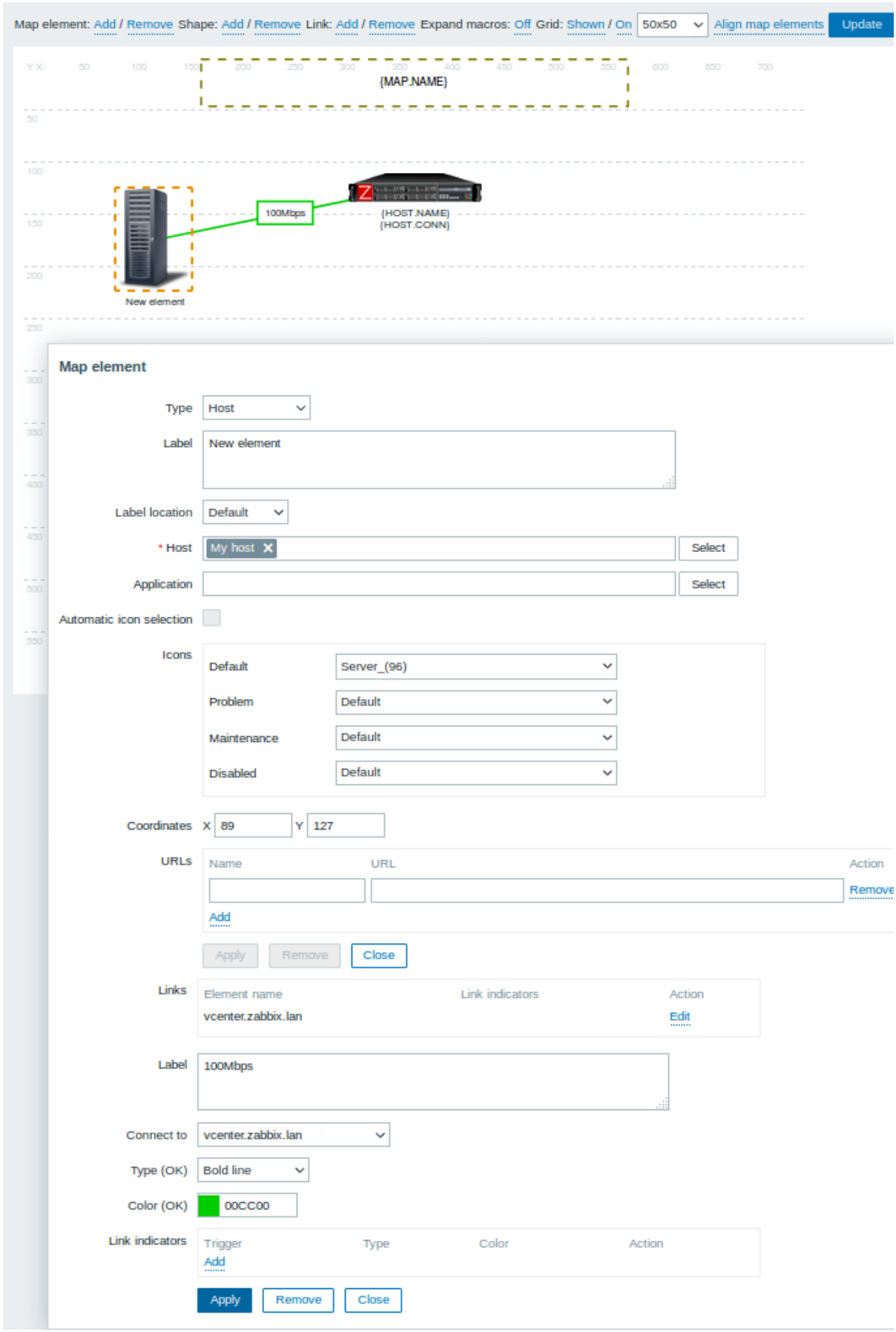
Once you select more than one element, the element property form shifts to the mass-update mode so you can change attributes of selected elements in one go. To do so, mark the attribute using the checkbox and enter a new value for it. You may use macros here (for example, {HOST.NAME} for the element label).



Linking elements

Once you have put some elements on the map, it is time to start linking them. To link two elements you must first select them. With the elements selected, click on *Add* next to Link.

With a link created, the single element form now contains an additional *Links* section. Click on *Edit* to edit link attributes.



Link attributes:

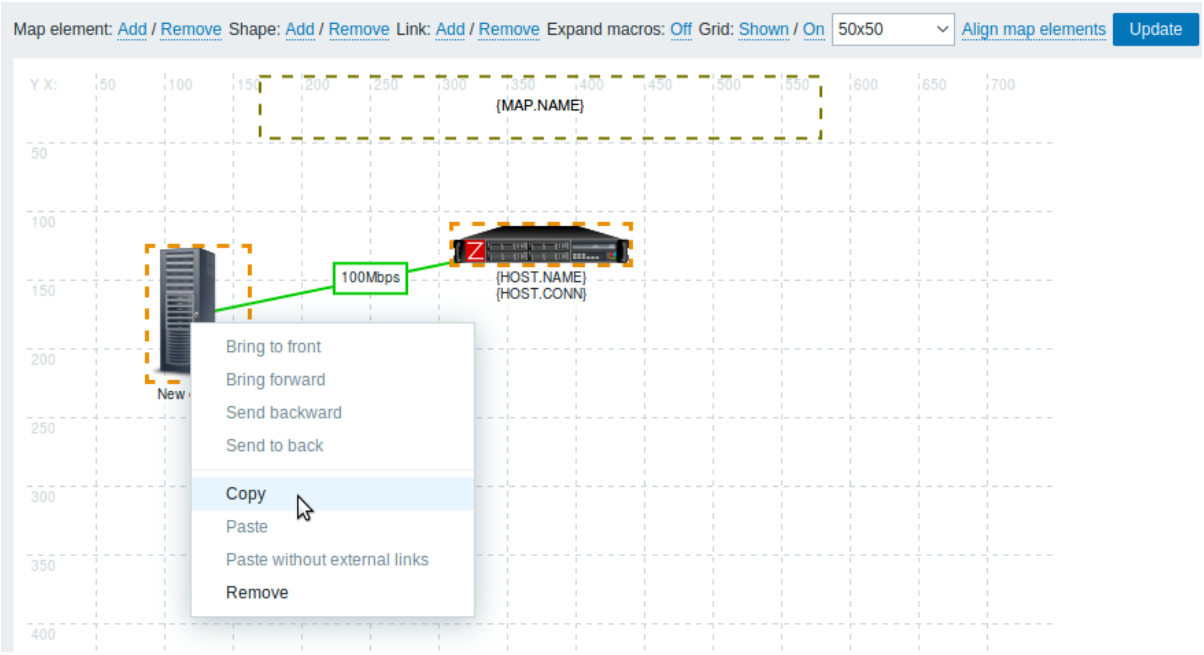
Parameter	Description
Label	Label that will be rendered on top of the link. Expression macros are supported in this field, but only with `avg`, `last`, `min` and `max` functions, with time as parameter (for example, `{?avg(/host/key,1h)}`).

Parameter	Description
Connect to	The element that the link connects to.
Type (OK)	Default link style: **Line** - single line **Bold line** - bold line **Dot** - dots **Dashed line** - dashed line
Color (OK)	Default link color.
Link indicators	List of triggers linked to the link. In case a trigger has status PROBLEM, its style is applied to the link.

Moving and copy-pasting elements

Several selected elements can be **moved** to another place in the map by clicking on one of the selected elements, holding down the mouse button, and moving the cursor to the desired location.

One or more elements can be **copied** by selecting the elements, then clicking on a selected element with the right mouse button and selecting *Copy* from the menu.



To paste the elements, click on a map area with the right mouse button and select *Paste* from the menu. The *Paste without external links* option will paste the elements retaining only the links that are between the selected elements.

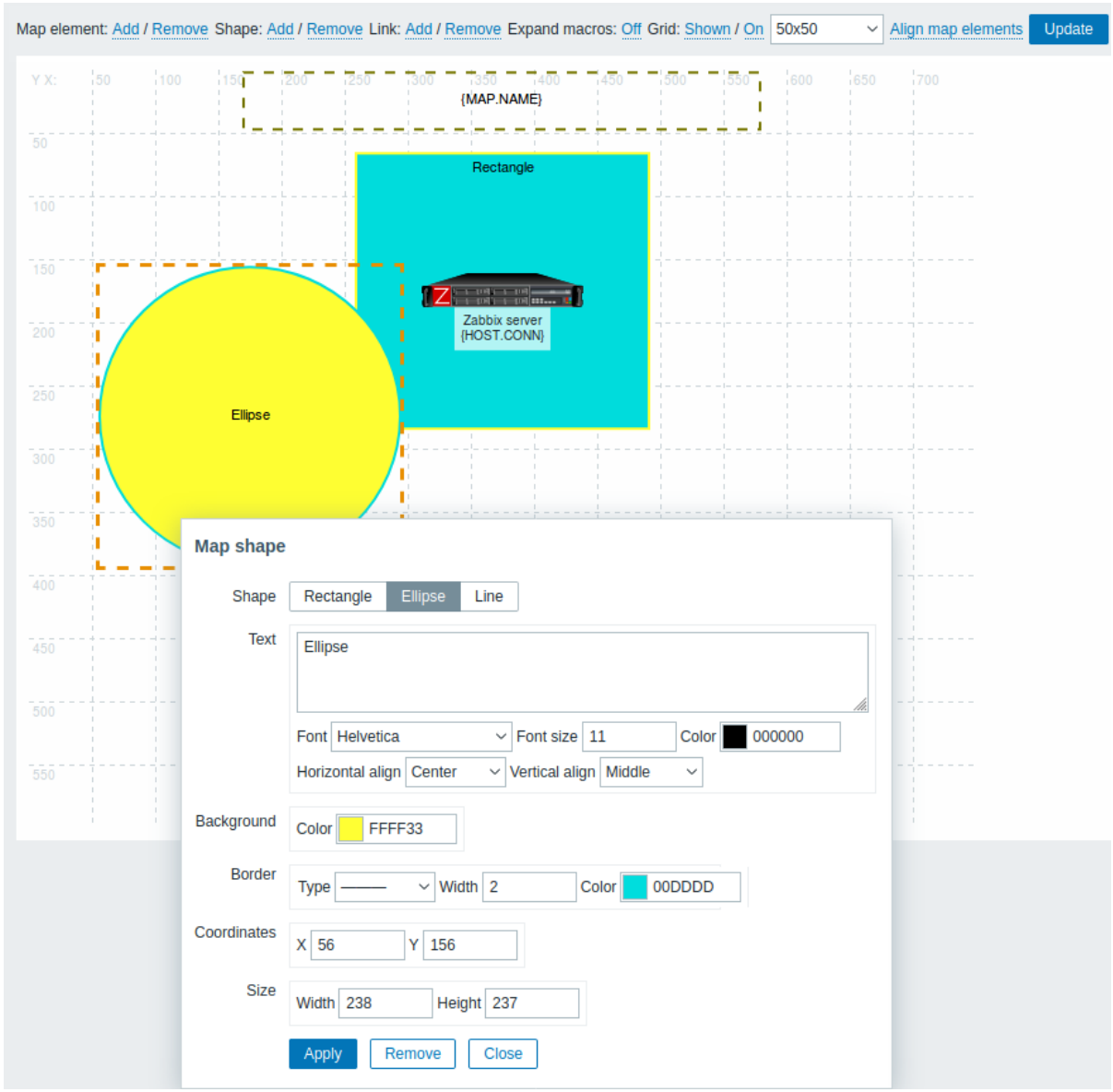
Copy-pasting works within the same browser window. Keyboard shortcuts are not supported.

Adding shapes

In addition to map elements, it is also possible to add some shapes. Shapes are not map elements; they are just a visual representation. For example, a rectangle shape can be used as a background to group some hosts. Rectangle and ellipse shapes can be added.

To add a shape, click on *Add* next to Shape. The new shape will appear at the top left corner of the map. Drag and drop it wherever you like.

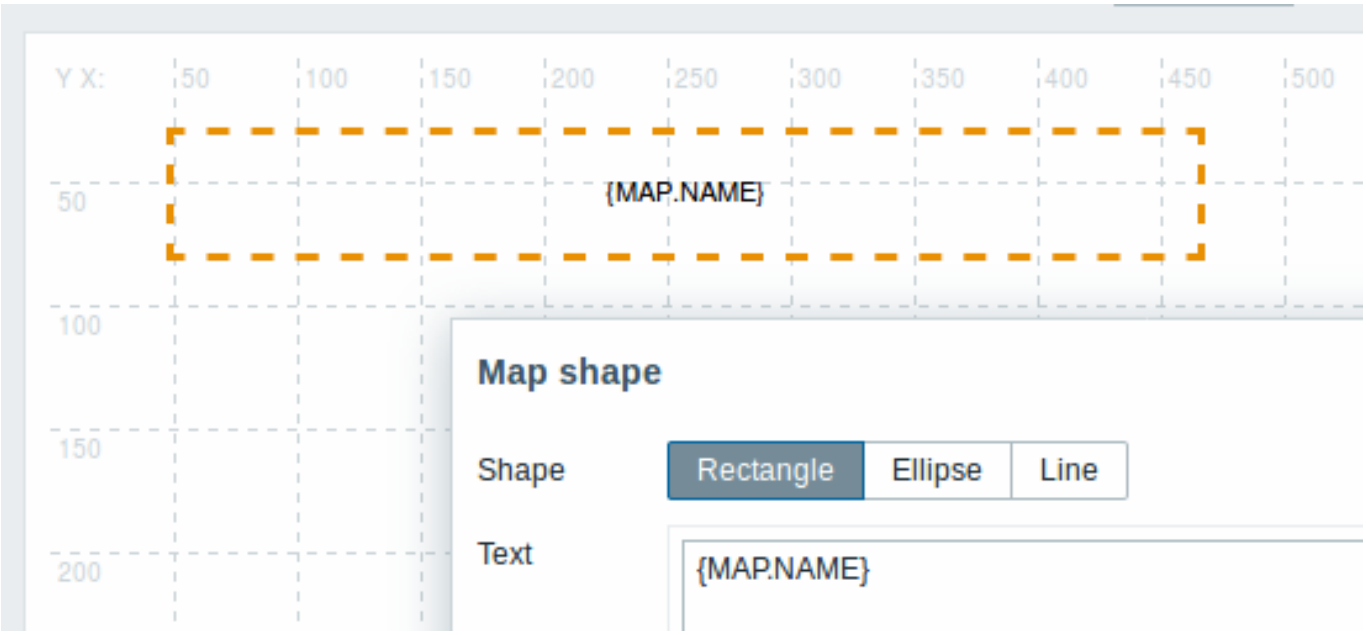
A new shape is added with default colors. By clicking on the shape, a form is displayed and you can customize the way a shape looks, add text, etc.



To select shapes, select one and then hold down *Ctrl* to select the others. With several shapes selected, common properties can be mass updated, similarly as with elements.

Text can be added in the shapes. Expression macros are supported in the text, but only with `avg`, `last`, `min` and `max` functions, with time as parameter (for example, `{?avg(/host/key,1h)}`).

To display text only the shape can be made invisible by removing the shape border (select 'None' in the *Border* field). For example, take note of how the {MAP.NAME} macro, visible in the screenshot above, is actually a rectangle shape with text, which can be seen when clicking on the macro:



{MAP.NAME} resolves to the configured map name when viewing the map.

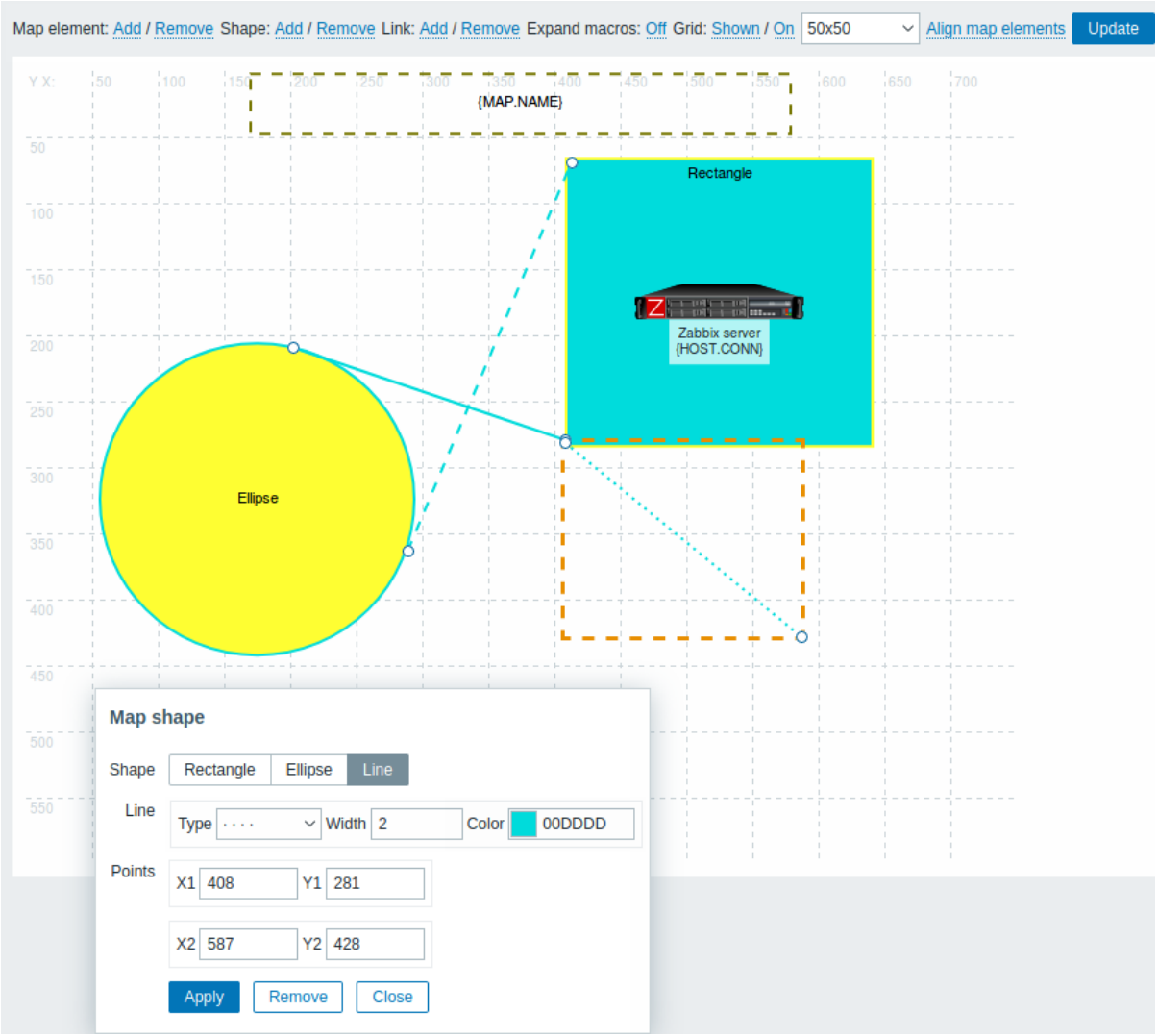
If hyperlinks are used in the text, they become clickable when viewing the map.

Line wrapping for text is always ”on” within shapes. However, within an ellipse, the lines are wrapped as though the ellipse were a rectangle. Word wrapping is not implemented, so long words (words that do not fit the shape) are not wrapped, but are masked (constructor page) or clipped (other pages with maps).

Adding lines

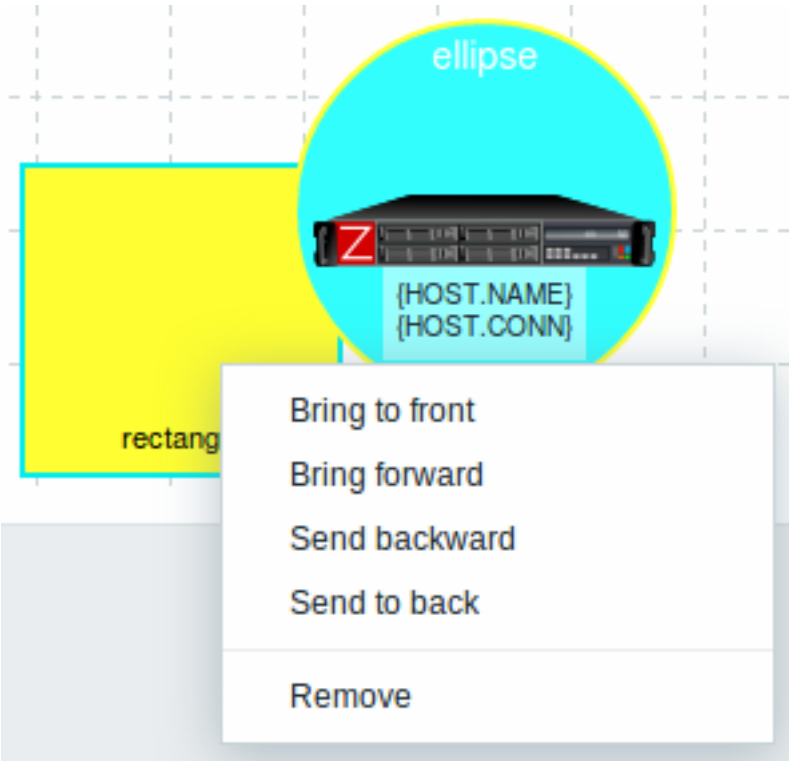
In addition to shapes, it is also possible to add some lines. Lines can be used to link elements or shapes in a map.

To add a line, click on *Add* next to Shape. A new shape will appear at the top left corner of the map. Select it and click on *Line* in the editing form to change the shape into a line. Then adjust line properties, such as line type, width, color, etc.



Ordering shapes and lines

To bring one shape in front of the other (or vice versa) click on the shape with the right mouse button bringing up the map shape menu.

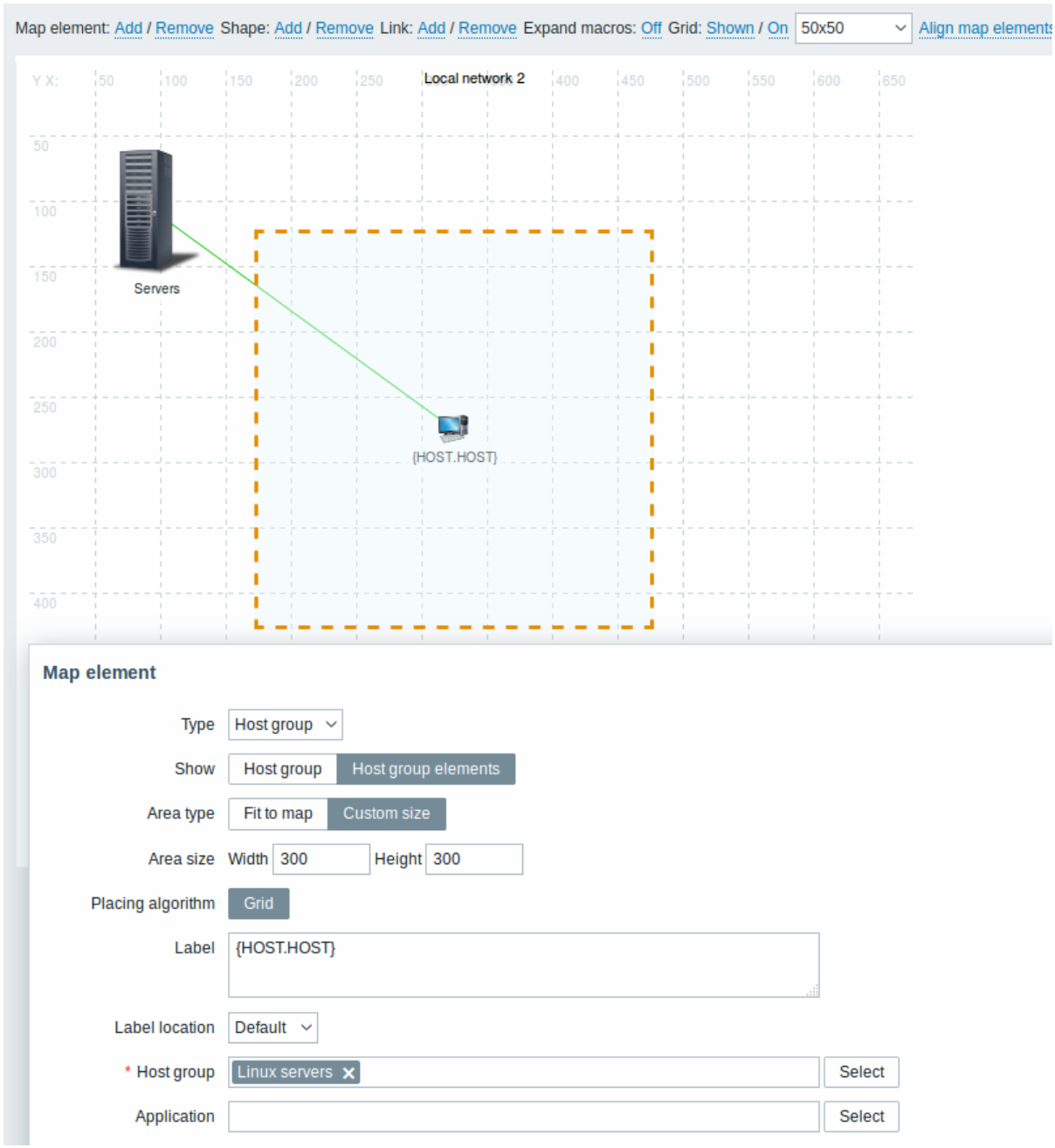


2 Host group elements

Overview

This section explains how to add a “Host group” type element when configuring a network map.

Configuration



All mandatory input fields are marked with a red asterisk.

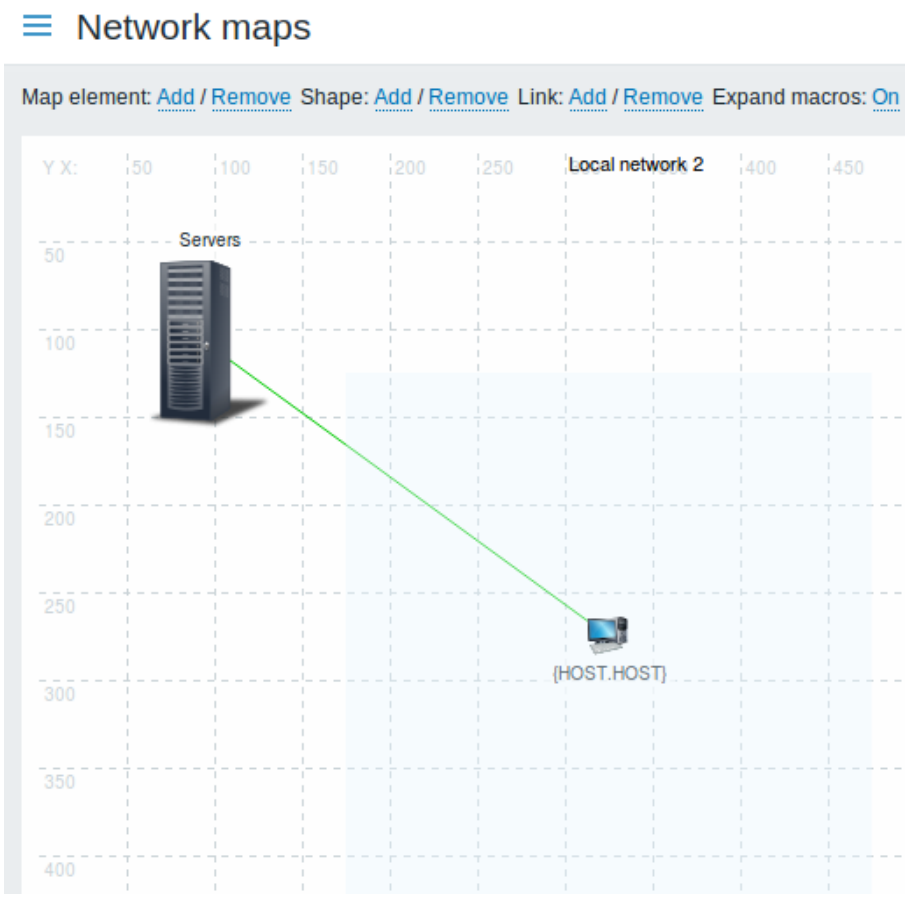
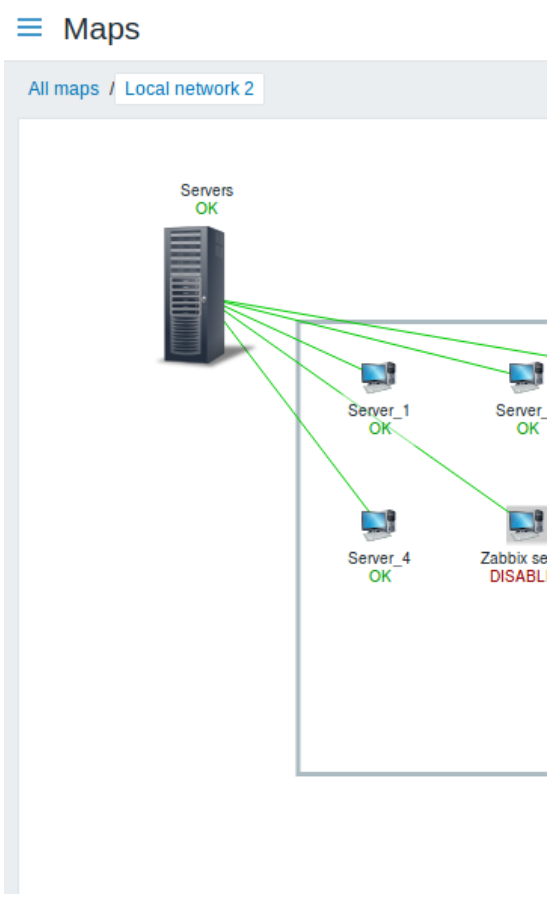
This table consists of parameters typical for *Host group* element type:

Parameter	Description
Type	Select Type of the element: **Host group** - icon representing the status of all triggers of all hosts belonging to the selected group
Show	Show options: **Host group** - selecting this option will result as one single icon displaying corresponding information about the certain host group **Host group elements** - selecting this option will result as multiple icons displaying corresponding information about every single element (host) of the certain host group

Parameter	Description
Area type	This setting is available if the “Host group elements” parameter is selected: **Fit to map** - all host group elements are equally placed within the map **Custom size** - a manual setting of the map area for all the host group elements to be displayed
Area size	This setting is available if “Host group elements” parameter and “Area type” parameter are selected: **Width** - numeric value to be entered to specify map area width **Height** - numeric value to be entered to specify map area height
Placing algorithm	**Grid** – only available option of displaying all the host group elements
Label	Icon label, any string. Macros and multiline strings can be used in labels. If the type of the map element is “Host group” specifying certain macros has an impact on the map view displaying corresponding information about every single host. For example, if {HOST.IP} macro is used, the edit map view will only display the macro {HOST.IP} itself while map view will include and display each host’s unique IP address

Viewing host group elements

This option is available if the ”Host group elements” show option is chosen. When selecting ”Host group elements” as the *show* option, you will at first see only one icon for the host group. However, when you save the map and then go to the map view, you will see that the map includes all the elements (hosts) of the certain host group:

Map editing view	Map view
	

Notice how the {HOST.NAME} macro is used. In map editing, the macro name is unresolved, while in map view all the unique names of the hosts are displayed.

3 Link indicators

Overview

You can assign some triggers to a link between elements in a network map. When these triggers go into a problem state, the link can reflect that.

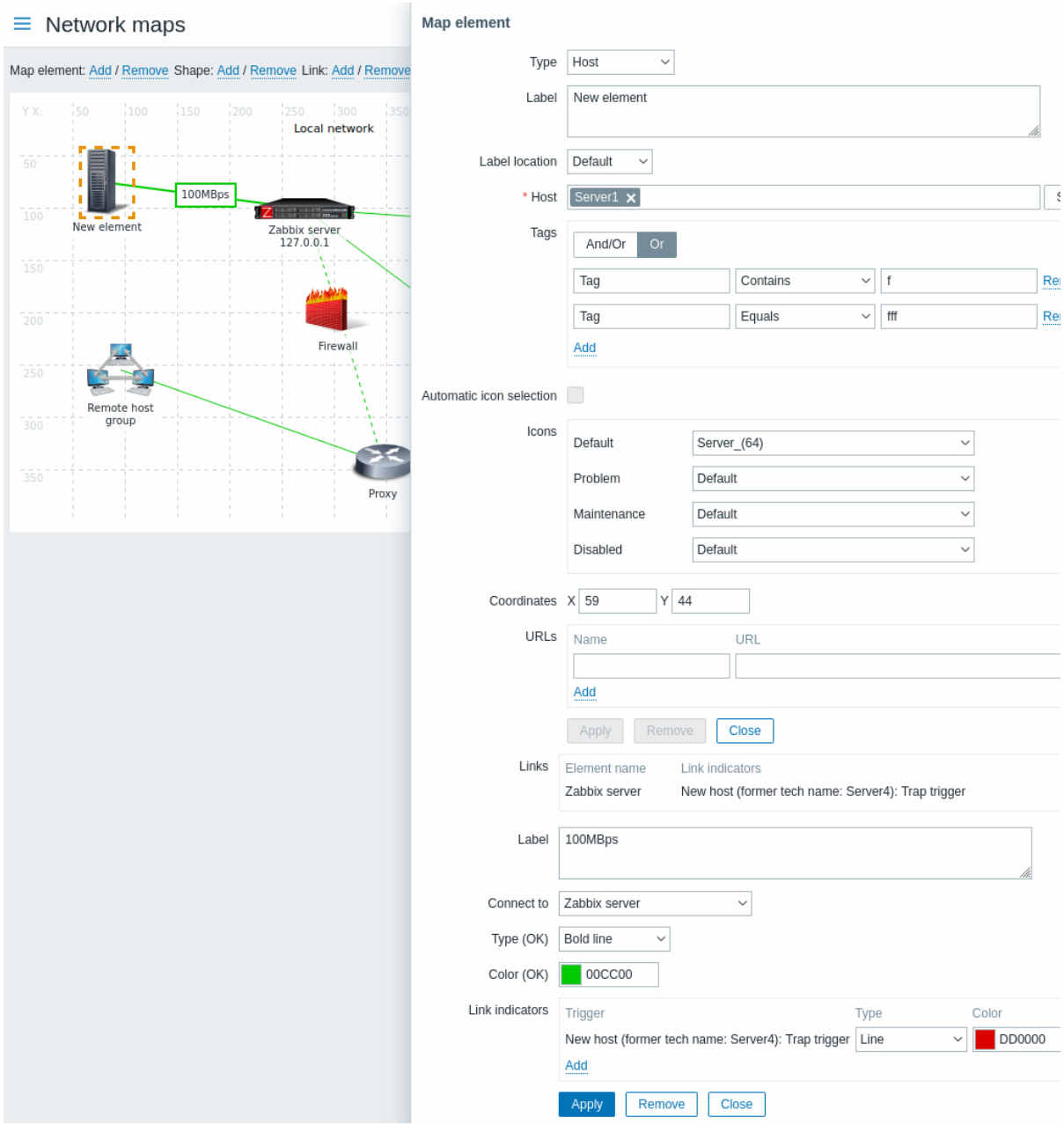
When you configure a link, you set the default link type and color. When you assign triggers to a link, you can assign different link types and colors with these triggers.

Should any of these triggers go into a problem state, their link style and color will be displayed on the link. So maybe your default link was a green line. Now, with the trigger in the problem state, your link may become bold red (if you have defined it so).

Configuration

To assign triggers as link indicators, do the following:

- select a map element
- click on *Edit* in the *Links* section for the appropriate link
- click on *Add* in the *Link indicators* block and select one or more triggers



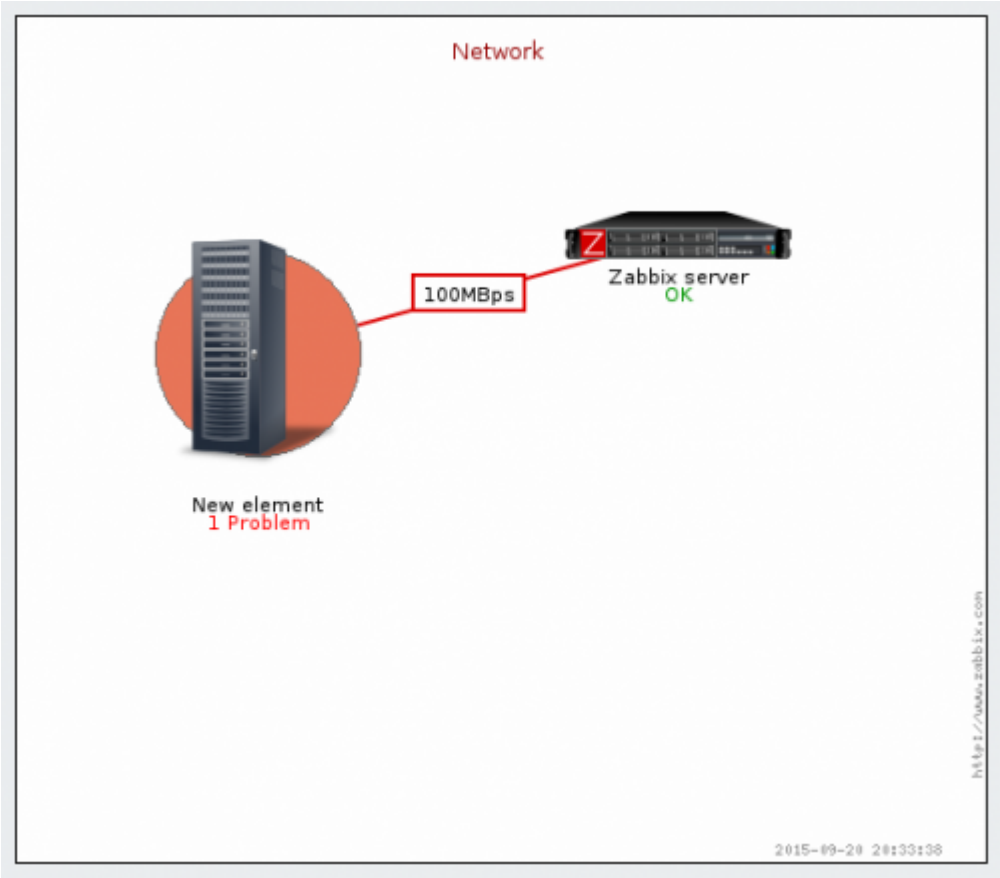
All mandatory input fields are marked with a red asterisk.

Added triggers can be seen in the *Link indicators* list.

You can set the link type and color for each trigger directly from the list. When done, click on *Apply*, close the form and click on *Update* to save the map changes.

Display

In *Monitoring* → *Maps* the respective color will be displayed on the link if the trigger goes into a problem state.

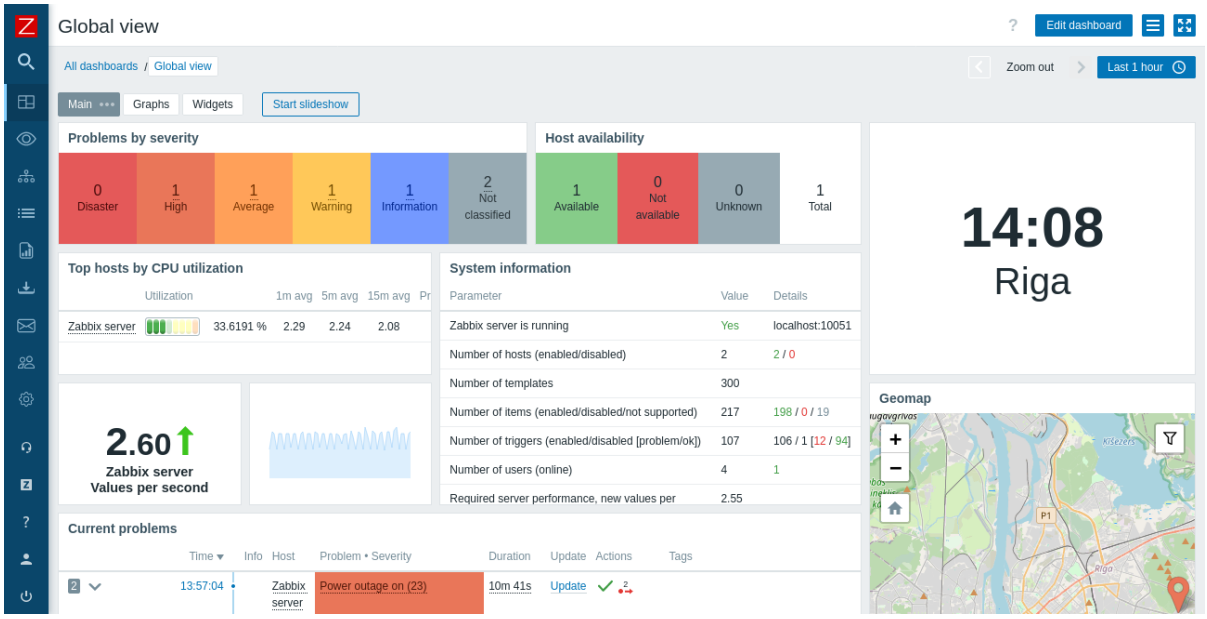


Note:
If multiple triggers go into a problem state, the problem with the highest severity will determine the link style and color. If multiple triggers with the same severity are assigned to the same map link, the one with the lowest ID takes precedence. Note also that:

1. *Minimum trigger severity* and *Show suppressed problem* settings from map configuration affect which problems are taken into account.
2. In the case of triggers with multiple problems (multiple problem generation), each problem may have a severity that differs from trigger severity (changed manually), may have different tags (due to macros), and may be suppressed.

3 Dashboards

Dashboards and their widgets provide a strong visualization platform with such tools as modern graphs, maps, slideshows, and many more.



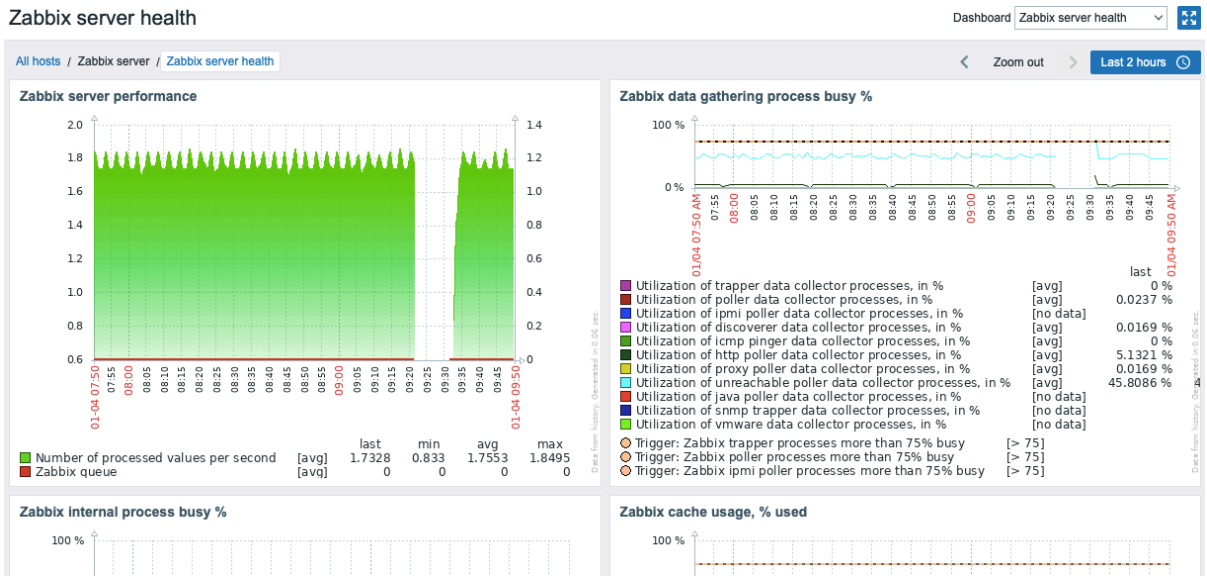
4 Host dashboards

Overview

Host dashboards look similar to global dashboards, however, host dashboards display data about the host only. Host dashboards have no owner.

Host dashboards are configured on the template level and then are generated for a host, once the template is linked to the host. Widgets of host dashboards can only be copied to host dashboards of the same template. Widgets from global dashboards cannot be copied onto host dashboards.

Host dashboards *cannot* be configured or directly accessed in the *Dashboards* section, which is reserved for global dashboards. The ways to access host dashboards are listed below in this section.



When viewing host dashboards you may switch between the configured dashboards using the dropdown in the upper right corner. To switch to *Monitoring→Hosts* section, click *All hosts* navigation link below the dashboard name in the upper left corner.

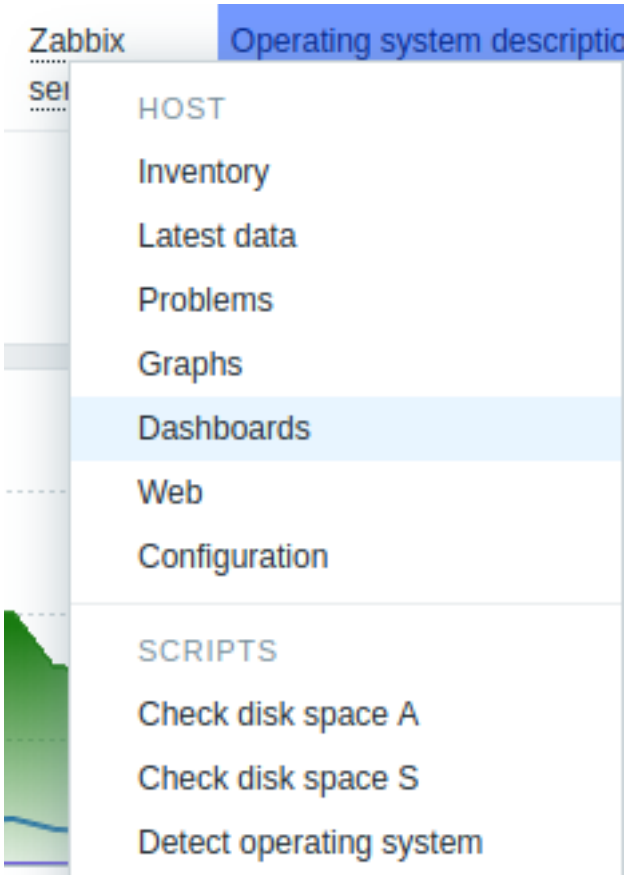
Widgets of the host dashboards cannot be edited.

Note that host dashboards used to be host screens before Zabbix 5.2. When importing an older template containing screens, the screen import will be ignored.

Accessing host dashboards

Access to host dashboards is provided:

- From the host menu that is available in many frontend locations:
  - click on the host name and then select *Dashboards* from the menu



- When searching for a host name in global search:
  - click on the *Dashboards* link provided in search results
- When clicking on a host name in *Inventory* → *Hosts*:
  - click on the *Dashboards* link provided

8 Templates and template groups

Overview

The use of templates is an excellent way of reducing one's workload and streamlining the Zabbix configuration. A template is a set of entities that can be conveniently applied to multiple hosts.

The entities may be:

- items
- triggers
- graphs
- dashboards
- low-level discovery rules
- web scenarios

As many hosts in real life are identical or fairly similar so it naturally follows that the set of entities (items, triggers, graphs,...) you have created for one host, may be useful for many. Of course, you could copy them to each new host, but that would be a lot of manual work. Instead, with templates you can copy them to one template and then apply the template to as many hosts as needed.

When a template is linked to a host, all entities (items, triggers, graphs,...) of the template are added to the host. Templates are assigned to each individual host directly (and not to a host group).

Templates are often used to group entities for particular services or applications (like Apache, MySQL, PostgreSQL, Postfix...) and then applied to hosts running those services.

Another benefit of using templates is when something has to be changed for all the hosts. Changing something on the template level once will propagate the change to all the linked hosts.

Templates are organized in template groups.

Proceed to creating and configuring a template.

9 Templates out of the box

Overview

Zabbix strives to provide a growing list of useful out-of-the-box templates. Out-of-the-box templates come preconfigured and thus are a useful way for speeding up the deployment of monitoring jobs.

The templates are available:

- In new installations - in *Data collection* → *Templates*;
- If you are upgrading Zabbix, the upgraded installation might lack newer templates. You can find them in the Zabbix Git repository - select the version you upgraded to. To add a new template, download the template file, open Zabbix frontend, go to *Data collection* → *Templates*, and import the file.

Please use the sidebar to access information about specific template types and operation requirements.

See also:

- Template import
- Linking a template
- Known issues

1 Zabbix agent template operation

Steps to ensure correct operation of templates that collect metrics with Zabbix agent:

1. Make sure that Zabbix agent is installed on the host. For active checks, also make sure that the host is added to the 'ServerActive' parameter of the agent configuration file.
2. Link the template to a target host (if the template is not available in your Zabbix installation, you may need to import the template's .xml file first - see Templates out-of-the-box section for instructions).
3. If necessary, adjust the values of template macros.
4. Configure the instance being monitored to allow sharing data with Zabbix.

A detailed description of a template, including the full list of macros, items and triggers is available in the template's Readme.md file (accessible by clicking on a template name).

The following templates are available:

- Apache by Zabbix agent
- HAProxy by Zabbix agent
- IIS by Zabbix agent
- IIS by Zabbix agent active
- Microsoft Exchange Server 2016 by Zabbix agent
- Microsoft Exchange Server 2016 by Zabbix agent active
- MySQL by Zabbix agent
- Nginx by Zabbix agent
- PHP-FPM by Zabbix agent
- PostgreSQL by Zabbix agent
- RabbitMQ cluster by Zabbix agent

2 Zabbix agent 2 template operation

Steps to ensure correct operation of templates that collect metrics with Zabbix agent 2:

1. Make sure that the agent 2 is installed on the host, and that the installed version contains the required plugin. In some cases, you may need to upgrade the agent 2 first.
2. Link the template to a target host (if the template is not available in your Zabbix installation, you may need to import the template's import file first - see Templates out-of-the-box section for instructions).
3. If necessary, adjust the values of template macros. Note that user macros can be used to override configuration parameters.
4. Configure the instance being monitored to allow sharing data with Zabbix.

Attention:
Zabbix agent 2 templates work in conjunction with the plugins. While the basic configuration can be done by simply adjusting user macros, the deeper customization can be achieved by configuring the plugin itself. For example, if a plugin supports named sessions, it is possible to monitor several entities of the same kind (e.g. MySQL1 and MySQL2) by specifying named session with own URI, username and password for each entity in the configuration file.

A detailed description of a template, including the full list of macros, items and triggers is available in the template's Readme.md file (accessible by clicking on a template name).

The following templates are available:

- Ceph by Zabbix agent 2
- Docker
- Memcached
- MongoDB cluster by Zabbix agent 2
- MongoDB node by Zabbix agent 2
- MySQL by Zabbix agent 2
- Oracle by Zabbix agent 2
- PostgreSQL by Zabbix agent 2
- Redis
- SMART by Zabbix agent 2
- SMART by Zabbix agent 2 active
- Systemd by Zabbix agent 2
- Website certificate by Zabbix agent 2

3 HTTP template operation

Steps to ensure correct operation of templates that collect metrics with HTTP agent:

1. Create a host in Zabbix and specify an IP address or DNS name of the monitoring target as the main interface. This is needed for the {HOST.CONN} macro to resolve properly in the template items.
2. Link the template to the host created in step 1 (if the template is not available in your Zabbix installation, you may need to import the template's .xml file first - see Templates out-of-the-box section for instructions).
3. If necessary, adjust the values of template macros.
4. Configure the instance being monitored to allow sharing data with Zabbix.

A detailed description of a template, including the full list of macros, items and triggers is available in the template's Readme.md file (accessible by clicking on a template name).

The following templates are available:

- Acronis Cyber Protect Cloud by HTTP
- Apache by HTTP
- Asterisk by HTTP
- AWS by HTTP
- AWS Cost Explorer by HTTP
- AWS EC2 by HTTP
- AWS ECS Cluster by HTTP (available since 6.4.5)
- AWS ECS Serverless Cluster by HTTP (available since 6.4.5)
- AWS ELB Application Load Balancer by HTTP
- AWS ELB Network Load Balancer by HTTP
- AWS RDS instance by HTTP
- AWS S3 bucket by HTTP
- Azure by HTTP
- Cisco Meraki organization by HTTP
- Cisco SD-WAN by HTTP
- ClickHouse by HTTP
- Cloudflare by HTTP
- CockroachDB by HTTP
- Control-M enterprise manager by HTTP
- Control-M server by HTTP
- DELL PowerEdge R720 by HTTP
- DELL PowerEdge R740 by HTTP
- DELL PowerEdge R820 by HTTP
- DELL PowerEdge R840 by HTTP

- Elasticsearch Cluster by HTTP
- Envoy Proxy by HTTP
- Etcd by HTTP
- FortiGate by HTTP
- GitLab by HTTP
- Google Cloud Platform (GCP) by HTTP
- Hadoop by HTTP
- HAProxy by HTTP
- HashiCorp Consul Cluster by HTTP
- HashiCorp Consul Node by HTTP
- HashiCorp Nomad by HTTP
- HashiCorp Vault by HTTP
- Hikvision camera by HTTP
- HPE iLO by HTTP
- HPE MSA 2040 Storage by HTTP
- HPE MSA 2060 Storage by HTTP
- HPE Primera by HTTP
- HPE Synergy by HTTP
- InfluxDB by HTTP
- Jenkins by HTTP
- Kubernetes API server by HTTP
- Kubernetes cluster state by HTTP
- Kubernetes Controller manager by HTTP
- Kubernetes kubelet by HTTP
- Kubernetes nodes by HTTP
- Kubernetes Scheduler by HTTP
- MantisBT by HTTP
- Microsoft SharePoint by HTTP
- NetApp AFF A700 by HTTP
- Nextcloud by HTTP
- NGINX by HTTP
- NGINX Plus by HTTP
- OpenStack by HTTP
- OpenWeatherMap by HTTP
- Oracle Cloud by HTTP
- PHP-FPM by HTTP
- Proxmox VE by HTTP
- RabbitMQ cluster by HTTP
- TiDB by HTTP
- TiDB PD by HTTP
- TiDB TiKV by HTTP
- Travis CI by HTTP
- Veeam Backup Enterprise Manager by HTTP
- Veeam Backup and Replication by HTTP
- VMware SD-WAN VeloCloud by HTTP
- YugabyteDB by HTTP
- ZooKeeper by HTTP

4 IPMI template operation

IPMI templates do not require any specific setup. To start monitoring, link the template to a target host (if the template is not available in your Zabbix installation, you may need to import the template's .xml file first - see Templates out-of-the-box section for instructions).

A detailed description of a template, including the full list of macros, items and triggers is available in the template's Readme.md file (accessible by clicking on a template name).

Available template:

- Chassis by IPMI

5 JMX template operation

Steps to ensure correct operation of templates that collect metrics by JMX:

1. Make sure Zabbix Java gateway is installed and set up properly.
2. Link the template to the target host. The host should have JMX interface set up.
If the template is not available in your Zabbix installation, you may need to import the template's .xml file first - see Templates out-of-the-box section for instructions.
3. If necessary, adjust the values of template macros.
4. Configure the instance being monitored to allow sharing data with Zabbix.

A detailed description of a template, including the full list of macros, items and triggers is available in the template's Readme.md file (accessible by clicking on a template name).

The following templates are available:

- Apache ActiveMQ by JMX
- Apache Cassandra by JMX
- Apache Kafka by JMX
- Apache Tomcat by JMX
- GridGain by JMX
- Ignite by JMX
- Jira Data Center by JMX
- WildFly Domain by JMX
- WildFly Server by JMX

6 ODBC template operation

Steps to ensure correct operation of templates that collect metrics via ODBC monitoring:

1. Make sure that required ODBC driver is installed on Zabbix server or proxy.
2. Link the template to a target host (if the template is not available in your Zabbix installation, you may need to import the template's .xml file first - see Templates out-of-the-box section for instructions).
3. If necessary, adjust the values of template macros.
4. Configure the instance being monitored to allow sharing data with Zabbix.

A detailed description of a template, including the full list of macros, items and triggers is available in the template's Readme.md file (accessible by clicking on a template name).

The following templates are available:

- MSSQL by ODBC
- MySQL by ODBC
- Oracle by ODBC
- PostgreSQL by ODBC

7 Standardized templates for network devices

Overview

In order to provide monitoring for network devices such as switches and routers, we have created two so-called models: for the network device itself (its chassis basically) and for network interface.

Since Zabbix 3.4 templates for many families of network devices are provided. All templates cover (where possible to get these items from the device):

- Chassis fault monitoring (power supplies, fans and temperature, overall status)
- Chassis performance monitoring (CPU and memory items)
- Chassis inventory collection (serial numbers, model name, firmware version)
- Network interface monitoring with IF-MIB and EtherLike-MIB (interface status, interface traffic load, duplex status for Ethernet)

These templates are available:

- In new installations - in *Data collection* → *Templates*;
- If you are upgrading from previous versions, you can find these templates in the *zabbix/templates* directory of the downloaded latest Zabbix version. While in *Data collection* → *Templates* you can import them manually from this directory.

If you are importing the new out-of-the-box templates, you may want to also update the `@Network interfaces for discovery` global regular expression to:

```
Result is FALSE: ^Software Loopback Interface
Result is FALSE: ^(In)?[lL]oop[bB]ack[0-9._]*$
Result is FALSE: ^NULL[0-9.]*$
Result is FALSE: ^[lL]o[0-9.]*$
Result is FALSE: ^[sS]ystem$
Result is FALSE: ^Nu[0-9.]*$
```

to filter out loopbacks and null interfaces on most systems.

Devices

List of device families for which templates are available:

Template name	Vendor	Device family	Known models	OS	MIBs used	**Tags**
Alcatel Timetra TiMOS SNMP	Alcatel	Alcatel Timetra	ALCATEL SR 7750	TiMOS	TIMETRA-SYSTEM-MIB,TIMETRA-CHASSIS-MIB	Certified
Brocade FC SNMP	Brocade	Brocade FC switches	Brocade 300 SAN Switch-	-	SW-MIB,ENTITY-MIB	Performance Fault
Brocade_Foundry Stackable SNMP	Brocade	Brocade ICX	Brocade ICX6610, Brocade ICX7250-48, Brocade ICX7450-48F		FOUNDRY-SN-AGENT-MIB, FOUNDRY-SN-STACKING-MIB	Certified
Brocade_Foundry Nonstackable SNMP	Brocade, Foundry	Brocade MLX, Foundry	Brocade MLXe, Foundry FLS648, Foundry FWSX424		FOUNDRY-SN-AGENT-MIB	Performance Fault
Check Point Next Generation Firewall by SNMP	Check Point	Next Genera-tion Firewall	-	Gaia	HOST-RESOURCES-MIB, CHECKPOINT-MIB, UCD-SNMP-MIB, SNMPv2-MIB, IF-MIB	Certified
Cisco Catalyst 3750<device model> SNMP	Cisco	Cisco Catalyst 3750	Cisco Catalyst 3750V2-24FS, Cisco Catalyst 3750V2-24PS, Cisco Catalyst 3750V2-24TS, Cisco Catalyst SNMP, Cisco Catalyst SNMP		CISCO-MEMORY-POOL-MIB, IF-MIB, EtherLike-MIB, SNMPv2-MIB, CISCO-PROCESS-MIB, CISCO-ENVMON-MIB, ENTITY-MIB	Certified
Cisco IOS SNMP	Cisco	Cisco IOS ver > 12.2 3.5	Cisco C2950	IOS	CISCO-PROCESS-MIB,CISCO-MEMORY-POOL-MIB,CISCO-ENVMON-MIB	Certified
Cisco IOS versions 12.0_3_T-12.2_3.5 SNMP	Cisco	Cisco IOS > 12.0 3 T and 12.2 3.5	-	IOS	CISCO-PROCESS-MIB,CISCO-MEMORY-POOL-MIB,CISCO-ENVMON-MIB	Certified
Cisco IOS prior to 12.0_3_T SNMP	Cisco	Cisco IOS 12.0 3 T	-	IOS	OLD-CISCO-CPU-MIB,CISCO-MEMORY-POOL-MIB	Certified
D-Link DES_DGS Switch SNMP	D-Link	DES/DGX switches	D-Link DES-xxxx/DGS-xxxx,DLINK DGS-3420-26SC	-	DLINK-AGENT-MIB,EQUIPMENT-MIB,ENTITY-MIB	Certified

Template name	Vendor	Device family	Known models	OS	MIBs used	Tags
D-Link DES 7200 SNMP	D-Link	DES-7xxx	D-Link DES 7206	-	ENTITY-MIB,MY-SYSTEM-MIB,MY-PROCESS-MIB,MY-MEMORY-MIB	Performance Fault Interfaces
Dell Force S-Series SNMP	Dell	Dell Force S-Series	S4810		F10-S-SERIES-CHASSIS-MIB	Certified
Extreme Exos SNMP	Extreme	Extreme EXOS	X670V-48x	EXOS	EXTREME-SYSTEM-MIB,EXTREME-SOFTWARE-MONITOR-MIB	Certified
FortiGate by SNMP	Fortinet	FortiGate (NGFW)	-	FortiOS	HOST-RESOURCES-MIB FORTINET-FORTIGATE-MIB FORTINET-CORE-MIB SNMPv2-MIB IF-MIB ENTITY-MIB	Performance Inventory
Huawei VRP SNMP	Huawei	Huawei VRP	S2352P-EI	-	ENTITY-MIB,HUAWEI-ENTITY-EXTENT-MIB	Certified
Intel_Qlogic Infiniband SNMP	Intel/QLogic	Intel/QLogic Infini-band devices	Infiniband 12300		ICS-CHASSIS-MIB	Fault Inventory
Juniper SNMP	Juniper	MX,SRX,EX models	Juniper MX240, Juniper EX4200-24F	JunOS	JUNIPER-MIB	Certified
Mellanox SNMP	Mellanox	Mellanox Infini-band devices	SX1036	MLNX-OS	HOST-RESOURCES-MIB,ENTITY-MIB,ENTITY-SENSOR-MIB,MELLANOX-MIB	Certified

Template name	Vendor	Device family	Known models	OS	MIBs used	**Tags**
MikroTik CCR<device model> SNMP	MikroTik	MikroTik Cloud Core Routers (CCR series)	Separate dedicated templates are available for MikroTik CCR1009-7G-1C-1S+, MikroTik CCR1009-7G-1C-1S+PC, MikroTik CCR1009-7G-1C-PC, MikroTik CCR1016-12G, MikroTik CCR1016-12S-1S+, MikroTik CCR1036-12G-4S-EM, MikroTik CCR1036-12G-4S, MikroTik CCR1036-8G-2S+, MikroTik CCR1036-8G-2S+EM, MikroTik CCR1072-1G-8S+, MikroTik CCR2004-16G-2S+, MikroTik CCR2004-1G-12S+2XS	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified

Template name	Vendor	Device family	Known models	OS	MIBs used	Tags
MikroTik CRS<device model> SNMP	MikroTik	MikroTik Cloud Router Switches (CRS series)	Separate dedicated templates are available for MikroTik CRS106-1C-5S, MikroTik CRS109-8G-1S-2HnD-IN, MikroTik CRS112-8G-4S-IN, MikroTik CRS112-8P-4S-IN, MikroTik CRS125-24G-1S-2HnD-IN, MikroTik CRS212-1G-10S-1S+IN, MikroTik CRS305-1G-4S+IN, MikroTik CRS309-1G-8S+IN, MikroTik CRS312-4C+8XG-RM, MikroTik CRS317-1G-16S+RM, MikroTik CRS326-24G-2S+IN, MikroTik CRS326-24G-2S+RM, MikroTik CRS326-24S+2Q+RM, MikroTik CRS328-24P-4S+RM, MikroTik CRS328-4C-20S-4S+RM, MikroTik CRS354-48G-4S+2Q+RM, MikroTik CRS354-48P-4S+2Q+RM	RouterOS/SwitchOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified
MikroTik CSS<device model> SNMP	MikroTik	MikroTik Cloud Smart Switches (CSS series)	Separate dedicated templates are available for MikroTik CSS326-24G-2S+RM, MikroTik CSS610-8G-2S+IN	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified
MikroTik FiberBox SNMP	MikroTik	MikroTik FiberBox	MikroTik FiberBox	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified
MikroTik hEX <device model> SNMP	MikroTik	MikroTik hEX	Separate dedicated templates are available for MikroTik hEX, MikroTik hEX lite, MikroTik hEX PoE, MikroTik hEX PoE lite, MikroTik hEX S	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified

Template name	Vendor	Device family	Known models	OS	MIBs used	Tags
MikroTik netPower <device model> SNMP	MikroTik	MikroTik net-Power	Separate dedicated templates are available for MikroTik netPower 15FR, MikroTik netPower 16P SNMP, MikroTik netPower Lite 7R	RouterOS/SwitchOS, SwitchOS Lite	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified
MikroTik PowerBox <device model> SNMP	MikroTik	MikroTik Power-Box	Separate dedicated templates are available for MikroTik PowerBox, MikroTik PowerBox Pro	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified
MikroTik RB<device model> SNMP	MikroTik	MikroTik RB series routers	Separate dedicated templates are available for MikroTik RB1100AHx4, MikroTik RB1100AHx4 Dude Edition, MikroTik RB2011iL-IN, MikroTik RB2011iL-RM, MikroTik RB2011iLS-IN, MikroTik RB2011UiAS-IN, MikroTik RB2011UiAS-RM, MikroTik RB260GS, MikroTik RB3011UiAS-RM, MikroTik RB4011iGS+RM, MikroTik RB5009UG+S+IN	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified
MikroTik SNMP	MikroTik	MikroTik RouterOS devices	MikroTik CCR1016-12G, MikroTik RB2011UAS-2HnD, MikroTik 912UAG-5HPnD, MikroTik 941-2nD, MikroTik 951G-2HnD, MikroTik 1100AHx2	RouterOS	MIKROTIK-MIB,HOST-RESOURCES-MIB	Certified
QTech QSW SNMP	QTech	Qtech devices	Qtech QSW-2800-28T	-	QTECH-MIB,ENTITY-MIB	Performance Inventory

Template name	Vendor	Device family	Known models	OS	MIBs used	Tags
Ubiquiti AirOS SNMP	Ubiquiti	Ubiquiti AirOS wireless devices	NanoBridge,NanoStation,Unifi	AirOS	FROGFOOT-RESOURCES-MIB,IEEE802dot11-MIB	Performance
HP Comware HH3C SNMP	HP	HP (H3C) Comware	HP A5500-24G-4SFP HI Switch		HH3C-ENTITY-EXT-MIB,ENTITY-MIB	Certified
HP Enterprise Switch SNMP	HP	HP Enterprise Switch	HP ProCurve J4900B Switch 2626, HP J9728A 2920-48G Switch		STATISTICS-MIB,NETSWITCH-MIB,HP-ICF-CHASSIS,ENTITY-MIB,SEMI-MIB	Certified
TP-LINK SNMP	TP-LINK	TP-LINK	T2600G-28TS v2.0		TPLINK-SYSMONITOR-MIB,TPLINK-SYSINFO-MIB	Performance Inventory
Netgear Fastpath SNMP	Netgear	Netgear Fastpath	M5300-28G		FASTPATH-SWITCHING-MIB,FASTPATH-BOXSERVICES-PRIVATE-MIB	Fault Inventory

Template design

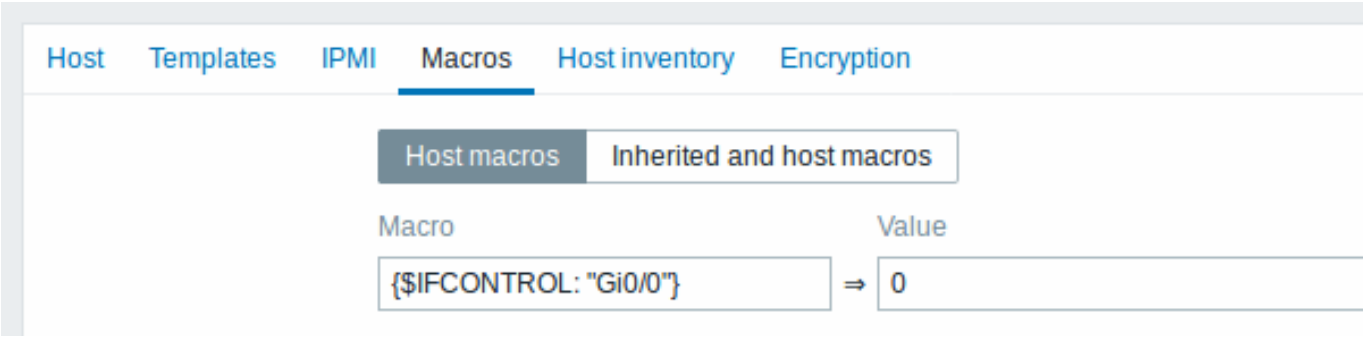
Templates were designed with the following in mind:

- User macros are used as much as possible so triggers can be tuned by the user;
- Low-level discovery is used as much as possible to minimize the number of unsupported items;
- All templates depend on Template ICMP Ping so all devices are also checked by ICMP;
- Items don't use any MIBs - SNMP OIDs are used in items and low-level discoveries. So it's not necessary to load any MIBs into Zabbix for templates to work;
- Loopback network interfaces are filtered when discovering as well as interfaces with ifAdminStatus = down(2)
- 64bit counters are used from IF-MIB::ifXTable where possible. If it is not supported, default 32bit counters are used instead.

All discovered network interfaces have a trigger that monitors its operational status (link), for example:

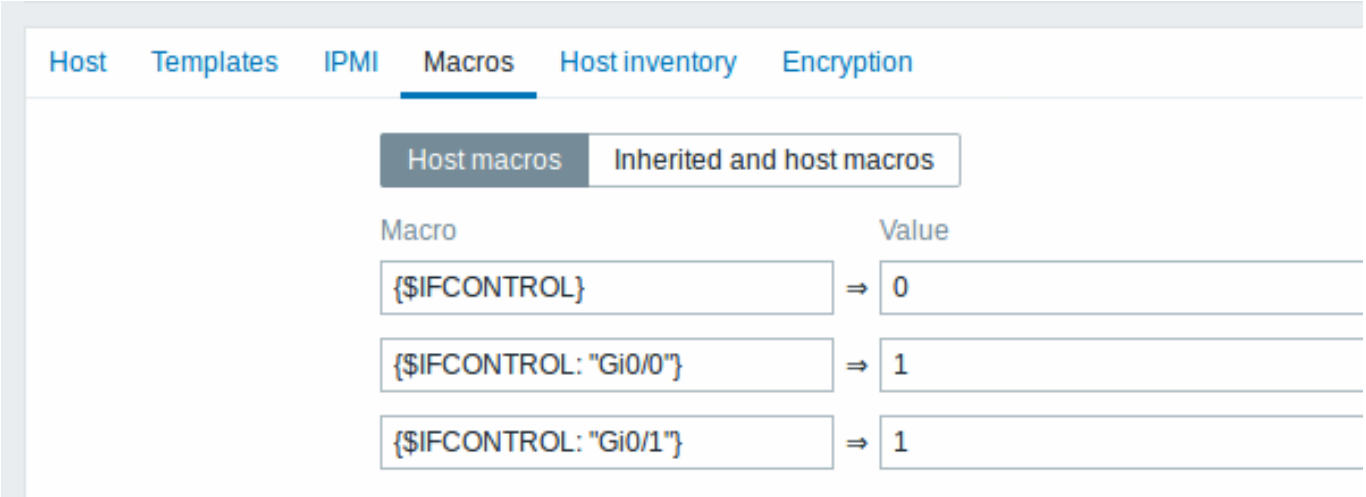
```
{$IFCONTROL:"{#IFNAME}"}=1 and last(/Alcatel Timetra TiMOS SNMP/net.if.status[ifOperStatus.{#SNMPINDEX}])
```

- If you do no want to monitor this condition for a specific interface create a user macro with context with the value 0. For example:



where Gi0/0 is {#IFNAME}. That way the trigger is not used any more for this specific interface.

- You can also change the default behavior for all triggers not to fire and activate this trigger only to limited number of interfaces like uplinks:



Tags

- Performance – device family MIBs provide a way to monitor CPU and memory items;
- Fault - device family MIBs provide a way to monitor at least one temperature sensor;
- Inventory – device family MIBs provide a way to collect at least the device serial number and model name;
- Certified – all three main categories above are covered.

10 Notifications upon events

Overview

Assuming that we have configured some items and triggers and now are getting some events happening as a result of triggers changing state, it is time to consider some actions.

To begin with, we would not want to stare at the triggers or events list all the time. It would be much better to receive notification if something significant (such as a problem) has happened. Also, when problems occur, we would like to see that all the people concerned are informed.

That is why sending notifications is one of the primary actions offered by Zabbix. Who and when should be notified upon a certain event can be defined.

To be able to send and receive notifications from Zabbix you have to:

- define some media
- configure an action that sends a message to one of the defined media

Actions consist of *conditions* and *operations*. Basically, when conditions are met, operations are carried out. The two principal operations are sending a message (notification) and executing a remote command.

For discovery and autoregistration created events, some additional operations are available. Those include adding or removing a host, linking a template etc.

1 Media types

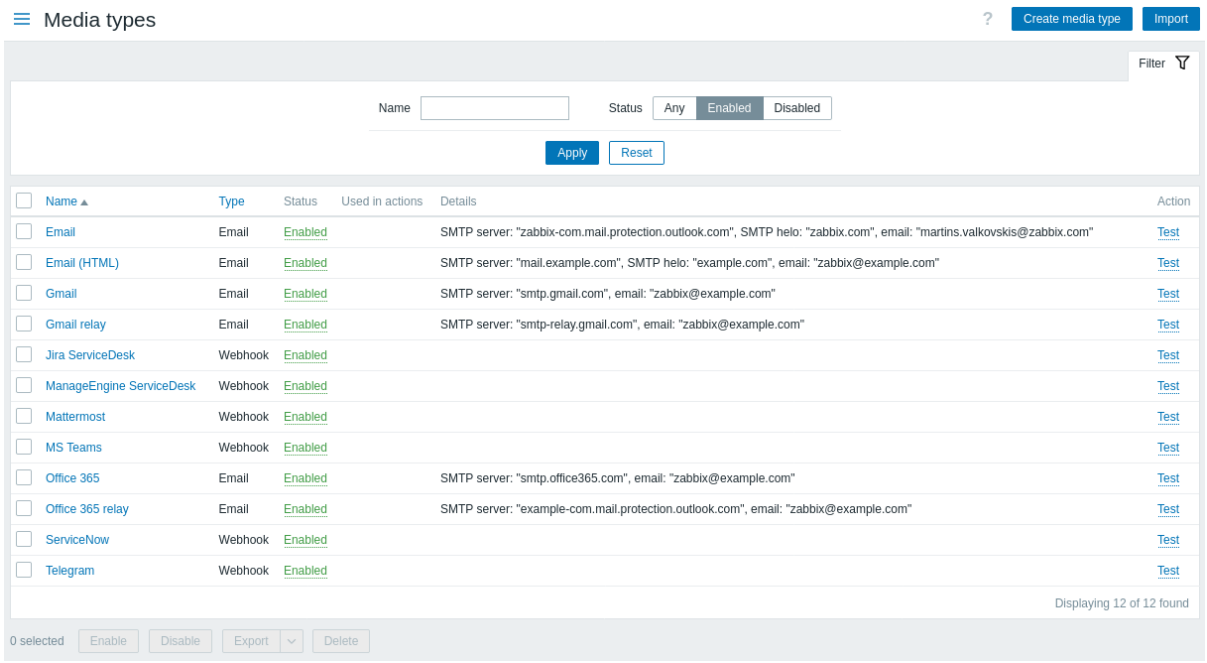
Overview

Media are the delivery channels used for sending notifications and alerts from Zabbix.

You can configure several media types:

- Email
- SMS
- Custom alertscripts
- Webhook

Media types are configured in *Alerts* → *Media types*.

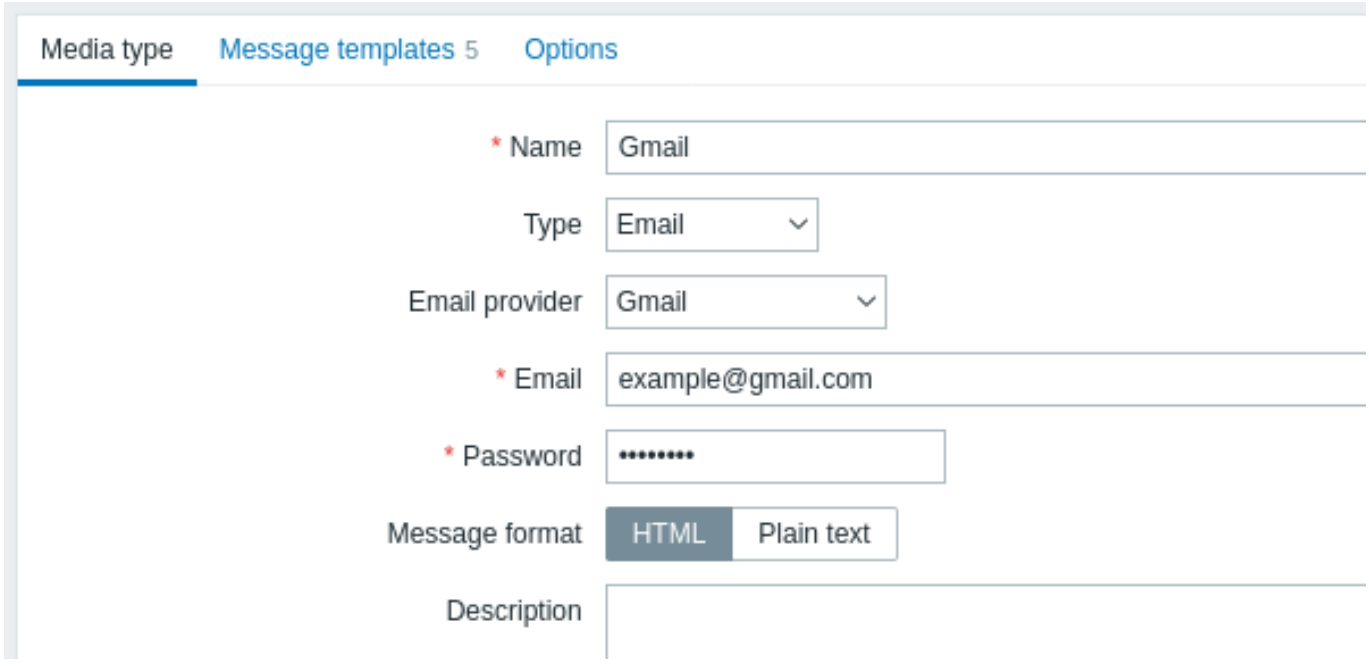


Some media types come pre-defined in the default dataset. You just need to finetune their parameters to get them working.

Automated Gmail/Office365 media types

Gmail or Office365 users may benefit from easier media type configuration. The *Email provider* field in the mail media type configuration allows to select pre-configured options for Gmail and Office 365.

When selecting the Gmail/Office365 related options, it is only required to supply the sender email address/password to create a working media type.



As soon as the email address/password is supplied, Zabbix will be able to automatically fill all required settings for Gmail/Office365 media types with the actual/recommended values, i.e. *SMTP server*, *SMTP server port*, *SMTP helo*, and *Connection security*. Because of this automation, these fields are not even shown, however, it is possible to see the SMTP server and email details in the media type list (see the *Details* column).

Note also that:

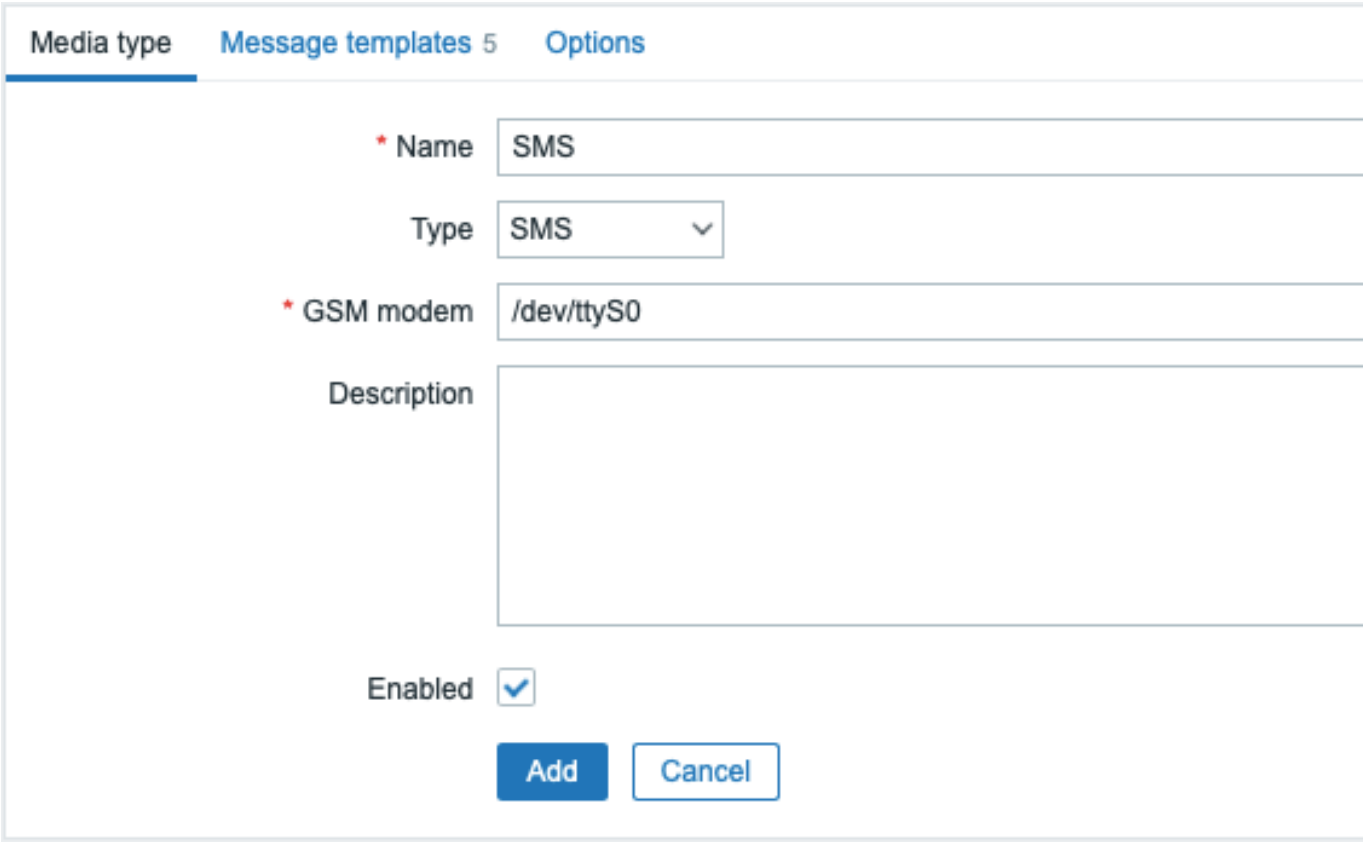
- The password is not required for the relay options
- For Office365 relay, the domain name of the provided email address will be used to dynamically fill the SMTP server (i.e. replace "example.com" in `example-com.mail.protection.outlook.com` with the real value)

To test if a configured media type works, click on the *Test* link in the last column (see media type testing for Email, Webhook, or Script for more details).

To create a new media type, click on the *Create media type* button. A media type configuration form is opened.

Common parameters

Some parameters are common for all media types.



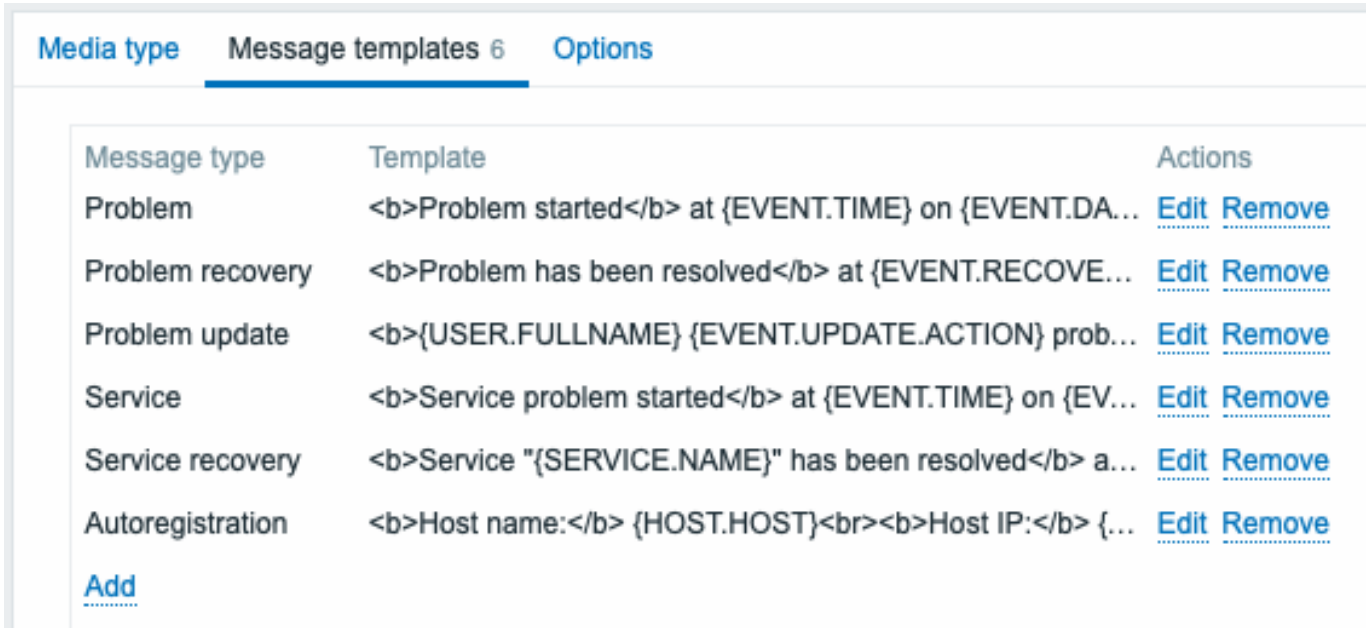
In the **Media type** tab the common general attributes are:

Parameter	Description
Name	Name of the media type.
Type	Select the type of media.
Description	Enter a description.
Enabled	Mark the checkbox to enable the media type.

See the individual pages of media types for media-specific parameters.

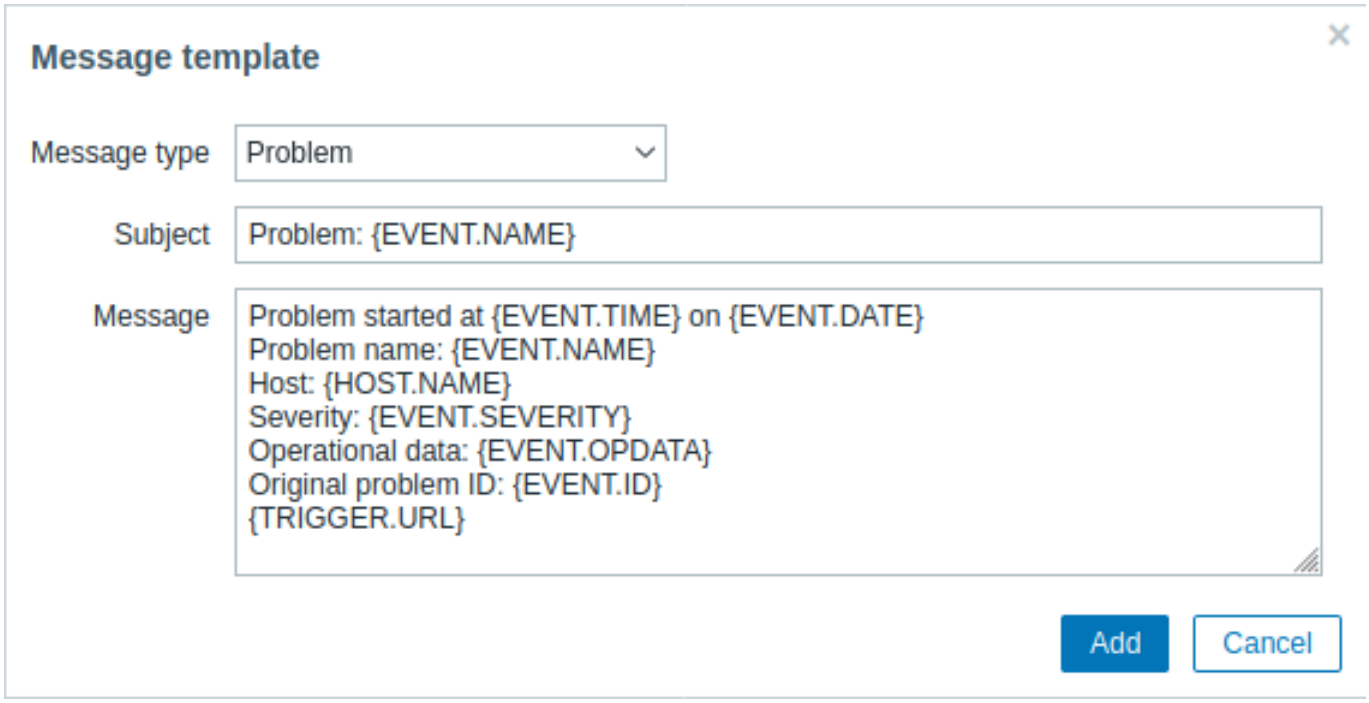
The **Message templates** tab allows to set default notification messages for all or some of the following event types:

- Problem
- Problem recovery
- Problem update
- Service
- Service recovery
- Service update
- Discovery
- Autoregistration
- Internal problem
- Internal problem recovery



To customize message templates:

- In the *Message templates* tab click on Add: a *Message template* popup window will open.
- Select required *Message type* and edit *Subject* and *Message* texts.
- Click on *Add* to save the message template



Message template parameters:

Parameter	Description
Message type	Type of an event for which the default message should be used. Only one default message can be defined for each event type.
Subject	Subject of the default message. The subject may contain macros. It is limited to 255 characters. Subject is not available for SMS media type.
Message	The default message. It is limited to certain amount of characters depending on the database type (see Sending messages for more information). The message may contain supported macros. In problem and problem update messages, expression macros are supported (for example, `{?avg(/host/key,1h)}`).

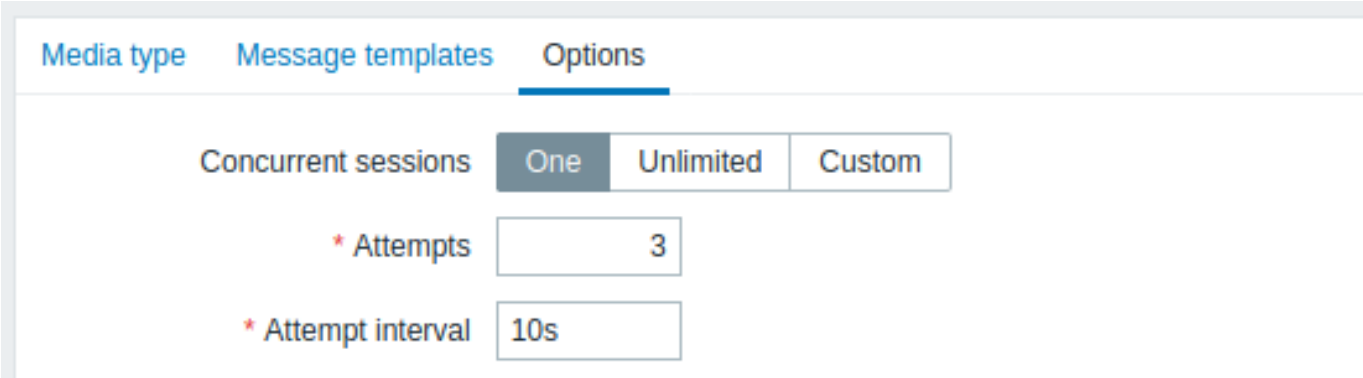
To make changes to an existing message template: in the *Actions* column click on Edit to edit the template or click on Remove to delete the message template.

It is possible to define a custom message template for a specific action (see action operations for details). Custom messages defined in the action configuration will override default media type message template.

Warning:
Defining message templates is mandatory for all media types, including webhooks or custom alert scripts that do not use default messages for notifications. For example, an action "Send message to Pushover webhook" will fail to send problem notifications, if the Problem message for the Pushover webhook is not defined.

The **Options** tab contains alert processing settings. The same set of options is configurable for each media type.

All media types are processed in parallel. While the maximum number of concurrent sessions is configurable per media type, the total number of alerter processes on the server can only be limited by the StartAlerters parameter. Alerts generated by one trigger are processed sequentially. So multiple notifications may be processed simultaneously only if they are generated by multiple triggers.



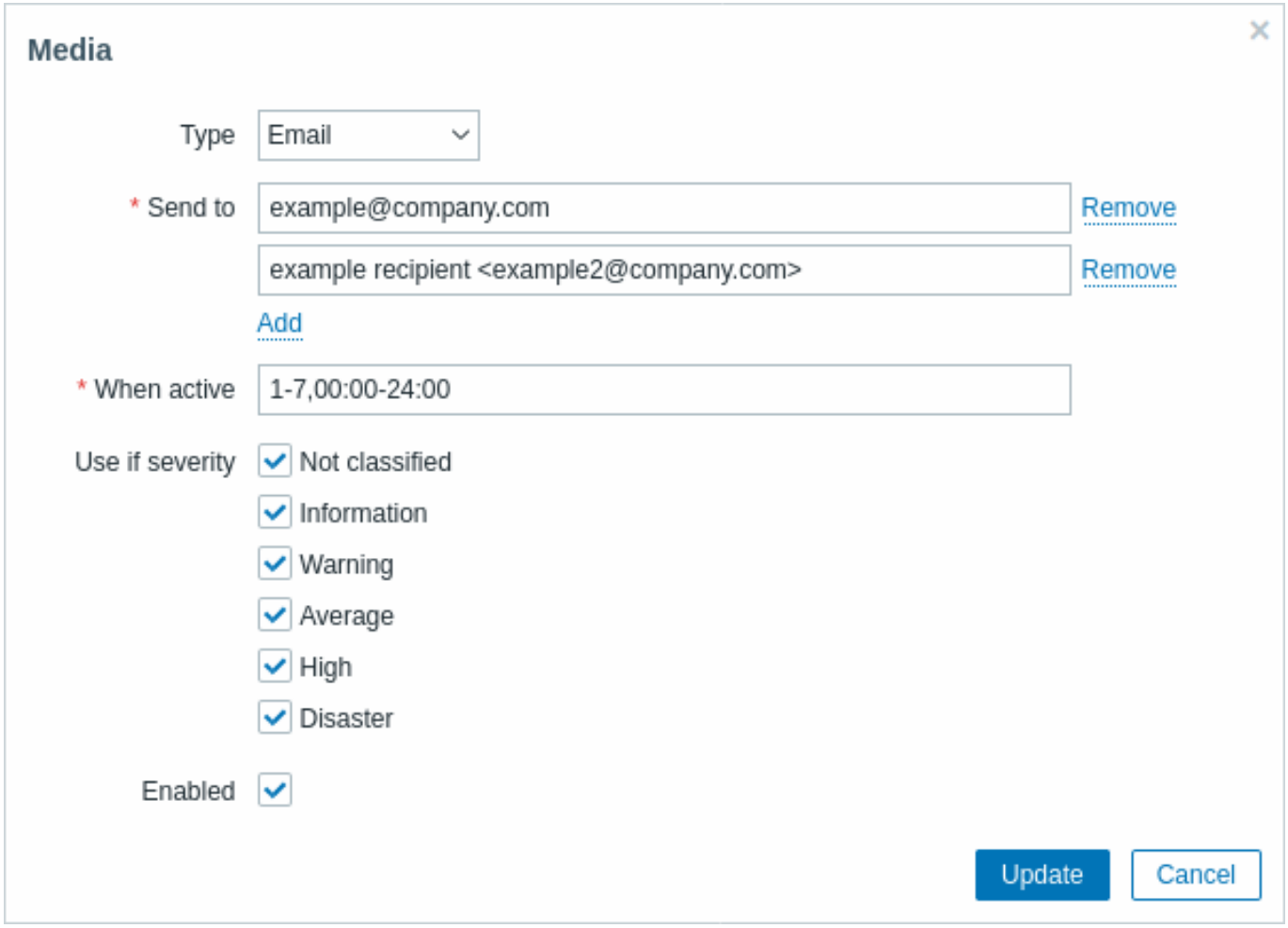
Parameter	Description
Concurrent sessions	Select the number of parallel alerter sessions for the media type: **One** - one session **Unlimited** - unlimited number of sessions **Custom** - select a custom number of sessions Unlimited/high values mean more parallel sessions and increased capacity for sending notifications. Unlimited/high values should be used in large environments where lots of notifications may need to be sent simultaneously. If more notifications need to be sent than there are concurrent sessions, the remaining notifications will be queued; they will not be lost.
Attempts	Number of attempts for trying to send a notification. Up to 100 attempts can be specified; the default value is '3'. If '1' is specified Zabbix will send the notification only once and will not retry if the sending fails.
Attempt interval	Frequency of trying to resend a notification in case the sending failed, in seconds (0-3600). If '0' is specified, Zabbix will retry immediately. Time suffixes are supported, e.g. 5s, 3m, 1h.

User media

To receive notifications of a media type, a medium (email address/phone number/webhook user ID etc) for this media type must be defined in the user profile. For example, an action sending messages to user "Admin" using webhook "X" will always fail to send anything if the webhook "X" medium is not defined in the user profile.

To define user media:

- Go to your user profile, or go to *Users* → *Users* and open the user properties form
- In the Media tab, click on Add



User media attributes:

Parameter	Description
Type	The drop-down list contains the names of enabled media types. Note that when editing a medium of a disabled media type, the type will be displayed in red.
Send to	Provide required contact information where to send messages. For an email media type it is possible to add several addresses by clicking on Add below the address field. In this case, the notification will be sent to all email addresses provided. It is also possible to specify recipient name in the *Send to* field of the email recipient in a format 'Recipient name <address1@company.com>'. Note that if a recipient name is provided, an email address should be wrapped in angle brackets (<>). UTF-8 characters in the name are supported, quoted pairs and comments are not. For example: *John Abercroft <manager@nycdatacenter.com>* and *manager@nycdatacenter.com* are both valid formats. Incorrect examples: *John Doe zabbix@company.com*, *%%"Zabbix\@\<H(comment)Q\>" zabbix@company.com %%*.
When active	You can limit the time when messages are sent, for example, set the working days only (1-5,09:00-18:00). Note that this limit is based on the user time zone. If the user time zone is changed and is different from the system time zone this limit may need to be adjusted accordingly so as not to miss important messages. See the Time period specification page for description of the format.
Use if severity	Mark the checkboxes of trigger severities that you want to receive notifications for. *Note* that the default severity ('Not classified') **must be** checked if you want to receive notifications for non-trigger events. After saving, the selected trigger severities will be displayed in the corresponding severity colors, while unselected ones will be grayed out.
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

1 Email

Overview

To configure email as the delivery channel for messages, you need to configure email as the media type and assign specific addresses to users.

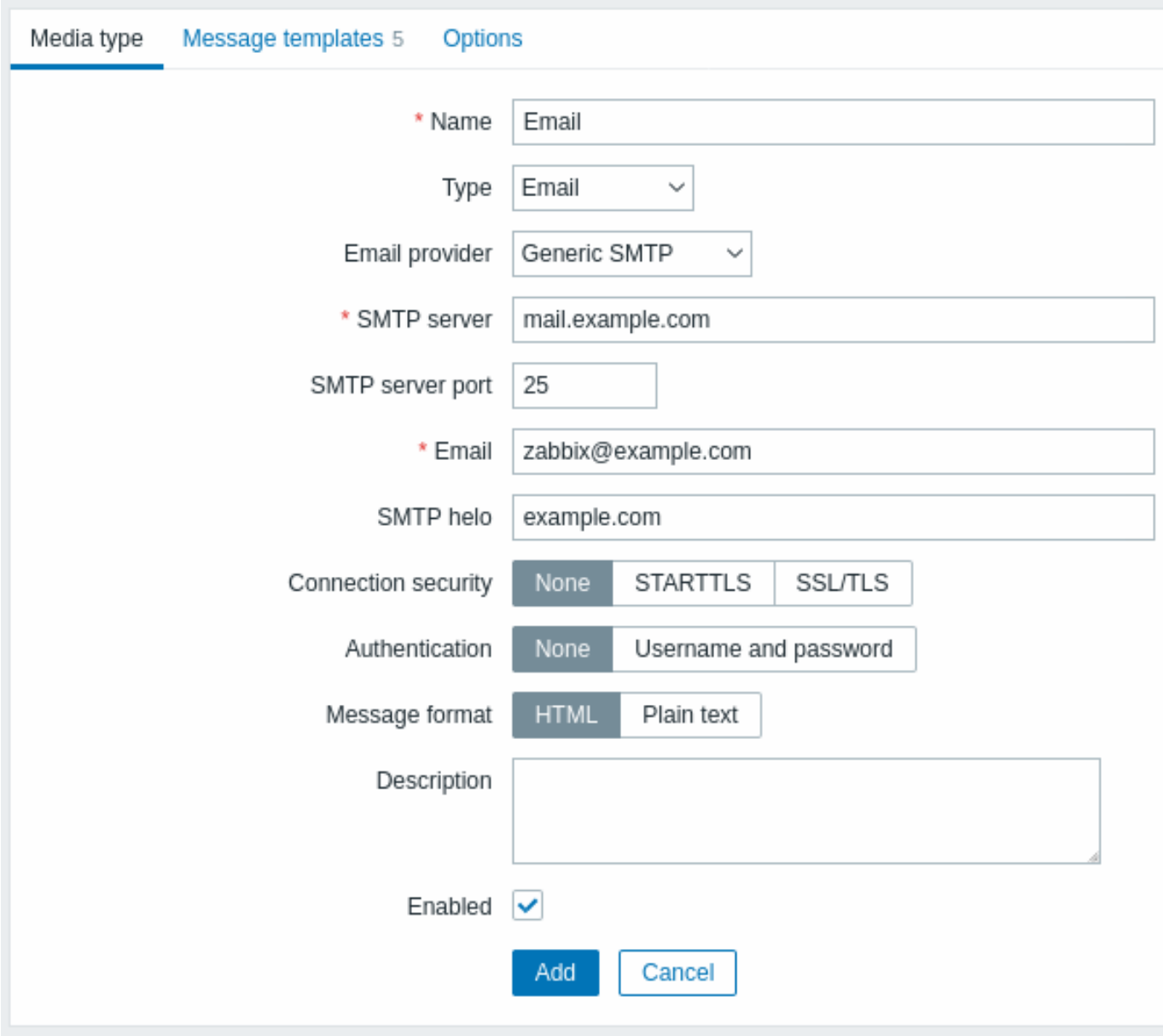
Note:
Multiple notifications for single event will be grouped together on the same email thread.

Configuration

To configure email as the media type:

- Go to *Alerts* → *Media types*
- Click on *Create media type* (or click on *Email* in the list of pre-defined media types).

The **Media type** tab contains general media type attributes:



All mandatory input fields are marked with a red asterisk.

The following parameters are specific for the email media type:

Parameter	Description
Email provider	Select the email provider: *Generic SMTP*, *Gmail*, *Gmail relay*, *Office365*, or *Office365 relay*. If you select the Gmail/Office365-related options, you will only need to supply the sender email address and password; such options as *SMTP server*, *SMTP server port*, *SMTP helo*, and *Connection security* will be automatically filled by Zabbix. See also: Automated Gmail/Office365 media types.
SMTP server	Set an SMTP server to handle outgoing messages. This field is available if *Generic SMTP* is selected as the email provider.
SMTP server port	Set the SMTP server port to handle outgoing messages. This field is available if *Generic SMTP* is selected as the email provider.
Email	The address entered here will be used as the **From** address for the messages sent. Adding a sender display name (like "Zabbix_info" in *Zabbix_info <zabbix@company.com>* in the screenshot above) with the actual email address is supported since Zabbix 2.2 version. There are some restrictions on display names in Zabbix emails in comparison to what is allowed by RFC 5322, as illustrated by examples: Valid examples: *zabbix@company.com* (only email address, no need to use angle brackets) *Zabbix_info <zabbix@company.com>* (display name and email address in angle brackets) *∑Ω-monitoring <zabbix@company.com>* (UTF-8 characters in display name) Invalid examples: *Zabbix HQ zabbix@company.com* (display name present but no angle brackets around email address) *"Zabbix\@\<H(comment)Q\>" <zabbix@company.com>* (although valid by RFC 5322, quoted pairs and comments are not supported in Zabbix emails)
SMTP helo	Set a correct SMTP helo value, normally a domain name. If empty, the domain name of the email will be sent (i. e. what comes after @ in the *Email* field). If it is impossible to fetch the domain name, a debug-level warning will be logged and the server hostname will be sent as the domain for HELO command. This field is available if *Generic SMTP* is selected as the email provider.
Connection security	Select the level of connection security: **None** - do not use the CURLOPT_USE_SSL option **STARTTLS** - use the CURLOPT_USE_SSL option with CURLUSESSL_ALL value **SSL/TLS** - use of CURLOPT_USE_SSL is optional
SSL verify peer	Mark the checkbox to verify the SSL certificate of the SMTP server. The value of "SSLCALocation" server configuration directive should be put into CURLOPT_CAPATH for certificate validation. This sets cURL option CURLOPT_SSL_VERIFYPEER.
SSL verify host	Mark the checkbox to verify that the *Common Name* field or the *Subject Alternate Name* field of the SMTP server certificate matches. This sets cURL option CURLOPT_SSL_VERIFYHOST.
Authentication	Select the level of authentication: **None** - no cURL options are set (since 3.4.2) **Username and password** - implies "AUTH=*" leaving the choice of authentication mechanism to cURL (until 3.4.2) **Normal password** - CURLOPT_LOGIN_OPTIONS is set to "AUTH=PLAIN"
Username	User name to use in authentication. This sets the value of CURLOPT_USERNAME. User macros supported as of Zabbix 6.4.15.
Password	Password to use in authentication. This sets the value of CURLOPT_PASSWORD. User macros supported as of Zabbix 6.4.15.
Message format	Select message format: **HTML** - send as HTML **Plain text** - send as plain text

Attention:
To enable SMTP authentication options, Zabbix server must be both compiled with the `--with-libcurl` compilation option (with cURL 7.20.0 or higher) and use the `libcurl-full` packages during runtime.

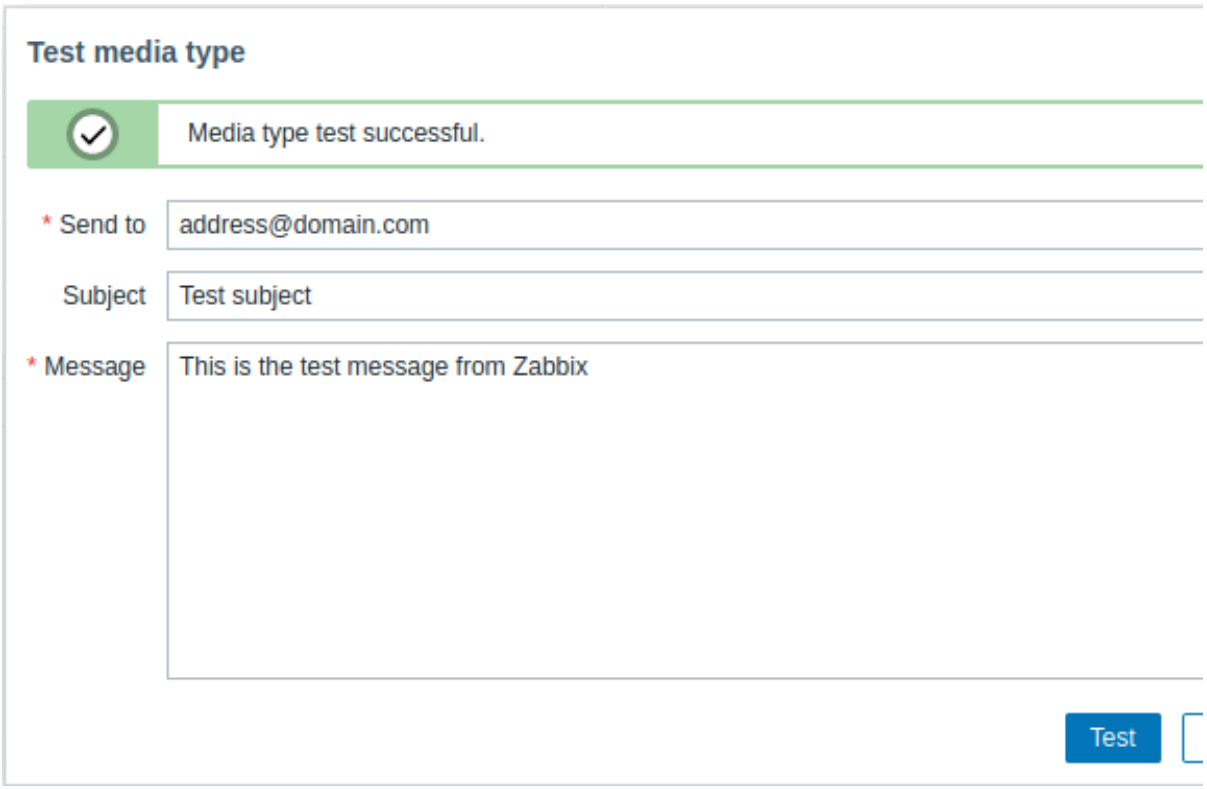
See also common media type parameters for details on how to configure default messages and alert processing options.

Media type testing

To test whether a configured email media type works correctly:

- Locate the relevant email in the list of media types.
- Click on *Test* in the last column of the list (a testing window will open).
- Enter a *Send to* recipient address, message body and, optionally, subject.
- Click on *Test* to send a test message.

Test success or failure message will be displayed in the same window:



User media

Once the email media type is configured, go to the *Users* → *Users* section and edit user profile to assign email media to the user. Steps for setting up user media, being common for all media types, are described on the Media types page.

2 SMS

Overview

Zabbix supports the sending of SMS messages using a serial GSM modem connected to Zabbix server's serial port.

Make sure that:

- The speed of the serial device (normally /dev/ttyS0 under Linux) matches that of the GSM modem. Zabbix does not set the speed of the serial link. It uses default settings.
- The 'zabbix' user has read/write access to the serial device. Run the command ls –l /dev/ttyS0 to see current permissions of the serial device.
- The GSM modem has PIN entered and it preserves it after power reset. Alternatively you may disable PIN on the SIM card. PIN can be entered by issuing command AT+CPIN="NNNN" (NNNN is your PIN number, the quotes must be present) in a terminal software, such as Unix minicom or Windows HyperTerminal.

Zabbix has been tested with these GSM modems:

- Siemens MC35
- Teltonika ModemCOM/G10

To configure SMS as the delivery channel for messages, you also need to configure SMS as the media type and enter the respective phone numbers for the users.

Configuration

To configure SMS as the media type:

- Go to *Alerts* → *Media types*
- Click on *Create media type* (or click on *SMS* in the list of pre-defined media types).

The following parameters are specific for the SMS media type:

Parameter	Description
GSM modem	Set the serial device name of the GSM modem.

See common media type parameters for details on how to configure default messages and alert processing options. Note that parallel processing of sending SMS notifications is not possible.

User media

Once the SMS media type is configured, go to the *Users* → *Users* section and edit user profile to assign SMS media to the user. Steps for setting up user media, being common for all media types, are described on the Media types page.

3 Custom alert scripts

Overview

If you are not satisfied with the existing media types for sending alerts, there is an alternative way to do that. You can create a script that will handle the notification your way.

Custom alert scripts are executed on Zabbix server. These scripts must be located in the directory specified in the server configuration file `AlertScriptsPath` parameter.

Here is an example of a custom alert script:

```
#####!/bin/bash

to=$1
subject=$2
body=$3
host=$4
value=$5

cat <<EOF | mail -s "$subject" "$to"
$body

Host: $host
Value: $value
EOF
```

Attention:
Starting from version 3.4 Zabbix checks for the exit code of the executed commands and scripts. Any exit code, which is different from **0**, is considered as a command execution error. In such cases, Zabbix will try to repeat failed execution.

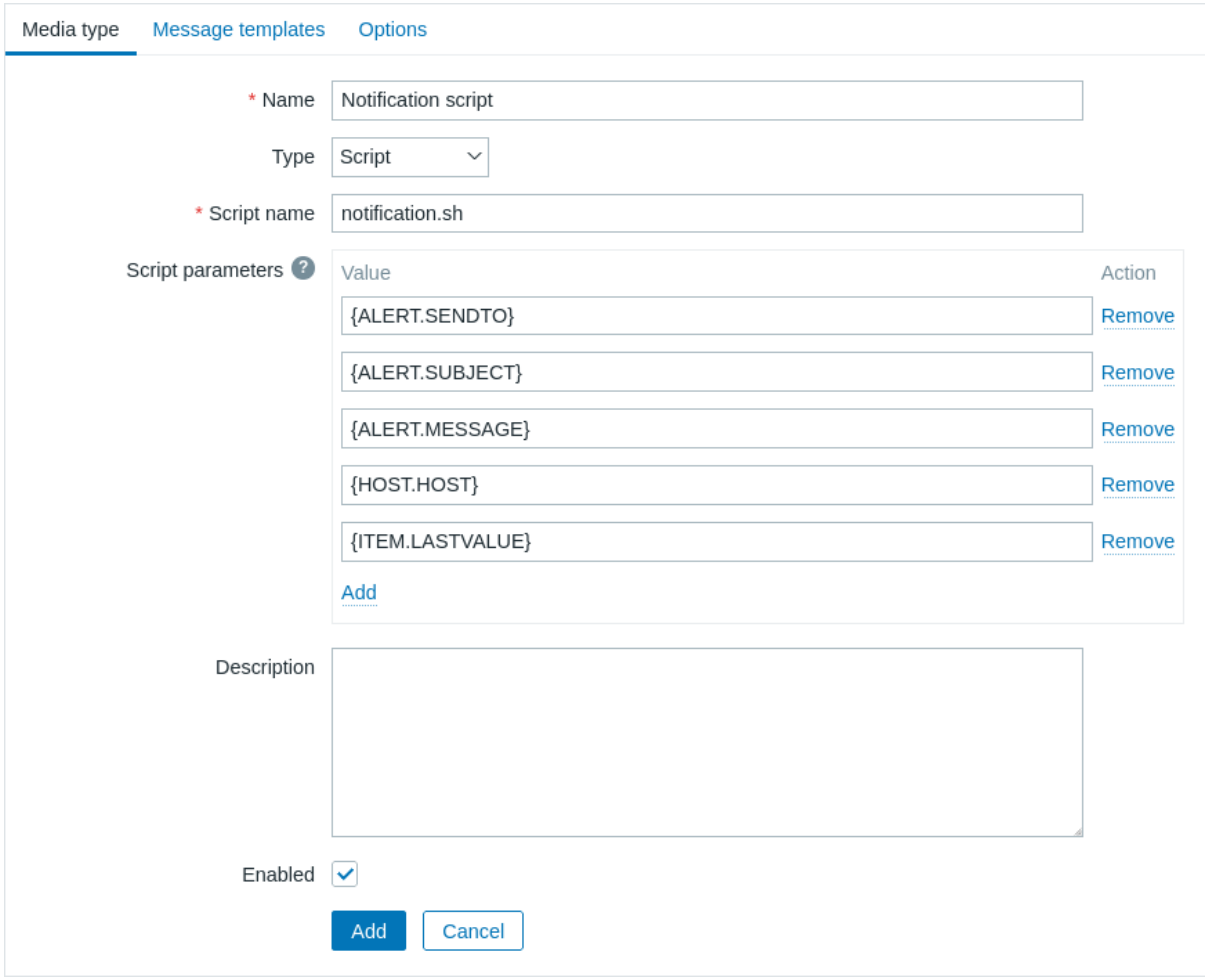
Environment variables are not preserved or created for the script, so they should be handled explicitly.

Configuration

To configure custom alert scripts as a media type:

1. Go to *Alerts* → *Media types*.
2. Click on *Create media type*.

The **Media type** tab contains general media type attributes:



All mandatory input fields are marked with a red asterisk.

The following parameters are specific for the script media type:

Parameter	Description
Script name	Enter the name of the script file (e.g., notification.sh) that is located in the directory specified in the server configuration file `AlertScriptsPath` parameter.
Script parameters	Add optional script parameters that will be passed to the script as command-line arguments in the order in which they are defined. Script parameters support {ALERT.SENDTO}, {ALERT.SUBJECT}, {ALERT.MESSAGE} macros, and, since Zabbix 6.4.0, all macros that are supported in notifications, as well as user macros.

See common media type parameters for details on how to configure default messages and alert processing options.

Warning:
Even if an alert script does not use default messages, the message templates for operation types used by this media type must still be defined. Otherwise, a notification will not be sent.

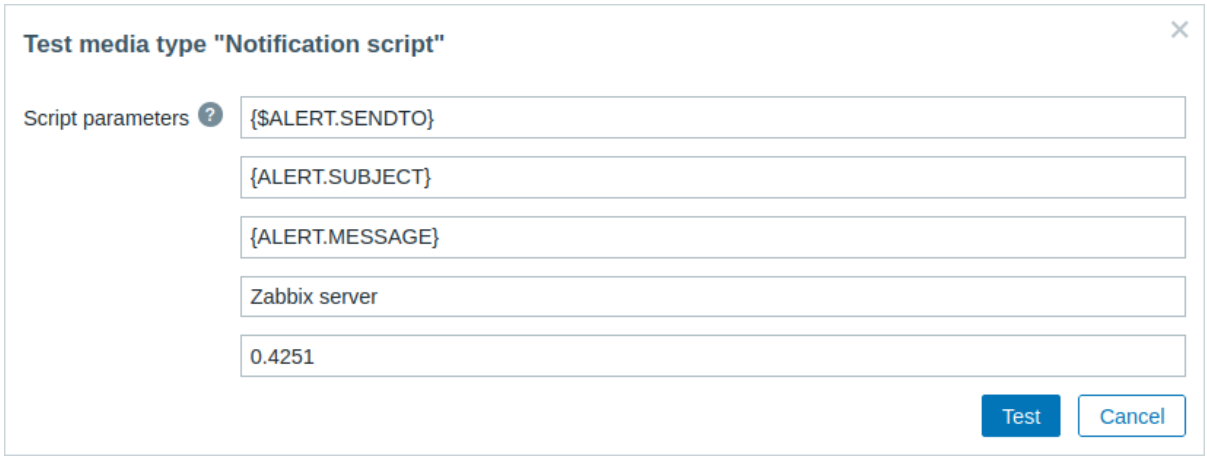
Attention:
If more than one script media type is configured, these scripts may be processed in parallel by the alerter processes. The total number of alerter processes is limited by the server configuration file `StartAlerters` parameter.

Media type testing

To test a configured script media type:

1. Locate the relevant script in the list of media types.
2. Click on *Test* in the last column of the list; a testing form will open in a pop-up window. The testing form will contain the same number of parameters that are configured for the script media type.

3. Edit the script parameter values if needed. Editing only affects the test procedure; the actual values will not be changed.
4. Click on *Test*.



Note:
When testing a configured script media type, {ALERT.SENDTO}, {ALERT.SUBJECT}, {ALERT.MESSAGE} and user macros will resolve to their values, but macros that are related to events (e.g., {HOST.HOST}, {ITEM.LASTVALUE}, etc.) will not resolve, as during testing there is no related event to get the details from. Note that macros within {ALERT.SUBJECT} and {ALERT.MESSAGE} macros will also not resolve. For example, if the value of {ALERT.SUBJECT} is composed of "Problem: {EVENT.NAME}" then the {EVENT.NAME} macro will not be resolved.

User media

Once the media type is configured, go to the *Users* → *Users* section and edit a user profile by assigning this media type to the user. Steps for setting up user media, being common for all media types, are described on the Media types page.

Note that when defining the user media, the *Send to* field cannot be empty. If this field is not used in the alert script, enter any combination of supported characters to bypass validation requirements.

4 Webhook

Overview

The webhook media type is useful for making HTTP calls using custom JavaScript code for straightforward integration with external software such as helpdesk systems, chats, or messengers. You may choose to import an integration provided by Zabbix or create a custom integration from scratch.

Integrations

The following integrations are available, allowing predefined webhook media types to be used for pushing Zabbix notifications to:

- brevis.one
- Discord
- Event-Driven Ansible
- Express.ms messenger
- Github issues
- GLPi
- iLert
- iTop
- Jira
- Jira Service Desk
- ManageEngine ServiceDesk
- Mantis Bug Tracker
- Mattermost
- Microsoft Teams
- LINE
- Opsgenie
- OTRS
- Pagerduty
- Pushover
- Redmine

- Rocket.Chat
- ServiceNow
- SIGNL4
- Slack
- SolarWinds
- SysAid
- Telegram
- TOPdesk
- VictorOps
- Zammad
- Zendesk

Note:
In addition to the services listed here, Zabbix can be integrated with **Spiceworks** (no webhook is required). To convert Zabbix notifications into Spiceworks tickets, create an email media type and enter Spiceworks helpdesk email address (e.g. help@zabbix.on.spiceworks.com) in the profile settings of a designated Zabbix user.

Configuration

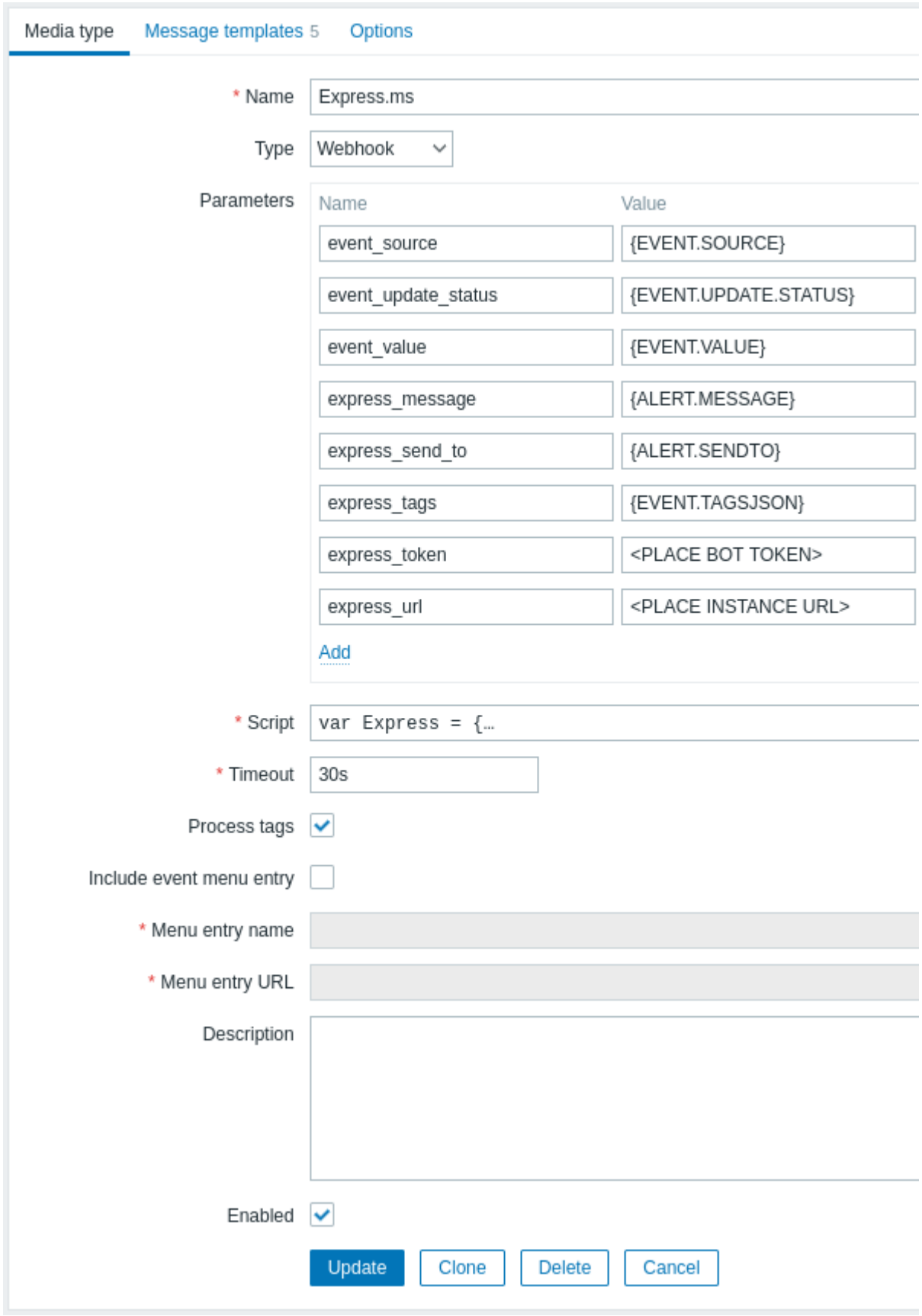
To start using a webhook integration:

1. Locate required .xml file in the `templates/media` directory of the downloaded Zabbix version or download it from Zabbix git repository
2. Import the file into your Zabbix installation. The webhook will appear in the list of media types.
3. Configure the webhook according to instructions in the *Readme.md* file (you may click on a webhook's name above to quickly access *Readme.md*).

To create a custom webhook from scratch:

- Go to *Alerts* → *Media types*
- Click on *Create media type*

The **Media type** tab contains various attributes specific for this media type:



All mandatory input fields are marked with a red asterisk.

The following parameters are specific for the webhook media type:

Parameter	Description
Parameters	Specify the webhook variables as the attribute and value pairs. For preconfigured webhooks, a list of parameters varies, depending on the service. Check the webhook's *Readme.md* file for parameter description. For new webhooks, several common variables are included by default (URL:<empty>, HTTPProxy:<empty>, To:{ALERT.SENDTO}, Subject:{ALERT.SUBJECT}, Message:{ALERT.MESSAGE}), feel free to keep or remove them. Webhook parameters support user macros, all macros that are supported in problem notifications and, additionally, {ALERT.SENDTO}, {ALERT.SUBJECT}, and {ALERT.MESSAGE} macros. If you specify an HTTP proxy, the field supports the same functionality as in the item configuration HTTP proxy field. The proxy string may be prefixed with `[scheme]://` to specify which kind of proxy is used (e.g. https, socks4, socks5; see documentation).
Script	Enter JavaScript code in the block that appears when clicking in the parameter field (or on the view/edit button next to it). This code will perform the webhook operation. The script is a function code that accepts parameter - value pairs. The values should be converted into JSON objects using JSON.parse() method, for example: `var params = JSON.parse(value);`. The code has access to all parameters, it may perform HTTP GET, POST, PUT and DELETE requests and has control over HTTP headers and request body. The script must contain a return operator, otherwise it will not be valid. It may return OK status along with an optional list of tags and tag values (see *Process tags* option) or an error string. Note that the script is executed only after an alert is created. If the script is configured to return and process tags, these tags will not get resolved in {EVENT.TAGS} and {EVENT.RECOVERY.TAGS} macros in the initial problem message and recovery messages because the script has not had the time to run yet. *Note*: Using local variables instead of global ones is recommended to make sure that each script operates on its own data and that there are no collisions between simultaneous calls (see known issues). See also: Webhook development guidelines, Webhook script examples, Additional JavaScript objects.
Timeout	JavaScript execution timeout (1-60s, default 30s). Time suffixes are supported, e.g. 30s, 1m.
Process tags	Mark the checkbox to process returned JSON property values as tags. These tags are added to any existing problem tags. Note that when using webhook tags, the webhook must return a JSON object containing at least an empty tags object: `var result = {tags: {}};` Examples of tags that can be returned: *Jira ID: PROD-1234*, *Responsible: John Smith*, *Processed:<no value>*
Include event menu entry	Mark the checkbox to include an entry in the event menu linking to a created external ticket. An entry will be included for each webhook that is enabled and has this checkbox marked. Note that if the *Menu entry name* and *Menu entry URL* parameters contain any {EVENT.TAGS.<tag name>} macros, an entry will be included only if these macros can be resolved (that is, the event has these tags defined). If marked, the webhook should not be used for sending notifications to different users (consider creating a dedicated user instead) and should not be used in multiple alert actions for a single problem event.
Menu entry name	Specify the menu entry name. {EVENT.TAGS.<tag name>} macro is supported. This field is only mandatory if *Include event menu entry* is marked.
Menu entry URL	Specify the underlying URL of the menu entry. {EVENT.TAGS.<tag name>} macro is supported. This field is only mandatory if *Include event menu entry* is marked.

See common media type parameters for details on how to configure default messages and alert processing options.

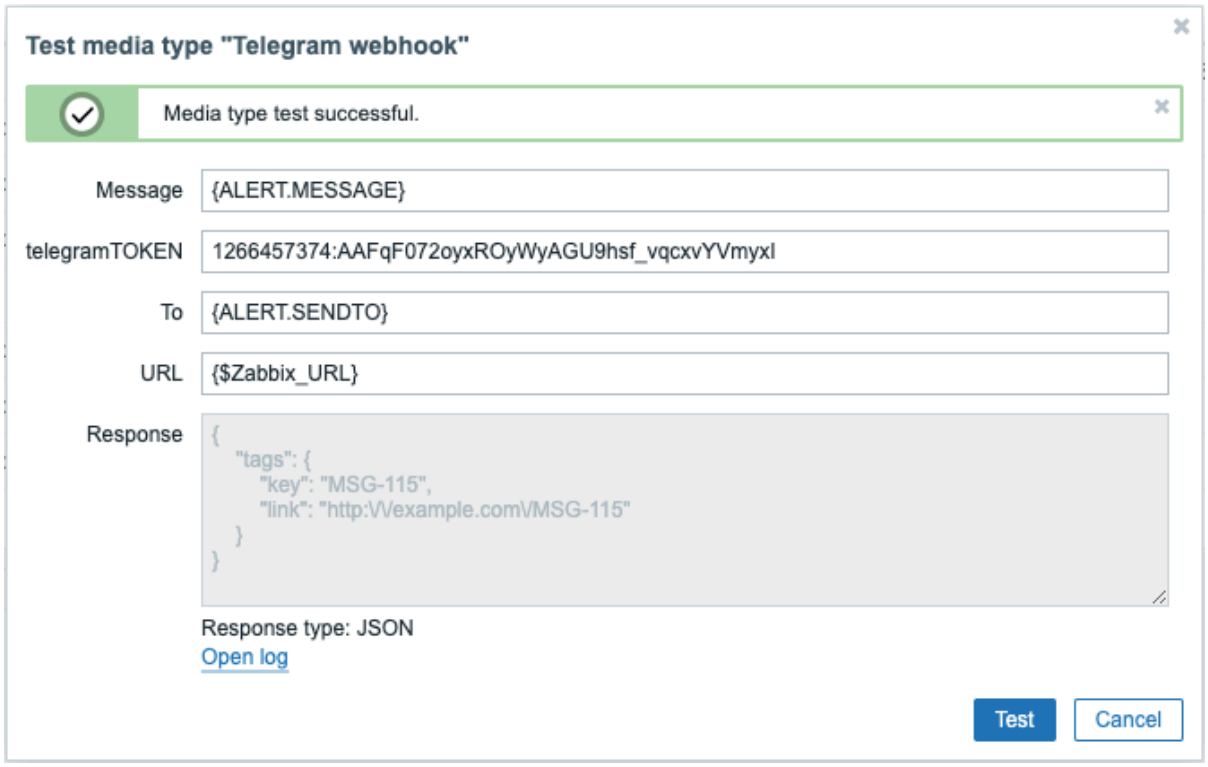
Warning:
Even if a webhook doesn't use default messages, message templates for operation types used by this webhook must still be defined.

Media type testing

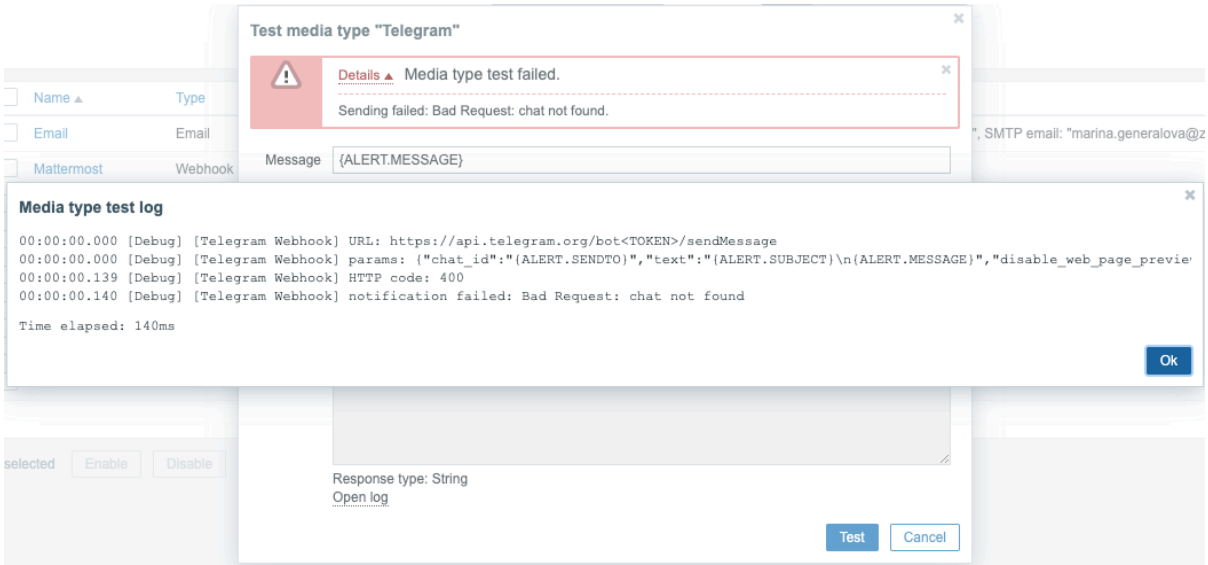
To test a configured webhook media type:

- Locate the relevant webhook in the list of media types.
- Click on *Test* in the last column of the list (a testing window will open).
- Edit the webhook parameter values, if needed.
- Click on *Test*.

By default, webhook tests are performed with parameters entered during configuration. However, it is possible to change attribute values for testing. Replacing or deleting values in the testing window affects the test procedure only, the actual webhook attribute values will remain unchanged.



To view media type test log entries without leaving the test window, click on *Open log* (a new popup window will open).



If the webhook test is successful:

- *"Media type test successful."* message is displayed
- Server response appears in the gray *Response* field

- Response type (JSON or String) is specified below the *Response* field

If the webhook test fails:

- *"Media type test failed."* message is displayed, followed by additional failure details.

User media

Once the media type is configured, go to the *Users → Users* section and assign the webhook media to an existing user or create a new user to represent the webhook. Steps for setting up user media for an existing user, being common for all media types, are described on the Media types page.

If a webhook uses tags to store ticket\message ID, avoid assigning the same webhook as a media to different users as doing so may cause webhook errors (applies to the majority of webhooks that utilize *Include event menu entry* option). In this case, the best practice is to create a dedicated user to represent the webhook:

1. After configuring the webhook media type, go to the *Users → Users* section and create a dedicated Zabbix user to represent the webhook - for example, with a username *Slack* for the Slack webhook. All settings, except media, can be left at their defaults as this user will not be logging into Zabbix.
2. In the user profile, go to a tab *Media* and add a webhook with the required contact information. If the webhook does not use a *Send to* field, enter any combination of supported characters to bypass validation requirements.
3. Grant this user at least read permissions to all hosts for which it should send the alerts.

When configuring alert action, add this user in the *Send to users* field in Operation details - this will tell Zabbix to use the webhook for notifications from this action.

Configuring alert actions

Actions determine which notifications should be sent via the webhook. Steps for configuring actions involving webhooks are the same as for all other media types with these exceptions:

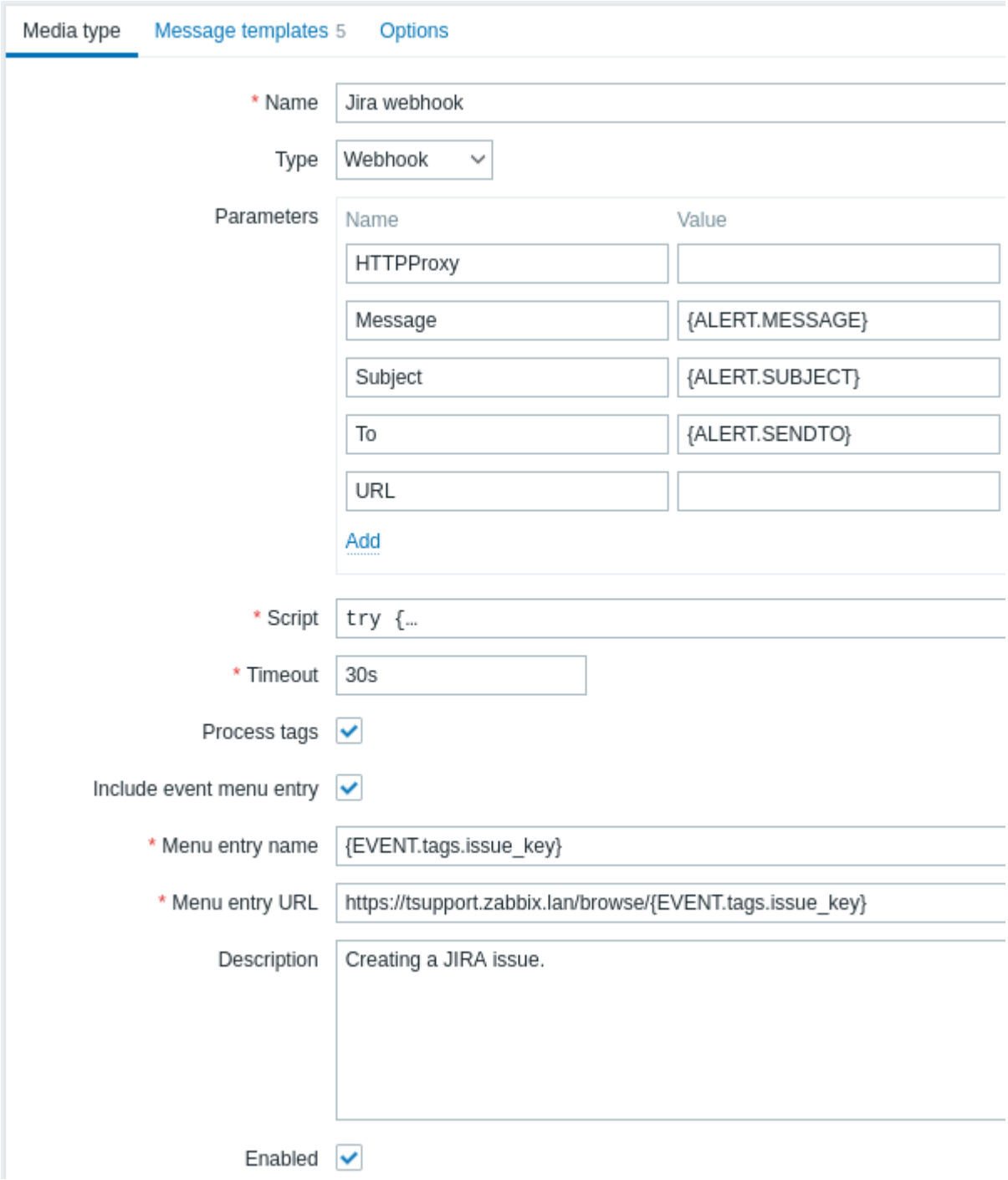
- If a webhook uses webhook tags to store ticket\message ID and handle update\resolve operations, avoid using the same webhook in multiple alert actions for a single problem event. If {EVENT.TAGS.<tag name>} exists and gets updated in the webhook, its resulting value will be undefined. To avoid this, use a new tag name in the webhook for storing updated values. This applies to Jira, Jira Service Desk, Mattermost, Opsgenie, OTRS, Redmine, ServiceNow, Slack, Zammad, and Zendesk webhooks provided by Zabbix and to most webhooks utilizing the *Include event menu entry* option. Note, however, that a single webhook can be used in multiple operations or escalation steps of the same action, as well as in different actions that will not be triggered by the same problem event due to different conditions.
- When using a webhook in actions for internal events, ensure to mark the *Custom message* checkbox and define a custom message in the action operation configuration. Otherwise, a notification will not be sent.

1 Webhook script examples

Overview

Though Zabbix offers a large number of webhook integrations available out-of-the-box, you may want to create your own webhooks instead. This section provides examples of custom webhook scripts (used in the *Script* parameter). See webhook section for description of other webhook parameters.

Jira webhook (custom)



This script will create a JIRA issue and return some info on the created issue.

```
try {
    Zabbix.log(4, '[ Jira webhook ] Started with params: ' + value);

    var result = {
            'tags': {
                'endpoint': 'jira'
            }
        },
        params = JSON.parse(value),
        req = new HttpRequest(),
        fields = {},
        resp;

    if (params.HTTPProxy) {
        req.setProxy(params.HTTPProxy);
    }
```

```
    req.addHeader('Content-Type: application/json');
    req.addHeader('Authorization: Basic ' + params.authentication);

    fields.summary = params.summary;
    fields.description = params.description;
    fields.project = {key: params.project_key};
    fields.issuetype = {id: params.issue_id};

    resp = req.post('https://jira.example.com/rest/api/2/issue/',
        JSON.stringify({"fields": fields})
    );

    if (req.getStatus() != 201) {
        throw 'Response code: ' + req.getStatus();
    }

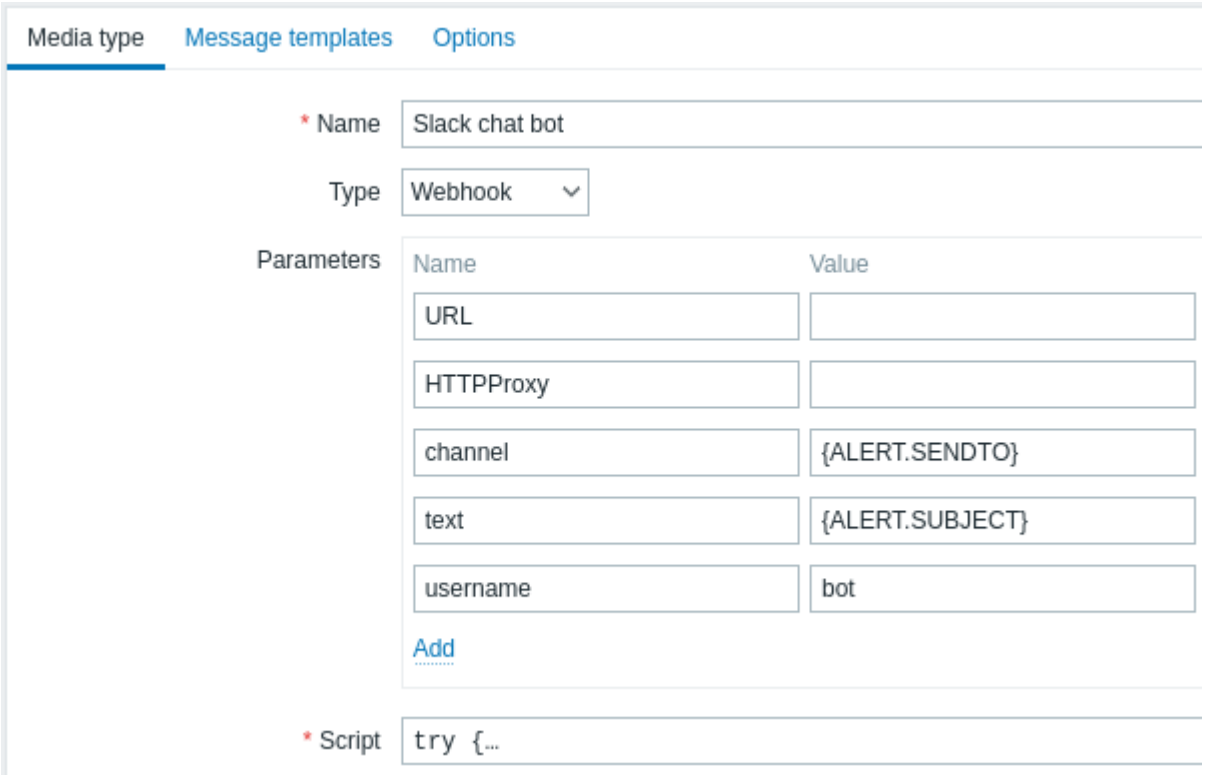
    resp = JSON.parse(resp);
    result.tags.issue_id = resp.id;
    result.tags.issue_key = resp.key;

    return JSON.stringify(result);
}
catch (error) {
    Zabbix.log(4, '[ Jira webhook ] Issue creation failed json : ' + JSON.stringify({"fields": fields}));
    Zabbix.log(3, '[ Jira webhook ] issue creation failed : ' + error);

    throw 'Failed with error: ' + error;
}
```

Slack webhook (custom)

This webhook will forward notifications from Zabbix to a Slack channel.




```
try {
    var params = JSON.parse(value),
        req = new HttpRequest(),
        response;

    if (params.HTTPProxy) {
```

```
        req.setProxy(params.HTTPProxy);
    }

    req.addHeader('Content-Type: application/x-www-form-urlencoded');

    Zabbix.log(4, '[ Slack webhook ] Webhook request with value=' + value);

    response = req.post(params.hook_url, 'payload=' + encodeURIComponent(value));
    Zabbix.log(4, '[ Slack webhook ] Responded with code: ' + req.getStatus() + '. Response: ' + response)

    try {
        response = JSON.parse(response);
    }
    catch (error) {
        if (req.getStatus() < 200 || req.getStatus() >= 300) {
            throw 'Request failed with status code ' + req.getStatus();
        }
        else {
            throw 'Request success, but response parsing failed.';
        }
    }

    if (req.getStatus() !== 200 || !response.ok || response.ok === 'false') {
        throw response.error;
    }

    return 'OK';
}
catch (error) {
    Zabbix.log(3, '[ Slack webhook ] Sending failed. Error: ' + error);

    throw 'Failed with error: ' + error;
}
```

2 Actions

Overview

If you want some operations taking place as a result of events (for example, notifications sent), you need to configure actions.

Actions can be defined in response to events of all supported types:

- Trigger actions - for events when trigger status changes from *OK* to *PROBLEM* and back
- Service actions - for events when service status changes from *OK* to *PROBLEM* and back
- Discovery actions - for events when network discovery takes place
- Autoregistration actions - for events when new active agents auto-register (or host metadata changes for registered ones)
- Internal actions - for events when items become unsupported or triggers go into an unknown state

The key differences of service actions are:

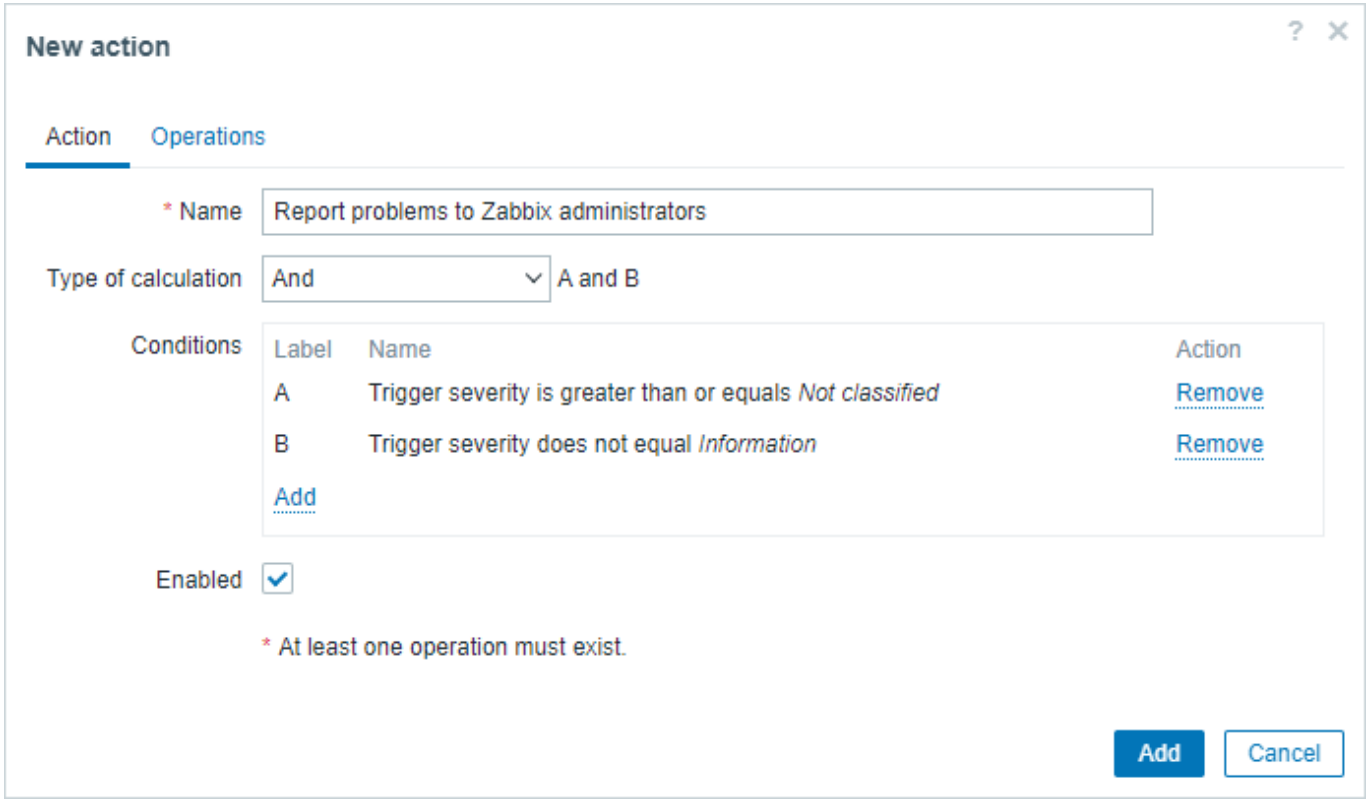
- User access to service actions depends on access rights to services granted by user's role
- Service actions support different set of conditions

Configuring an action

To configure an action, do the following:

- Go to *Alerts* → *Actions* and select the required action type from the submenu (you can switch to another type later, using the title dropdown)
- Click on *Create action*
- Name the action
- Choose conditions upon which operations are carried out
- Choose the operations to carry out

General action attributes:



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Unique action name.
Type of calculation	Select the evaluation option for action conditions (with more than one condition): **And** - all conditions must be met. **Or** - enough if one condition is met. **And/Or** - combination of the two: AND with different condition types and OR with the same condition type. **Custom expression** - a user-defined calculation formula for evaluating action conditions.
Conditions	List of action conditions. Click on *Add* to add a new condition. If no conditions are configured, the action will run for every event that corresponds to the action type being configured.
Enabled	Mark the checkbox to enable the action. Otherwise, it will be disabled.

1 Conditions

Overview

It is possible to define that an action is executed only if the event matches a defined set of conditions. Conditions are set when configuring the action.

Condition matching is case-sensitive.

Trigger actions

The following conditions can be used in trigger-based actions:

Condition type	Supported operators	Description
Host group	equals does not equal	Specify host groups or host groups to exclude. **equals** - event belongs to this host group. **does not equal** - event does not belong to this host group. Specifying a parent host group implicitly selects all nested host groups. To specify the parent group only, all nested groups have to be additionally set with the **does not equal** operator.

Condition type	Supported operators	Description
Template	equals does not equal	Specify templates or templates to exclude. **equals** - event belongs to a trigger inherited from this template. **does not equal** - event does not belong to a trigger inherited from this template.
Host	equals does not equal	Specify hosts or hosts to exclude. **equals** - event belongs to this host. **does not equal** - event does not belong to this host.
Tag name	equals does not equal contains does not contain	Specify event tag or event tag to exclude. **equals** - event has this tag. **does not equal** - event does not have this tag. **contains** - event has a tag containing this string. **does not contain** - event does not have a tag containing this string.
Tag value	equals does not equal contains does not contain	Specify event tag and value combination or tag and value combination to exclude. **equals** - event has this tag and value. **does not equal** - event does not have this tag and value. **contains** - event has a tag and value containing these strings. **does not contain** - event does not have a tag and value containing these strings.
Trigger	equals does not equal	Specify triggers or triggers to exclude. **equals** - event is generated by this trigger. **does not equal** - event is generated by any other trigger, except this one.
Event name	contains does not contain	Specify a string in the name of the event generated by the trigger or a string to exclude. By default, the event name matches the trigger name unless a custom event name is specified in trigger configuration. **contains** - event name contains this string. **does not contain** - this string is excluded from the event name. Note: Entered value will be compared to event name with all macros expanded.
Trigger severity	equals does not equal is greater than or equals is less than or equals	Specify trigger severity. **equals** - equal to trigger severity. **does not equal** - not equal to trigger severity. **is greater than or equals** - more or equal to trigger severity. **is less than or equals** - less or equal to trigger severity.
Time period	in not in	Specify a time period or a time period to exclude. **in** - event time is within the time period. **not in** - event time is not within the time period. See the time period specification page for description of the format. User macros are supported, since Zabbix 3.4.0.
Problem is suppressed	no yes	Specify if the problem is suppressed (not shown) because of host maintenance. **no** - problem is not suppressed. **yes** - problem is suppressed.

Service actions

The following conditions can be used in service actions:

Condition type	Supported operators	Description
Service	equals does not equal	Specify a service or a service to exclude. **equals** - event belongs to this service. **does not equal** - event does not belong to this service. Specifying a parent service implicitly selects all child services. To specify the parent service only, all nested services have to be additionally set with the **does not equal** operator.

Condition type	Supported operators	Description
Service name	contains does not contain	Specify a string in the service name or a string to exclude. **contains** - event is generated by a service, containing this string in the name. **does not contain** - this string cannot be found in the service name.
Service tag name	equals does not equal contains does not contain	Specify an event tag or an event tag to exclude. Service event tags can be defined in the service configuration section *Tags*. **equals** - event has this tag. **does not equal** - event does not have this tag. **contains** - event has a tag containing this string. **does not contain** - event does not have a tag containing this string.
Service tag value	equals does not equal contains does not contain	Specify an event tag and value combination or a tag and value combination to exclude. Service event tags can be defined in the service configuration section *Tags*. **equals** - event has this tag and value. **does not equal** - event does not have this tag and value. **contains** - event has a tag and value containing these strings. **does not contain** - event does not have a tag and value containing these strings.

Attention:
Make sure to define message templates for Service actions in the *Alerts* → *Media types* menu. Otherwise, the notifications will not be sent.

Discovery actions

The following conditions can be used in discovery-based events:

Condition type	Supported operators	Description
Host IP	equals does not equal	Specify an IP address range or a range to exclude for a discovered host. **equals** - host IP is in the range. **does not equal** - host IP is not in the range. It may have the following formats: Single IP: 192.168.1.33 Range of IP addresses: 192.168.1-10.1-254 IP mask: 192.168.4.0/24 List: 192.168.1.1-254, 192.168.2.1-100, 192.168.2.200, 192.168.4.0/24 Support for spaces in the list format is provided since Zabbix 3.0.0.
Service type	equals does not equal	Specify a service type of a discovered service or a service type to exclude. **equals** - matches the discovered service. **does not equal** - does not match the discovered service. Available service types: SSH, LDAP, SMTP, FTP, HTTP, HTTPS *(available since Zabbix 2.2 version)*, POP, NNTP, IMAP, TCP, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping, telnet *(available since Zabbix 2.2 version)*.
Service port	equals does not equal	Specify a TCP port range of a discovered service or a range to exclude. **equals** - service port is in the range. **does not equal** - service port is not in the range.
Discovery rule	equals does not equal	Specify a discovery rule or a discovery rule to exclude. **equals** - using this discovery rule. **does not equal** - using any other discovery rule, except this one.
Discovery check	equals does not equal	Specify a discovery check or a discovery check to exclude. **equals** - using this discovery check. **does not equal** - using any other discovery check, except this one.
Discovery object	equals	Specify the discovered object. **equals** - equal to discovered object (a device or a service).

Condition type	Supported operators	Description
Discovery status	equals	**Up** - matches 'Host Up' and 'Service Up' events. **Down** - matches 'Host Down' and 'Service Down' events. **Discovered** - matches 'Host Discovered' and 'Service Discovered' events. **Lost** - matches 'Host Lost' and 'Service Lost' events.
Uptime/Downtime	is greater than or equals is less than or equals	Uptime for 'Host Up' and 'Service Up' events. Downtime for 'Host Down' and 'Service Down' events. **is greater than or equals** - is more or equal to. Parameter is given in seconds. **is less than or equals** - is less or equal to. Parameter is given in seconds.
Received value	equals does not equal is greater than or equals is less than or equals contains does not contain	Specify the value received from an agent (Zabbix, SNMP) check in a discovery rule. String comparison. If several Zabbix agent or SNMP checks are configured for a rule, received values for each of them are checked (each check generates a new event which is matched against all conditions). **equals** - equal to the value. **does not equal** - not equal to the value. **is greater than or equals** - more or equal to the value. **is less than or equals** - less or equal to the value. **contains** - contains the substring. Parameter is given as a string. **does not contain** - does not contain the substring. Parameter is given as a string.
Proxy	equals does not equal	Specify a proxy or a proxy to exclude. **equals** - using this proxy. **does not equal** - using any other proxy except this one.

Note:
Service checks in a discovery rule, which result in discovery events, do not take place simultaneously. Therefore, if **multiple** values are configured for `Service type`, `Service port` or `Received value` conditions in the action, they will be compared to one discovery event at a time, but **not** to several events simultaneously. As a result, actions with multiple values for the same check types may not be executed correctly.

Autoregistration actions

The following conditions can be used in actions based on active agent autoregistration:

Condition type	Supported operators	Description
Host metadata	contains does not contain matches does not match	Specify host metadata or host metadata to exclude. **contains** - host metadata contains the string. **does not contain** - host metadata does not contain the string. Host metadata can be specified in an agent configuration file. **matches** - host metadata matches regular expression. **does not match** - host metadata does not match regular expression.
Host name	contains does not contain matches does not match	Specify a host name or a host name to exclude. **contains** - host name contains the string. **does not contain** - host name does not contain the string. **matches** - host name matches regular expression. **does not match** - host name does not match regular expression.
Proxy	equals does not equal	Specify a proxy or a proxy to exclude. **equals** - using this proxy. **does not equal** - using any other proxy except this one.

Internal event actions

The following conditions can be set for actions based on internal events:

Condition type	Supported operators	Description
Event type	equals	**Item in "not supported" state** - matches events where an item goes from a 'normal' to 'not supported' state. **Low-level discovery rule in "not supported" state** - matches events where a low-level discovery rule goes from a 'normal' to 'not supported' state. **Trigger in "unknown" state** - matches events where a trigger goes from a 'normal' to 'unknown' state.
Host group	equals does not equal	Specify host groups or host groups to exclude. **equals** - event belongs to this host group. **does not equal** - event does not belong to this host group.
Tag name	equals does not equal contains does not contain	Specify event tag or event tag to exclude. **equals** - event has this tag. **does not equal** - event does not have this tag. **contains** - event has a tag containing this string. **does not contain** - event does not have a tag containing this string.
Tag value	equals does not equal contains does not contain	Specify event tag and value combination or tag and value combination to exclude. **equals** - event has this tag and value. **does not equal** - event does not have this tag and value. **contains** - event has a tag and value containing these strings. **does not contain** - event does not have a tag and value containing these strings.
Template	equals does not equal	Specify templates or templates to exclude. **equals** - event belongs to an item/trigger/low-level discovery rule inherited from this template. **does not equal** - event does not belong to an item/trigger/low-level discovery rule inherited from this template.
Host	equals does not equal	Specify hosts or hosts to exclude. **equals** - event belongs to this host. **does not equal** - event does not belong to this host.

Type of calculation

The following options of calculating conditions are available:

- **And** - all conditions must be met

Note that using "And" calculation is disallowed between several triggers when they are selected as a `Trigger=` condition. Actions can only be executed based on the event of one trigger.

- **Or** - enough if one condition is met
- **And/Or** - combination of the two: AND with different condition types and OR with the same condition type, for example:

Host group equals Oracle servers
Host group equals MySQL servers
Event name contains 'Database is down'
Event name contains 'Database is unavailable'

is evaluated as

(Host group equals Oracle servers **or** Host group equals MySQL servers**) and (**Event name contains 'Database is down' **or** Event name contains 'Database is unavailable'**)**

- **Custom expression** - a user-defined calculation formula for evaluating action conditions. It must include all conditions (represented as uppercase letters A, B, C, ...) and may include spaces, tabs, brackets ( ), **and** (case sensitive), **or** (case sensitive), **not** (case sensitive).

While the previous example with `And/Or` would be represented as (A or B) and (C or D), in a custom expression you may as well have multiple other ways of calculation:

(A and B) and (C or D)
(A and B) or (C and D)
((A or B) and C) or D
(not (A or B) and C) or not D
etc.

Actions disabled due to deleted objects

If a certain object (host, template, trigger, etc.) used in an action condition/operation is deleted, the condition/operation is removed and the action is disabled to avoid incorrect execution of the action. The action can be re-enabled by the user.

This behavior takes place when deleting:

- host groups (”host group” condition, ”remote command” operation on a specific host group);
- hosts (”host” condition, ”remote command” operation on a specific host);
- templates (”template” condition, ”link to template” and ”unlink from template” operations);
- triggers (”trigger” condition);
- discovery rules (when using ”discovery rule” and ”discovery check” conditions).

Note:
If a remote command has many target hosts, and we delete one of them, only this host will be removed from the target list, the operation itself will remain. But, if it’s the only host, the operation will be removed, too. The same goes for ”link to template” and ”unlink from template” operations.

Actions are not disabled when deleting a user or user group used in a ”send message” operation.

2 Operations

Overview

You can define the following operations for all events:

- Send a message
- Execute a remote command

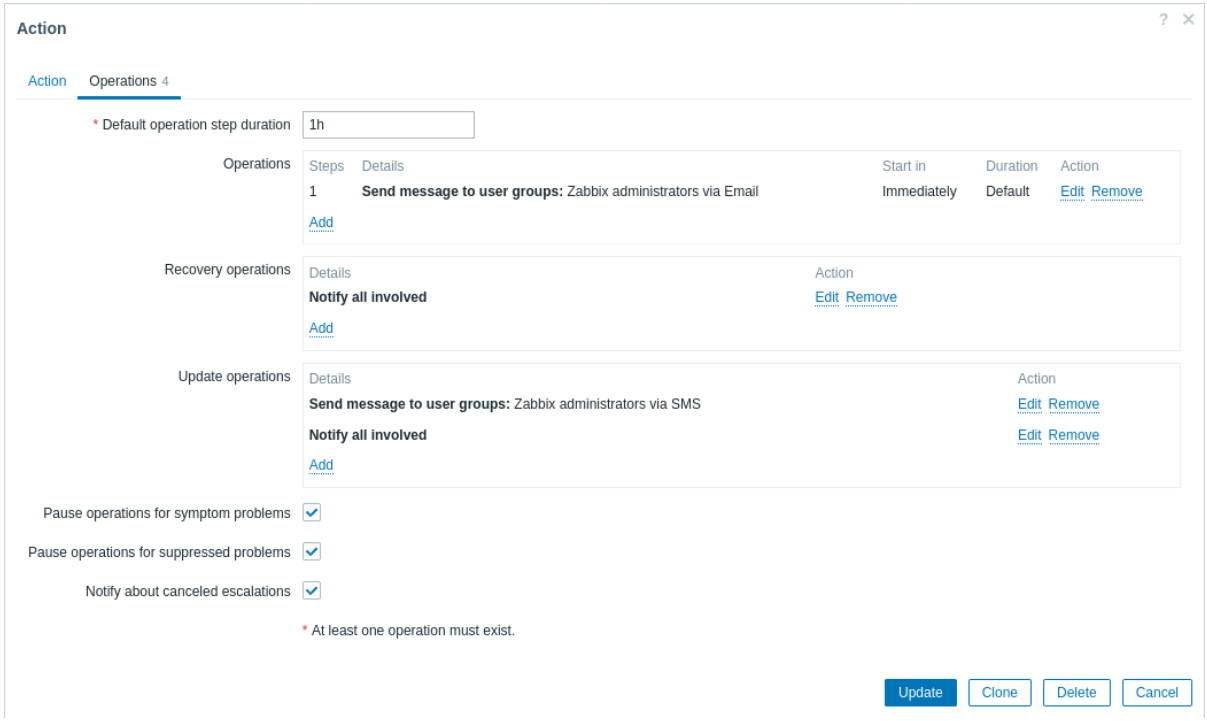
Attention:
Zabbix server does not create alerts if access to the host is explicitly ”denied” for the user defined as action operation recipient or if the user has no rights defined to the host at all.

For discovery and autoregistration events, additional operations are available:

- Add host
- Remove host
- Enable host
- Disable host
- Add to host group
- Remove from host group
- Link to template
- Unlink from template
- Set host inventory mode

Configuring an operation

To configure an operation, go to the *Operations* tab in action configuration.



General operation attributes:

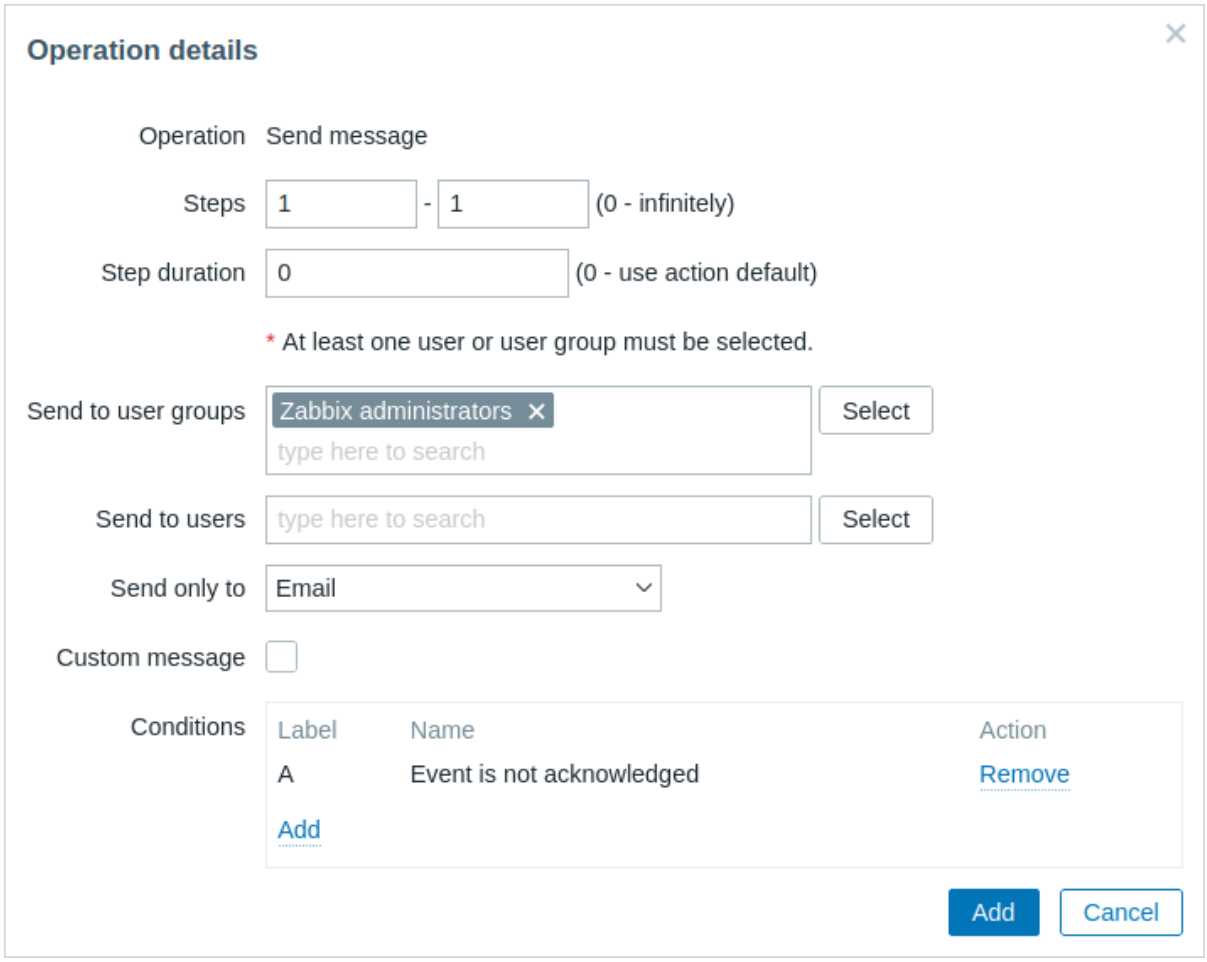
Parameter	Description
Default operation step duration	Duration of one operation step by default (60 seconds to 1 week). For example, an hour-long step duration means that if an operation is carried out, an hour will pass before the next step. Time suffixes are supported, e.g. 60s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0.
Operations	Action operations (if any) are displayed, with these details: **Steps** - escalation step(s) to which the operation is assigned. **Details** - type of operation and its recipient/target. The operation list also displays the media type (email, SMS or script) used as well as the name and surname (in parentheses after the username) of a notification recipient. **Start in** - how long after an event the operation is performed. **Duration (sec)** - step duration is displayed. *Default* is displayed if the step uses default duration, and a time is displayed if custom duration is used. **Action** - links for editing and removing an operation are displayed.
Recovery operations	Action operations (if any) are displayed, with these details: **Details** - type of operation and its recipient/target. The operation list also displays the media type (email, SMS or script) used as well as the name and surname (in parentheses after the username) of a notification recipient. **Action** - links for editing and removing an operation are displayed.
Update operations	Action operations (if any) are displayed, with these details: **Details** - type of operation and its recipient/target. The operation list also displays the media type (email, SMS or script) used as well as the name and surname (in parentheses after the username) of a notification recipient. **Action** - links for editing and removing an operation are displayed.
Pause operations for symptom problems	Mark this checkbox to pause operations (after the first operation) for symptom problems. Note that this setting affects only problem escalations; recovery and update operations will not be affected. This option is available for *Trigger actions* only.
Pause operations for suppressed problems	Mark this checkbox to delay the start of operations for the duration of a maintenance period. When operations are started, after the maintenance, all operations are performed including those for the events during the maintenance. Note that this setting affects only problem escalations; recovery and update operations will not be affected. If you unmark this checkbox, operations will be executed without delay even during a maintenance period. This option is not available for *Service actions*.

Parameter	Description
Notify about canceled escalations	Unmark this checkbox to disable notifications about canceled escalations (when host, item, trigger or action is disabled).

All mandatory input fields are marked with a red asterisk.

To configure details of a new operation, click on Add in the *Operations* block. To edit an existing operation, click on Edit next to the operation. A pop-up window will open where you can edit the operation step details.

Operation details



Parameter	Description
Operation	Select the operation: **Send message** - send message to user. **<remote command name>** - execute a remote command. Commands are available for execution if previously defined in global scripts with *Action operation* selected as its scope. More operations are available for discovery and autoregistration based events (see above).
Steps	Select the step(s) to assign the operation to in an escalation schedule: **From** - execute starting with this step. **To** - execute until this step (0=infinity, execution will not be limited).
Step duration	Custom duration for these steps (0=use default step duration). Time suffixes are supported, e.g. 60s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. Several operations can be assigned to the same step. If these operations have different step duration defined, the shortest one is taken into account and applied to the step.

Parameter	Description
Operation type: send message	
Send to user groups	Select user groups to send the message to. The user group must have at least "read" permissions to the host in order to be notified.
Send to users	Select users to send the message to. The user must have at least "read" permissions to the host in order to be notified.
Send only to	Send message to all defined media types or a selected one only.
Custom message	If selected, the custom message can be configured. For notifications about internal events via webhooks, custom message is mandatory.
Subject	Subject of the custom message. The subject may contain macros. It is limited to 255 characters.
Message	The custom message. The message may contain macros. It is limited to certain amount of characters depending on the type of database (see Sending message for more information).
Operation type: remote command	
Target list	Select targets to execute the command on: **Current host** - command is executed on the host of the trigger that caused the problem event. This option will not work if there are multiple hosts in the trigger. **Host** - select host(s) to execute the command on. **Host group** - select host group(s) to execute the command on. Specifying a parent host group implicitly selects all nested host groups. Thus the remote command will also be executed on hosts from nested groups. A command on a host is executed only once, even if the host matches more than once (e.g. from several host groups; individually and from a host group). The target list is meaningless if a custom script is executed on Zabbix server. Selecting more targets in this case only results in the script being executed on the server more times. Note that for global scripts, the target selection also depends on the *Host group* setting in global script configuration. *Target list* option is not available for *Service actions* because in this case remote commands are always executed on Zabbix server.
Conditions	Condition for performing the operation: **Event is not acknowledged** - only when the event is unacknowledged. **Event is acknowledged** - only when the event is acknowledged. *Conditions* option is only available for *Trigger actions*.

When done, click *Add* to add the operation to the list of *Operations*.

1 Sending message

Overview

Sending a message is one of the best ways of notifying people about a problem. That is why it is one of the primary actions offered by Zabbix.

Configuration

To be able to send and receive notifications from Zabbix you have to:

- define the media to send a message to

If the operation takes place outside of the When active time period defined for the selected media in the user configuration, the message will not be sent.

The default trigger severity ('Not classified') **must be** checked in user media configuration if you want to receive notifications for non-trigger events such as discovery, active agent autoregistration or internal events.

- configure an action operation that sends a message to one of the defined media

Attention:
Zabbix sends notifications only to those users that have at least 'read' permissions to the host that generated the event. At least one host of a trigger expression must be accessible.

You can configure custom scenarios for sending messages using escalations.

To successfully receive and read emails from Zabbix, email servers/clients must support standard 'SMTP/MIME email' format since Zabbix sends UTF-8 data (If the subject contains ASCII characters only, it is not UTF-8 encoded.). The subject and the body of the message are base64-encoded to follow 'SMTP/MIME email' format standard.

Message limit after all macros expansion is the same as message limit for Remote commands.

Tracking messages

You can view the status of messages sent in *Monitoring* → *Problems*.

In the *Actions* column you can see summarized information about actions taken. In there green numbers represent messages sent, red ones - failed messages. *In progress* indicates that an action is initiated. *Failed* informs that no action has executed successfully.

If you click on the event time to view event details, you will also see the *Message actions* block containing details of messages sent (or not sent) due to the event.

In *Reports* → *Action log* you will see details of all actions taken for those events that have an action configured.

2 Remote commands

Overview

With remote commands you can define that a certain pre-defined command is automatically executed on the monitored host upon some condition.

Thus remote commands are a powerful mechanism for smart pro-active monitoring.

In the most obvious uses of the feature you can try to:

- Automatically restart some application (web server, middleware, CRM) if it does not respond
- Use IPMI 'reboot' command to reboot some remote server if it does not answer requests
- Automatically free disk space (removing older files, cleaning /tmp) if running out of disk space
- Migrate a VM from one physical box to another depending on the CPU load
- Add new nodes to a cloud environment upon insufficient CPU (disk, memory, whatever) resources

Configuring an action for remote commands is similar to that for sending a message, the only difference being that Zabbix will execute a command instead of sending a message.

Remote commands can be executed by Zabbix server, proxy or agent. Remote commands on Zabbix agent can be executed directly by Zabbix server or through Zabbix proxy. Both on Zabbix agent and Zabbix proxy remote commands are disabled by default. They can be enabled by:

- adding an `AllowKey=system.run[*]` parameter in agent configuration;
- setting the EnableRemoteCommands parameter to '1' in proxy configuration.

Remote commands executed by Zabbix server are run as described in Command execution including exit code checking.

Remote commands are executed even if the target host is in maintenance.

Remote command limit

Remote command limit after resolving all macros depends on the type of database and character set (non-ASCII characters require more than one byte to be stored):

Database	Limit in characters	Limit in bytes
MySQL	65535	65535
Oracle Database	2048	4000
PostgreSQL	65535	not limited
SQLite (only Zabbix proxy)	65535	not limited

Remote command execution output (return value) is limited to 16MB (including trailing whitespace that is truncated). IPMI remote command limit is based on the installed IPMI library. Note that database limits apply to all remote commands.

Configuration

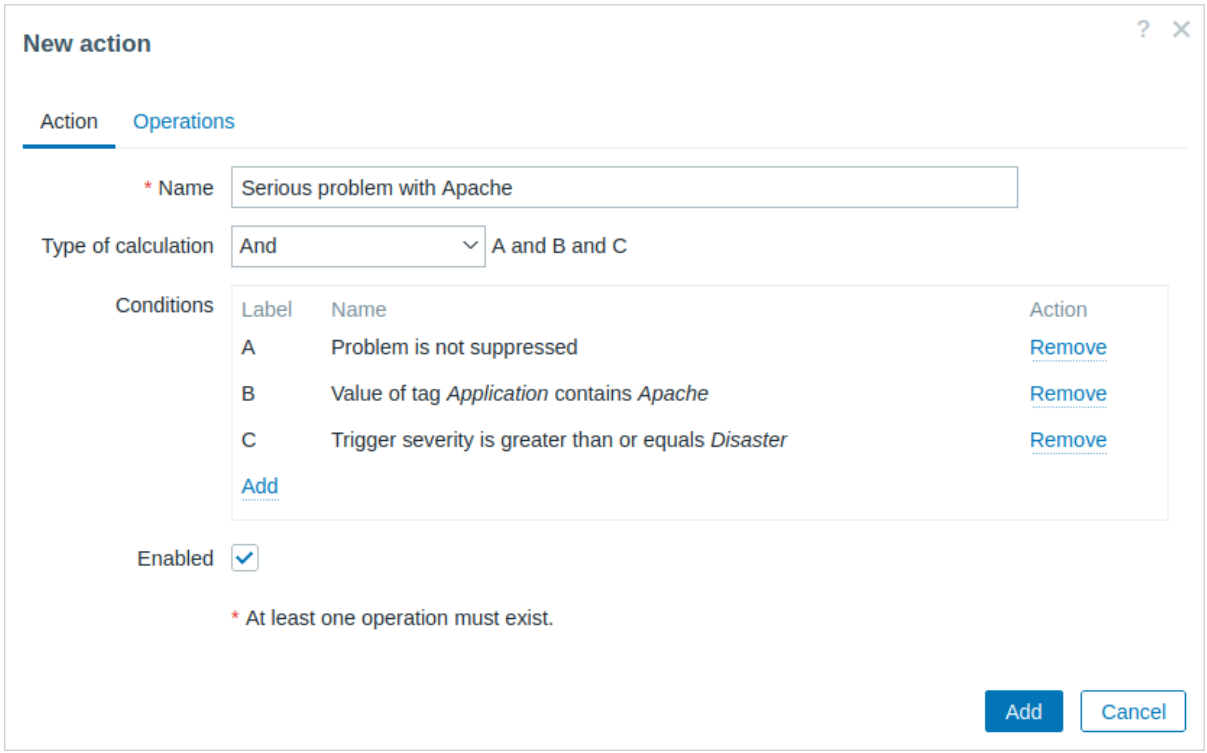
Those remote commands that are executed on Zabbix agent (custom scripts) must be first enabled in the agent configuration.

Make sure that the AllowKey=system.run[<command>,*] parameter is added for each allowed command in agent configuration to allow specific command with nowait mode. Restart agent daemon if changing this parameter.

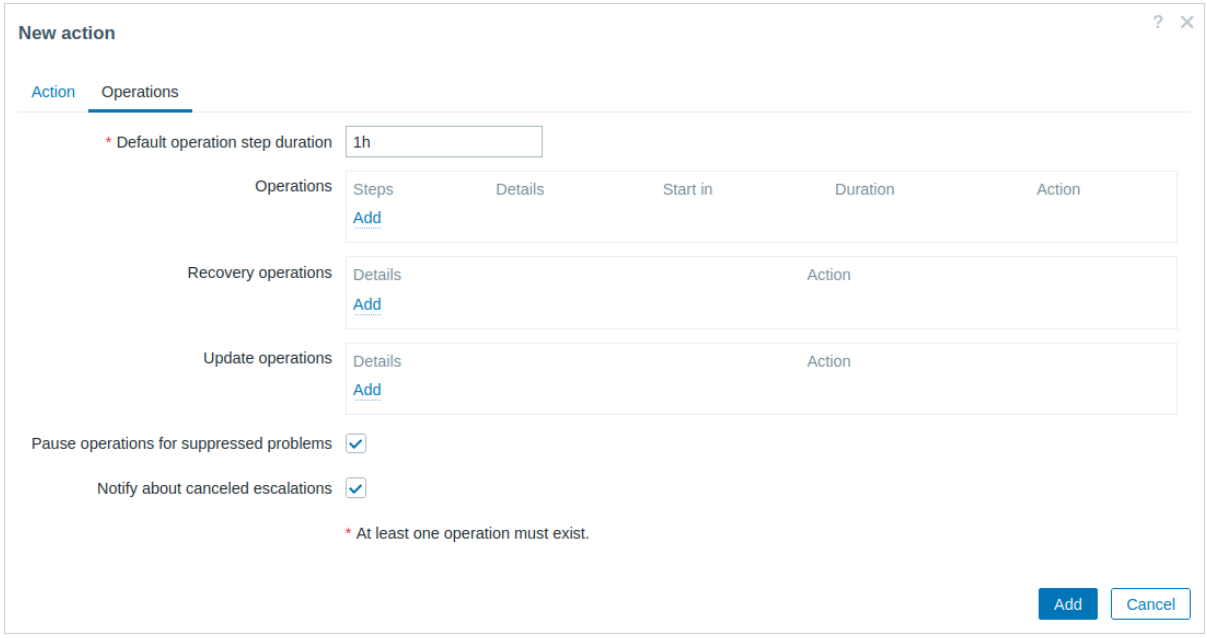
Attention:
Remote commands do not work with active Zabbix agents.

Then, when configuring a new action in *Alerts* → *Actions* → *Trigger actions*:

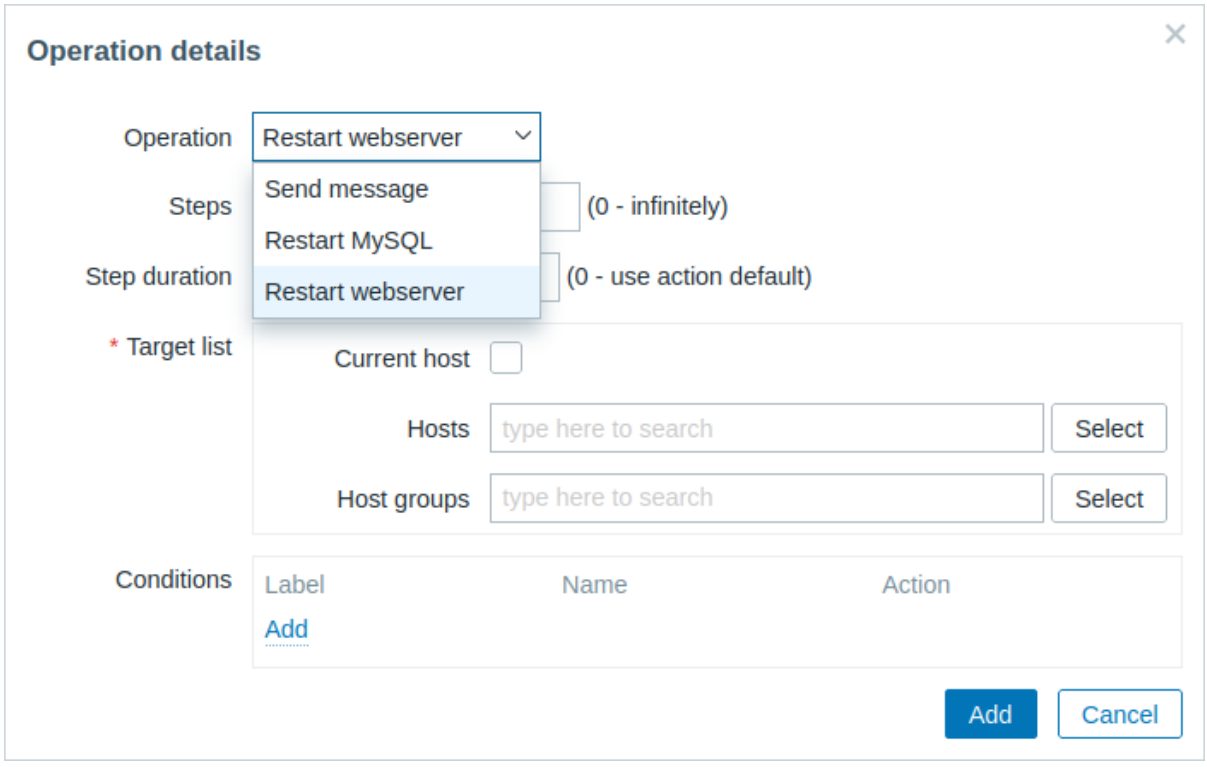
1. Define the appropriate conditions, for example, set that the action is activated upon any disaster problems with one of Apache applications.



2. In the *Operations* tab, click on *Add* in the *Operations*, *Recovery operations*, or *Update operations* block.



3. Select one of the predefined scripts from the *Operation* dropdown list and set the *Target list* for the script.



Predefined scripts

Scripts that are available for action operations (webhook, script, SSH, Telnet, IPMI) are defined in global scripts.

For example:

```
sudo /etc/init.d/apache restart
```

In this case, Zabbix will try to restart an Apache process. With this command, make sure that the command is executed on Zabbix agent (click the *Zabbix agent* button against *Execute on*).

Attention:
Note the use of **sudo** - Zabbix user does not have permissions to restart system services by default. See below for hints on how to configure **sudo**.

Note:
Zabbix agent should run on the remote host and accept incoming connections. Zabbix agent executes commands in background.

Remote commands on Zabbix agent are executed without timeout by the system.run[,nowait] key and are not checked for execution results. On Zabbix server and Zabbix proxy, remote commands are executed with timeout as set in the TrapperTimeout parameter of zabbix_server.conf or zabbix_proxy.conf file and are checked for execution results.

Access permissions

Make sure that the 'zabbix' user has execute permissions for configured commands. One may be interested in using **sudo** to give access to privileged commands. To configure access, execute as root:

```
# visudo
```

Example lines that could be used in *sudoers* file:

```
# allows 'zabbix' user to run all commands without password.
zabbix ALL=NOPASSWD: ALL

# allows 'zabbix' user to restart apache without password.
zabbix ALL=NOPASSWD: /etc/init.d/apache restart
```

Note:
On some systems *sudoers* file will prevent non-local users from executing commands. To change this, comment out **requiretty** option in */etc/sudoers*.

Remote commands with multiple interfaces

If the target system has multiple interfaces of the selected type (Zabbix agent or IPMI), remote commands will be executed on the default interface.

It is possible to execute remote commands via SSH and Telnet using another interface than the Zabbix agent one. The available interface to use is selected in the following order:

- Zabbix agent default interface
- SNMP default interface
- JMX default interface
- IPMI default interface

IPMI remote commands

For IPMI remote commands the following syntax should be used:

```
<command> [<value>]
```

where

- <command> - one of IPMI commands without spaces
- <value> - 'on', 'off' or any unsigned integer. <value> is an optional parameter.

Examples

Examples of global scripts that may be used as remote commands in action operations.

Example 1

Restart of Windows on certain condition.

In order to automatically restart Windows upon a problem detected by Zabbix, define the following script:

Script parameter	Value
Scope	'Action operation'
Type	'Script'
Command	c:\windows\system32\shutdown.exe -r -f

Example 2

Restart the host by using IPMI control.

Script parameter	Value
Scope	'Action operation'
Type	'IPMI'
Command	reset

Example 3

Power off the host by using IPMI control.

Script parameter	Value
Scope	'Action operation'
Type	'IPMI'
Command	power off

3 Additional operations

Overview

In this section you may find some details of additional operations for discovery/autoregistration events.

Adding host

Hosts are added during the discovery process, as soon as a host is discovered, rather than at the end of the discovery process.

Note:
As network discovery can take some time due to many unavailable hosts/services having patience and using reasonable IP ranges is advisable.

When adding a host, its name is decided by the standard **gethostbyname** function. If the host can be resolved, resolved name is used. If not, the IP address is used. Besides, if IPv6 address must be used for a host name, then all ":" (colons) are replaced by "_" (underscores), since colons are not allowed in host names.

Attention:
If performing discovery by a proxy, currently hostname lookup still takes place on Zabbix server.

Attention:
If a host already exists in Zabbix configuration with the same name as a newly discovered one, versions of Zabbix prior to 1.8 would add another host with the same name. Zabbix 1.8.1 and later adds **_N** to the hostname, where **N** is increasing number, starting with 2.

4 Using macros in messages

Overview

In message subjects and message text you can use macros for more efficient problem reporting.

In addition to a number of built-in macros, user macros and expression macros are also supported. A full list of macros supported by Zabbix is available.

Examples

Examples here illustrate how you can use macros in messages.

Example 1

Message subject:

```
Problem: {TRIGGER.NAME}
```

When you receive the message, the message subject will be replaced by something like:

```
Problem: Processor load is too high on Zabbix server
```

Example 2

Message:

```
Processor load is: {?last(/zabbix.zabbix.com/system.cpu.load[,avg1])}
```

When you receive the message, the message will be replaced by something like:

```
Processor load is: 1.45
```

Example 3

Message:

```
Latest value: {?last(/{HOST.HOST}/{ITEM.KEY})}
MAX for 15 minutes: {?max(/{HOST.HOST}/{ITEM.KEY},15m)}
MIN for 15 minutes: {?min(/{HOST.HOST}/{ITEM.KEY},15m)}
```

When you receive the message, the message will be replaced by something like:

```
Latest value: 1.45
MAX for 15 minutes: 2.33
MIN for 15 minutes: 1.01
```

Example 4

Message:

```
http://<server_ip_or_name>/zabbix/tr_events.php?triggerid={TRIGGER.ID}&eventid={EVENT.ID}
```

When you receive the message, it will contain a link to the *Event details* page, which provides information about the event, its trigger, and a list of latest events generated by the same trigger.

Example 5

Informing about values from several hosts in a trigger expression.

Message:

```
Problem name: {TRIGGER.NAME}
Trigger expression: {TRIGGER.EXPRESSION}

1. Item value on {HOST.NAME1}: {ITEM.VALUE1} ({ITEM.NAME1})
2. Item value on {HOST.NAME2}: {ITEM.VALUE2} ({ITEM.NAME2})
```

When you receive the message, the message will be replaced by something like:

```
Problem name: Processor load is too high on a local host
Trigger expression: last(/Myhost/system.cpu.load[percpu,avg1])>5 or last(/Myotherhost/system.cpu.load[perc

1. Item value on Myhost: 0.83 (Processor load (1 min average per core))
2. Item value on Myotherhost: 5.125 (Processor load (1 min average per core))
```

Example 6

Receiving details of both the problem event and recovery event in a recovery message:

Message:

```
Problem:

Event ID: {EVENT.ID}
Event value: {EVENT.VALUE}
Event status: {EVENT.STATUS}
Event time: {EVENT.TIME}
Event date: {EVENT.DATE}
Event age: {EVENT.AGE}
Event acknowledgment: {EVENT.ACK.STATUS}
Event update history: {EVENT.UPDATE.HISTORY}

Recovery:

Event ID: {EVENT.RECOVERY.ID}
```

```
Event value: {EVENT.RECOVERY.VALUE}
Event status: {EVENT.RECOVERY.STATUS}
Event time: {EVENT.RECOVERY.TIME}
Event date: {EVENT.RECOVERY.DATE}
Operational data: {EVENT.OPDATA}
```

When you receive the message, the macros will be replaced by something like:

```
Problem:

Event ID: 21874
Event value: 1
Event status: PROBLEM
Event time: 13:04:30
Event date: 2018.01.02
Event age: 5m
Event acknowledgment: Yes
Event update history: 2018.01.02 13:05:51 "John Smith (Admin)"
Actions: acknowledged.

Recovery:

Event ID: 21896
Event value: 0
Event status: OK
Event time: 13:10:07
Event date: 2018.01.02
Operational data: Current value is 0.83
```

Attention:
Separate notification macros for the original problem event and recovery event are supported since Zabbix 2.2.0.

3 Recovery operations

Overview

Recovery operations allow you to be notified when problems are resolved.

Both messages and remote commands are supported in recovery operations. While several operations can be added, escalation is not supported - all operations are assigned to a single step and therefore will be performed simultaneously.

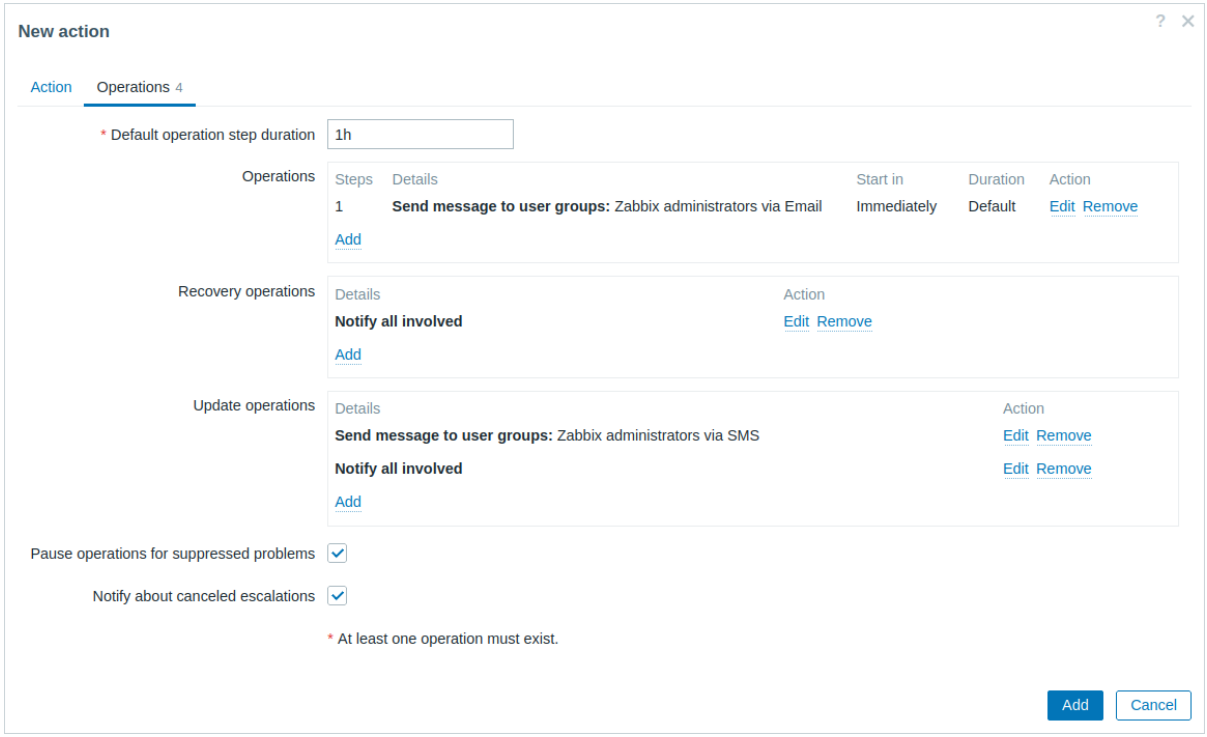
Use cases

Some use cases for recovery operations are as follows:

1. Notify on a recovery all users that were notified on the problem:
    - Select *Notify all involved* as operation type.
2. Have multiple operations upon recovery: send a notification and execute a remote command:
    - Add operation types for sending a message and executing a command.
3. Open a ticket in external helpdesk/ticketing system and close it when the problem is resolved:
    - Create an external script that communicates with the helpdesk system.
    - Create an action having operation that executes this script and thus opens a ticket.
    - Have a recovery operation that executes this script with other parameters and closes the ticket.
    - Use the {EVENT.ID} macro to reference the original problem.

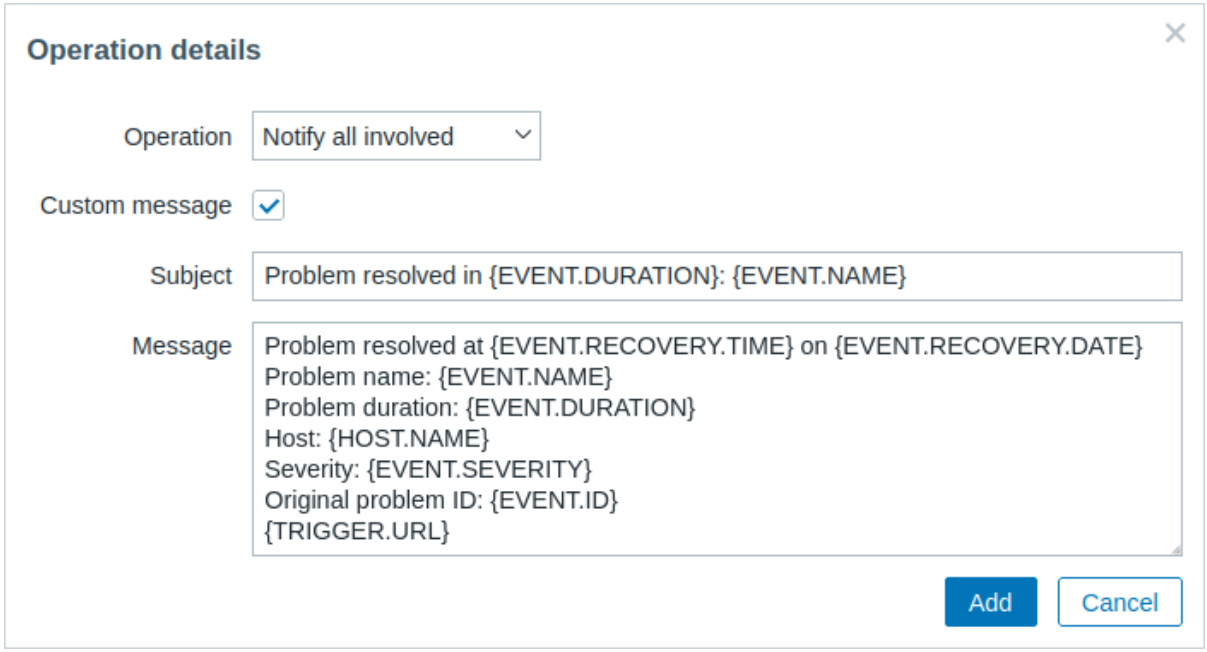
Configuring a recovery operation

To configure a recovery operation, go to the *Operations* tab in action configuration.



To configure details of a new recovery operation, click on Add in the *Recovery operations* block. To edit an existing operation, click on Edit next to the operation. A pop-up window will open where you can edit the operation step details.

Recovery operation details



Three operation types are available for recovery events:

- **Send message** - send recovery message to specified user.
- **Notify all involved** - send recovery message to all users who were notified on the problem event.
- **<remote command name>** - execute a remote command. Commands are available for execution if previously defined in global scripts with *Action operation* selected as its scope.

Parameters for each operation type are described below. All mandatory input fields are marked with a red asterisk. When done, click on *Add* to add operation to the list of *Recovery operations*.

Note:
Note that if the same recipient is defined in several operation types without specified *Custom message*, duplicate notifications are not sent.

Operation type: send message

Parameter		Description
Send to user groups		Select user groups to send the recovery message to. The user group must have at least ”read” permissions to the host in order to be notified.
Send to users		Select users to send the recovery message to. The user must have at least ”read” permissions to the host in order to be notified.
Send only to		Send default recovery message to all defined media types or a selected one only.
Custom message		If selected, a custom message can be defined.
	Subject	Subject of the custom message. The subject may contain macros.
	Message	The custom message. The message may contain macros.

Operation type: remote command

Parameter	Description
Target list	Select targets to execute the command on: **Current host** - command is executed on the host of the trigger that caused the problem event. This option will not work if there are multiple hosts in the trigger. **Host** - select host(s) to execute the command on. **Host group** - select host group(s) to execute the command on. Specifying a parent host group implicitly selects all nested host groups. Thus the remote command will also be executed on hosts from nested groups. A command on a host is executed only once, even if the host matches more than once (e.g. from several host groups; individually and from a host group). The target list is meaningless if the command is executed on Zabbix server. Selecting more targets in this case only results in the command being executed on the server more times. Note that for global scripts, the target selection also depends on the *Host group* setting in global script configuration.

Operation type: notify all involved

Parameter		Description
Custom message		If selected, a custom message can be defined.
	Subject	Subject of the custom message. The subject may contain macros.
	Message	The custom message. The message may contain macros.

4 Update operations

Overview

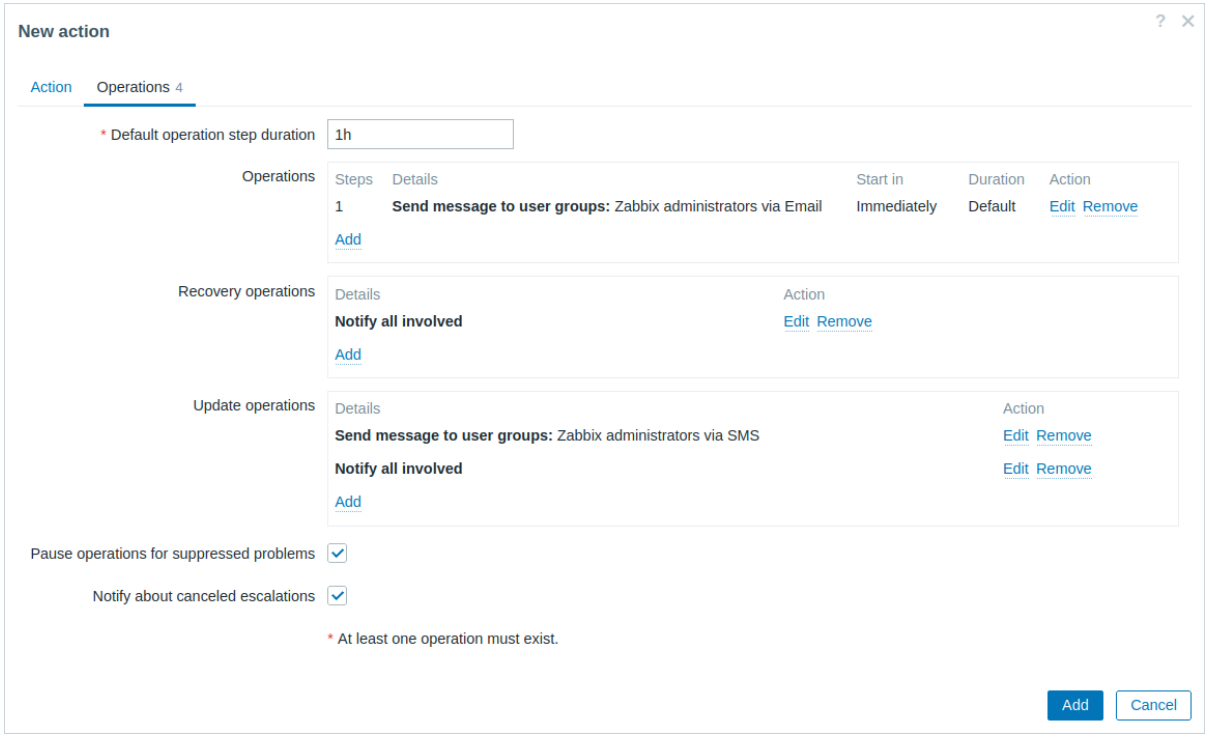
Update operations are available in actions with the following event sources:

- *Triggers* - when problems are updated by other users, i.e. commented upon, acknowledged, severity has been changed, closed (manually);
- *Services* - when the severity of a service has changed but the service is still not recovered.

Both messages and remote commands are supported in update operations. While several operations can be added, escalation is not supported - all operations are assigned to a single step and therefore will be performed simultaneously.

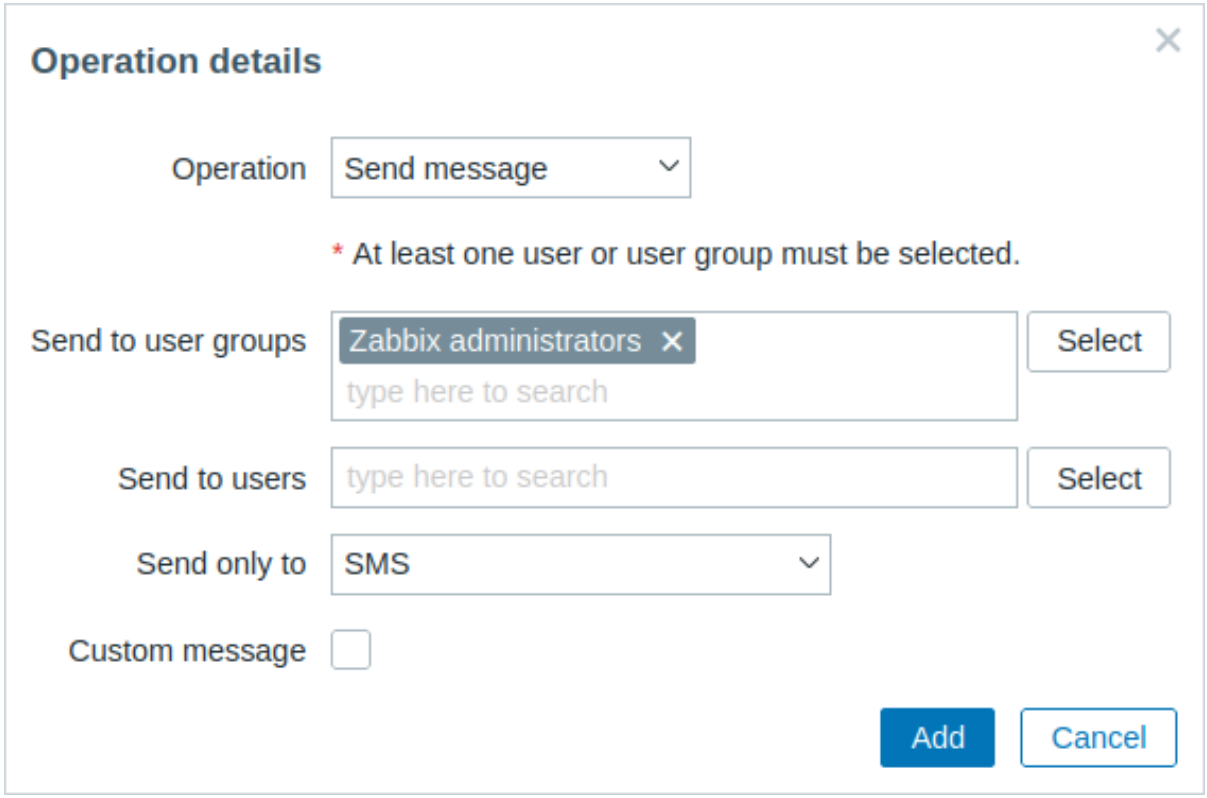
Configuring an update operation

To configure an update operation go to the *Operations* tab in action configuration.



To configure details of a new update operation, click on Add in the *Update operations* block. To edit an existing operation, click on Edit next to the operation. A pop-up window will open where you can edit the operation step details.

Update operation details



Update operations offer the same set of parameters as Recovery operations.

5 Escalations

Overview

With escalations you can create custom scenarios for sending notifications or executing remote commands.

In practical terms it means that:

- Users can be informed about new problems immediately.
- Notifications can be repeated until the problem is resolved.
- Sending a notification can be delayed.
- Notifications can be escalated to another ”higher” user group.
- Remote commands can be executed immediately or when a problem is not resolved for a lengthy period.

Actions are escalated based on the **escalation step**. Each step has a duration in time.

You can define both the default duration and a custom duration of an individual step. The minimum duration of one escalation step is 60 seconds.

You can start actions, such as sending notifications or executing commands, from any step. Step one is for immediate actions. If you want to delay an action, you can assign it to a later step. For each step, several actions can be defined.

The number of escalation steps is not limited.

Escalations are defined when configuring an operation. Escalations are supported for problem operations only, not recovery.

Miscellaneous aspects of escalation behavior

Let’s consider what happens in different circumstances if an action contains several escalation steps.

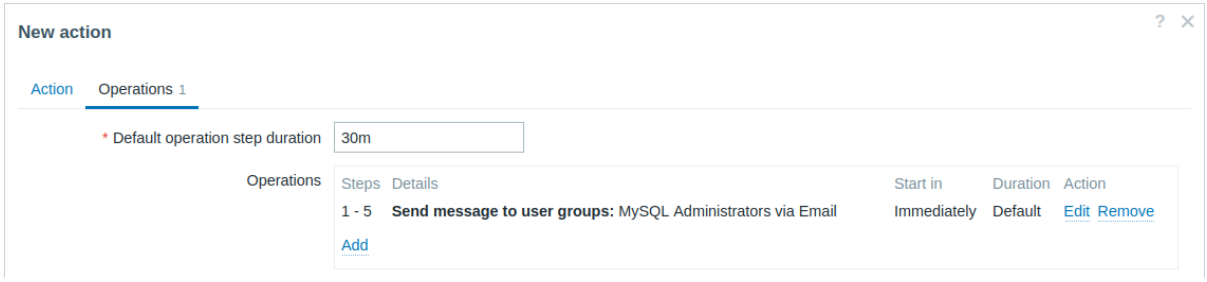
Situation	Behavior
The host in question goes into maintenance after the initial problem notification is sent	Depending on the *Pause operations for suppressed problems* setting in action configuration, all remaining escalation steps are executed either with a delay caused by the maintenance period or without delay. A maintenance period does not cancel operations.
*The time period defined in the **Time period** action condition ends after the initial notification is sent*	All remaining escalation steps are executed. The *Time period* condition cannot stop operations; it has effect with regard to when actions are started/not started, not operations.
A problem starts during maintenance and continues (is not resolved) after maintenance ends	Depending on the *Pause operations for suppressed problems* setting in action configuration, all escalation steps are executed either from the moment maintenance ends or immediately.
A problem starts during a no-data maintenance and continues (is not resolved) after maintenance ends	It must wait for the trigger to fire, before all escalation steps are executed.
Different escalations follow in close succession and overlap	The execution of each new escalation supersedes the previous escalation, but for at least one escalation step that is always executed on the previous escalation. This behavior is relevant in actions upon events that are created with EVERY problem evaluation of the trigger.
During an escalation in progress (like a message being sent), based on any type of event: - the action is disabled Based on trigger event: - the trigger is disabled - the host or item is disabled Based on internal event about triggers: - the trigger is disabled Based on internal event about items/low-level discovery rules: - the item is disabled - the host is disabled	The message in progress is sent and then one more message on the escalation is sent. The follow-up message will have the cancellation text at the beginning of the message body (*NOTE: Escalation canceled*) naming the reason (for example, *NOTE: Escalation canceled: action ’<Action name>’ disabled*). This way the recipient is informed that the escalation is canceled and no more steps will be executed. This message is sent to all who received the notifications before. The reason of cancellation is also logged to the server log file (starting from Debug Level 3=Warning). Note that the *Escalation canceled* message is also sent if operations are finished, but recovery operations are configured and are not executed yet.
During an escalation in progress (like a message being sent) the action is deleted	No more messages are sent. The information is logged to the server log file (starting from Debug Level 3=Warning), for example: `escalation canceled: action id:334 deleted`

Escalation examples

Example 1

Sending a repeated notification once every 30 minutes (5 times in total) to a "MySQL Administrators" group. To configure:

- In *Operations* tab, set the *Default operation step duration* to "30m" (30 minutes).
- Set the escalation *Steps* to be from "1" to "5".
- Select the "MySQL Administrators" group as the recipients of the message.



Notifications will be sent at 0:00, 0:30, 1:00, 1:30, 2:00 hours after the problem starts (unless, of course, the problem is resolved sooner).

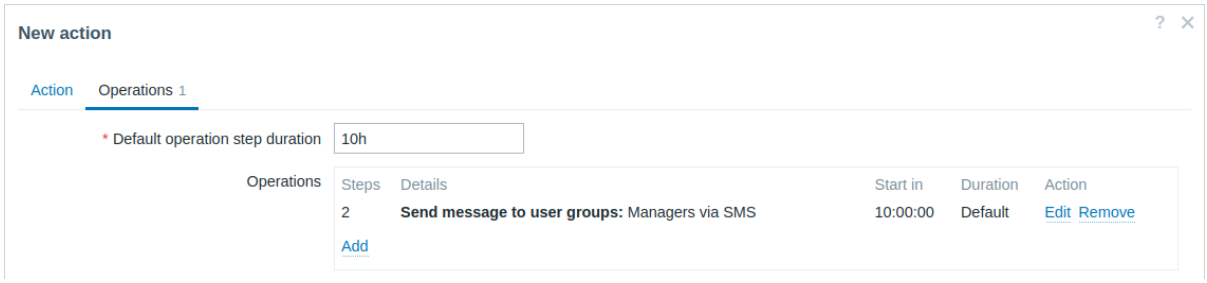
If the problem is resolved and a recovery message is configured, it will be sent to those who received at least one problem message within this escalation scenario.

Note:
If the trigger that generated an active escalation is disabled, Zabbix sends an informative message about it to all those that have already received notifications.

Example 2

Sending a delayed notification about a long-standing problem. To configure:

- In *Operations* tab, set the *Default operation step duration* to "10h" (10 hours).
- Set the escalation *Steps* to be from "2" to "2".



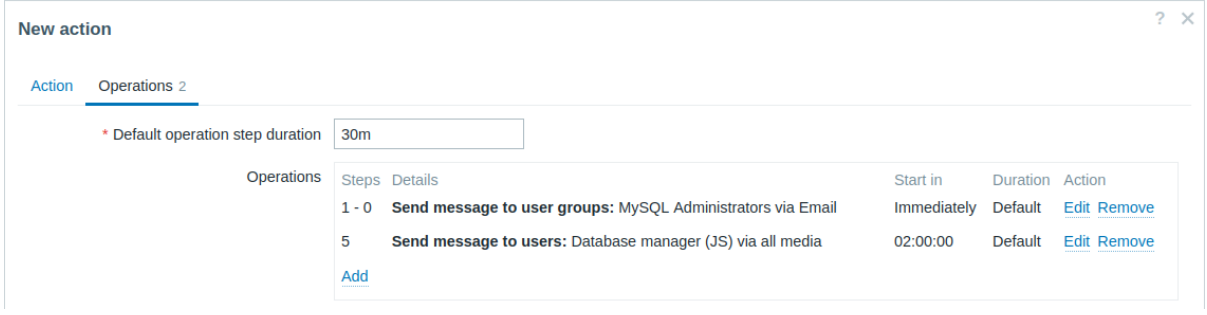
A notification will only be sent at Step 2 of the escalation scenario, or 10 hours after the problem starts.

You can customize the message text to something like "The problem is more than 10 hours old".

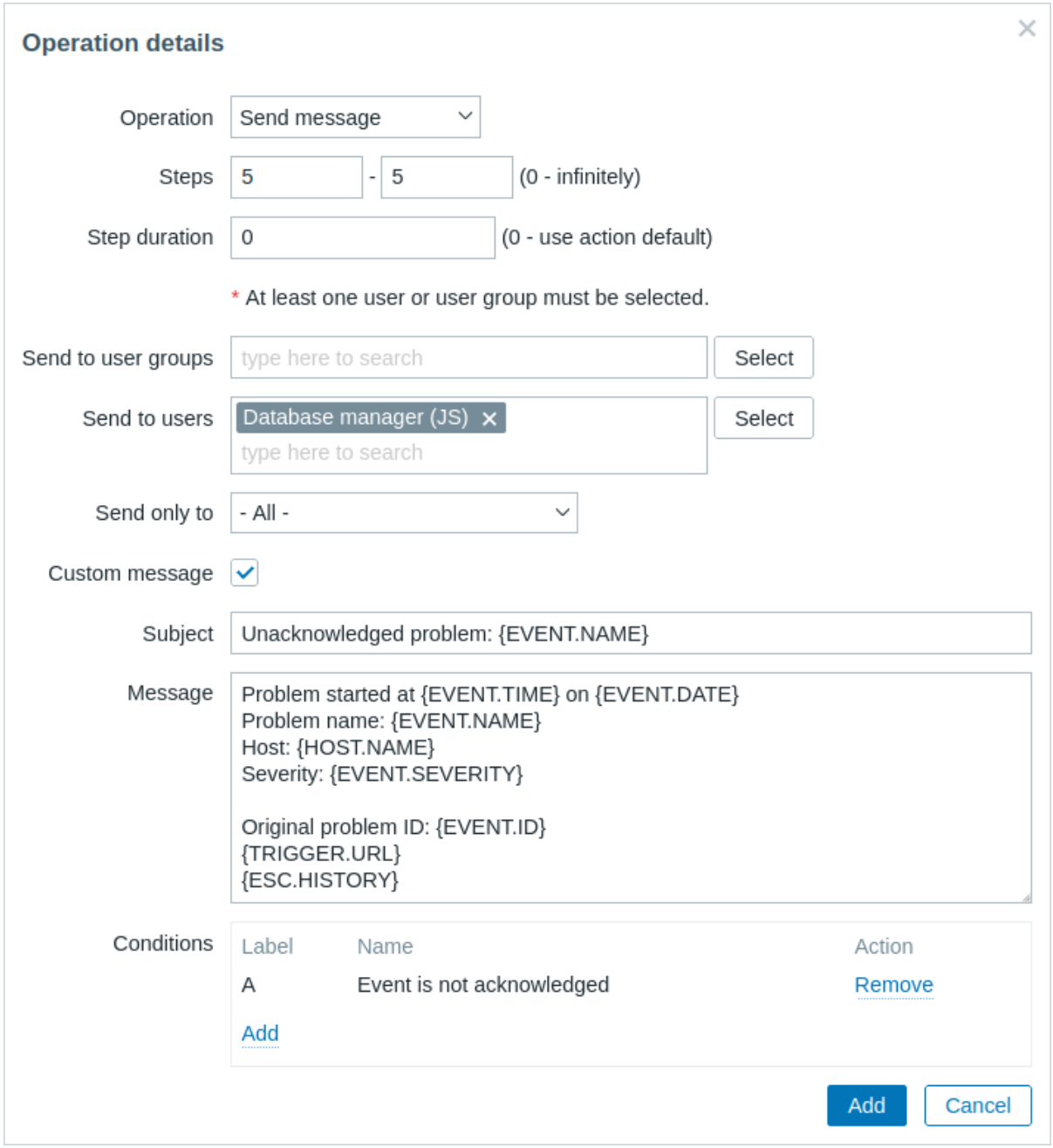
Example 3

Escalating the problem to the Boss.

In the first example above we configured periodical sending of messages to MySQL administrators. In this case, the administrators will get four messages before the problem will be escalated to the Database manager. Note that the manager will get a message only in case the problem is not acknowledged yet, supposedly no one is working on it.



Details of Operation 2:



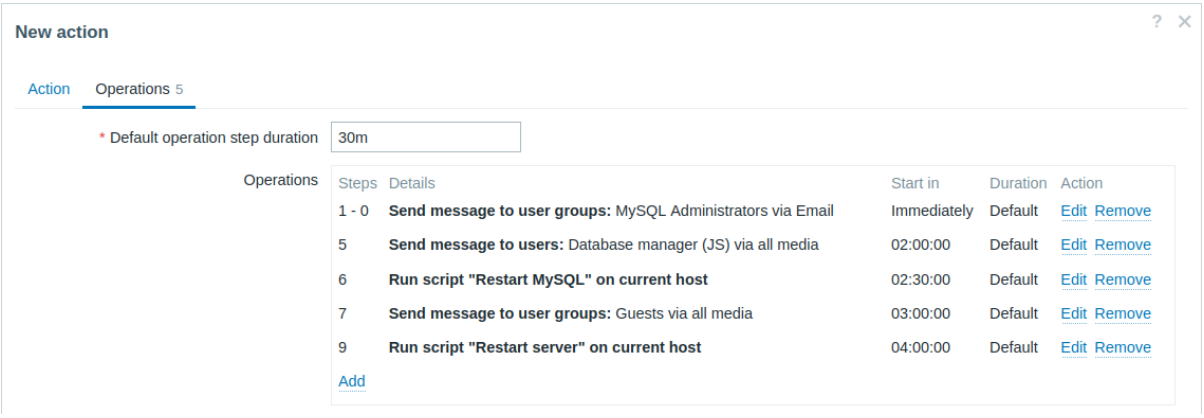
Note the use of {ESC.HISTORY} macro in the customized message. The macro will contain information about all previously executed steps on this escalation, such as notifications sent and commands executed.

Example 4

A more complex scenario. After multiple messages to MySQL administrators and escalation to the manager, Zabbix will try to restart the MySQL database. It will happen if the problem exists for 2:30 hours and it hasn't been acknowledged.

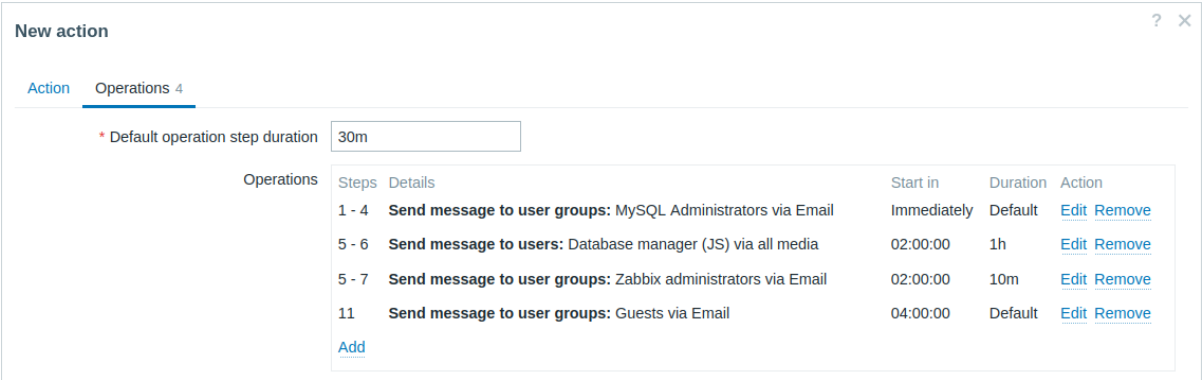
If the problem still exists, after another 30 minutes Zabbix will send a message to all guest users.

If this does not help, after another hour Zabbix will reboot server with the MySQL database (second remote command) using IPMI commands.



Example 5

An escalation with several operations assigned to one step and custom intervals used. The default operation step duration is 30 minutes.



Notifications will be sent as follows:

- To MySQL administrators at 0:00, 0:30, 1:00, 1:30 after the problem starts.
- To Database manager at 2:00 and 2:10. (and not at 3:00; seeing that steps 5 and 6 overlap with the next operation, the shorter custom step duration of 10 minutes in the next operation overrides the longer step duration of 1 hour tried to set here).
- To Zabbix administrators at 2:00, 2:10, 2:20 after the problem starts (the custom step duration of 10 minutes working).
- To guest users at 4:00 hours after the problem start (the default step duration of 30 minutes returning between steps 8 and 11).

3 Receiving notification on unsupported items

Overview

Receiving notifications on unsupported items is supported since Zabbix 2.2.

It is part of the concept of internal events in Zabbix, allowing users to be notified on these occasions. Internal events reflect a change of state:

- when items go from 'normal' to 'unsupported' (and back);
- when triggers go from 'normal' to 'unknown' (and back);
- when low-level discovery rules go from 'normal' to 'unsupported' (and back).

This section presents a how-to for **receiving notification** when an item turns unsupported.

Configuration

Overall, the process of setting up the notification should feel familiar to those who have set up alerts in Zabbix before.

Step 1

Configure some media, such as email, SMS, or script to use for the notifications. Refer to the corresponding sections of the manual to perform this task.

Attention:
For notifying on internal events the default severity ('Not classified') is used, so leave it checked when configuring user media if you want to receive notifications for internal events.

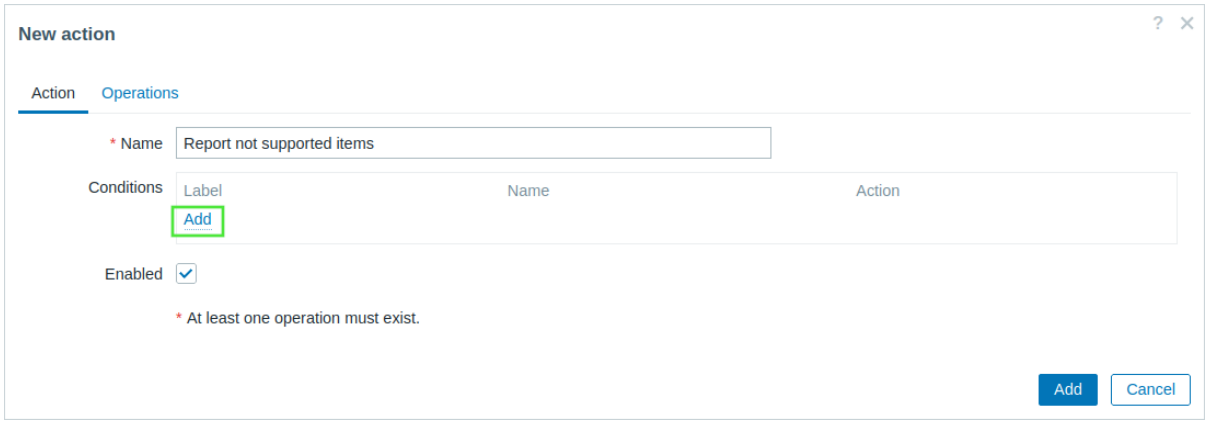
Step 2

Go to *Alerts* → *Actions* → *Internal actions*.

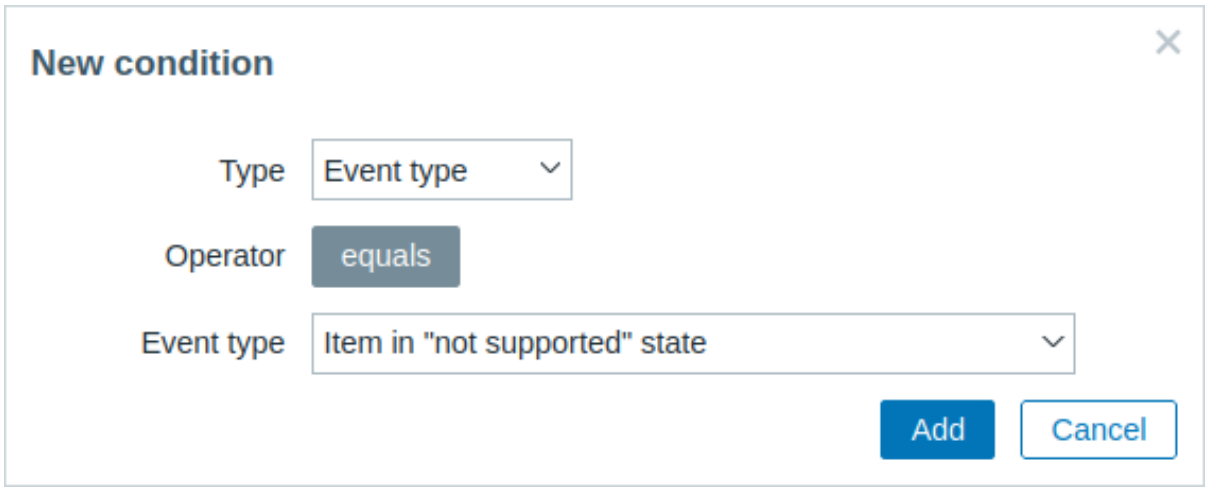
Click on *Create action* at the top right corner of the page to open an action configuration form.

Step 3

In the *Action* tab enter a name for the action. Then click on *Add* in the *Conditions* block to add a new condition.



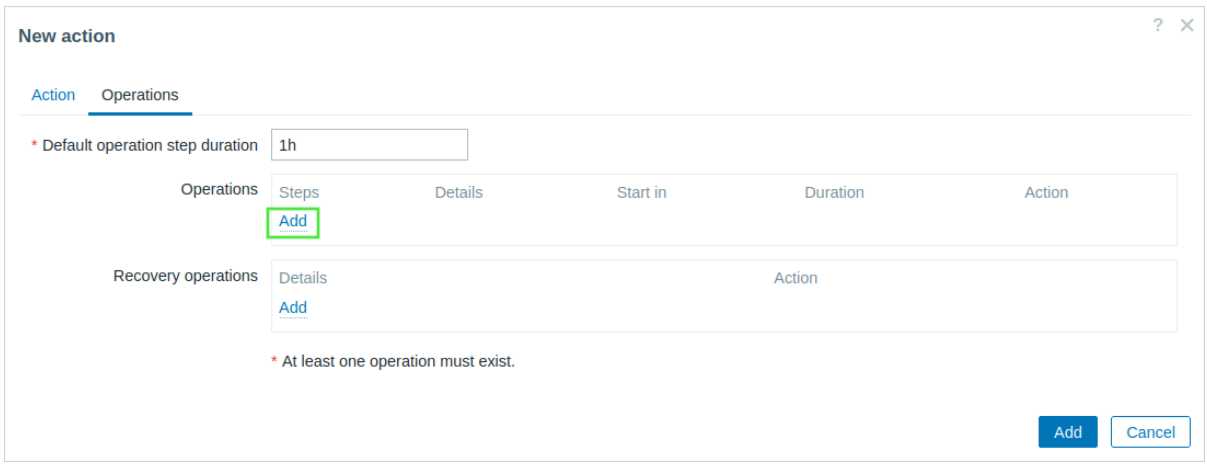
In the *New condition* pop-up window select "Event type" as the condition type and then select "Item in 'not supported' state" as the event type.



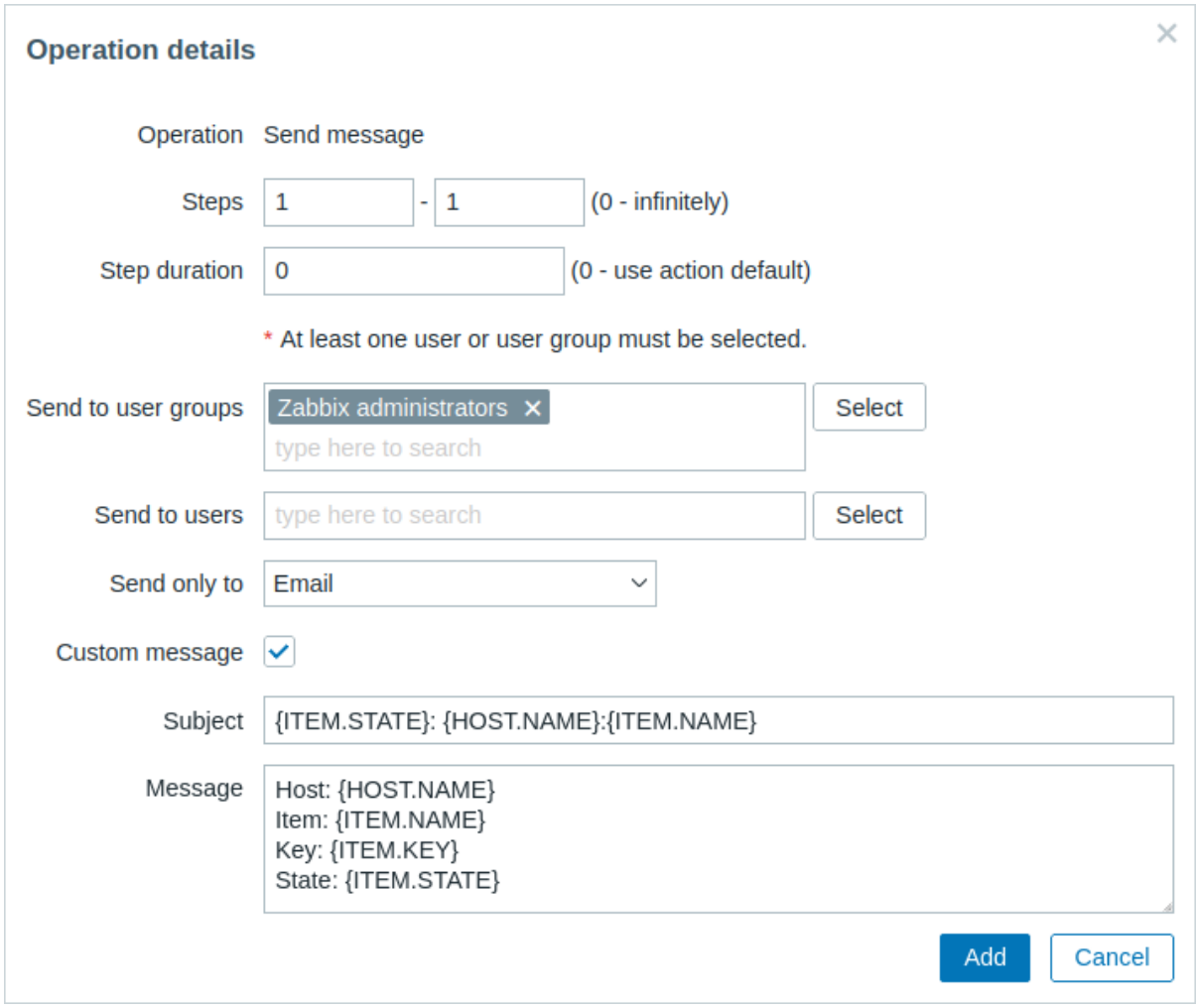
Don't forget to click on *Add* to actually list the condition in the *Conditions* block.

Step 4

In the *Operations* tab, click on *Add* in the *Operations* block to add a new operation.



Select some recipients of the message (user groups/users) and the media type (or "All") to use for delivery. Check the *Custom message* checkbox if you wish to enter the custom subject/content of the problem message.

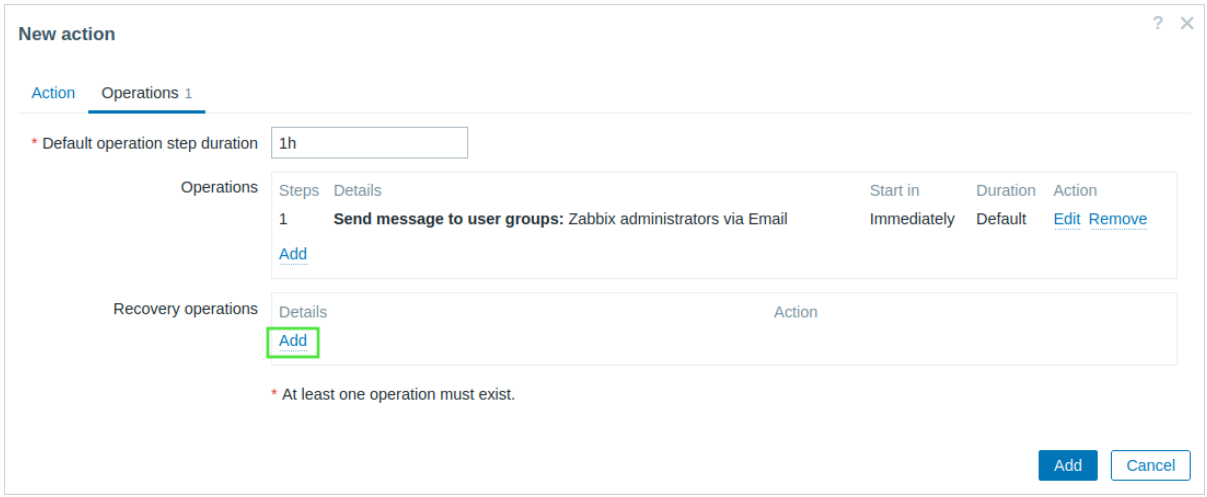


Click on *Add* to actually list the operation in the *Operations* block.

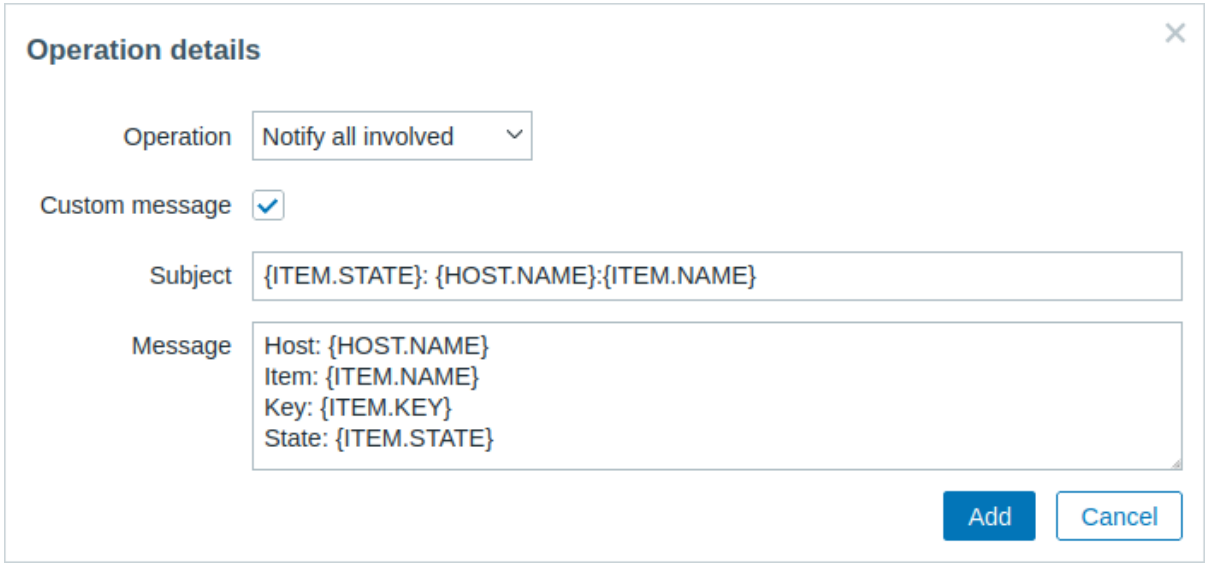
If you wish to receive more than one notification, set the operation step duration (interval between messages sent) and add another step.

Step 5

The *Recovery operations* block allows to configure a recovery notification when an item goes back to the normal state. Click on *Add* in the *Recovery operations* block to add a new recovery operation.



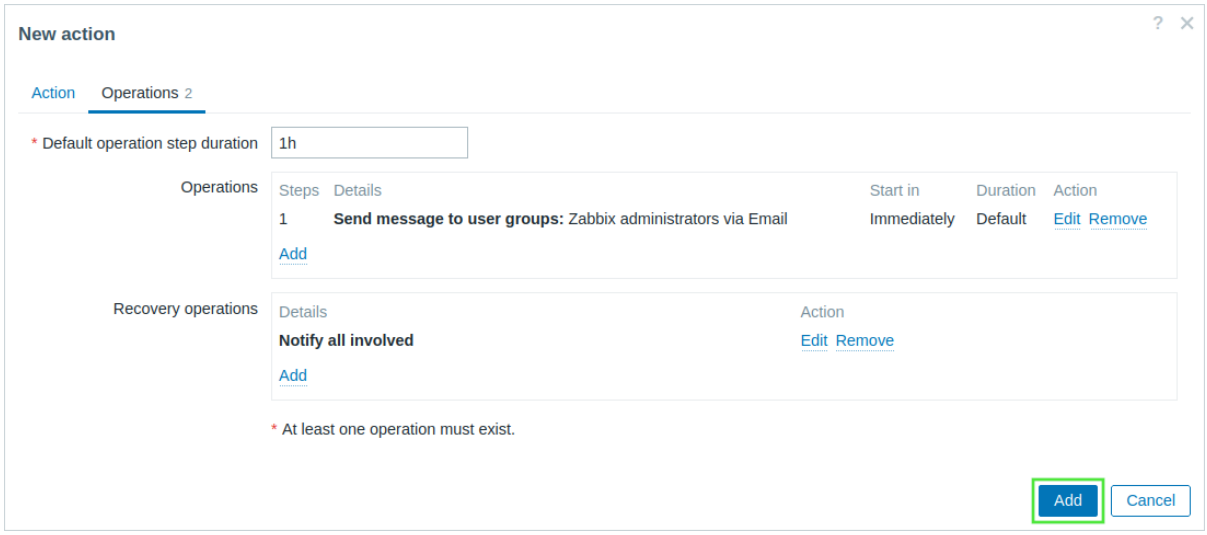
Select the operation type "Notify all involved". Select *Custom message* checkbox if you wish to enter the custom subject/content of the problem message.



Click on *Add* in the *Operation details* pop-up window to actually list the operation in the *Recovery operations* block.

Step 6

When finished, click on the *Add* button at the bottom of the form.



And that's it, you're done! Now you can look forward to receiving your first notification from Zabbix if some item turns unsupported.

11 Macros

Overview

Zabbix supports a number of built-in macros which may be used in various situations. These macros are variables, identified by a specific syntax:

`{MACRO}`

Macros resolve to a specific value depending on the context.

Effective use of macros allows to save time and make Zabbix configuration more transparent.

In one of typical uses, a macro may be used in a template. Thus a trigger on a template may be named "Processor load is too high on {HOST.NAME}". When the template is applied to the host, such as Zabbix server, the name will resolve to "Processor load is too high on Zabbix server" when the trigger is displayed in the Monitoring section.

Macros may be used in item key parameters. A macro may be used for only a part of the parameter, for example `item.key[server_{HOST.HOST}_local]`. Double-quoting the parameter is not necessary as Zabbix will take care of any ambiguous special symbols, if present in the resolved macro.

There are other types of macros in Zabbix.

Zabbix supports the following macros:

- {MACRO} - built-in macro (see full list)
- {<macro>.<func>(<params>)} - macro functions
- {$MACRO} - user-defined macro, optionally with context
- {#MACRO} - macro for low-level discovery
- {?EXPRESSION} - expression macro

1 Macro functions

Overview

Macro functions offer the ability to customize macro values.

Sometimes a macro may resolve to a value that is not necessarily easy to work with. It may be long or contain a specific substring of interest that you would like to extract. This is where macro functions can be useful.

The syntax of a macro function is:

```
{<macro>.<func>(<params>)}
```

where:

- <macro> - the macro to customize (for example {ITEM.VALUE} or {#LLDMACRO})
- <func> - the function to apply
- <params> - a comma-delimited list of function parameters. Parameters must be quoted if they start with " " (space), " or contain), ,.

For example:

```
{{TIME}.fmttime(format,time_shift)}
{{ITEM.VALUE}.regsub(pattern, output)}
{{#LLDMACRO}.regsub(pattern, output)}
```

Supported macro functions

Optional function parameters are indicated by < >.

FUNCTION	Description	Parameters	Supported for
fmtnum(digits)	Number formatting to control the number of digits printed after the decimal point.	**digits** - the number of digits after decimal point. Valid range: 0-20 (since Zabbix 6.4.6). No trailing zeros will be produced.	{ITEM.VALUE} {ITEM.LASTVALUE} Expression macros
fmttime(format,<time_shift>)			

FUNCTION		
Time formatting.	**format** - mandatory format string, compatible with strftime function formatting **time_shift** - the time shift applied to the time before formatting; should start with `-<N><time_unit>` or `+<N><time_unit>`, where *N* - the number of time units to add or subtract; *time_unit* - h (hour), d (day), w (week), M (month) or y (year). Since Zabbix 5.4, time_shift parameter supports multi-step time operations and may include `/<time_unit>` for shifting to the beginning of the time unit (`/d` - midnight, `/w` - 1st day of the week (Monday), `/M` - 1st day of the month, etc.). Examples: `-1w` - exactly 7 days back; `-1w/w` - Monday of the previous week; `-1w/w+1d` - Tuesday of the previous week. Note that time operations are calculated from left to right without priorities. For example, `-1M/d+1h/w` will be parsed as `((-1M/d)+1h)/w`.	{TIME}
iregsub (pattern,output) Substring extraction by a regular expression match (case insensitive).	**pattern** - the regular expression to match **output** - the output options. **\1 - \9** placeholders are supported to capture groups. **\0** returns the matched text.	{ITEM.VALUE} {ITEM.LASTVALUE} Low-level discovery macros (except in low-level discovery rule filter)
regsub (pattern,output) Substring extraction by a regular expression match (case sensitive).	**pattern** - the regular expression to match **output** - the output options. **\1 - \9** placeholders are supported to capture groups. **\0** returns the matched text.	{ITEM.VALUE} {ITEM.LASTVALUE} Low-level discovery macros (except in low-level discovery rule filter)

If a function is used in a supported location, but applied to a macro not supporting macro functions, then the macro evaluates to 'UNKNOWN'.

If pattern is not a correct regular expression then the macro evaluates to 'UNKNOWN' (excluding low-level discovery macros where the function will be ignored in that case and macro will remain unexpanded)

Examples

The ways in which macro functions can be used to customize macro values is illustrated in the following examples on received values:

Received value	Macro	Output
24.3413523	{{ITEM.VALUE}.fmtnum(2)}	24.34
24.3413523	{{ITEM.VALUE}.fmtnum(0)}	24

Received value	Macro	Output
12:36:01	{{TIME}.fmttime(%B)}	October
12:36:01	{{TIME}.fmttime(%d %B,-1M/M)}	1 September
123Log line	{{ITEM.VALUE}.regsub(^[0-9]+, Problem)}	Problem
123 Log line	{{ITEM.VALUE}.regsub("^([0-9]+)", "Problem ID: \1)}	Problem ID: 123
Log line	{{ITEM.VALUE}.regsub(".*", "Problem ID: \1")}	”Problem ID: ”
MySQL crashed errno 123	{{ITEM.VALUE}.regsub("^(\w+).*?([0-9]+)", " Problem ID: \1_\2 ")}	”Problem ID: MySQL_123 ”
123 Log line	{{ITEM.VALUE}.regsub("([1-9]+", "Problem ID: \1")}	*UNKNOWN* (invalid regular expression)
customername_1	{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}	customername
customername_1	{{#IFALIAS}.regsub("(.*)_([0-9]+)", \2)}	1
customername_1	{{#IFALIAS}.regsub("(.*)_([0-9]+", \1)}	{{#IFALIAS}.regsub("(.*)_([0-9]+", \1)} (invalid regular expression)
customername_1	{$MACRO:"{{#IFALIAS}.regsub(\"(.*)_([0-9]+)\", \1)}"}	{$MACRO:"customername"}
customername_1	{$MACRO:"{{#IFALIAS}.regsub(\"(.*)_([0-9]+)\", \2)}"}	{$MACRO:"1"}
customername_1	{$MACRO:"{{#IFALIAS}.regsub(\"(.*)_([0-9]+\", \1)}"}	{$MACRO:"{{#IFALIAS}.regsub(\"(.*)_([0-9]+\", \1)}"} (invalid regular expression)
customername_1	"{$MACRO:\"{{#IFALIAS}.regsub(\\"(.*)_([0-9]+)\\", \1)}\"}"	"{$MACRO:\"customername\"}"
customername_1	"{$MACRO:\"{{#IFALIAS}.regsub(\\"(.*)_([0-9]+)\\", \2)}\"}"	"{$MACRO:\"1\"}"
customername_1	"{$MACRO:\"{{#IFALIAS}.regsub(\\"(.*)_([0-9]+\\", \1)}\"}"	"{$MACRO:\"{{#IFALIAS}.regsub(\\"(.*)_([0-9]+\\", \1)}\"}" (invalid regular expression)

Seeing full item values

Long values of resolved {ITEM.VALUE} and {ITEM.LASTVALUE} macros for text/log items are truncated to 20 characters in some frontend locations. To see the full values of these macros you may use macro functions, e.g.:

{{ITEM.VALUE}.regsub("(.*)", \1)}
 {{ITEM.LASTVALUE}.regsub("(.*)", \1)}

See also: {ITEM.VALUE} and {ITEM.LASTVALUE} macro details.

2 User macros

Overview

User macros are supported in Zabbix for greater flexibility, in addition to the macros supported out-of-the-box.

User macros can be defined on global, template and host level. These macros have a special syntax:

{$MACRO}

Zabbix resolves macros according to the following precedence:

1. host level macros (checked first)
2. macros defined for first level templates of the host (i.e., templates linked directly to the host), sorted by template ID
3. macros defined for second level templates of the host, sorted by template ID
4. macros defined for third level templates of the host, sorted by template ID, etc.
5. global macros (checked last)

In other words, if a macro does not exist for a host, Zabbix will try to find it in the host templates of increasing depth. If still not found, a global macro will be used, if exists.

Warning:
If a macro with the **same name** exists on multiple linked templates of the same level, the macro from the template with the lowest ID will be used. Thus having macros with the same name in multiple templates is a configuration risk.

If Zabbix is unable to find a macro, the macro will not be resolved.

Attention:
Macros (including user macros) are left unresolved in the Configuration section (for example, in the trigger list) by design to make complex configuration more transparent.

User macros can be used in:

- item key parameter
- item update intervals and flexible intervals
- trigger name and description
- trigger expression parameters and constants (see examples)
- many other locations - see the full list

Common use cases of global and host macros

- use a global macro in several locations; then change the macro value and apply configuration changes to all locations with one click
- take advantage of templates with host-specific attributes: passwords, port numbers, file names, regular expressions, etc.

Note:
It is advisable to use host macros instead of global macros because adding, updating or deleting global macros forces incremental configuration update for all hosts. For more information, see Upgrade notes for 6.4.0.

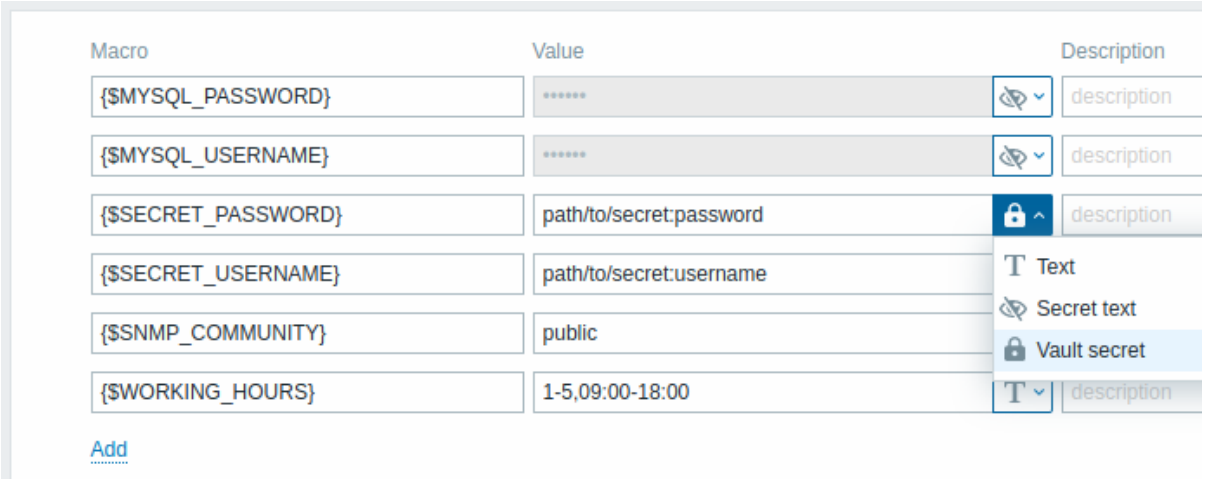
Configuration

To define user macros, go to the corresponding location in the frontend:

- for global macros, visit *Administration* → *Macros*
- for host and template level macros, open host or template properties and look for the *Macros* tab

Note:
If a user macro is used in items or triggers in a template, it is suggested to add that macro to the template even if it is defined on a global level. That way, if the macro type is *text* exporting the template to XML and importing it in another system will still allow it to work as expected. Values of secret macros are not exported.

A user macro has the following attributes:



Parameter	Description
Macro	Macro name. The name must be wrapped in curly brackets and start with a dollar sign. Example: {$FRONTEND_URL}. The following characters are allowed in the macro names: **A-Z** (uppercase only) , **0-9** , _ , **.**

Parameter	Description
Value	Macro value. Three value types are supported: **Text** (default) - plain-text value **Secret text** - the value is masked with asterisks **Vault secret** - the value contains a path/query to a vault secret. To change the value type click on the button at the end of the value input field. Maximum length of a user macro value is 2048 characters (255 characters in versions before 5.2.0).
Description	Text field used to provide more information about this macro.

Attention:
In trigger expressions user macros will resolve if referencing a parameter or constant. They will NOT resolve if referencing a host, item key, function, operator or another trigger expression. Secret macros cannot be used in trigger expressions.

Examples

Example 1

Use of host-level macro in the "Status of SSH daemon" item key:

```
net.tcp.service[ssh,,{$SSH_PORT}]
```

This item can be assigned to multiple hosts, providing that the value of **{$SSH_PORT}** is defined on those hosts.

Example 2

Use of host-level macro in the "CPU load is too high" trigger:

```
last(/ca_001/system.cpu.load[,avg1])>{$MAX_CPULOAD}
```

Such a trigger would be created on the template, not edited in individual hosts.

Note:
If you want to use the amount of values as the function parameter (for example, **max(/host/key,#3)**), include hash mark in the macro definition like this: SOME_PERIOD => #3

Example 3

Use of two macros in the "CPU load is too high" trigger:

```
min(/ca_001/system.cpu.load[,avg1],{$CPULOAD_PERIOD})>{$MAX_CPULOAD}
```

Note that a macro can be used as a parameter of trigger function, in this example function **min()**.

Example 4

Synchronize the agent unavailability condition with the item update interval:

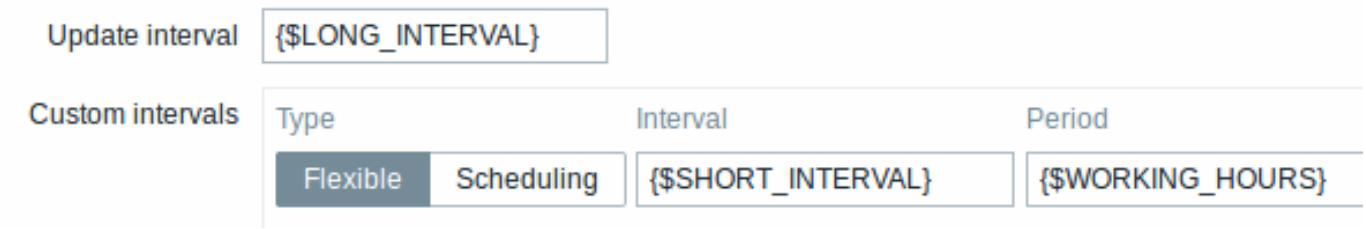
- define {$INTERVAL} macro and use it in the item update interval;
- use {$INTERVAL} as parameter of the agent unavailability trigger:

```
nodata(/ca_001/agent.ping,{$INTERVAL})=1
```

Example 5

Centralize configuration of working hours:

- create a global {$WORKING_HOURS} macro equal to 1-5,09:00-18:00;
- use it in the *Working time* field in *Administration → General → GUI*;
- use it in the *When active* field in *Users → Users*, *Media* tab of a user;
- use it to set up more frequent item polling during working hours:

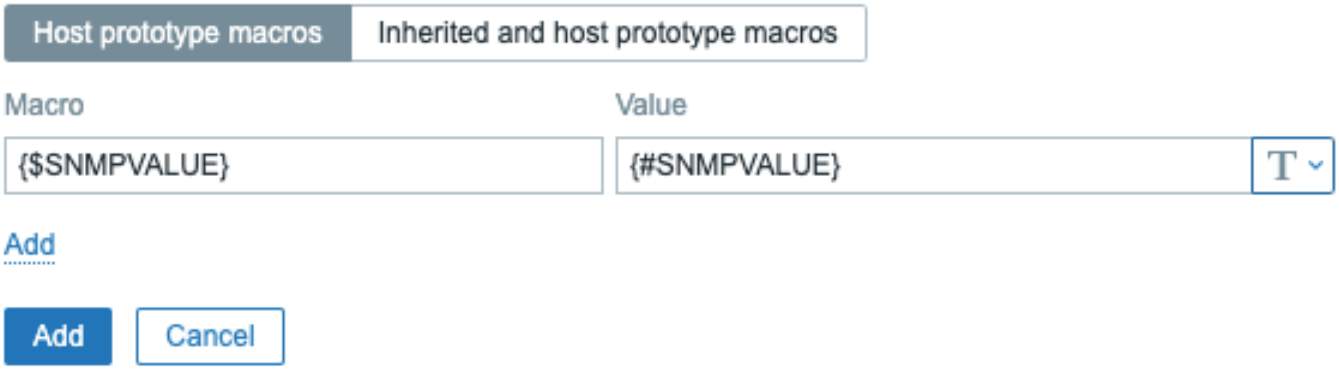


- use it in the *Time period* action condition;
- adjust the working time in *Administration* → *Macros*, if needed.

Example 6

Use host prototype macro to configure items for discovered hosts:

- on a host prototype define user macro {$SNMPVALUE} with {#SNMPVALUE} low-level discovery macro as a value:



- assign *Generic SNMPv2* template to the host prototype;
- use {$SNMPVALUE} in the *SNMP OID* field of *Generic SNMPv2* template items.

User macro context

See user macros with context.

3 User macros with context

Overview

An optional context can be used in user macros, allowing to override the default value with a context-specific one.

The context is appended to the macro name; the syntax depends on whether the context is a static text value:

```
{$MACRO:"static text"}
```

or a regular expression:

```
{$MACRO:regex:"regular expression"}
```

Note that a macro with regular expression context can only be defined in user macro configuration. If the `regex:` prefix is used elsewhere as user macro context, like in a trigger expression, it will be treated as static context.

Context quoting is optional (see also important notes).

Macro context examples:

Example	Description
`{$LOW_SPACE_LIMIT}`	User macro without context.
`{$LOW_SPACE_LIMIT:/tmp}`	User macro with context (static string).
`{$LOW_SPACE_LIMIT:regex:"^/tmp$"}`	User macro with context (regular expression). Same as `{$LOW_SPACE_LIMIT:/tmp}`.
`{$LOW_SPACE_LIMIT:regex:"^/var/log/.*$"}`	User macro with context (regular expression). Matches all strings prefixed with /var/log/.

Use cases

User macros with context can be defined to accomplish more flexible thresholds in trigger expressions (based on the values retrieved by low-level discovery). For example, you may define the following macros:

- {$LOW_SPACE_LIMIT} = 10
- {$LOW_SPACE_LIMIT:/home} = 20
- {$LOW_SPACE_LIMIT:regex:"^/[a-z]+$"} = 30

Then a low-level discovery macro may be used as macro context in a trigger prototype for mounted file system discovery:

```
last(/host/vfs.fs.size[{#FSNAME},pfree])<{$LOW_SPACE_LIMIT:"{#FSNAME}"}
```

After the discovery different low-space thresholds will apply in triggers depending on the discovered mount points or file system types. Problem events will be generated if:

- /home folder has less than 20% of free disk space
- folders that match the regexp pattern (like /etc, /tmp or /var) have less than 30% of free disk space
- folders that don't match the regexp pattern and are not /home have less than 10% of free disk space

Important notes

- If more than one user macro with context exists, Zabbix will try to match the simple context macros first and then context macros with regular expressions in an undefined order.

Warning:
Do not create different context macros matching the same string to avoid undefined behavior.

- If a macro with its context is not found on host, linked templates or globally, then the macro without context is searched for.
- Only low-level discovery macros are supported in the context. Any other macros are ignored and treated as plain text.

Technically, macro context is specified using rules similar to item key parameters, except macro context is not parsed as several parameters if there is a , character:

- Macro context must be quoted with " if the context contains a } character or starts with a " character. Quotes inside quoted context must be escaped with the \ character.
- The \ character itself is not escaped, which means it's impossible to have a quoted context ending with the \ character - the macro {$MACRO:"a:\b\c\"} is invalid.
- The leading spaces in context are ignored, the trailing spaces are not:
  - For example {$MACRO:A} is the same as {$MACRO: A}, but not {$MACRO:A }.
- All spaces before leading quotes and after trailing quotes are ignored, but all spaces inside quotes are not:
  - Macros {$MACRO:"A"}, {$MACRO: "A"}, {$MACRO:"A" } and {$MACRO: "A" } are the same, but macros {$MACRO:"A"} and {$MACRO:" A "} are not.

The following macros are all equivalent, because they have the same context: {$MACRO:A}, {$MACRO: A} and {$MACRO:"A"}. This is in contrast with item keys, where 'key[a]', 'key[a]' and 'key["a"]' are the same semantically, but different for uniqueness purposes.

4 Secret user macros

Zabbix provides two options for protecting sensitive information in user macro values:

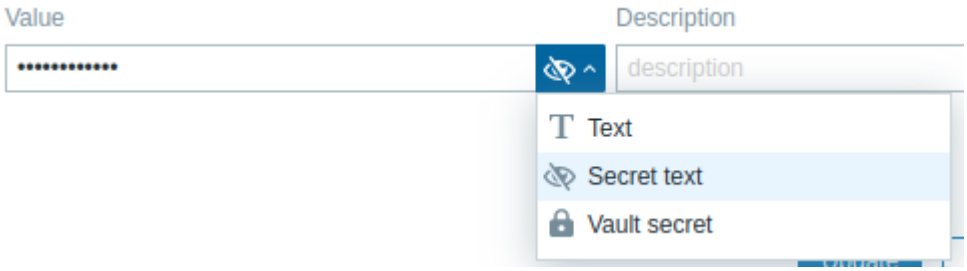
- Secret text
- Vault secret

Note that while the value of a secret macro is hidden, the value can be revealed through the use in items. For example, in an external script an 'echo' statement referencing a secret macro may be used to reveal the macro value to the frontend because Zabbix server has access to the real macro value.

Secret macros cannot be used in trigger expressions.

Secret text Values of secret text macros are masked by the asterisks.

To make macro value 'secret', click on the button at the end of the value field and select the option *Secret text*.



Once the configuration is saved, it will no longer be possible to view the value.

The macro value will be displayed as asterisks.

To enter a new value, hover over the value field and press Set new value button (appears on hover).

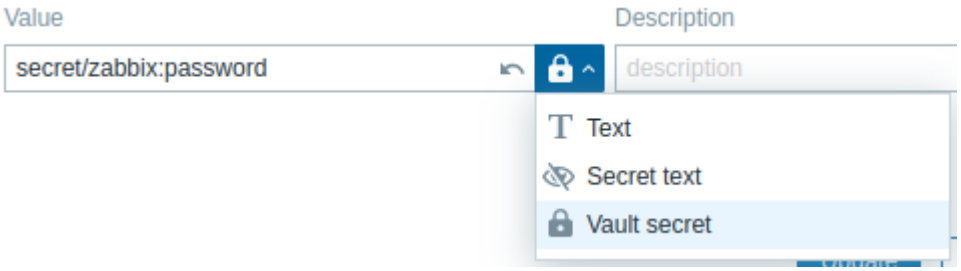


If you change macro value type or press *Set new value*, current value will be erased. To revert the original value, use the backwards arrow at the right end of the *Value* field (only available before saving new configuration). Reverting the value will not expose it.

Note:
URLs that contain a secret macro will not work as the macro in them will be resolved as "******".

Vault secret With Vault secret macros, the actual macro value is stored in an external secret management software (vault).

To configure a Vault secret macro, click on the button at the end of the Value field and select the option *Vault secret*.



The macro value should point to a vault secret. The input format depends on the vault provider. For provider-specific configuration examples, see:

- HashiCorp
- CyberArk

Vault secret values are retrieved by Zabbix server on every refresh of configuration data and then stored in the configuration cache.

To manually trigger refresh of secret values from a vault, use the 'secrets_reload' command-line option.

Zabbix proxy receives values of vault secret macros from Zabbix server on each configuration sync and stores them in its own configuration cache. The proxy never retrieves macro values from the vault directly. That means a Zabbix proxy cannot start data collection after a restart until it receives the configuration data update from Zabbix server for the first time.

Encryption must be enabled between Zabbix server and proxy; otherwise a server warning message is logged.

Warning:
If a macro value cannot be retrieved successfully, the corresponding item using the value will turn unsupported.

5 Low-level discovery macros

Overview

There is a type of macro used within the low-level discovery (LLD) function:

`{#MACRO}`

It is a macro that is used in an LLD rule and returns real values of the file system name, network interface, SNMP OID, etc.

These macros can be used for creating item, trigger and graph *prototypes*. Then, when discovering real file systems, network interfaces etc., these macros are substituted with real values and are the basis for creating real items, triggers and graphs.

These macros are also used in creating host and host group *prototypes* in virtual machine discovery.

Some low-level discovery macros come "pre-packaged" with the LLD function in Zabbix - {#FSNAME}, {#FSTYPE}, {#IFNAME}, {#SNMPINDEX}, {#SNMPVALUE}. However, adhering to these names is not compulsory when creating a custom low-level discovery rule. Then you may use any other LLD macro name and refer to that name.

Supported locations

LLD macros can be used:

- in the low-level discovery rule filter
- for item prototypes in
  - name
  - key parameters
  - unit
  - update interval[1]
  - history storage period[1]
  - trend storage period[1]
  - item value preprocessing steps
  - SNMP OID
  - IPMI sensor field
  - calculated item expression, in:
    - expression constants and function parameters
    - item key parameters
  - SSH script and Telnet script
  - database monitoring SQL query
  - JMX item endpoint field
  - description
  - HTTP agent URL field
  - HTTP agent HTTP query fields field
  - HTTP agent request body field
  - HTTP agent required status codes field
  - HTTP agent headers field key and value
  - HTTP agent HTTP authentication username field
  - HTTP agent HTTP authentication password field
  - HTTP agent HTTP proxy field
  - HTTP agent HTTP SSL certificate file field
  - HTTP agent HTTP SSL key file field
  - HTTP agent HTTP SSL key password field
  - HTTP agent HTTP timeout[1] field
  - tags
- for trigger prototypes in
  - name
  - operational data
  - expression (only in constants and function parameters)
  - URL
  - description
  - tags
- for graph prototypes in
  - name
- for host prototypes in
  - name
  - visible name
  - custom interface fields: IP, DNS, port, SNMP v1/v2 community, SNMP v3 context name, SNMP v3 security name, SNMP v3 authentication passphrase, SNMP v3 privacy passphrase
  - host group prototype name
  - host tag value
  - host macro value
  - (see the full list)

In all those places, except the low-level discovery rule filter, LLD macros can be used inside static user macro context.

Using macro functions

Macro functions are supported with low-level discovery macros (except in low-level discovery rule filter), allowing to extract a certain part of the macro value using a regular expression.

For example, you may want to extract the customer name and interface number from the following LLD macro for the purposes of event tagging:

`{#IFALIAS}=customername_1`

To do so, the `regsub` macro function can be used with the macro in the event tag value field of a trigger prototype:



Note that commas are not allowed in unquoted item key parameters, so the parameter containing a macro function has to be quoted. The backslash (\) character should be used to escape double quotes inside the parameter. Example:

```
net.if.in["{{#IFALIAS}.regsub(\"(.*)_([0-9]+)\", \1)}",bytes]
```

For more information on macro function syntax, see: Macro functions

Macro functions are supported in low-level discovery macros since Zabbix 4.0.

Footnotes

1 In the fields marked with [1] a single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported.

6 Expression macros

Overview

Expression macros are useful for formula calculations. They are calculated by expanding all macros inside and evaluating the resulting expression.

Expression macros have a special syntax:

`{?EXPRESSION}`

The syntax in EXPRESSION is the same as in trigger expressions (see usage limitations below).

{HOST.HOST<1-9>} and {ITEM.KEY<1-9>} macros are supported inside expression macros.

Usage

In the following locations:

- graph names
- map element labels
- map shape labels
- map link labels

only a **single** function, from the following set: `avg`, `last`, `max`, `min`, is allowed as an expression macro, e.g.:

`{?avg(/{HOST.HOST}/{ITEM.KEY},1h)}`

Expressions such as `{?last(/host/item1)/last(/host/item2)}`, `{?count(/host/item1,5m)}` and `{?last(/host/item1)*10}` are incorrect in these locations.

However, in:

- trigger event names
- trigger-based notifications and commands
- problem update notifications and commands

complex expressions are allowed, e.g.:

`{?trendavg(/host/item1,1M:now/M)/trendavg(/host/item1,1M:now/M-1y)*100}`

Note:
When using expression macros in templates please do not specify the template name but rather use {HOST.HOST<N>}, because template names are not substituted with hosts during linkage. You can also omit host reference altogether for the first host, e.g. `{?avg(//item1,1h)}`

See also:

- Supported macros for a list of supported locations of the expression macro
- Example of using an expression macro in the event name

12 Users and user groups

Overview

All users in Zabbix access the Zabbix application through the web-based frontend. Each user is assigned a unique login name and a password.

All user passwords are encrypted and stored in the Zabbix database. Users cannot use their user id and password to log directly into the UNIX server unless they have also been set up accordingly to UNIX. Communication between the web server and the user browser can be protected using SSL.

With a flexible user permission schema you can restrict and differentiate rights to:

- access administrative Zabbix frontend functions
- perform certain actions in the frontend
- access monitored hosts in hostgroups
- use specific API methods

1 Configuring a user

Overview

The initial Zabbix installation has two predefined users:

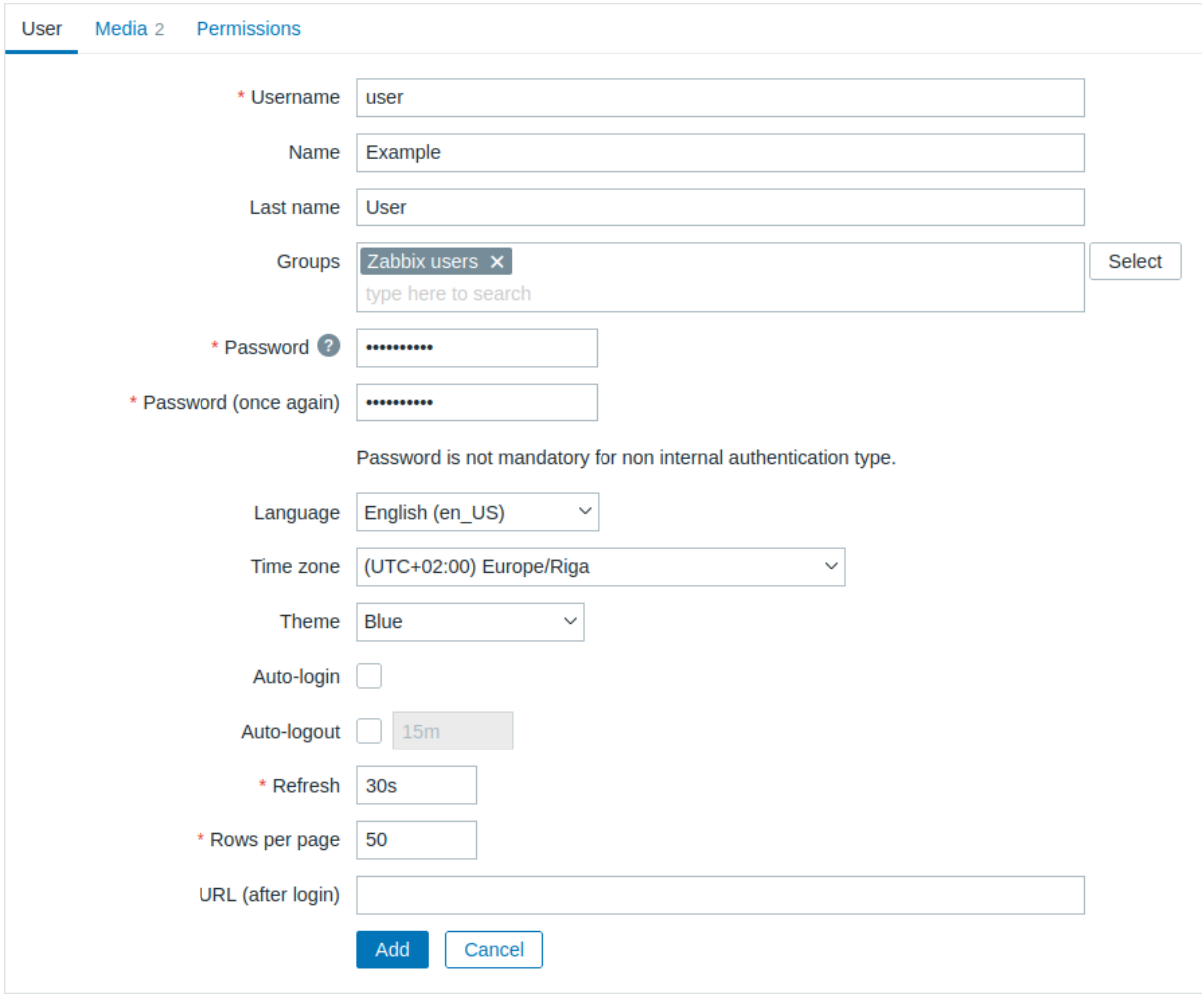
- *Admin* - a Zabbix superuser with full permissions.
- *guest* - a special Zabbix user. The 'guest' user is disabled by default. If you add it to the Guests user group, you may log in with this user and access monitoring pages in Zabbix. (Note that, before Zabbix 6.4.13, automatic guest login was possible.) By default, 'guest' has no permissions on Zabbix objects.

To configure a user:

- Go to *Users* → *Users*.
- Click on *Create user* (or on a user name to edit an existing user).
- Edit user attributes in the form.

General attributes

The *User* tab contains general user attributes:



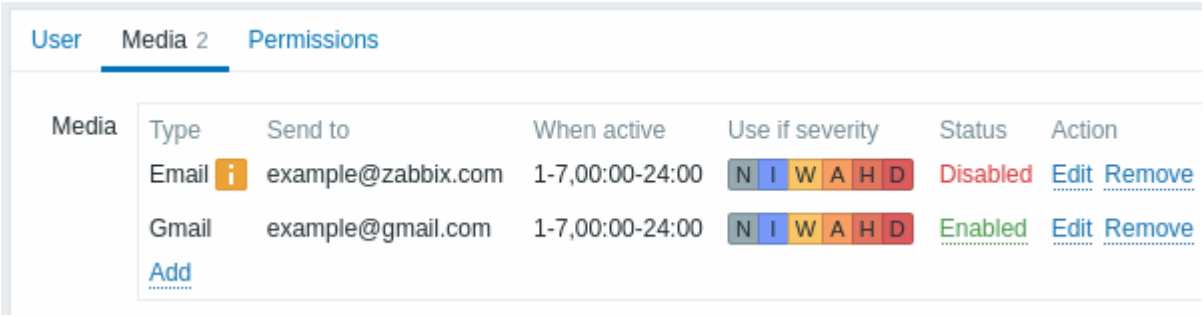
All mandatory input fields are marked with a red asterisk.

Parameter	Description
Username	Unique username, used as the login name.
Name	User first name (optional). If not empty, visible in acknowledgment information and notification recipient information.
Last name	User last name (optional). If not empty, visible in acknowledgment information and notification recipient information.
Groups	Select user groups the user belongs to. Starting with Zabbix 3.4.3 this field is auto-complete so starting to type the name of a user group will offer a dropdown of matching groups. Scroll down to select. Alternatively, click on *Select* to add groups. Click on 'x' to remove the selected. Adherence to user groups determines what host groups and hosts the user will have access to.
Password	Two fields for entering the user password, or a *Change password* button if the user already exists. Clicking on the *Change password* button opens two fields for entering a new password. For the user with the *Super admin role* changing own password, clicking on the *Change password* button opens an additional field for entering the current (old) password. On a successful password change, the user for which the password was changed will be logged out of all active sessions. Note that the password can only be changed for users using Zabbix internal authentication.
Language	Language of the Zabbix frontend. The php gettext extension is required for the translations to work.
Time zone	Select the time zone to override global time zone on user level or select **System default** to use global time zone settings.
Theme	Defines how the frontend looks like: **System default** - use default system settings **Blue** - standard blue theme **Dark** - alternative dark theme **High-contrast light** - light theme with high contrast **High-contrast dark** - dark theme with high contrast
Auto-login	Mark this checkbox to make Zabbix remember the user and log the user in automatically for 30 days. Browser cookies are used for this.

Parameter	Description
Auto-logout	With this checkbox marked the user will be logged out automatically, after the set amount of seconds (minimum 90 seconds, maximum 1 day). Time suffixes are supported, e.g. 90s, 5m, 2h, 1d. Note that this option will not work: * If the "Show warning if Zabbix server is down" global configuration option is enabled and Zabbix frontend is kept open. * When Monitoring menu pages perform background information refreshes. * If logging in with the *Remember me for 30 days* option checked.
Refresh	Set the refresh rate used for graphs, plain text data, etc. Can be set to 0 to disable.
Rows per page	You can determine how many rows per page will be displayed in lists.
URL (after login)	You can make Zabbix transfer the user to a specific URL after successful login, for example, to *Problems* page.

User media

The *Media* tab contains a listing of all media defined for the user. Media are used for sending notifications.



Click on *Add* to assign media to the user.

If the media type has been disabled:

- A yellow info icon is displayed after the name.
- *Disabled* is displayed in the Status column.

See the Media types section for details on configuring user media.

Permissions

The *Permissions* tab contains information on the following elements:

- User role that can only be changed by a *Super admin* user.

Warning:
Since Zabbix 6.4.4, users cannot be created without a user role (except with Zabbix User API). Previously created users which do not have a role may still be edited without assigning a role to them. However, once a role is assigned, it can only be changed, not removed.

 Note that users without a role can log into Zabbix only using LDAP or SAML authentication, provided their LDAP/SAML information matches the user group mappings configured in Zabbix.

- User type (*User*, *Admin*, *Super admin*) that is defined in the user role configuration.
- Host and template groups that the user has access to.
  - *User* and *Admin* type users, by default, do not have access to any groups, templates, and hosts. To grant such access, users must be included in user groups configured with permissions to the relevant entities.
- Access rights to sections and elements of Zabbix frontend, modules, and API methods.
  - Elements with allowed access are displayed in green color, while those with denied access - in light gray color.
- Rights to perform specific actions.
  - Actions that the user is allowed to perform are displayed in green color, while those that are denied - in light gray color.

See the Permissions page for details.

2 Permissions

Overview

Permissions in Zabbix depend on the user type, customized user roles and access to hosts, which is specified based on the user group.

User types

Permissions in Zabbix depend, primarily, on the user type:

- *User* - has limited access rights to menu sections (see below) and no access to any resources by default. Any permissions to host or template groups must be explicitly assigned;
- *Admin* - has incomplete access rights to menu sections (see below). The user has no access to any host groups by default. Any permissions to host or template groups must be explicitly given;
- *Super admin* - has access to all menu sections. The user has a read-write access to all host and template groups. Permissions cannot be revoked by denying access to specific groups.

Menu access

The following table illustrates access to Zabbix menu sections per user type:

Menu section		User	Admin	Super admin
Dashboards		+	+	+
Monitoring		+	+	+
	Problems	+	+	+
	Hosts	+	+	+
	Latest data	+	+	+
	Maps	+	+	+
	Discovery		+	+
Services		+	+	+
	Services	+	+	+
	SLA		+	+
	SLA report	+	+	+
Inventory		+	+	+
	Overview	+	+	+
	Hosts	+	+	+
Reports		+	+	+
	System information			+
	Scheduled reports		+	+
	Availability report	+	+	+
	Triggers top 100	+	+	+
	Audit log			+
	Action log			+
	Notifications		+	+
Data collection			+	+
	Template groups		+	+
	Host groups		+	+
	Templates		+	+
	Hosts		+	+
	Maintenance		+	+
	Event correlation			+
	Discovery		+	+
Alerts			+	+
	Trigger actions		+	+
	Service actions		+	+
	Discovery actions		+	+
	Autoregistration actions		+	+
	Internal actions		+	+
	Media types			+
	Scripts			+
Users				+
	User groups			+
	User roles			+
	Users			+
	API tokens			+
	Authentication			+
Administration				+
	General			+

Menu section		User	Admin	Super admin
	Audit log			+
	Housekeeping			+
	Proxies			+
	Macros			+
	Queue			+

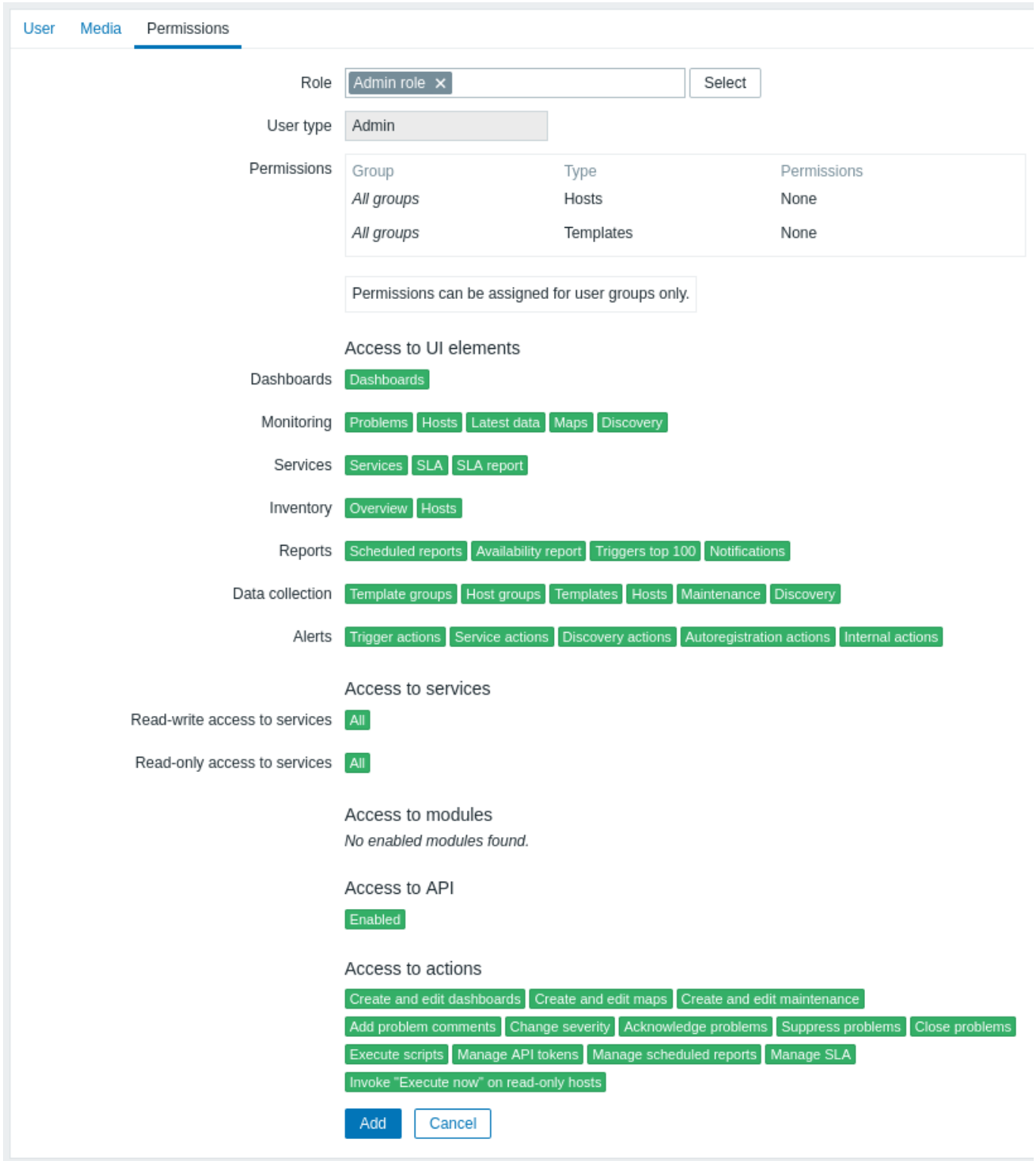
User roles

User roles allow to make custom adjustments to the permissions defined by the user type. While no permissions can be added (that would exceed those of the user type), some permissions can be revoked.

Furthermore, a user role determines access not only to menu sections, but also to services, modules, API methods and various actions in the frontend.

User roles are configured in the *Users* → *User roles* section by Super admin users.

User roles are assigned to users in the user configuration form, *Permissions* tab, by Super admin users.



Access to hosts

Access to any host and template data in Zabbix is granted to user groups on the host/template group level only.

That means that an individual user cannot be directly granted access to a host (or host group). It can only be granted access to a host by being part of a user group that is granted access to the host group that contains the host.

Similarly, a user can only be granted access to a template by being part of a user group that is granted access to the template group that contains the template.

3 User groups

Overview

User groups allow to group users both for organizational purposes and for assigning permissions to data. Permissions to viewing and configuring data of host groups and template groups are assigned to user groups, not individual users.

It may often make sense to separate what information is available for one group of users and what - for another. This can be accomplished by grouping users and then assigning varied permissions to host and template groups.

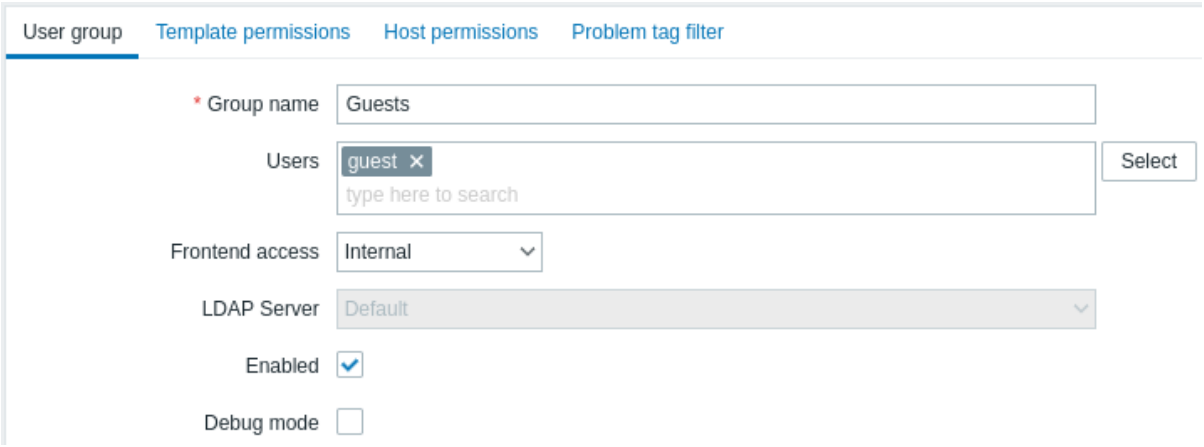
A user can belong to any number of groups.

Configuration

To configure a user group:

- Go to *Users* → *User groups*
- Click on *Create user group* (or on the group name to edit an existing group)
- Edit group attributes in the form

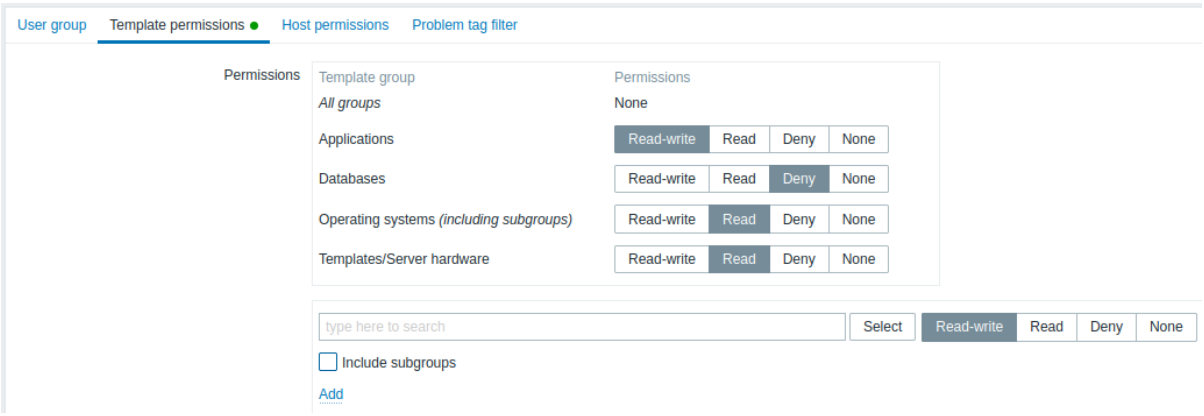
The **User group** tab contains general group attributes:



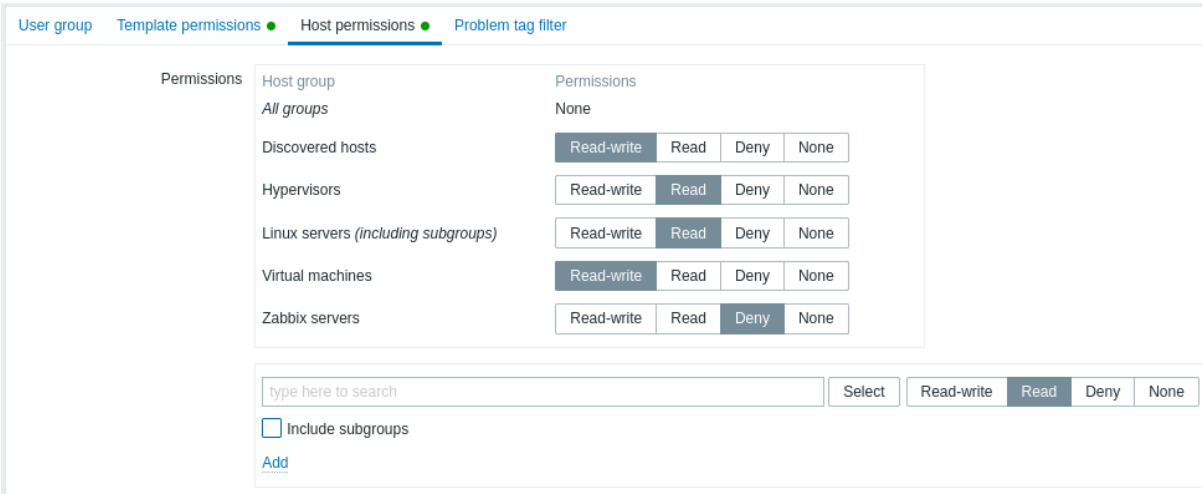
All mandatory input fields are marked with a red asterisk.

Parameter	Description
Group name	Unique group name.
Users	To add users to the group start typing the name of an existing user. When the dropdown with matching user names appears, scroll down to select. Alternatively you may click the *Select* button to select users in a popup.
Frontend access	How the users of the group are authenticated. **System default** - use default authentication method (set globally) **Internal** - use Zabbix internal authentication (even if LDAP authentication is used globally). Ignored if HTTP authentication is the global default. **LDAP** - use LDAP authentication (even if internal authentication is used globally). Ignored if HTTP authentication is the global default. **Disabled** - access to Zabbix frontend is forbidden for this group
LDAP server	Select which LDAP server to use to authenticate the user. This field is enabled only if *Frontend access* is set to LDAP or System default.
Enabled	Status of user group and group members. *Checked* - user group and users are enabled *Unchecked* - user group and users are disabled
Debug mode	Mark this checkbox to activate debug mode for the users.

The **Template permissions** tab allows to specify user group access to template group (and thereby template) data:



The **Host permissions** tab allows to specify user group access to host group (and thereby host) data:



Template permissions and **Host permissions** tabs support the same set of parameters.

Current permissions to groups are displayed in the *Permissions* block.

If current permissions of the group are inherited by all nested groups, this is indicated after the group name (”*including subgroups*”). Note that a *Super admin* user can enforce nested groups to have the same level of permissions as the parent group; this can be done in the host/template group configuration form.

You may change the level of access to a group:

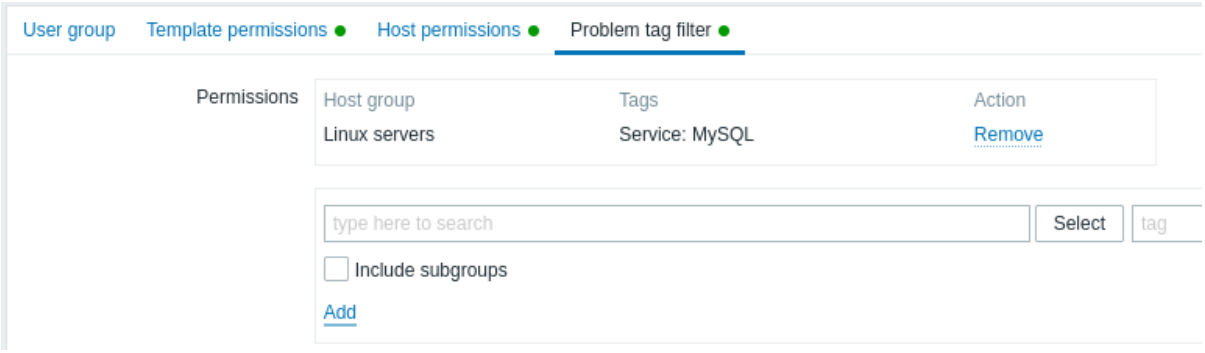
- **Read-write** - read-write access to a group;
- **Read** - read-only access to a group;
- **Deny** - access to a group denied;
- **None** - no permissions are set.

Use the selection field below to select groups and the level of access to them. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. If you wish to see all groups, click on *Select*. If you wish to include nested groups, mark the *Include subgroups* checkbox. Click on Add to add the selected groups to the list of group permissions.

Attention:
Adding a parent group with the *Include subgroups* checkbox marked will override (and remove from the list) previously configured permissions of all related nested groups. Adding a group with *None* as the level of access selected will remove the group from the list if the group is already in the list.

Note:
If a user group grants **Read-write** permissions to a host, and **None** to a template, the user will not be able to edit templated items on the host, and template name will be displayed as *Inaccessible template*.

The **Problem tag filter** tab allows setting tag-based permissions for user groups to see problems filtered by tag name and value:



To select a host group to apply a tag filter for, click *Select* to get the complete list of existing host groups or start to type the name of a host group to get a dropdown of matching groups. Only host groups will be displayed, because problem tag filter cannot be applied to template groups.

To apply tag filters to nested host groups, mark the *Include subgroups* checkbox.

Tag filter allows to separate the access to host group from the possibility to see problems.

For example, if a database administrator needs to see only ”MySQL” database problems, it is required to create a user group for database administrators first, then specify ”Service” tag name and ”MySQL” value.



If ”Service” tag name is specified and value field is left blank, the user group will see all problems with tag name ”Service” for the selected host group. If both tag name and value fields are blank, but a host group is selected, the user group will see all problems for the specified host group.

Make sure tag name and tag value are correctly specified, otherwise, the user group will not see any problems.

Let’s review an example when a user is a member of several user groups selected. Filtering in this case will use OR condition for tags.

User group A			**User group B**			**Visible result for a user (member) of both groups**
Tag filter						
Host group	*Tag name*	*Tag value*	*Host group*	*Tag name*	*Tag value*	
Linux servers	Service	MySQL	Linux servers	Service	Oracle	Service: MySQL or Oracle problems visible
Linux servers	*blank*	*blank*	Linux servers	Service	Oracle	All problems visible
not selected	*blank*	*blank*	Linux servers	Service	Oracle	Service:Oracle problems visible

Attention:
Adding a filter (for example, all tags in a certain host group ”Linux servers”) results in not being able to see the problems of other host groups.

Access from several user groups

A user may belong to any number of user groups. These groups may have different access permissions to hosts or templates.

Therefore, it is important to know what entities an unprivileged user will be able to access as a result. For example, let us consider how access to host **X** (in Hostgroup 1) will be affected in various situations for a user who is in user groups A and B.

- If Group A has only *Read* access to Hostgroup 1, but Group B *Read-write* access to Hostgroup 1, the user will get **Read-write** access to ’X’.

Attention:
“Read-write” permissions have precedence over “Read” permissions.

- In the same scenario as above, if ’X’ is simultaneously also in Hostgroup 2 that is **denied** to Group A or B, access to ’X’ will be **unavailable**, despite a *Read-write* access to Hostgroup 1.
- If Group A has no permissions defined and Group B has a *Read-write* access to Hostgroup 1, the user will get **Read-write** access to ’X’.
- If Group A has *Deny* access to Hostgroup 1 and Group B has a *Read-write* access to Hostgroup 1, the user will get access to ’X’ **denied**.

Other details

- An Admin level user with *Read-write* access to a host will not be able to link/unlink templates, if he has no access to the template group they belong to. With *Read* access to the template group he will be able to link/unlink templates to the host, however, will not see any templates in the template list and will not be able to operate with templates in other places.
- An Admin level user with *Read* access to a host will not see the host in the configuration section host list; however, the host triggers will be accessible in IT service configuration.

- Any non-Super Admin user (including 'guest') can see network maps as long as the map is empty or has only images. When hosts, host groups or triggers are added to the map, permissions are respected.
- Zabbix server will not send notifications to users defined as action operation recipients if access to the concerned host is explicitly "denied".

13 Storage of secrets

Overview Zabbix can be configured to retrieve sensitive information from a secure vault. The following secret management services are supported: HashiCorp Vault KV Secrets Engine - Version 2, CyberArk Vault CV12.

Secrets can be used for retrieving:

- user macro values
- database access credentials

Zabbix provides read-only access to the secrets in a vault, assuming that secrets are managed by someone else.

For information about specific vault provider configuration, see:

- HashiCorp configuration
- CyberArk configuration

Caching of secret values Vault secret macro values are retrieved by Zabbix server on every refresh of configuration data and then stored in the configuration cache. Zabbix proxy receives values of vault secret macros from Zabbix server on each configuration sync and stores them in its own configuration cache.

> **Attention:**
> Encryption must be enabled between Zabbix server and proxy; otherwise a server warning message is logged.

To manually trigger refresh of cached secret values from a vault, use the 'secrets_reload' command-line option.

For Zabbix frontend database credentials caching is disabled by default, but can be enabled by setting the option `$DB['VAULT_CACHE'] = true` in zabbix.conf.php. The credentials will be stored in a local cache using the filesystem temporary file directory. The web server must allow writing in a private temporary folder (for example, for Apache the configuration option `PrivateTmp=True` must be set). To control how often the data cache is refreshed/invalidated, use the ZBX_DATA_CACHE_TTL constant .

TLS configuration To configure TLS for communication between Zabbix components and the vault, add a certificate signed by a certificate authority (CA) to the system-wide default CA store. To use another location, specify the directory in the SSLCALocation Zabbix server/proxy configuration parameter, place the certificate file inside that directory, then run the CLI command:

```
$ c_rehash .
```

1 CyberArk configuration

This section explains how to configure Zabbix to retrieve secrets from CyberArk Vault CV12.

The vault should be installed and configured as described in the official CyberArk documentation.

To learn about configuring TLS in Zabbix, see *Storage of secrets*.

Database credentials

Access to a secret with database credentials is configured for each Zabbix component separately.

Server and proxies

To obtain database credentials from the vault for Zabbix server or proxy, specify the following configuration parameters in the configuration file:

- `Vault` - which vault provider should be used;
- `VaultURL` - vault server HTTP[S] URL;
- `VaultDBPath` - query to the vault secret containing database credentials which will be retrieved by keys "Content" and "UserName";
- `VaultTLSCertFile`, `VaultTLSKeyFile` - SSL certificate and key file names; setting up these options is not mandatory, but highly recommended.

Attention:
Zabbix server also uses the `Vault`, `VaultURL`, `VaultTLSCertFile` and `VaultTLSKeyFile` configuration parameters for vault authentication when processing vault secret macros.

Zabbix server and Zabbix proxy read the vault-related configuration parameters from *zabbix_server.conf* and *zabbix_proxy.conf* files upon startup.

Example

1. In *zabbix_server.conf*, specify the following parameters:

```
Vault=CyberArk
VaultURL=https://127.0.0.1:1858
VaultDBPath=AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_server_database
VaultTLSCertFile=cert.pem
VaultTLSKeyFile=key.pem
```

2. Zabbix will send the following API request to the vault:

```
curl \
--header "Content type: application/json" \
--cert cert.pem \
--key key.pem \
https://127.0.0.1:1858/AIMWebService/api/Accounts?AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbi
```

3. The vault response will contain the keys ”Content” and ”UserName”:

```
{
    "Content": <password>,
    "UserName": <username>,
    "Address": <address>,
    "Database": <Database>,
    "PasswordChangeInProcess":<PasswordChangeInProcess>
}
```

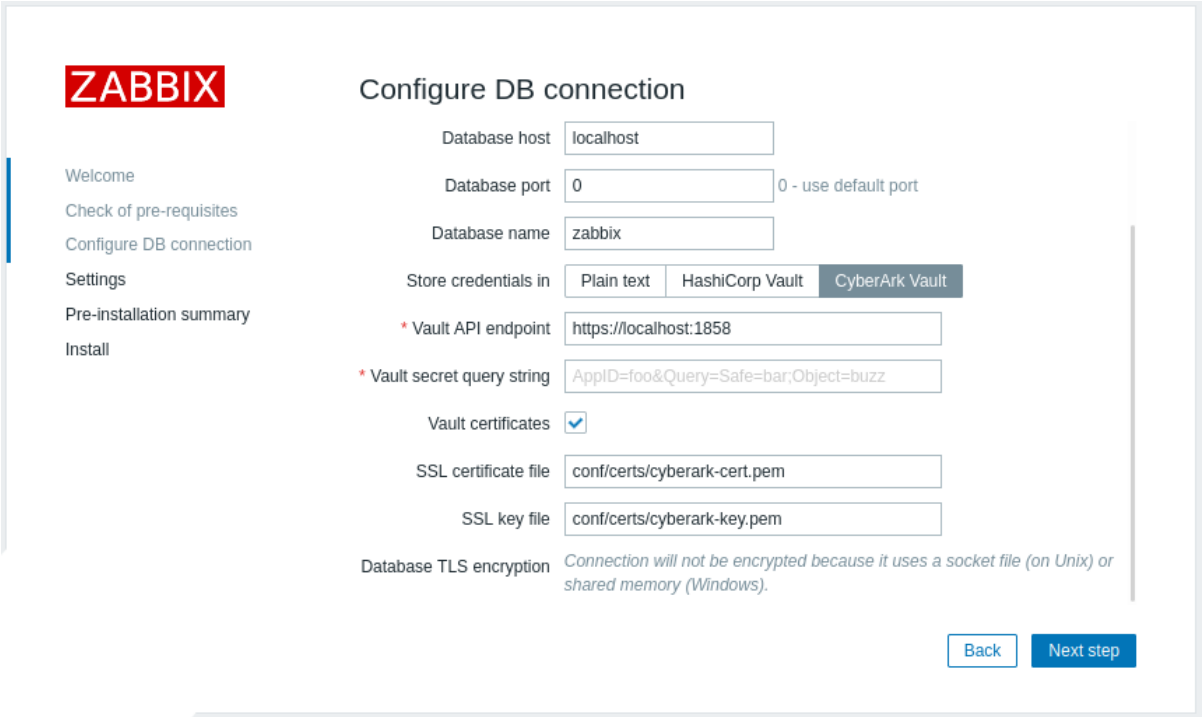
4. As a result, Zabbix will use the following credentials for database authentication:

- Username: <username>
- Password: <password>

Frontend

To obtain database credentials from the vault for Zabbix frontend, specify the following parameters during frontend installation.

1. At the *Configure DB Connection* step, set the *Store credentials in* parameter to ”CyberArk Vault”.



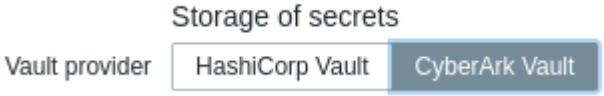
2. Then, fill in the additional parameters:

Parameter	Mandatory	Default value	Description
Vault API endpoint	yes	https://localhost:1858	Specify the URL for connecting to the vault in the format `scheme://host:port`
Vault secret query string	yes		A query, which specifies from where database credentials should be retrieved. Example: `AppID=foo&Query=Safe=bar;Object=buzz`
Vault certificates	no		After marking the checkbox, additional parameters will appear allowing to configure client authentication. While this parameter is optional, it is highly recommended to enable it for communication with the CyberArk Vault.
SSL certificate file	no	conf/certs/cyberark-cert.pem	Path to the SSL certificate file. The file must be in PEM format. If the certificate file also contains the private key, leave the SSL key file parameter empty.
SSL key file	no	conf/certs/cyberark-key.pem	Name of the SSL private key file used for client authentication. The file must be in PEM format.

User macro values

To use CyberArk Vault for storing *Vault secret* user macro values, make sure that:

- Zabbix server is configured to work with CyberArk Vault;
- the *Vault provider* parameter in *Administration → General → Other* is set to "CyberArk Vault".



Note:
Only Zabbix server requires access to *Vault secret* macro values from the vault. Other Zabbix components (proxy, frontend) do not need such access.

The macro value should contain a query (as `query:key`).

See *Vault secret macros* for detailed information on macro value processing by Zabbix.

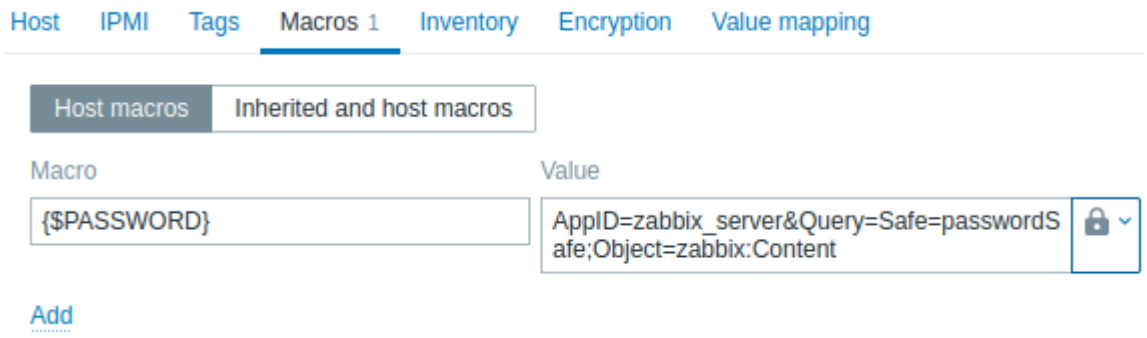
Query syntax

The colon symbol (":") is reserved for separating the query from the key.

If a query itself contains a forward slash or a colon, these symbols should be URL-encoded ("/" is encoded as "%2F", ":" is encoded as "%3A").

Example

1. In Zabbix, add a user macro {$PASSWORD} of type *Vault secret* and with the value `AppID=zabbix_server&Query=Safe=passwordS`



2. Zabbix will send the following API request to the vault:

```
curl \
--header "Content type: application/json" \
--cert cert.pem \
--key key.pem \
https://127.0.0.1:1858/AIMWebService/api/Accounts?AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbi
```

3. The vault response will contain the key "Content":

```
{
    "Content": <password>,
    "UserName": <username>,
    "Address": <address>,
    "Database" :<Database>,
    "PasswordChangeInProcess":<PasswordChangeInProcess>
}
```

4. As a result, Zabbix will resolve the macro {$PASSWORD} to the value - <password>

Update existing configuration

To update an existing configuration for retrieving secrets from a CyberArk Vault:

1. Update the Zabbix server or proxy configuration file parameters as described in the *Database credentials* section.
2. Update the DB connection settings by reconfiguring Zabbix frontend and specifying the required parameters as described in the *Frontend* section. To reconfigure Zabbix frontend, open the frontend setup URL in the browser:

- for Apache: http://<server_ip_or_name>/zabbix/setup.php
- for Nginx: http://<server_ip_or_name>/setup.php

Alternatively, these parameters can be set in the frontend configuration file (*zabbix.conf.php*):

```
$DB['VAULT']                    = 'CyberArk';
$DB['VAULT_URL']                = 'https://127.0.0.1:1858';
$DB['VAULT_DB_PATH']            = 'AppID=foo&Query=Safe=bar;Object=buzz';
$DB['VAULT_TOKEN']              = '';
$DB['VAULT_CERT_FILE']          = 'conf/certs/cyberark-cert.pem';
$DB['VAULT_KEY_FILE']           = 'conf/certs/cyberark-key.pem';
```

3. Configure user macros as described in the *User macro values* section, if necessary.

To update an existing configuration for retrieving secrets from a HashiCorp Vault, see *HashiCorp configuration*.

2 HashiCorp configuration

This section explains how to configure Zabbix to retrieve secrets from HashiCorp Vault KV Secrets Engine - Version 2.

The vault should be deployed and configured as described in the official HashiCorp documentation.

To learn about configuring TLS in Zabbix, see *Storage of secrets*.

Database credentials

Access to a secret with database credentials is configured for each Zabbix component separately.

Server and proxies

To obtain database credentials from the vault for Zabbix server or proxy, specify the following configuration parameters in the configuration file:

- `Vault` - which vault provider should be used;
- `VaultToken` - vault authentication token (see Zabbix server/proxy configuration file for details);
- `VaultURL` - vault server HTTP[S] URL;
- `VaultDBPath` - path to the vault secret containing database credentials; Zabbix server or proxy will retrieve the credentials by keys "password" and "username".

Attention:
Zabbix server also uses the `Vault`, `VaultToken` and `VaultURL` configuration parameters for vault authentication when processing vault secret macros.

Zabbix server and Zabbix proxy read the vault-related configuration parameters from *zabbix_server.conf* and *zabbix_proxy.conf* upon startup. Additionally, Zabbix server and Zabbix proxy will read the `VAULT_TOKEN` environment variable once during startup and will unset it so that it would not be available through forked scripts; it is an error if both `VaultToken` and `VAULT_TOKEN` parameters contain a value.

Example

1. In *zabbix_server.conf*, specify the following parameters:

```
Vault=HashiCorp
VaultToken=hvs.CAESIIG_PILmULFYOsEyWHxkZ2mF2a8VPKNLE8eHqd4autYGGh4KHGh2cy5aeTY0NFNSaUp3ZnpWbDF1RUNjUkNTZEg
VaultURL=https://127.0.0.1:8200
VaultDBPath=secret/zabbix/database
```

2. Run the following CLI commands to create the required secret in the vault:

```
#### Enable "secret/" mount point if not already enabled; note that "kv-v2" must be used.
$ vault secrets enable -path=secret/ kv-v2

#### Put new secrets with keys username and password under mount point "secret/" and path "secret/zabbix/d
$ vault kv put secret/zabbix/database username=zabbix password=<password>

#### Test that secret is successfully added.
$ vault kv get secret/zabbix/database

#### Finally test with Curl; note that "data" need to be manually added after mount point and "/v1" before
$ curl --header "X-Vault-Token: <VaultToken>" https://127.0.0.1:8200/v1/secret/data/zabbix/database
```

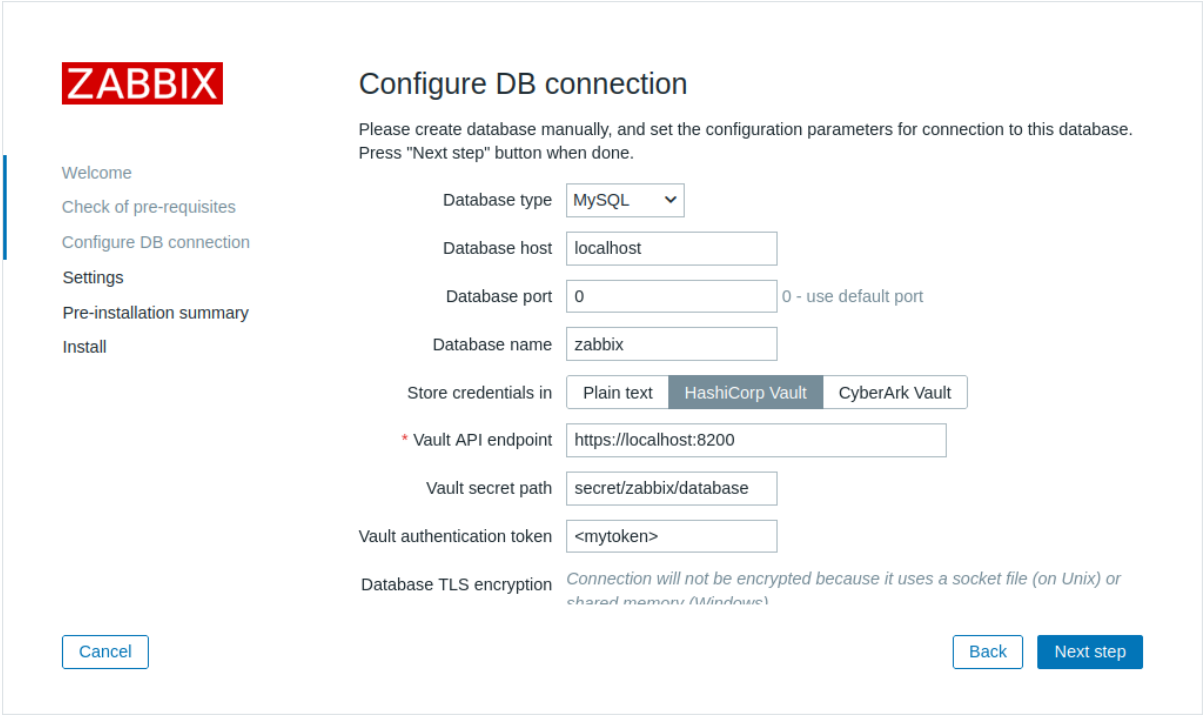
3. As a result, Zabbix server will retrieve the following credentials for database authentication:

- Username: zabbix
- Password: <password>

Frontend

To obtain database credentials from the vault for Zabbix frontend, specify the following parameters during frontend installation.

1. At the *Configure DB Connection* step, set the *Store credentials in* parameter to "HashiCorp Vault".



2. Then, fill in the additional parameters:

Parameter	Mandatory	Default value	Description
Vault API endpoint	yes	https://localhost:8200	Specify the URL for connecting to the vault in the format `scheme://host:port`
Vault secret path	no		A path to the secret from where credentials for the database shall be retrieved by the keys ”password” and ”username”. Example: `secret/zabbix/database`
Vault authentication token	no		Provide an authentication token for read-only access to the secret path. See HashiCorp documentation for information about creating tokens and vault policies.

User macro values

To use HashiCorp Vault for storing *Vault secret* user macro values, make sure that:

- Zabbix server is configured to work with HashiCorp Vault;
- the *Vault provider* parameter in *Administration → General → Other* is set to ”HashiCorp Vault” (default).



Note:
Only Zabbix server requires access to *Vault secret* macro values from the vault. Other Zabbix components (proxy, frontend) do not need such access.

The macro value should contain a reference path (as `path:key`, for example, `secret/zabbix:password`). The authentication token specified during Zabbix server configuration (by the `VaultToken` parameter) must provide read-only access to this path.

See *Vault secret macros* for detailed information on macro value processing by Zabbix.

Path syntax

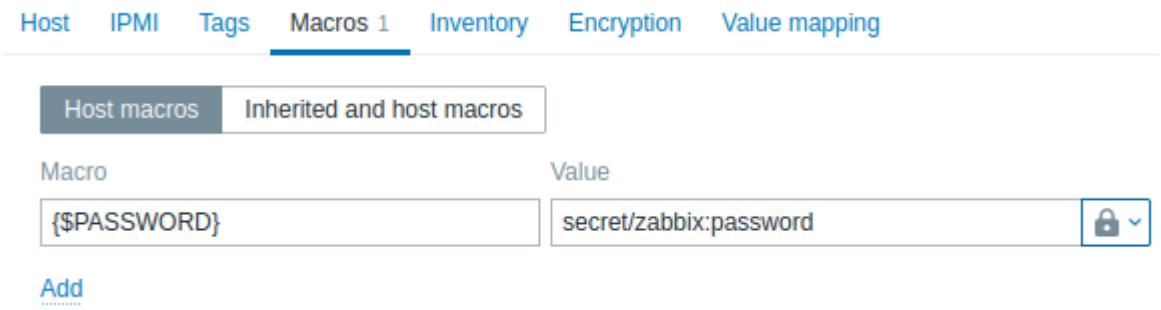
The symbols forward slash (”/”) and colon (”:”) are reserved.

A forward slash can only be used to separate a mount point from a path (e.g., *secret/zabbix* where the mount point is ”secret” and the path is ”zabbix”). In the case of Vault macros, a colon can only be used to separate a path/query from a key.

It is possible to URL-encode the forward slash and colon symbols if there is a need to create a mount point with the name that is separated by a forward slash (e.g., *foo/bar/zabbix*, where the mount point is "foo/bar" and the path is "zabbix", can be encoded as "foo%2Fbar/zabbix") and if a mount point name or path need to contain a colon.

Example

1. In Zabbix, add a user macro {$PASSWORD} of type "Vault secret" and with the value `secret/zabbix:password`



2. Run the following CLI commands to create required secret in the vault:

```
#### Enable "secret/" mount point if not already enabled; note that "kv-v2" must be used.
vault secrets enable -path=secret/ kv-v2

#### Put new secret with key password under mount point "secret/" and path "secret/zabbix".
vault kv put secret/zabbix password=<password>

#### Test that secret is successfully added.
vault kv get secret/zabbix

#### Finally test with Curl; note that "data" need to be manually added after mount point and "/v1" before
curl --header "X-Vault-Token: <VaultToken>" https://127.0.0.1:8200/v1/secret/data/zabbix
```

3. As a result, Zabbix will resolve the macro {$PASSWORD} to the value: <password>

Update existing configuration

To update an existing configuration for retrieving secrets from a HashiCorp Vault:

1. Update the Zabbix server or proxy configuration file parameters as described in the *Database credentials* section.
2. Update the DB connection settings by reconfiguring Zabbix frontend and specifying the required parameters as described in the *Frontend* section. To reconfigure Zabbix frontend, open the frontend setup URL in the browser:

- for Apache: http://<server_ip_or_name>/zabbix/setup.php
- for Nginx: http://<server_ip_or_name>/setup.php

Alternatively, these parameters can be set in the frontend configuration file (*zabbix.conf.php*):

```
$DB['VAULT']                    = 'HashiCorp';
$DB['VAULT_URL']                = 'https://localhost:8200';
$DB['VAULT_DB_PATH']            = 'secret/zabbix/database';
$DB['VAULT_TOKEN']              = '<mytoken>';
$DB['VAULT_CERT_FILE']          = '';
$DB['VAULT_KEY_FILE']           = '';
```

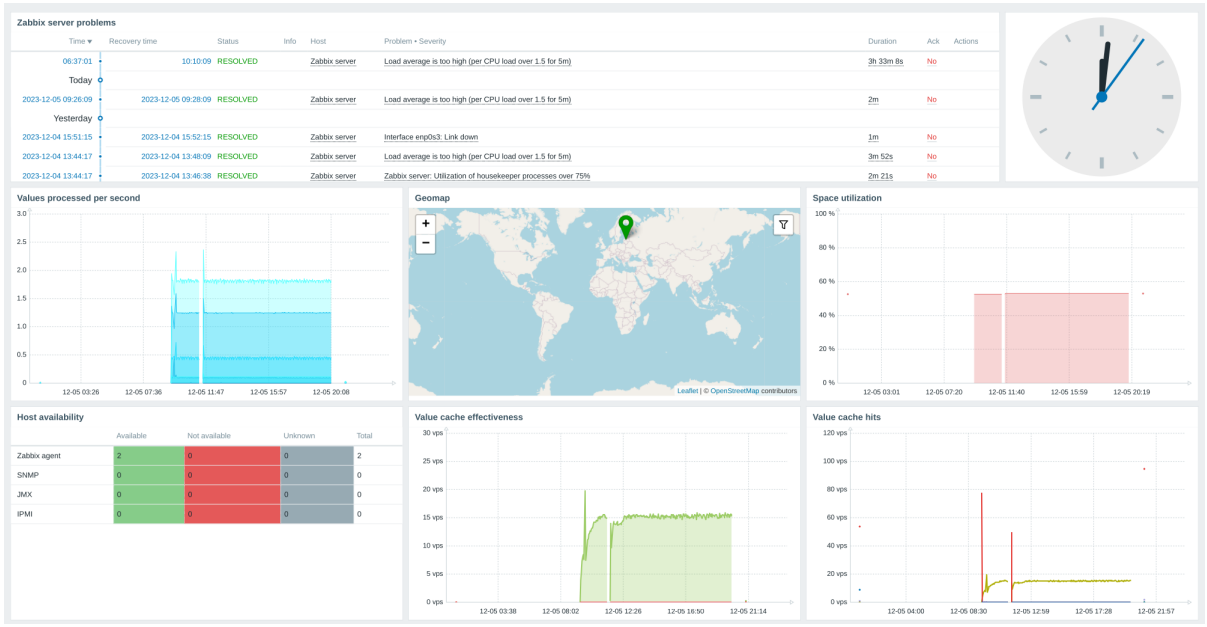
3. Configure user macros as described in the *User macro values* section, if necessary.

To update an existing configuration for retrieving secrets from a CyberArk Vault, see *CyberArk configuration*.

14 Scheduled reports

Overview

With the *Scheduled reports* feature, you can set up a PDF version of a given dashboard to be sent to specified recipients at recurring intervals.



Attention:
Currently the support of scheduled reports is experimental.

Note:
For multi-page dashboards, only the first page is included in the PDF report.

Pre-requisites:

- Zabbix web service must be installed and configured correctly to enable scheduled report generation - see Setting up scheduled reports for instructions.
- A user must have a user role of type *Admin* or *Super admin* with the following permissions:
  - *Scheduled reports* in the *Access to UI elements* block (to view reports)
  - *Manage scheduled reports* in the *Access to actions* block (to create/edit reports)

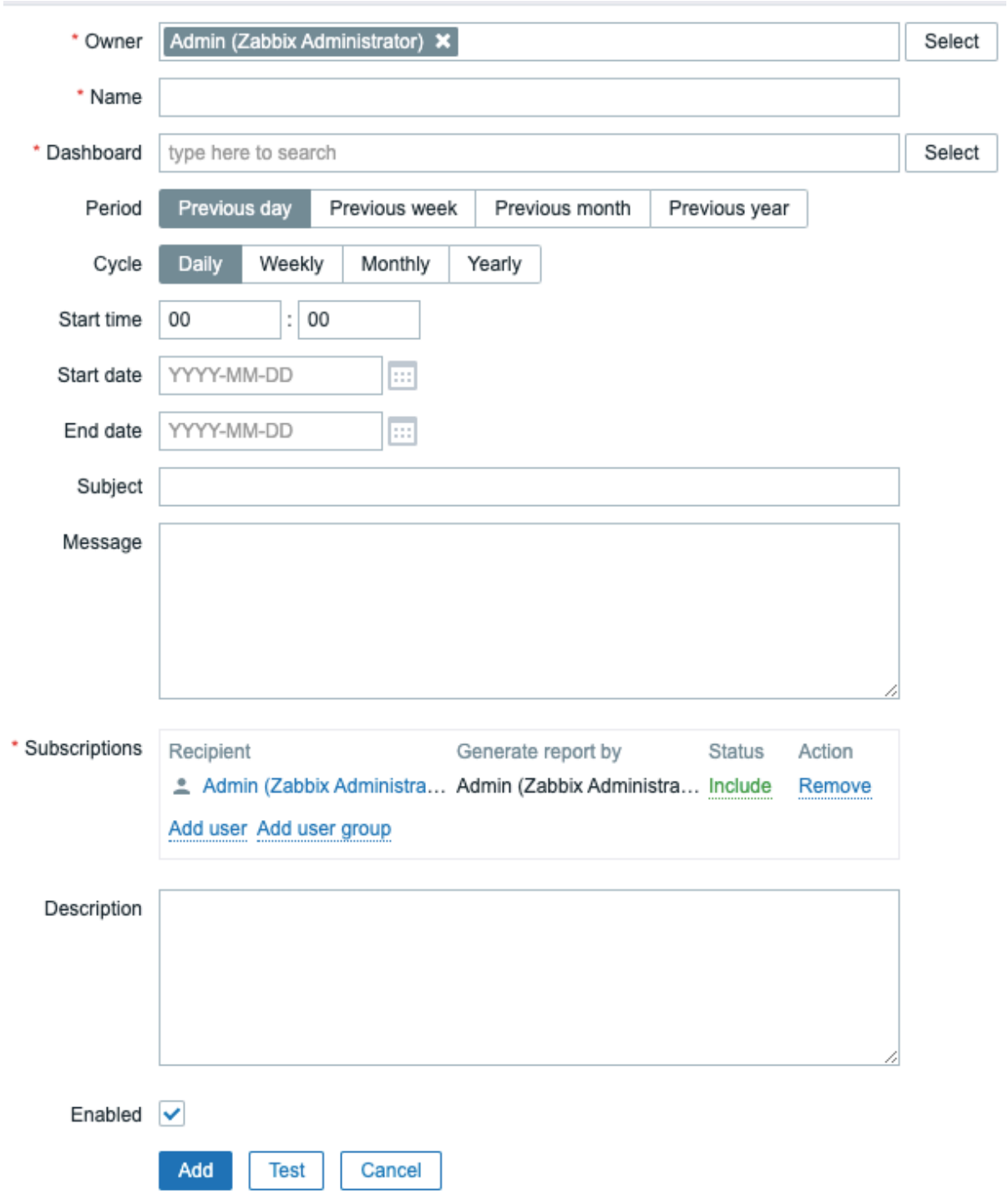
To create a scheduled report in Zabbix frontend, do the following:

- Go to: *Reports* → *Scheduled reports*
- Click *Create report* in the upper right corner of the screen
- Enter parameters of the report in the form

You can also create a report by opening an existing one, clicking the *Clone* button, and then saving it under a different name.

Configuration

The *Scheduled reports* tab contains general report attributes.



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Owner	User that creates a report. *Super admin* level users are allowed to change the owner. For *Admin* level users, this field is read-only.
Name	Name of the report; must be unique.
Dashboard	Dashboard on which the report is based; only one dashboard can be selected at a time. To select a dashboard, start typing the name - a list of matching dashboards will appear; scroll down to select. Alternatively, you may click *Select* next to the field and select a dashboard from the displayed list. If a dashboard contains multiple pages, only the first page will be sent as a report.
Period	Period for which the report will be prepared. Select the previous day, week, month, or year.
Cycle	Report generation frequency. The reports can be sent daily, weekly, monthly, or yearly. ”Weekly” mode allows to select days of the week when the report will be sent.
Start time	Time of day in the format hh:mm when the report will be prepared.
Repeat on	Days of the week when the report will be sent. This field is available only if *Cycle* is set to ”Weekly”.

Parameter	Description
Start date	Date when regular report generation should be started.
End date	Date when regular report generation should be stopped.
Subject	Subject of the report email. Supports {TIME} macro.
Message	Body of the report email. Supports {TIME} macro.
Subscriptions	List of report recipients. By default, includes only the report owner. Any Zabbix user with configured email media may be specified as a report recipient. Click *Add user* or *Add user group* to add more recipients. Click the username to edit settings: *Generate report by* - whether the report data should be generated based on the dashboard permissions of the current user or the recipient. *Status* - select ”Include” to send the report to the user or ”Exclude” to prevent sending the report to this user. At least one user must have the ”Include” status. The ”Exclude” status can be used to exclude specific users from a user group that is included. Note that users with insufficient permissions (that is, users with a role based on the *Admin* user type who are not members of the same user group as the recipient or report owner) will see ”Inaccessible user” or ”Inaccessible user group” instead of the actual names in the fields *Recipient* and *Generate report by*; the fields *Status* and *Action* will be displayed as read-only.
Enabled	Report status. Clearing this checkbox will disable the report.
Description	An optional description of the report. This description is for internal use and will not be sent to report recipients.

Form buttons

Buttons at the bottom of the form allow to perform several operations.

Add	Add a report. This button is only available for new reports.
Update	Update the properties of a report.
Clone	Create another report based on the properties of the current report.
Test	Test if report configuration is correct by sending a report to the current user.
Delete	Delete the report.
Cancel	Cancel the editing of report properties.

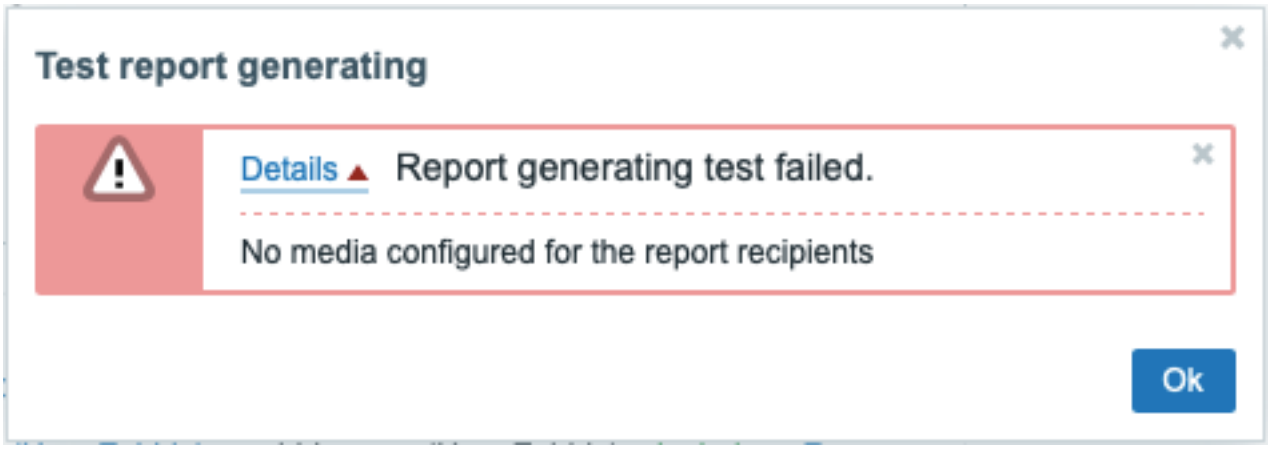
Testing

To test a report, click the *Test* button at the bottom of the report configuration form.

Note:
The *Test* button is not available if the report configuration form has been opened from the dashboard action menu.

If the configuration is correct, the test report is sent immediately to the current user. For test reports, subscribers and *Generate report by* user settings are ignored.

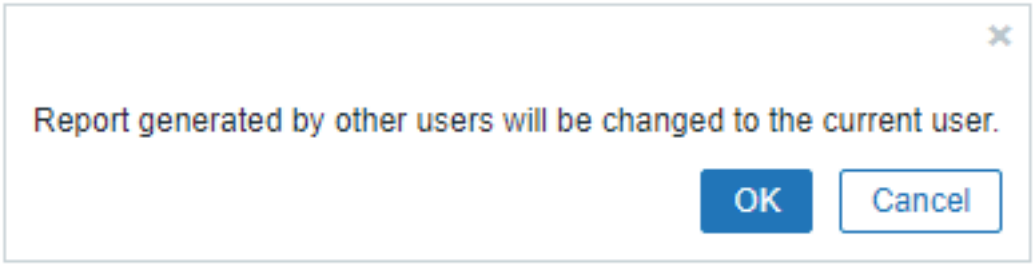
If the configuration is incorrect, an error message is displayed describing the possible cause.



Updating a report

To update an existing report, click the report name, make the required configuration changes, and then click the *Update* button.

If an existing report is updated by another user and this user changes the Dashboard, upon clicking the *Update* button, a warning message ”Report generated by other users will be changed to the current user” will be displayed.



Clicking *OK* at this step will lead to the following changes:

- *Generate report by* settings will be updated to display the user who edited the report last (unless *Generate report by* is set to the recipient).
- Users that have been displayed as ”Inaccessible user” or ”Inaccessible user group” will be deleted from the list of report subscribers.

Clicking *Cancel* will close the configuration form and cancel the report update.

Cloning a report

To quickly clone an existing report, click the *Clone* button at the bottom of an existing report configuration form. When cloning a report created by another user, the current user becomes the owner of the new report.

Report settings will be copied to the new report configuration form with respect to user permissions:

- If the user who clones a report has no permissions to a dashboard, the *Dashboard* field will be cleared.
- If the user who clones a report has no permissions to some users or user groups in the *Subscriptions* list, inaccessible recipients will not be cloned.
- *Generate report by* settings will be updated to display the current user (unless *Generate report by* is set to the recipient).

Change the required settings and the report name, then click *Add*.

15 Data export

Overview

Zabbix supports data export in real-time in two ways:

- export to files
- streaming to external systems

The following entities can be exported:

- trigger events
- item values
- trends (export to files only)

1 Export to files

Overview

It is possible to configure real-time exporting of trigger events, item values and trends in a newline-delimited JSON format.

Exporting is done into files, where each line of the export file is a JSON object. Value mappings are not applied.

In case of errors (data cannot be written to the export file or the export file cannot be renamed or a new one cannot be created after renaming it), the data item is dropped and never written to the export file. It is written only in the Zabbix database. Writing data to the export file is resumed when the writing problem is resolved.

For precise details on what information is exported, see the export protocol page.

Note that host/item can have no metadata (host groups, host name, item name) if the host/item was removed after the data was received, but before server exported data.

Configuration

Real-time export of trigger events, item values and trends is configured by specifying a directory for the export files - see the ExportDir parameter in server configuration.

Two other parameters are available:

- `ExportFileSize` may be used to set the maximum allowed size of an individual export file. When a process needs to write to a file it checks the size of the file first. If it exceeds the configured size limit, the file is renamed by appending .old to its name and a new file with the original name is created.

Attention:
A file will be created per each process that will write data (i.e. approximately 4-30 files). As the default size per export file is 1G, keeping large export files may drain the disk space fast.

- `ExportType` allows to specify which entity types (events, history, trends) will be exported.

2 Streaming to external systems

Overview

It is possible to stream item values and events from Zabbix to external systems over HTTP (see protocol details).

Warning:
This feature currently has experimental status.

The tag filter can be used to stream subsets of item values or events.

Two Zabbix server processes are responsible for data streaming: `connector manager` and `connector worker`. A Zabbix internal item `zabbix[connector_queue]` allows to monitor the count of values enqueued in the connector queue.

Configuration

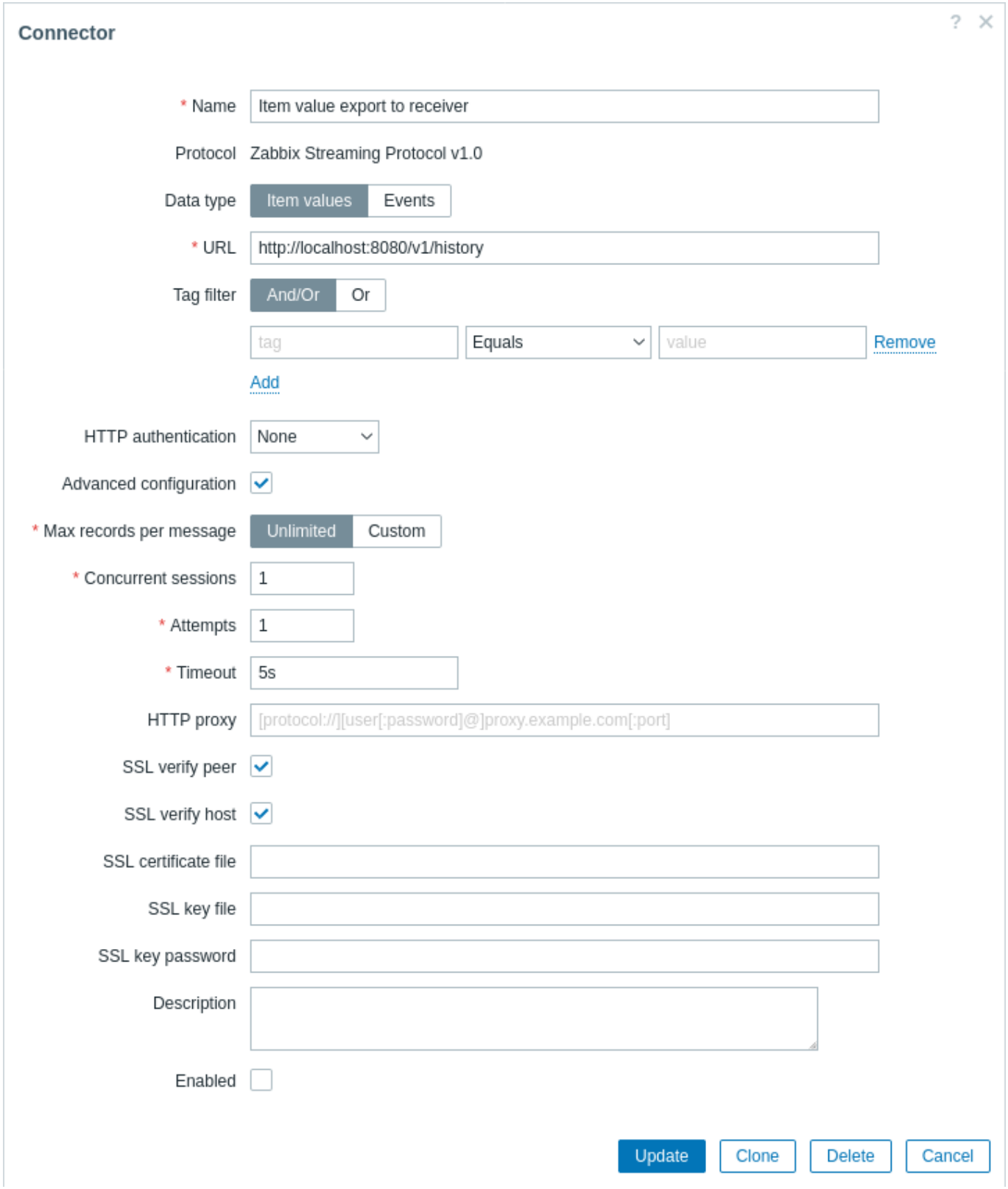
The following steps are required to configure data streaming to an external system:

1. Have a remote system set up for receiving data from Zabbix.

See the documentation of a simple receiver. The receiver currently logs the received information in events.ndjson and history.ndjson files.

2. Set the required number of connector workers in Zabbix by adjusting the `StartConnectors` parameter in `zabbix_server.conf`. The number of connector workers should match (or exceed if concurrent sessions are more than 1) the configured connector count in Zabbix frontend. Then, restart Zabbix server.

3. Configure a new connector in Zabbix frontend (*Administration* → *General* → *Connectors*) and reload the server cache with the `zabbix_server -R config_cache_reload` command.



Mandatory fields are marked by an asterisk.

Parameter	Description
Name	Enter the connector name.
Data type	Select the data type: *Item values* or *Events*.
URL	Enter the receiver URL. User macros are supported.

Parameter	Description
Tag filter	Export only values or events matching the tag filter. If not set, then export everything. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met
HTTP authentication	Select the authentication option: **None** - no authentication used; **Basic** - basic authentication is used; **NTLM** - NTLM (Windows NT LAN Manager) authentication is used; **Kerberos** - Kerberos authentication is used; **Digest** - Digest authentication is used; **Bearer** - Bearer authentication is used.
Username	Enter the user name. This field is available if *HTTP authentication* is set to Basic, NTLM, Kerberos, or Digest. User macros are supported.
Password	Enter the user password. This field is available if *HTTP authentication* is set to Basic, NTLM, Kerberos, or Digest. User macros are supported.
Bearer token	Enter the Bearer token. This field is available and required if *HTTP authentication* is set to Bearer. User macros are supported.
Advanced configuration	Mark this checkbox to display advanced configuration options.
Max records per message	Specify the maximum number of values or events that can be sent within one message.
Concurrent sessions	Select the number of sender processes to run for this connector. Up to 100 sessions can be specified; the default value is '1'.
Attempts	Number of attempts for trying to send the data. Up to 5 attempts can be specified; the default value is '1'.
Timeout	Specify the message timeout (1-60 seconds, default 5 seconds). Time suffixes are supported, e.g. 30s, 1m. User macros are supported.
HTTP proxy	You can specify an HTTP proxy to use, using the format `[protocol://][username[:password]@]proxy.example.com[:port]`. The optional `protocol://` prefix may be used to specify alternative proxy protocols (the protocol prefix support was added in cURL 7.21.7). With no protocol specified, the proxy will be treated as an HTTP proxy. By default, 1080 port will be used. If specified, the proxy will overwrite proxy related environment variables like http_proxy, HTTPS_PROXY. If not specified, the proxy will not overwrite proxy-related environment variables. The entered value is passed on "as is", no sanity checking takes place. You may also enter a SOCKS proxy address. If you specify the wrong protocol, the connection will fail and the item will become unsupported. *Note* that only simple authentication is supported with HTTP proxy. User macros are supported.
SSL verify peer	Mark the checkbox to verify the SSL certificate of the web server. The server certificate will be automatically taken from system-wide certificate authority (CA) location. You can override the location of CA files using Zabbix server or proxy configuration parameter SSLCALocation.
SSL verify host	Mark the checkbox to verify that the *Common Name* field or the *Subject Alternate Name* field of the web server certificate matches. This sets the CURLOPT_SSL_VERIFYHOST cURL option.

Parameter	Description
SSL certificate file	Name of the SSL certificate file used for client authentication. The certificate file must be in PEM[1] format. If the certificate file contains also the private key, leave the *SSL key file* field empty. If the key is encrypted, specify the password in *SSL key password* field. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLCertLocation. User macros are supported.
SSL key file	Name of the SSL private key file used for client authentication. The private key file must be in PEM[1] format. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLKeyLocation. User macros are supported.
SSL key password	SSL private key file password. User macros are supported.
Description	Enter the connector description.
Enabled	Mark the checkbox to enable the connector.

Protocol

Communication between the server and the receiver is done over HTTP using REST API, NDJSON, "Content-Type: application/x-ndjson".

For more details, see newline-delimited JSON export protocol.

Server request

Example of sending item values:

```
POST /v1/history HTTP/1.1
Host: localhost:8080
Accept: */*
Accept-Encoding: deflate, gzip, br, zstd
Content-Length: 628
Content-Type: application/x-ndjson

{"host":{"host":"Zabbix server","name":"Zabbix server"},"groups":["Zabbix servers"],"item_tags":[{"tag":"f
{"host":{"host":"Zabbix server","name":"Zabbix server"},"groups":["Zabbix servers"],"item_tags":[{"tag":"f
{"host":{"host":"Zabbix server","name":"Zabbix server"},"groups":["Zabbix servers"],"item_tags":[{"tag":"b
```

Example of sending events:

```
POST /v1/events HTTP/1.1
Host: localhost:8080
Accept: */*
Accept-Encoding: deflate, gzip, br, zstd
Content-Length: 333
Content-Type: application/x-ndjson

{"clock":1673454303,"ns":800155804,"value":1,"eventid":5,"name":"trigger for foo being 0","severity":0,"ho
{"clock":1673454303,"ns":832290669,"value":0,"eventid":6,"p_eventid":5}
```

Receiver response

The response consists of the HTTP response status code and the JSON payload. The HTTP response status code must be "200" for requests that were handled successfully, other for failed requests.

Example of success:

```
localhost:8080/v1/history": HTTP/1.1 200 OK
Date: Wed, 11 Jan 2023 16:40:30 GMT
Content-Length: 0
```

Example with errors:

```
localhost:8080/v1/history": HTTP/1.1 422 Unprocessable Entity
Content-Type: application/json
Date: Wed, 11 Jan 2023 17:07:36 GMT
Content-Length: 55

{"error":"invalid character '{' after top-level value"}
```

8 Service monitoring

Overview Service monitoring is a business-level monitoring that can be used to get an overview of the entire IT infrastructure service tree, identify weak places of the infrastructure, calculate SLA of various IT services, and check out other information at a higher level. Service monitoring focuses on the overall availability of a service instead of low-level details, such as the lack of disk space, high processor load, etc. Since Zabbix 6.0, service monitoring also provides functionality to find the root cause of a problem if a service is not performing as expected.

Service monitoring allows to create a hierarchy representation of monitored data.

A very simple service structure may look like:

```
Service
|
|-Workstations
| |
| |-Workstation1
| |
| |-Workstation2
|
|-Servers
```

Each node of the structure has attribute status. The status is calculated and propagated to upper levels according to the selected algorithm. The status of individual nodes is affected by the status of the mapped problems. Problem mapping is accomplished with tagging.

Zabbix can send notifications or automatically execute a script on the Zabbix server in case service status change is detected. It is possible to define flexible rules whether a parent service should go into a 'Problem state' based on the statuses of child services. Services problem data can then be used to calculate SLA and send SLA reports based on the flexible set of conditions.

Service monitoring is configured in the Services menu, which consists of the following sections:

- Services

Services section allows to build a hierarchy of your monitored infrastructure by adding parent services, and then - child services to the parent services.

In addition to configuring service tree, this section provides an overview of the whole infrastructure and allows to quickly identify the problems that led to a service status change.

- SLA

In this section you can define service level agreements and set service level objectives for specific services.

- SLA report

In this section you can view SLA reports.

Service actions

You may also configure service actions.

Service actions are optional and allow to:

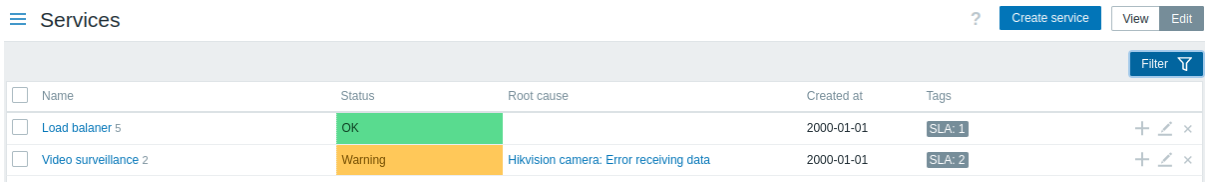
- send a notification that a service is down
- execute a remote command on a Zabbix server upon a service status change
- send a recovery notification when a service is up again.

See also:

- SLA monitoring configuration example
- Notes about upgrading services from Zabbix versions below 6.0

1 Service tree

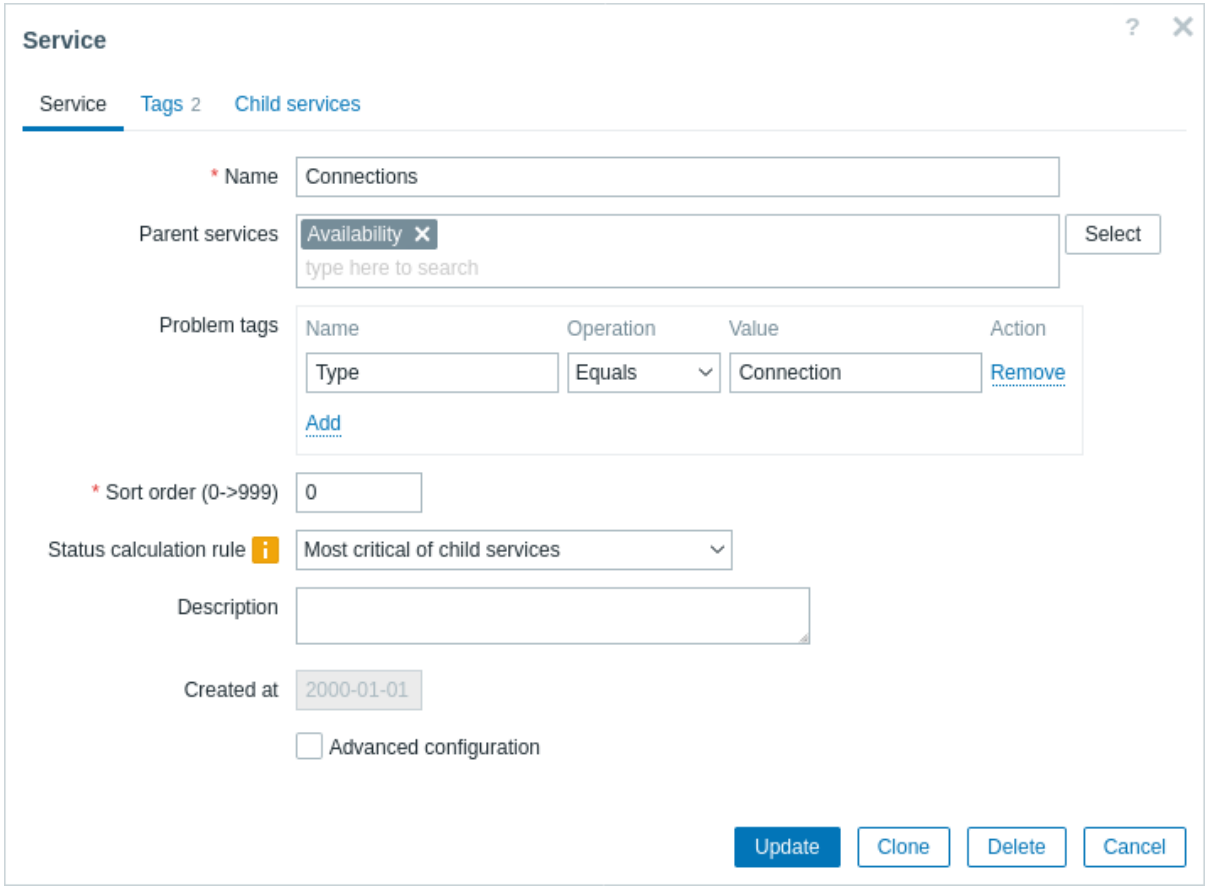
Service tree is configured in the *Services->Services* menu section. In the upper right corner, switch from View to the Edit mode.



To configure a new service, click on the *Create service* button in the top right-hand corner.

To quickly add a child service, you can alternatively press a plus icon next to the parent service. This will open the same service configuration form, but the Parent services parameter will be pre-filled.

Service configuration In the **Service** tab, specify required service parameters:

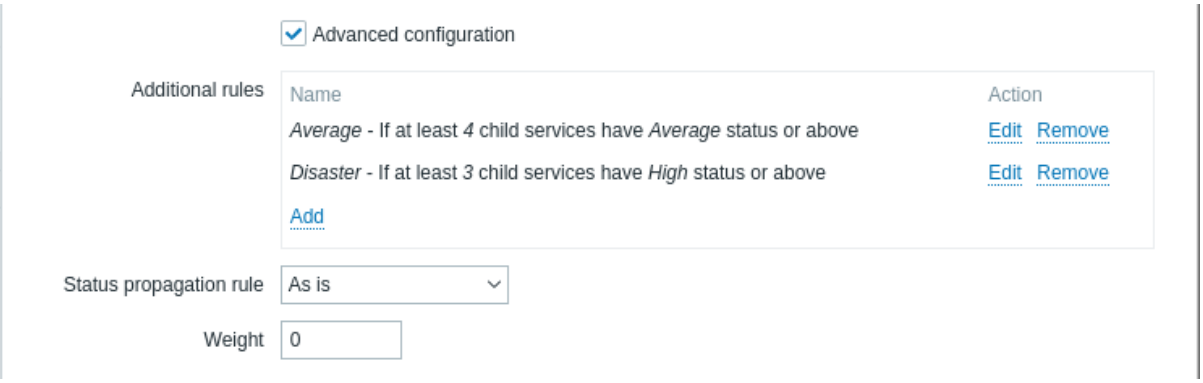


All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Service name.
Parent services	Parent services the service belongs to. Leave this field empty if you are adding the service of highest level. One service may have multiple parent services. In this case, it will be displayed in the service tree under each of the parent services.
Problem tags	Specify tags to map problem data to the service: **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) Tag name matching is always case-sensitive.
Sort order	Sort order for display, lowest comes first.
Status calculation rule	Rule for calculating service status: **Most critical if all children have problems** - the most critical problem in the child services is used to color the service status, if all children have problems **Most critical of child services** - the most critical problem in the child services is used to color the service status **Set status to OK** - do not calculate service status Mark the *Advanced configuration* checkbox below to configure additional status calculation rules.
Description	Service description.

Parameter	Description
Advanced configuration	Mark the checkbox to access advanced configuration options.

Advanced configuration



Parameter	Description
Additional rules	Click on *Add* to define additional status calculation rules.
Set status to	Set service status to either *OK* (default), *Not classified*, *Information*, *Warning*, *Average*, *High* or *Disaster* in case of a condition match.
Condition	Select the condition for direct child services: **if at least (N) child services have (Status) status or above** **if at least (N%) of child services have (Status) status or above** **if less than (N) child services have (Status) status or below** **if less than (N%) of child services have (Status) status or below** **if weight of child services with (Status) status or above is at least (W)** **if weight of child services with (Status) status or above is at least (N%)** **if weight of child services with (Status) status or below is less than (W)** **if weight of child services with (Status) status or below is less than (N%)** If several conditions are specified and the situation matches more than one condition, the highest severity will be set.
N (W)	Set the value of N or W (1-100000), or N% (1-100) in the condition.
Status	Select the value of *Status* in the condition: *OK* (default), *Not classified*, *Information*, *Warning*, *Average*, *High* or *Disaster*.
Status propagation rule	Rule for propagating the service status to the parent service: **As is** - the status is propagated without change **Increase by** - you may increase the propagated status by 1 to 5 severities **Decrease by** - you may decrease the propagated status by 1 to 5 severities **Ignore this service** - the status is not propagated to the parent service at all **Fixed status** - the status is propagated statically, i.e. as always the same
Weight	Weight of the service (integer in the range from 0 (default) to 1000000).

Note:
Additional status calculation rules can only be used to increase severity level over the level calculated according to the main *Status calculation rule* parameter. If according to additional rules the status should be Warning, but according to the *Status calculation rule* the status is Disaster - the service will have status Disaster.

The **Tags** tab contains service-level tags. Service-level tags are used to identify a service. Tags of this type are not used to map problems to the service (for that, use *Problem tags* from the first tab).

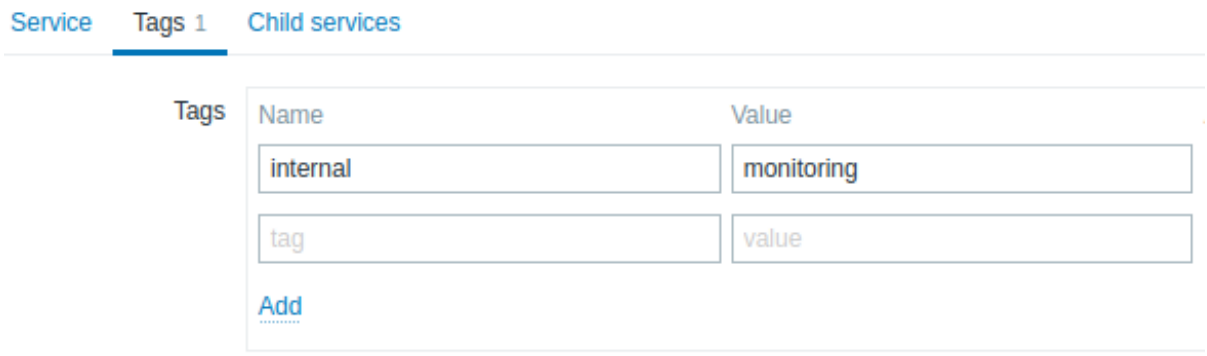
The **Child services** tab allows to specify dependant services. Click on *Add* to add a service from the list of existing services. If you want to add a new child service, save this service first, then click on a plus icon next to the service that you have just created.

Tags There are two different types of tags in services:

- Service tags
- Problem tags

Service tags

Service tags are used to match services with service actions and SLAs. These tags are specified at the *Tags* service configuration tab. For mapping SLAs, *OR* logic is used: a service will be mapped to an SLA if it has at least one matching tag. In service actions, mapping rules are configurable and can use either *AND*, *OR*, or *AND/OR* logic.

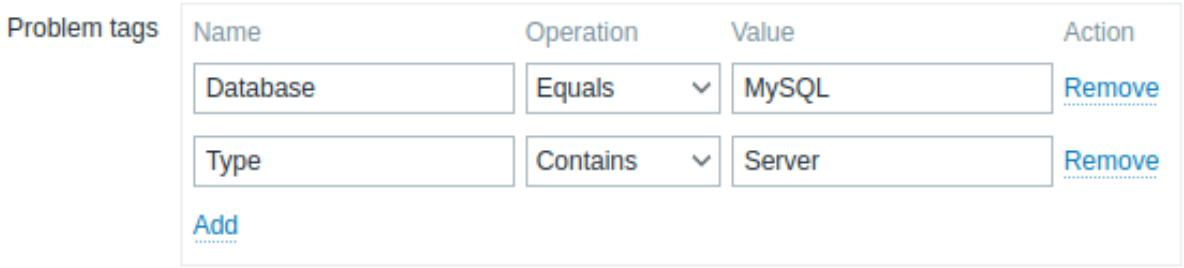


Problem tags

Problem tags are used to match problems and services. These tags are specified at the primary service configuration tab.

Only child services of the lowest hierarchy level may have problem tags defined and be directly correlated to problems. If problem tags match, the service status will change to the same status as the problem has. In case of several problems, a service will have the status of the most severe one. Status of a parent service is then calculated based on child services statuses according to Status calculation rules.

If several tags are specified, *AND* logic is used: a problem must have all tags specified in the service configuration to be mapped to the service.

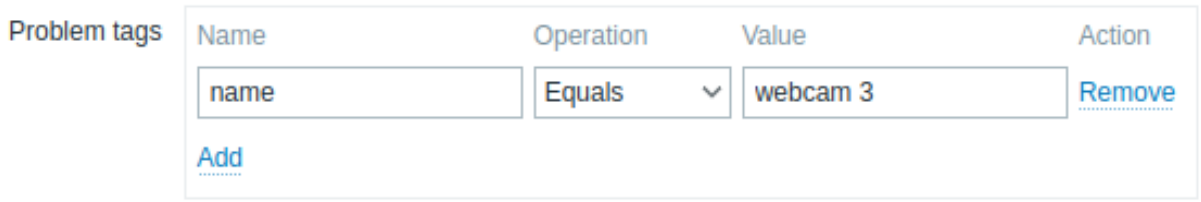


Note:
A problem in Zabbix inherits tags from the whole chain of templates, hosts, items, web scenarios, and triggers. Any of these tags can be used for matching problems to services.

Example:

Problem *Web camera 3 is down* has tags `type:video surveillance`, `floor:1st` and `name:webcam 3` and status *Warning*

The service **Web camera 3** has the only problem tag specified: `name:webcam 3`



Service status will change from *OK* to *Warning* when this problem is detected.

If the service **Web camera 3** had problem tags `name:webcam 3` and `floor:2nd`, its status would not be changed, when the problem is detected, because the conditions are only partially met.

Modifying existing services

Note:
The buttons described below are visible only when *Services* section is in the Edit mode.

To edit an existing service, press the pencil icon next to the service.

To clone an existing service, press the pencil icon to open its configuration and then press Clone button. When a service is cloned, its parent links are preserved, while the child links are not.

To delete a service, press on the x icon next to it. When you delete a parent service, its child services will not be deleted and will move one level higher in the service tree (1st level children will get the same level as the deleted parent service).

Two buttons below the list of services offer some mass-editing options:

- *Mass update* - mass update service properties
- *Delete* - delete the services

To use these options, mark the checkboxes before the respective services, then click on the required button.

2 SLA

Overview Once the services are created, you can start monitoring whether their performance is on track with service level agreement (SLA).

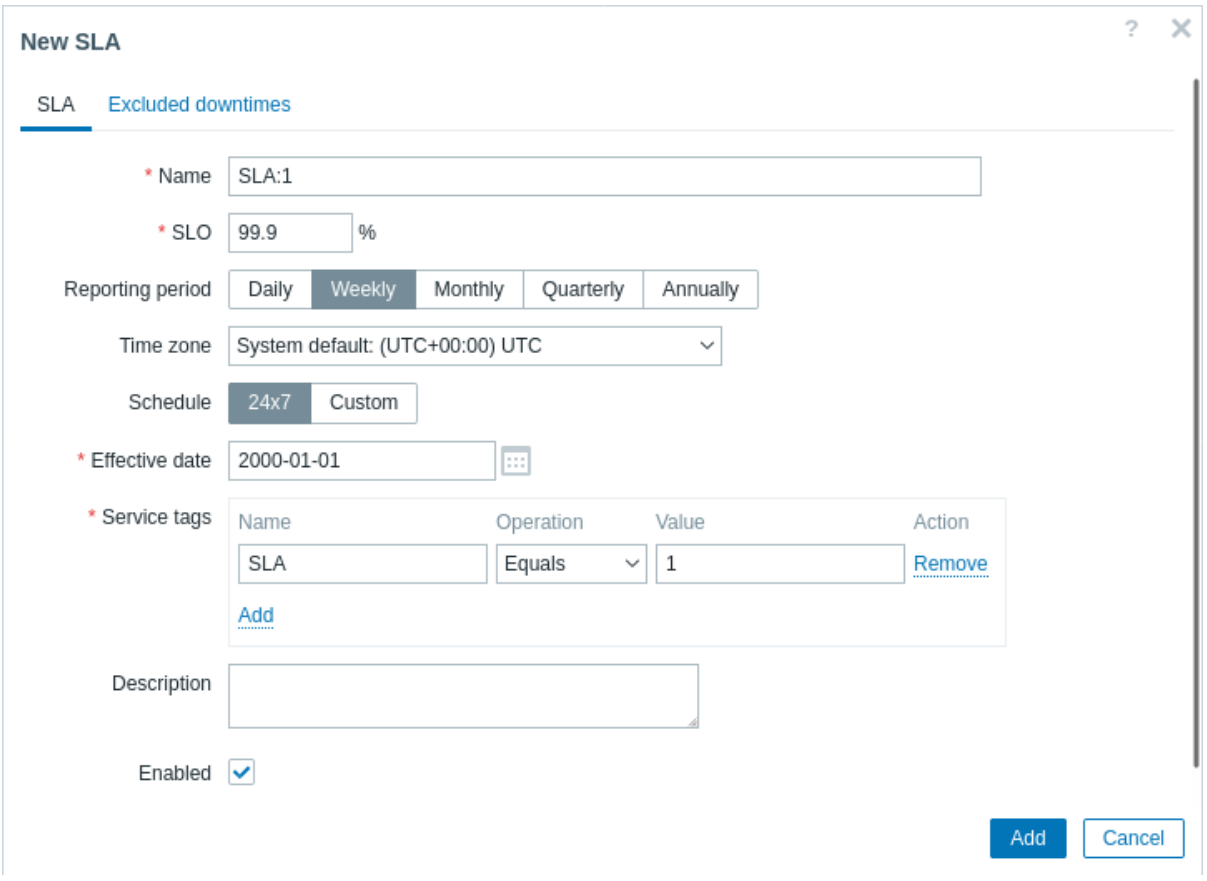
Services->SLA menu section allows to configure SLAs for various services. An SLA in Zabbix defines service level objective (SLO), expected uptime schedule and planned downtimes.

SLAs and services are matched by service tags. The same SLA may be applied to multiple services - performance will be measured for each matching service separately. A single service may have multiple SLAs assigned - data for each of the SLAs will be displayed separately.

In SLA reports Zabbix provides Service level indicator (SLI) data, which measures real service availability. Whether a service meets the SLA targets is determined by comparing SLO (expected availability in %) with SLI (real-life availability in %).

Configuration To create a new SLA, click on the *Create SLA* button.

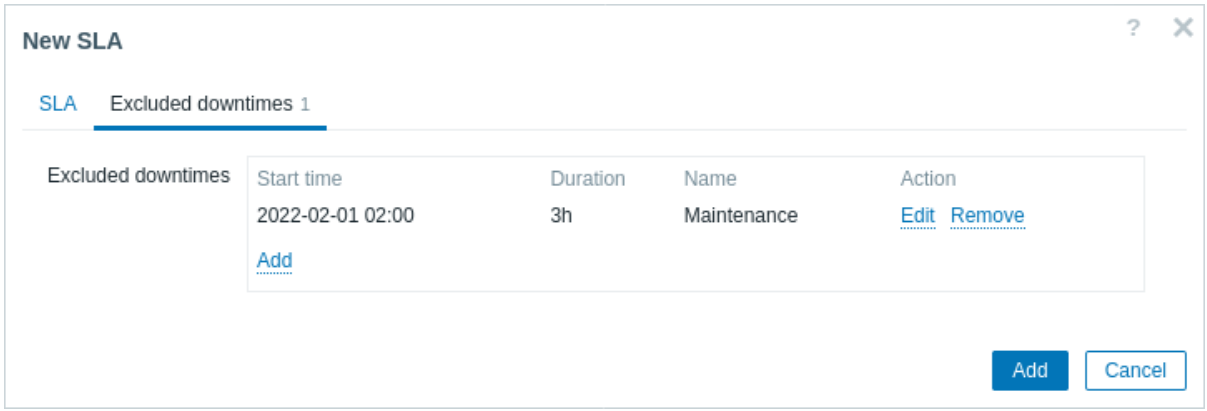
The **SLA** tab allows to specify general SLA parameters.



Parameter	Description
Name	Enter the SLA name.
SLO	Enter the service level objective (SLO) as percentage.

Parameter	Description
Reporting period	Selecting the period will affect what periods are used in the SLA report - *daily*, *weekly*, *monthly*, *quarterly*, or *annually*.
Time zone	Select the SLA time zone.
Schedule	Select the SLA schedule - 24x7 or custom.
Effective date	Select the date of starting SLA calculation.
Service tags	Add service tags to identify the services towards which this SLA should be applied. **Name** - service tag name, must be exact match, case-sensitive. **Operation** - select *Equals* if the tag value must match exactly (case-sensitive) or *Contains* if part of the tag value must match (case-insensitive). **Value** - service tag value to search for according to selected operation. The SLA is applied to a service, if at least one service tag matches.
Description	Add a description for the SLA.
Enabled	Mark the checkbox to enable the SLA calculation.

The **Excluded downtimes** tab allows to specify downtimes that are excluded from the SLA calculation.



Click on *Add* to configure excluded downtimes, then enter the period name, start date and duration.

SLA reports How a service performs compared to an SLA is visible in the SLA report. SLA reports can be viewed:

- from the *SLA* section by clicking on the SLA report hyperlink;
- from the *Services* section by clicking on the SLA name in the info tab;
- in the Dashboard widget *SLA report*.

Once an SLA is configured, the *Info* tab in the services section will also display some information about service performance.

3 Setup example

Overview This section describes a simple setup for monitoring Zabbix high availability cluster as a service.

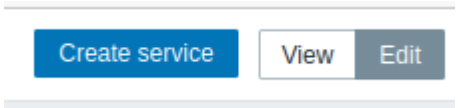
Pre-requisites Prior to configuring service monitoring, you need to have the hosts configured:

- *HA node 1* with at least one trigger and a tag (preferably set on a trigger level) `component:HA node 1`
- *HA node 2* with at least one trigger and a tag (preferably set on a trigger level) `component:HA node 2`

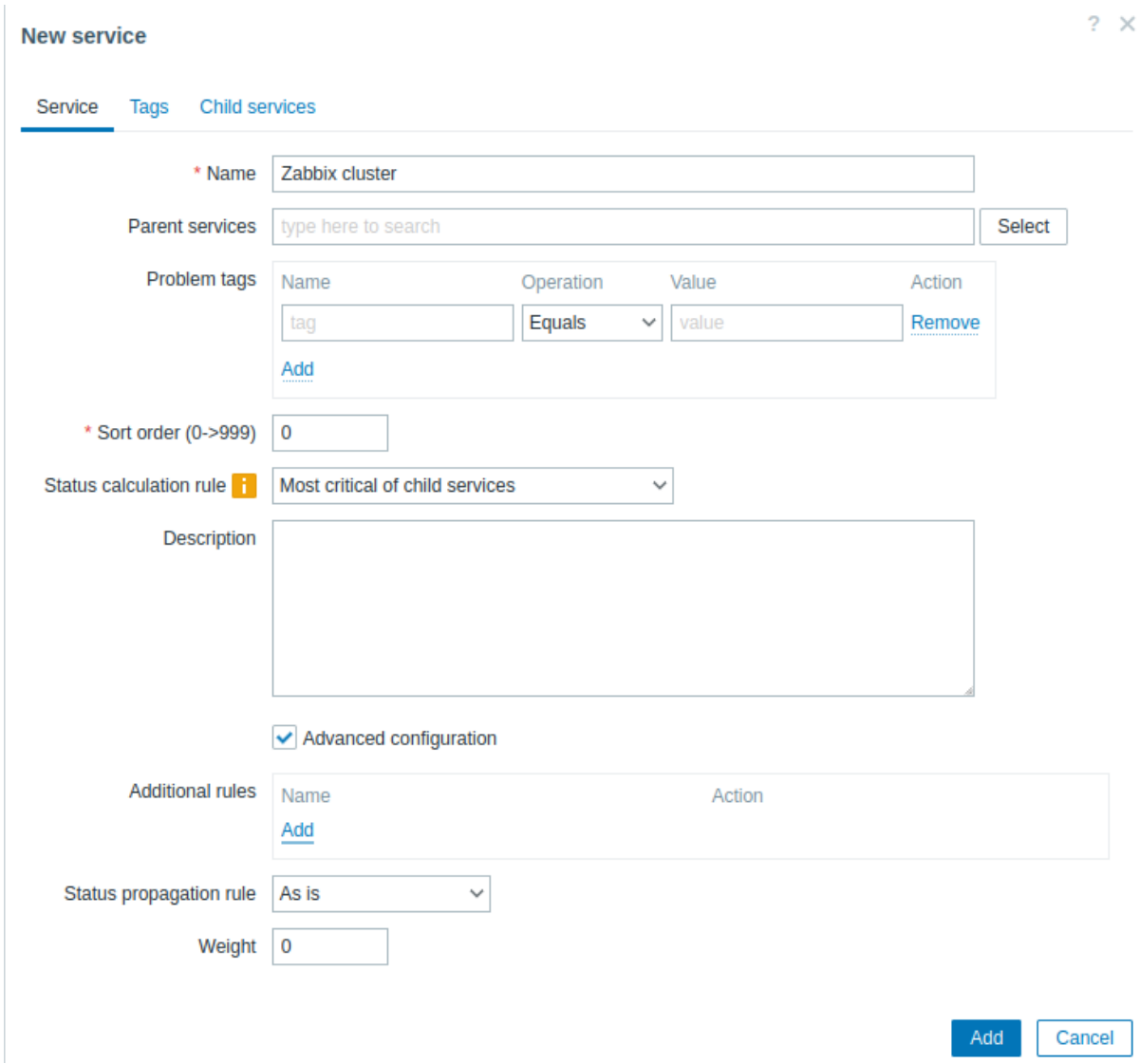
Service tree The next step is to build the service tree. In this example, the infrastructure is very basic and consists of three services: *Zabbix cluster* (parent) and two child services *Zabbix server node 1* and *Zabbix server node 2*.

```
Zabbix cluster
|
|- Zabbix server node 1
|- Zabbix server node 2
```

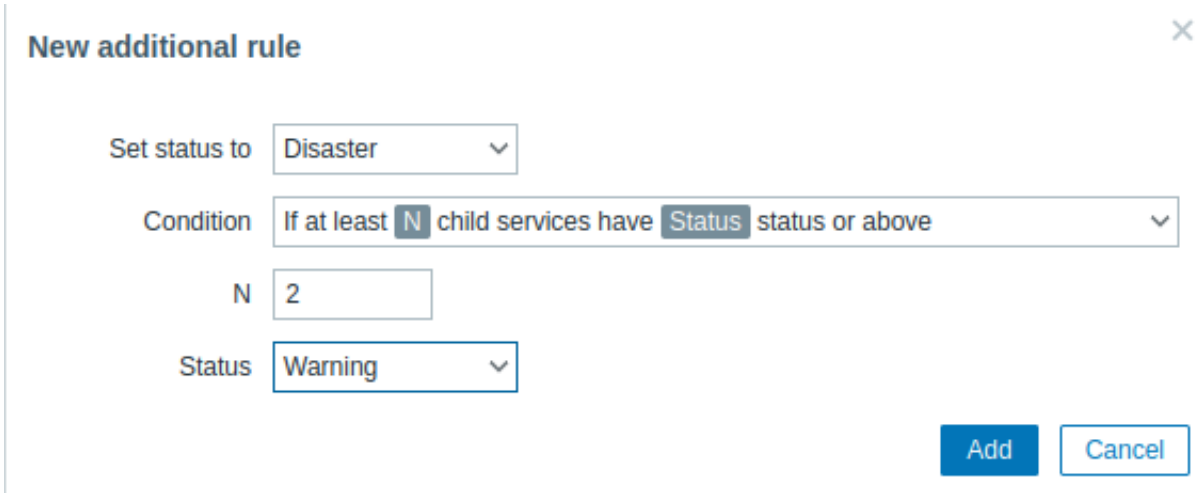
At the Services page, turn on *Edit* mode and press Create service:



In the service configuration window, enter name *Zabbix cluster* and mark the checkbox *Advanced configuration*.



Configure additional rule:

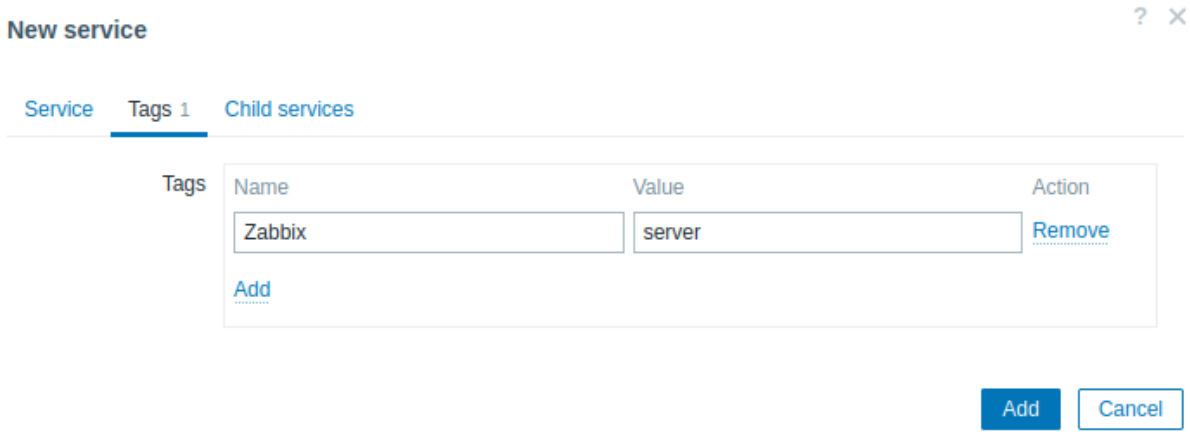


Zabbix cluster will have two child services - one for each of the HA nodes. If both HA nodes have problems of at least *Warning* status, parent service status should be set to *Disaster*. To achieve this, additional rule should be configured as:

- Set status to: Disaster
- Condition: If at least `N` child services have `Status` status or above

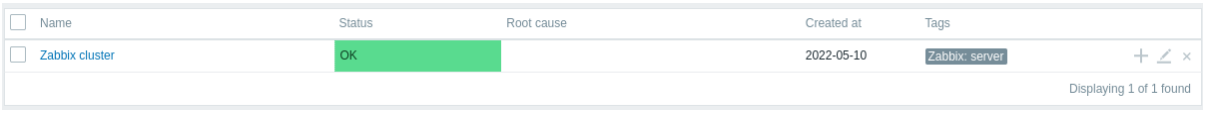
- N: 2
- Status: Warning

Switch to the *Tags* tab and add a tag `Zabbix:server`. This tag will be used later for service actions and SLA reports.



Save the new service.

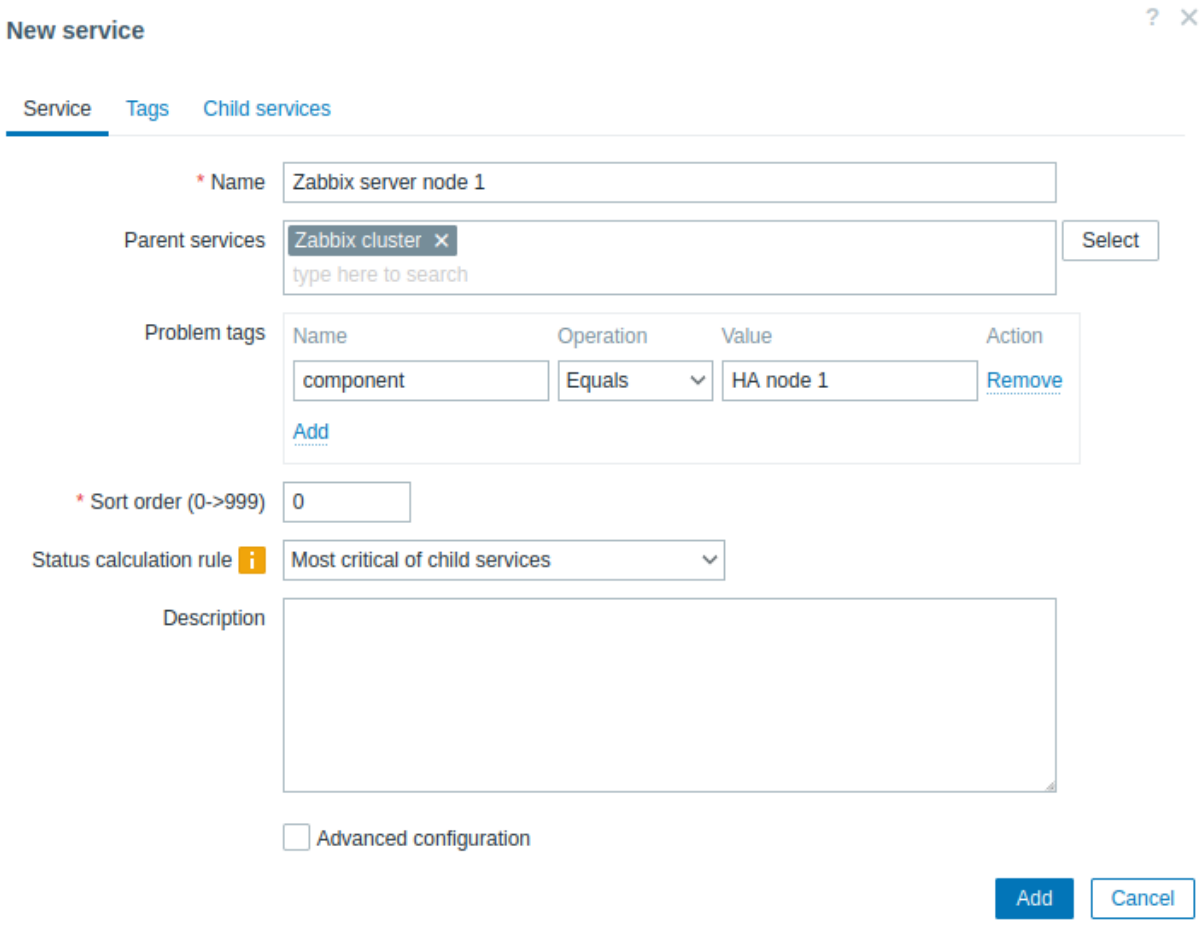
To add a child service, press on the plus icon next to the Zabbix cluster service (the icon is visible only in *Edit* mode).



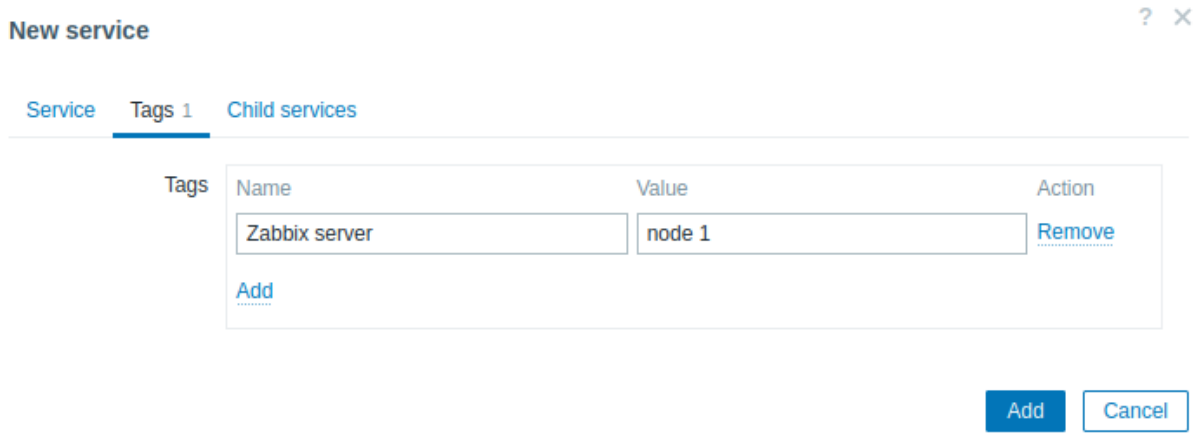
In the service configuration window, enter name *Zabbix server node 1*. Note that the Parent services parameter is already pre-filled with *Zabbix cluster*.

Availability of this service is affected by problems on the host *HA node 1*, marked with `component:HA node 1` problem tag. In the Problem tags parameter, enter:

- Name: component
- Operation: Equals
- Value: HA node 1



Switch to the *Tags* tab and add a service tag: `Zabbix server:node 1`. This tag will be used later for service actions and SLA reports.



Save the new service.

Create another child service of Zabbix cluster with name ”Zabbix server node 2”.

Set the Problem tags as:

- Name: component
- Operation: Equals
- Value: HA node 2

Switch to the *Tags* tab and add a service tag: `Zabbix server:node 2`.

Save the new service.

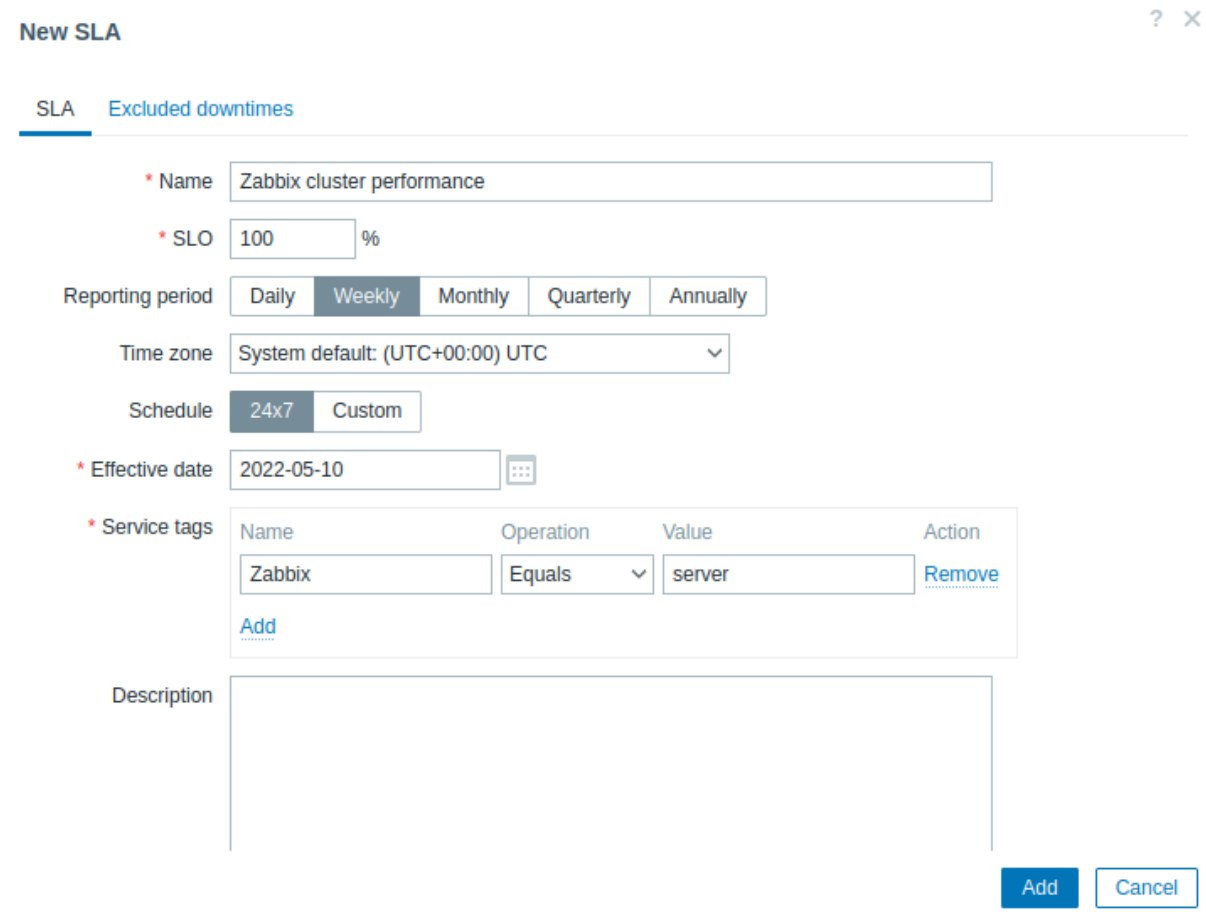
SLA In this example, expected Zabbix cluster performance is 100% excluding semi-annual one hour maintenance period.

First, you need to add a new service level agreement.

Go to the *Services->SLA* menu section and press Create SLA. Enter name *Zabbix cluster performance* and set the SLO to 100%.

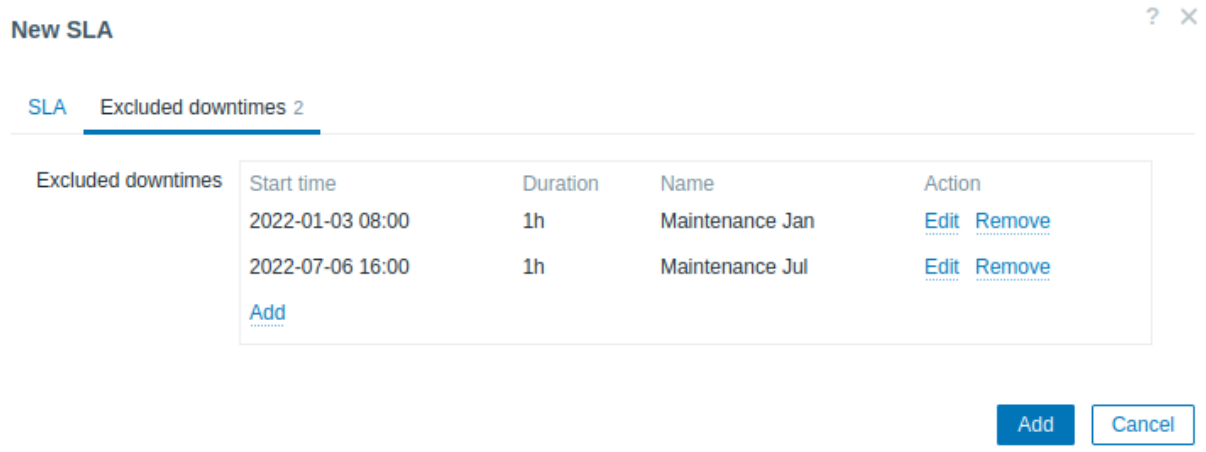
The service Zabbix cluster has a service tag `Zabbix:server`. To use this SLA for measuring performance of Zabbix cluster, in the *Service tags* parameter, specify:

- Name: Zabbix
- Operation: Equals
- Value: server



In a real-life setup, you can also update desired reporting period, time zone and start date or change the schedule from 24/7 to custom. For this example, the default settings are sufficient.

Switch to the *Excluded downtimes* tab and add downtimes for scheduled maintenance periods to exclude these periods from SLA calculation. In the Excluded downtimes section press the Add link, enter downtime name, planned start time and duration.

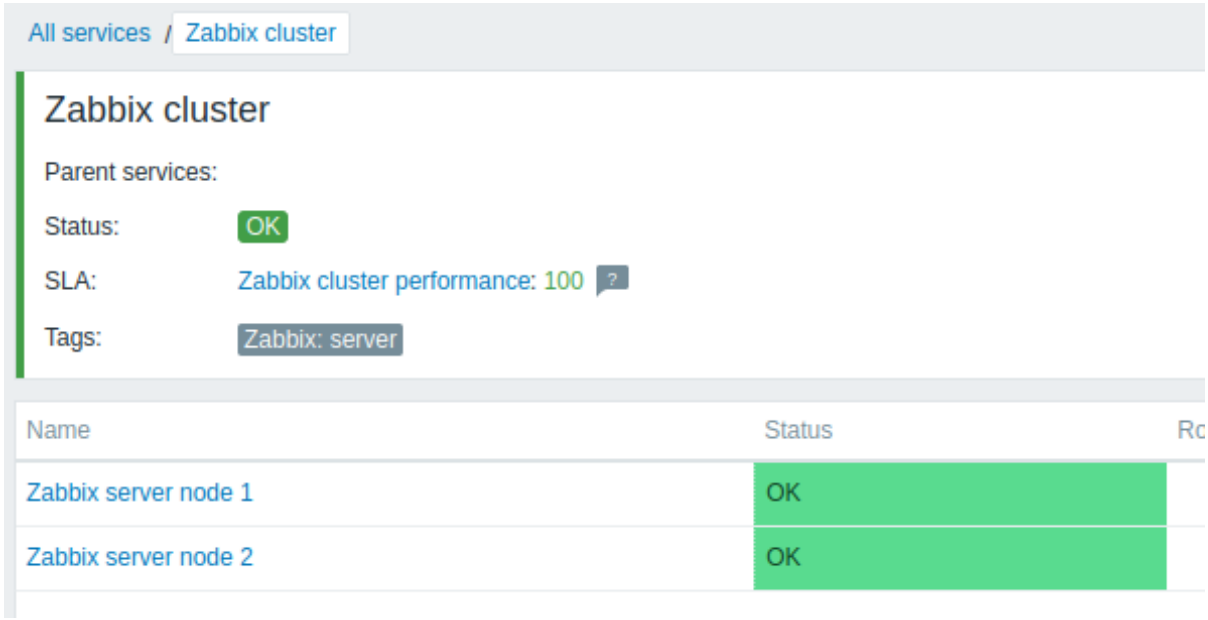


Press Add to save the new SLA.

Switch to the SLA reports section to view the SLA report for Zabbix cluster.

Year	SLO	SLI	Uptime	Downtime	Error budget
2022	100%	100	36m 53s	0	0

The SLA info can also be checked in the *Services* section.



9 Web monitoring

Overview With Zabbix you can check several availability aspects of web sites.

Attention:
To perform web monitoring Zabbix server must be initially configured with cURL (libcurl) support.

To activate web monitoring you need to define web scenarios. A web scenario consists of one or several HTTP requests or "steps". The steps are periodically executed by Zabbix server in a pre-defined order. If a host is monitored by proxy, the steps are executed by the proxy.

Web scenarios are attached to hosts/templates in the same way as items, triggers, etc. That means that web scenarios can also be created on a template level and then applied to multiple hosts in one move.

The following information is collected in any web scenario:

- average download speed per second for all steps of whole scenario
- number of the step that failed
- last error message

The following information is collected in any web scenario step:

- download speed per second
- response time
- response code

For more details, see web monitoring items.

Data collected from executing web scenarios is kept in the database. The data is automatically used for graphs, triggers and notifications.

Zabbix can also check if a retrieved HTML page contains a pre-defined string. It can execute a simulated login and follow a path of simulated mouse clicks on the page.

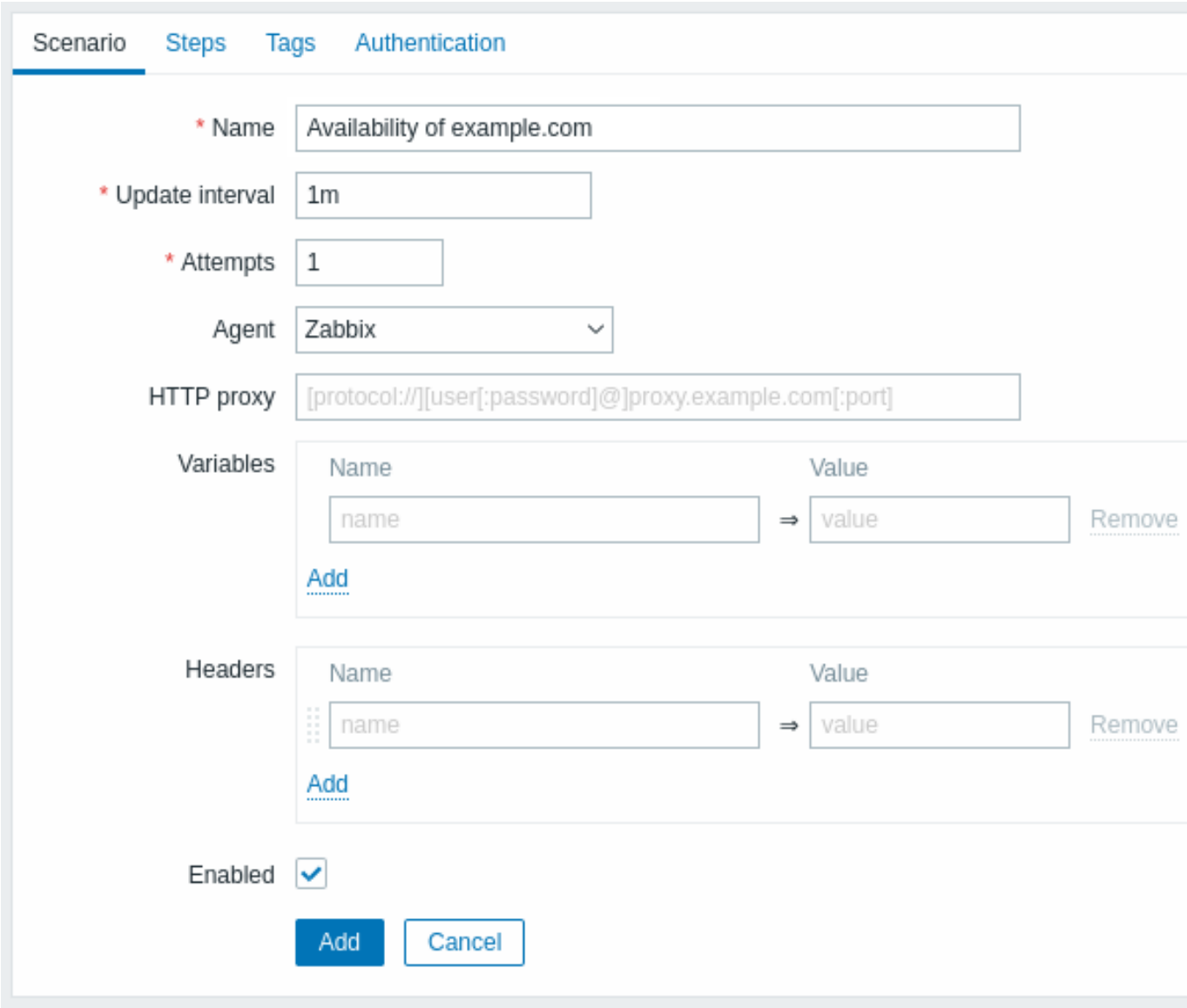
Zabbix web monitoring supports both HTTP and HTTPS. When running a web scenario, Zabbix will optionally follow redirects (see option *Follow redirects* below). Maximum number of redirects is hard-coded to 10 (using cURL option CURLOPT_MAXREDIRS). All cookies are preserved during the execution of a single scenario.

Configuring a web scenario To configure a web scenario:

- Go to: *Data collection* → *Hosts* (or *Templates*)
- Click on *Web* in the row of the host/template
- Click on *Create web scenario* to the right (or on the scenario name to edit an existing scenario)

- Enter parameters of the scenario in the form

The **Scenario** tab allows you to configure the general parameters of a web scenario.



All mandatory input fields are marked with a red asterisk.

Scenario parameters:

Parameter	Description
Name	Unique scenario name. User macros are supported. *Note* that if user macros are used, these macros will be left unresolved in web monitoring item names.
Update interval	How often the scenario will be executed. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d. User macros are supported. *Note* that if a user macro is used and its value is changed (e.g. 5m → 30s), the next check will be executed according to the previous value (farther in the future with the example values). New web scenarios will be checked within 60 seconds of their creation.
Attempts	The number of attempts for executing web scenario steps. In case of network problems (timeout, no connectivity, etc) Zabbix can repeat executing a step several times. The figure set will equally affect each step of the scenario. Up to 10 attempts can be specified, default value is 1. *Note*: Zabbix will not repeat a step because of a wrong response code or the mismatch of a required string.
Agent	Select a client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers. User macros can be used in this field.

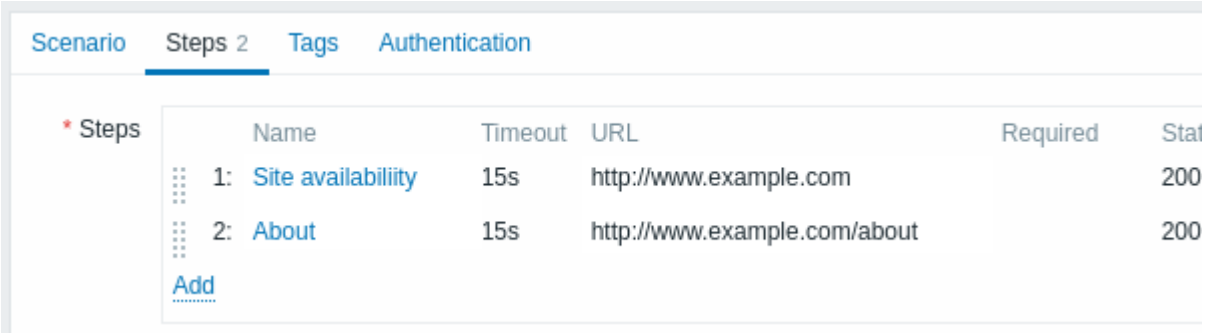
Parameter	Description
HTTP proxy	You can specify an HTTP proxy to use, using the format `[protocol://][username[:password]@]proxy.example.com[:port]`. This sets the CURLOPT_PROXY cURL option. The optional `protocol://` prefix may be used to specify alternative proxy protocols (the protocol prefix support was added in cURL 7.21.7). With no protocol specified, the proxy will be treated as an HTTP proxy. By default, 1080 port will be used. If specified, the proxy will overwrite proxy related environment variables like http_proxy, HTTPS_PROXY. If not specified, the proxy will not overwrite proxy-related environment variables. The entered value is passed on "as is", no sanity checking takes place. You may also enter a SOCKS proxy address. If you specify the wrong protocol, the connection will fail and the item will become unsupported. *Note* that only simple authentication is supported with HTTP proxy. User macros can be used in this field.
Variables	Variables that may be used in scenario steps (URL, post variables). They have the following format: **{macro1}**=value1 **{macro2}**=value2 **{macro3}**=regex:<regular expression> For example: {username}=Alexei {password}=kj3h5kJ34bd {hostid}=regex:hostid is ([0-9]+) The macros can then be referenced in the steps as {username}, {password} and {hostid}. Zabbix will automatically replace them with actual values. Note that variables with `regex:` need one step to get the value of the regular expression so the extracted value can only be applied to the step after. If the value part starts with `regex:` then the part after it is treated as a regular expression that searches the web page and, if found, stores the match in the variable. At least one subgroup must be present so that the matched value can be extracted. User macros and {HOST.*} macros are supported. Variables are automatically URL-encoded when used in query fields or form data for post variables, but must be URL-encoded manually when used in raw post or directly in URL.
Headers	HTTP Headers are used when performing a request. Default and custom headers can be used. Headers will be assigned using default settings depending on the Agent type selected from a drop-down list on a scenario level, and will be applied to all the steps, unless they are custom defined on a step level. **It should be noted that defining the header on a step level automatically discards all the previously defined headers, except for a default header that is assigned by selecting the 'User-Agent' from a drop-down list on a scenario level.** However, even the 'User-Agent' default header can be overridden by specifying it on a step level. To unset the header on a scenario level, the header should be named and attributed with no value on a step level. Headers should be listed using the same syntax as they would appear in the HTTP protocol, optionally using some additional features supported by the CURLOPT_HTTPHEADER cURL option. For example: Accept-Charset=utf-8 Accept-Language=en-US Content-Type=application/xml; charset=utf-8 User macros and {HOST.*} macros are supported.
Enabled	The scenario is active if this box is checked, otherwise - disabled.

Note that when editing an existing scenario, two extra buttons are available in the form:

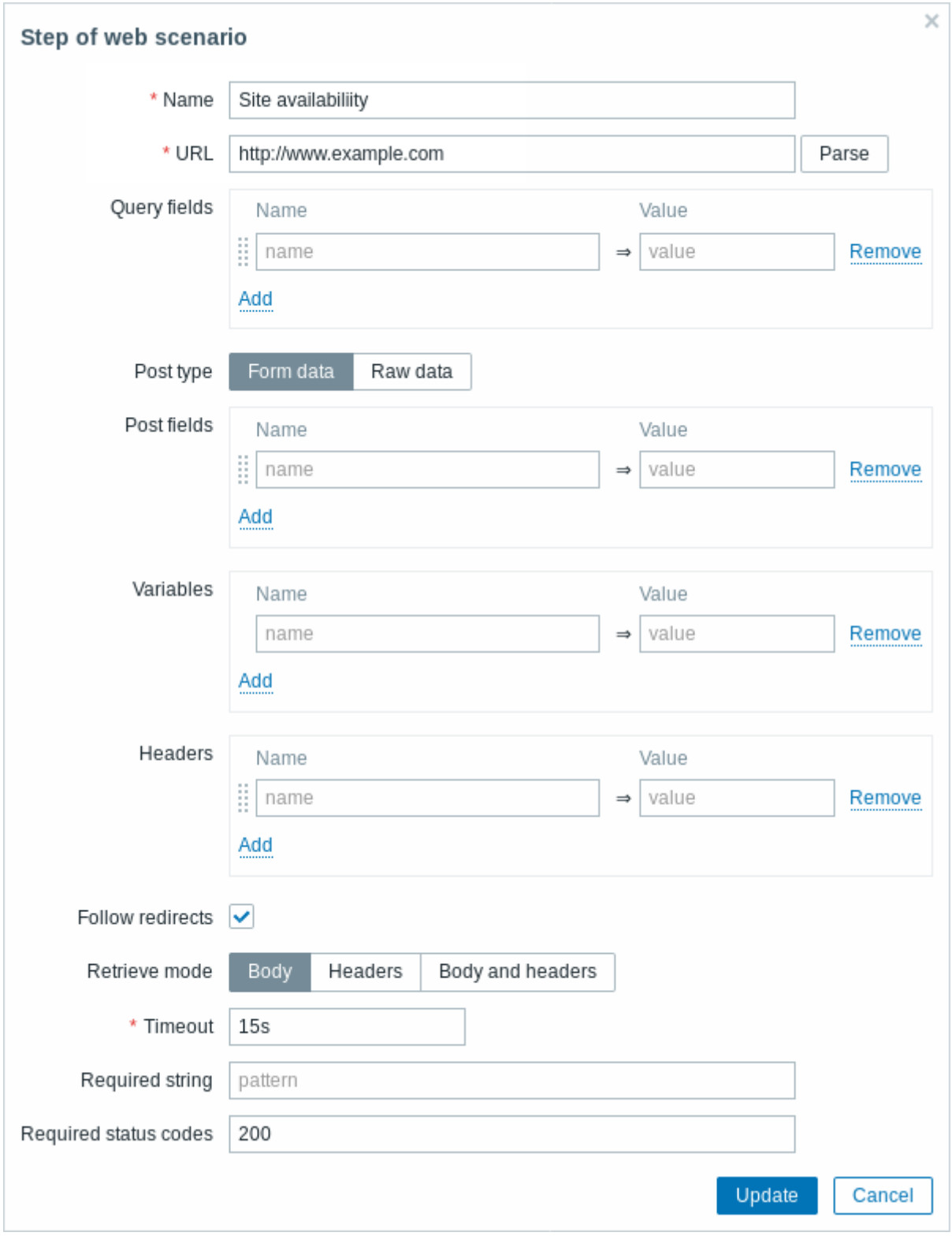
Clone	Create another scenario based on the properties of the existing one.
Clear history and trends	Delete history and trend data for the scenario. This will make the server perform the scenario immediately after deleting the data.

Note:
If *HTTP proxy* field is left empty, another way for using an HTTP proxy is to set proxy related environment variables.
For HTTP checks - set the **http_proxy** environment variable for the Zabbix server user. For example, `http_proxy=http://proxy_ip:proxy_port`.
For HTTPS checks - set the **HTTPS_PROXY** environment variable. For example, `HTTPS_PROXY=http://proxy_ip:proxy_port`. More details are available by running a shell command: *# man curl*.

The **Steps** tab allows you to configure the web scenario steps. To add a web scenario step, click on *Add* in the *Steps* block.



Note:
Secret user macros must not be used in URLs as they will resolve to "******".



Configuring steps

Step parameters:

Parameter	Description
Name	Unique step name. User macros are supported. *Note* that if user macros are used, these macros will be left unresolved in web monitoring item names.

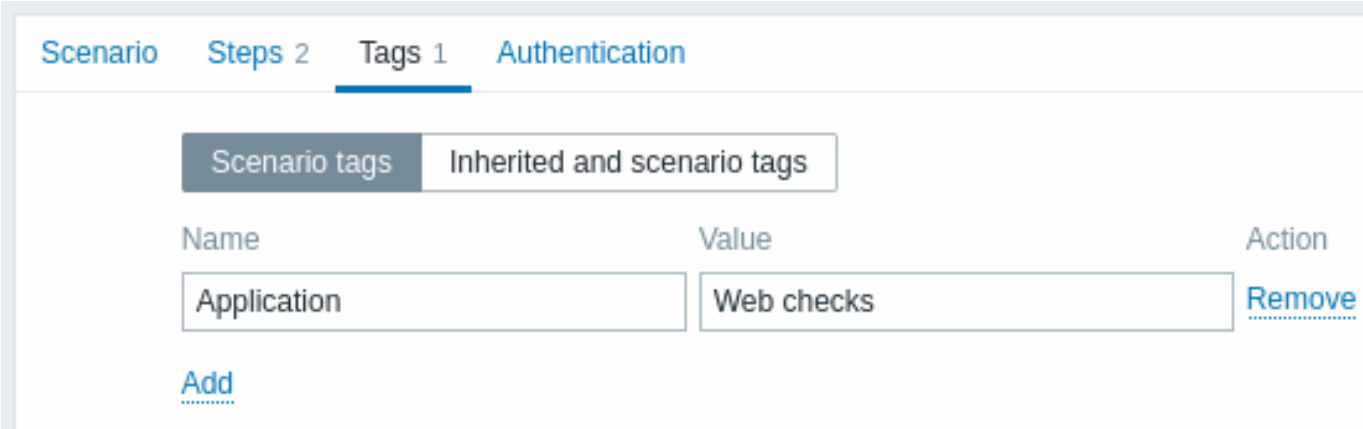
Parameter	Description
URL	URL to connect to and retrieve data. For example: https://www.example.com http://www.example.com/download Domain names can be specified in Unicode characters. They are automatically punycode-converted to ASCII when executing the web scenario step. The *Parse* button can be used to separate optional query fields (like ?name=Admin&password=mypassword) from the URL, moving the attributes and values into *Query fields* for automatic URL-encoding. Variables can be used in the URL, using the {macro} syntax. Variables can be URL-encoded manually using a {{macro}.urlencode()} syntax. User macros and {HOST.*} macros are supported. Limited to 2048 characters.
Query fields	HTTP GET variables for the URL. Specified as attribute and value pairs. Values are URL-encoded automatically. Values from scenario variables, user macros or {HOST.*} macros are resolved and then URL-encoded automatically. Using a {{macro}.urlencode()} syntax will double URL-encode them. User macros and {HOST.*} macros are supported.
Post	HTTP POST variables. In **Form data** mode, specified as attribute and value pairs. Values are URL-encoded automatically. Values from scenario variables, user macros or {HOST.*} macros are resolved and then URL-encoded automatically. In **Raw data** mode, attributes/values are displayed on a single line and concatenated with a **&** symbol. Raw values can be URL-encoded/decoded manually using a {{macro}.urlencode()} or {{macro}.urldecode()} syntax. For example: id=2345&userid={user} If {user} is defined as a variable of the web scenario, it will be replaced by its value when the step is executed. If you wish to URL-encode the variable, substitute {user} with {{user}.urlencode()}. User macros and {HOST.*} macros are supported.
Variables	Step-level variables that may be used for GET and POST functions. Specified as attribute and value pairs. Step-level variables override scenario-level variables or variables from the previous step. However, the value of a step-level variable only affects the step after (and not the current step). They have the following format: **{macro}**=value **{macro}**=regex:<regular expression> For more information see variable description on the scenario level. Variables are automatically URL-encoded when used in query fields or form data for post variables, but must be URL-encoded manually when used in raw post or directly in URL.
Headers	Custom HTTP headers that will be sent when performing a request. Specified as attribute and value pairs. A header defined on a step level will be used for that particular step. **It should be noted that defining the header on a step level automatically discards all the previously defined headers, except for a default header that is assigned by selecting the 'User-Agent' from a drop-down list on a scenario level.** However, even the 'User-Agent' default header can be overridden by specifying it on a step level. For example, assigning the name to a header, but setting no value will unset the default header on a scenario level. User macros and {HOST.*} macros are supported. This sets the CURLOPT_HTTPHEADER cURL option.
Follow redirects	Mark the checkbox to follow HTTP redirects. This sets the CURLOPT_FOLLOWLOCATION cURL option.
Retrieve mode	Select the retrieve mode: **Body** - retrieve only body from the HTTP response **Headers** - retrieve only headers from the HTTP response **Body and headers** - retrieve body and headers from the HTTP response

Parameter	Description
Timeout	Zabbix will not spend more than the set amount of time on processing the URL (from one second to maximum of 1 hour). Actually this parameter defines the maximum time for making connection to the URL and maximum time for performing an HTTP request. Therefore, Zabbix will not spend more than **2 x Timeout** seconds on the step. Time suffixes are supported, e.g. 30s, 1m, 1h. User macros are supported.
Required string	Required regular expression pattern. Unless retrieved content (HTML) matches the required pattern the step will fail. If empty, no check on required string is performed. For example: Homepage of Zabbix Welcome.*admin *Note*: Referencing regular expressions created in the Zabbix frontend is not supported in this field. User macros and {HOST.*} macros are supported.
Required status codes	List of expected HTTP status codes. If Zabbix gets a code which is not in the list, the step will fail. If empty, no check on status codes is performed. For example: 200,201,210-299 User macros are supported.

Note:
Any changes in web scenario steps will only be saved when the whole scenario is saved.

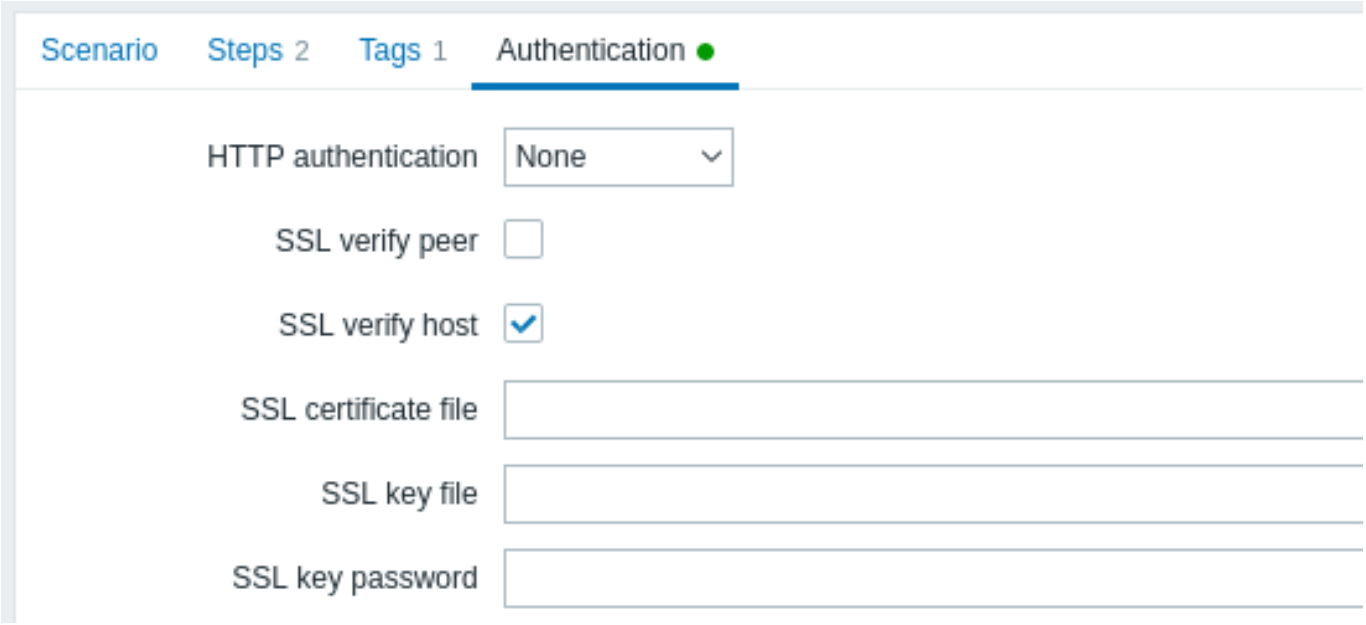
See also a real-life example of how web monitoring steps can be configured.

Configuring tags The **Tags** tab allows to define scenario-level tags.



Tagging allows to filter web scenarios and web monitoring items.

Configuring authentication The **Authentication** tab allows you to configure scenario authentication options. A green dot next to the tab name indicates that some type of HTTP authentication is enabled.



Authentication parameters:

Parameter	Description
Authentication	Authentication options. **None** - no authentication used. **Basic** - basic authentication is used. **NTLM** - NTLM (Windows NT LAN Manager) authentication is used. **Kerberos** - Kerberos authentication is used. See also: Configuring Kerberos with Zabbix. **Digest** - Digest authentication is used. Selecting an authentication method will provide two additional fields for entering a user name and password. User macros can be used in user and password fields.
SSL verify peer	Mark the checkbox to verify the SSL certificate of the web server. The server certificate will be automatically taken from system-wide certificate authority (CA) location. You can override the location of CA files using Zabbix server or proxy configuration parameter SSLCALocation. This sets the CURLOPT_SSL_VERIFYPEER cURL option.
SSL verify host	Mark the checkbox to verify that the *Common Name* field or the *Subject Alternate Name* field of the web server certificate matches. This sets the CURLOPT_SSL_VERIFYHOST cURL option.
SSL certificate file	Name of the SSL certificate file used for client authentication. The certificate file must be in PEM[1] format. If the certificate file contains also the private key, leave the *SSL key file* field empty. If the key is encrypted, specify the password in *SSL key password* field. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLCertLocation. `HOST.*` macros and user macros can be used in this field. This sets the CURLOPT_SSLCERT cURL option.
SSL key file	Name of the SSL private key file used for client authentication. The private key file must be in PEM[1] format. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLKeyLocation. `HOST.*` macros and user macros can be used in this field. This sets the CURLOPT_SSLKEY cURL option.
SSL key password	SSL private key file password. User macros can be used in this field. This sets the CURLOPT_KEYPASSWD cURL option.

Attention:
[1] Zabbix supports certificate and private key files in PEM format only. In case you have your certificate and private key data in PKCS #12 format file (usually with extension *.p12 or *.pfx) you may generate the PEM file from it using the following commands:

```
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes  -out ssl-cert.key
```

Note:
Zabbix server picks up changes in certificates without a restart.

Note:
If you have client certificate and private key in a single file just specify it in a "SSL certificate file" field and leave "SSL key file" field empty. The certificate and key must still be in PEM format. Combining certificate and key is easy:
`cat client.crt client.key > client.pem`

Display To view web scenarios configured for a host, go to *Monitoring* → *Hosts*, locate the host in the list and click on the *Web* hyperlink in the last column. Click on the scenario name to get detailed information.



An overview of web scenarios can also be displayed in *Dashboards* by the Web monitoring widget.

Recent results of the web scenario execution are available in the *Monitoring* → *Latest data* section.

Extended monitoring Sometimes it is necessary to log received HTML page content. This is especially useful if some web scenario step fails. Debug level 5 (trace) serves that purpose. This level can be set in server and proxy configuration files or using a runtime control option (`-R log_level_increase="http poller,N"`, where N is the process number). The following examples demonstrate how extended monitoring can be started provided debug level 4 is already set:

`Increase log level of all http pollers:`

```
shell> zabbix_server -R log_level_increase="http poller"

Increase log level of second http poller:
shell> zabbix_server -R log_level_increase="http poller,2"
```

If extended web monitoring is not required it can be stopped using the `-R log_level_decrease` option.

1 Web monitoring items

Overview

Some new items are automatically added for monitoring when web scenarios are created.

All items inherit tags from the web scenario.

Scenario items

As soon as a scenario is created, Zabbix automatically adds the following items for monitoring.

Item	Description
Download speed for scenario <Scenario>	This item will collect information about the download speed (bytes per second) of the whole scenario, i.e. average for all steps. Item key: web.test.in[Scenario„bps] Type: *Numeric(float)*
Failed step of scenario <Scenario>	This item will display the number of the step that failed on the scenario. If all steps are executed successfully, 0 is returned. Item key: web.test.fail[Scenario] Type: *Numeric(unsigned)*
Last error message of scenario <Scenario>	This item returns the last error message text of the scenario. A new value is stored only if the scenario has a failed step. If all steps are ok, no new value is collected. Item key: web.test.error[Scenario] Type: *Character*

The actual scenario name will be used instead of ”Scenario”.

Note:
If the scenario name contains user macros, these macros will be left unresolved in web monitoring item names.

 If the scenario name starts with a doublequote or contains a comma or a square bracket, it will be properly quoted in item keys. In other cases no additional quoting will be performed.

Note:
Web monitoring items are added with a 30 day history and a 90 day trend retention period.

These items can be used to create triggers and define notification conditions.

Example 1

To create a ”Web scenario failed” trigger, you can define a trigger expression:

```
last(/host/web.test.fail[Scenario])<>0
```

Make sure to replace ’Scenario’ with the real name of your scenario.

Example 2

To create a ”Web scenario failed” trigger with a useful problem description in the trigger name, you can define a trigger with name:

```
Web scenario "Scenario" failed: {ITEM.VALUE}
```

and trigger expression:

```
length(last(/host/web.test.error[Scenario]))>0 and last(/host/web.test.fail[Scenario])>0
```

Make sure to replace ’Scenario’ with the real name of your scenario.

Example 3

To create a ”Web application is slow” trigger, you can define a trigger expression:

```
last(/host/web.test.in[Scenario,,bps])<10000
```

Make sure to replace ’Scenario’ with the real name of your scenario.

Scenario step items

As soon as a step is created, Zabbix automatically adds the following items for monitoring.

Item	Description
Download speed for step <Step> of scenario <Scenario>	This item will collect information about the download speed (bytes per second) of the step. Item key: web.test.in[Scenario,Step,bps] Type: *Numeric(float)*
Response time for step <Step> of scenario <Scenario>	This item will collect information about the response time of the step in seconds. Response time is counted from the beginning of the request until all information has been transferred. Item key: web.test.time[Scenario,Step,resp] Type: *Numeric(float)*
Response code for step <Step> of scenario <Scenario>	This item will collect response codes of the step. Item key: web.test.rspcode[Scenario,Step] Type: *Numeric(unsigned)*

Actual scenario and step names will be used instead of ”Scenario” and ”Step” respectively.

Note:
Web monitoring items are added with a 30 day history and a 90 day trend retention period.

Note:
If scenario name starts with a doublequote or contains comma or square bracket, it will be properly quoted in item keys. In other cases no additional quoting will be performed.

These items can be used to create triggers and define notification conditions. For example, to create a ”Zabbix GUI login is too slow” trigger, you can define a trigger expression:

```
last(/zabbix/web.test.time[ZABBIX GUI,Login,resp])>3
```

2 Real-life scenario

Overview

This section presents a step-by-step real-life example of how web monitoring can be used.

Let’s use Zabbix web monitoring to monitor the web interface of Zabbix. We want to know if it is available, provides the right content and how quickly it works. To do that we also must log in with our user name and password.

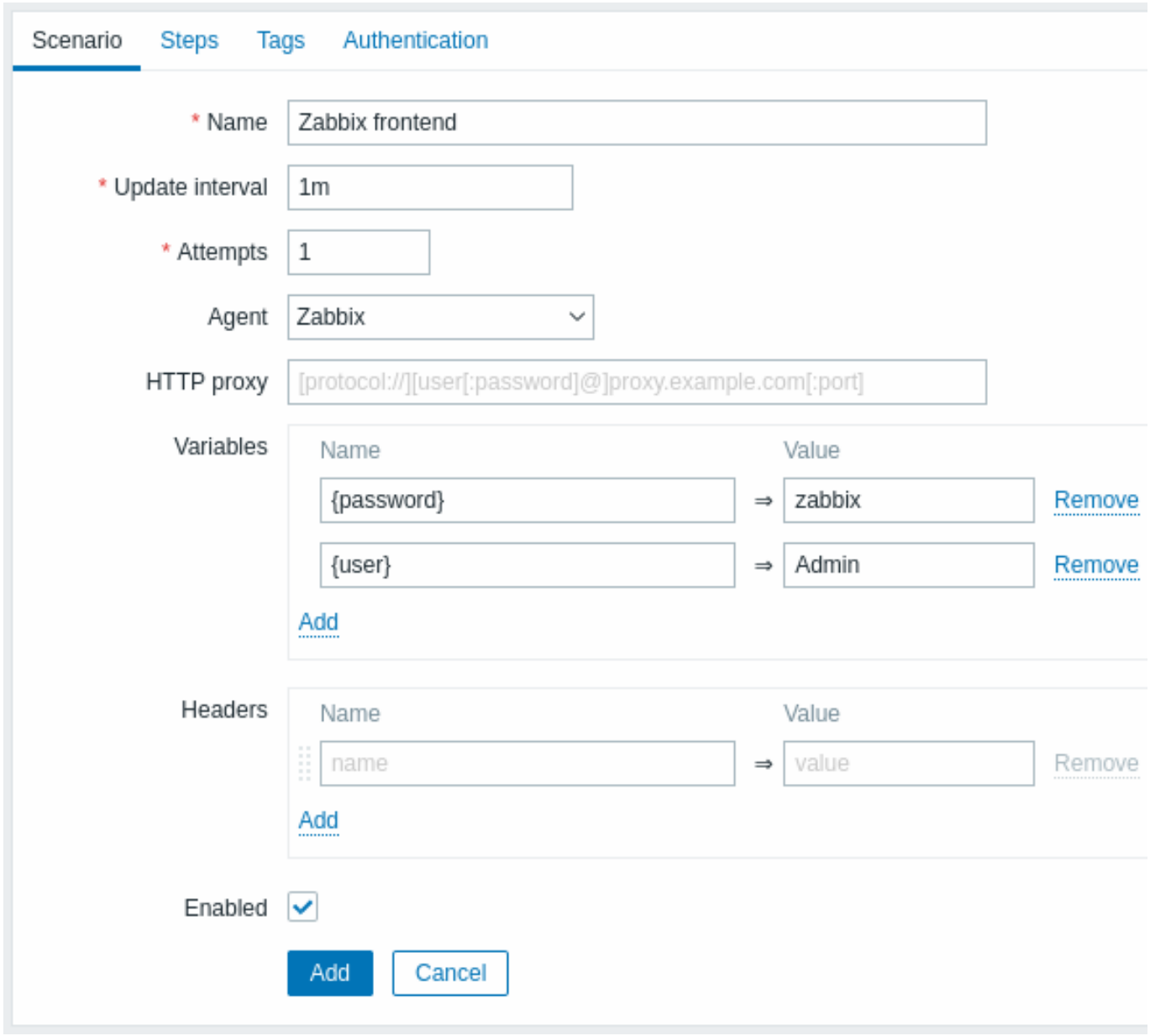
Scenario

Step 1

Add a new web scenario.

We will add a scenario to monitor the web interface of Zabbix. The scenario will execute a number of steps.

Go to *Data collection* → *Hosts*, pick a host and click on *Web* in the row of that host. Then click on *Create web scenario*.



All mandatory input fields are marked with a red asterisk.

In the new scenario form we will name the scenario as *Zabbix frontend*. We will also create two variables: {user} and {password}.

You may also want to add a new *Application:Zabbix frontend* tag in the Tags tab.

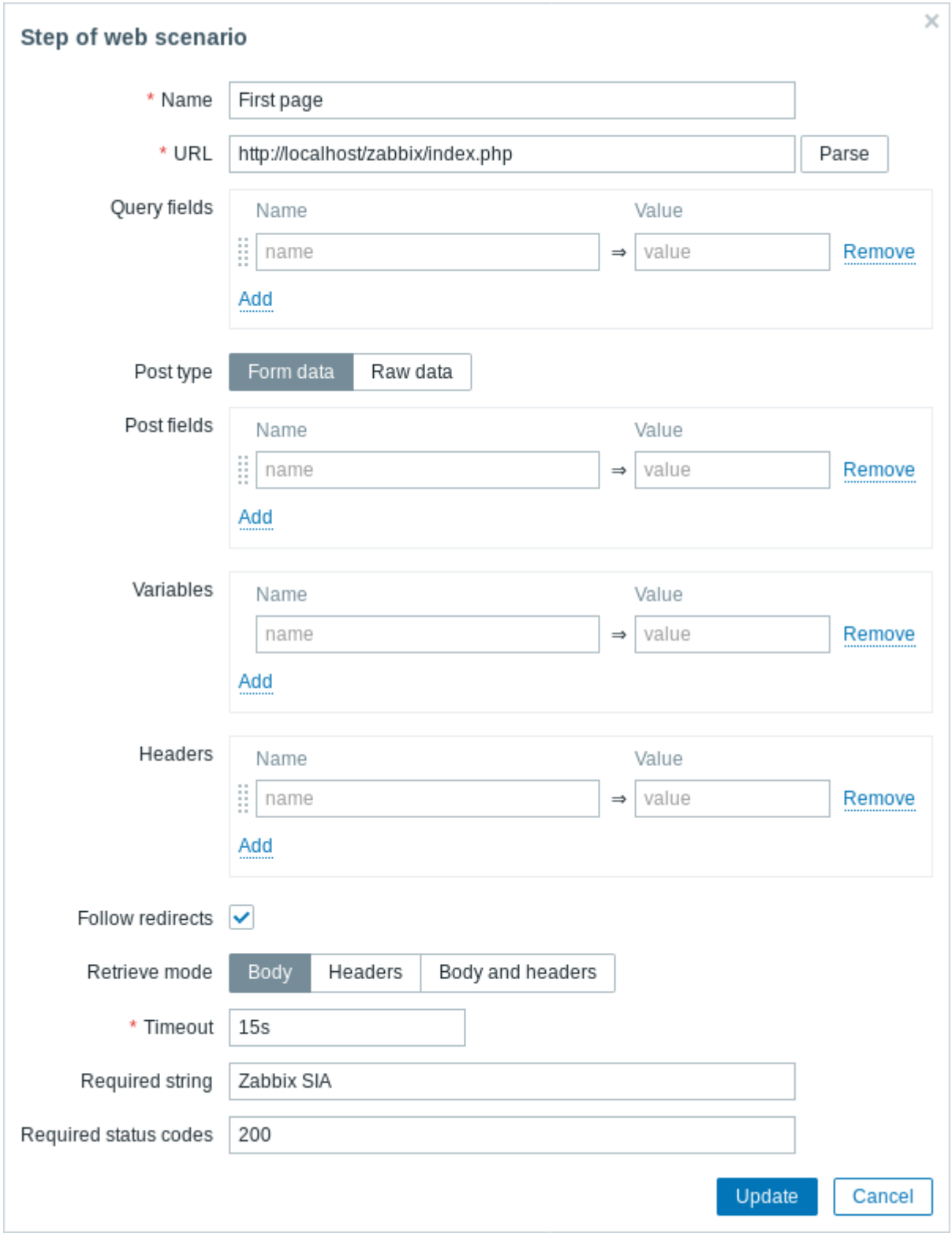
Step 2

Define steps for the scenario.

Click on *Add* button in the *Steps* tab to add individual steps.

Web scenario step 1

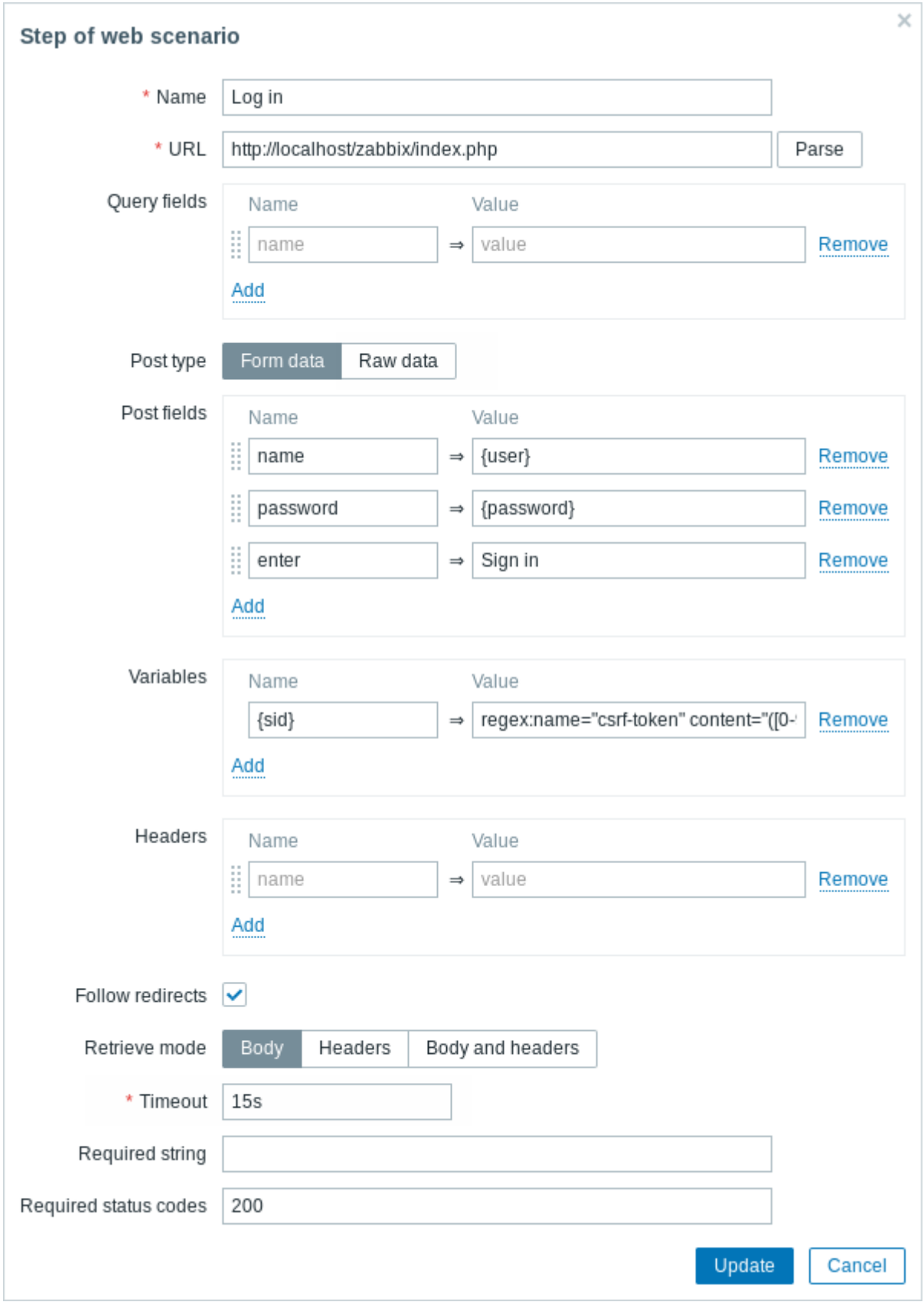
We start by checking that the first page responds correctly, returns with HTTP response code 200 and contains text ”Zabbix SIA”.



When done configuring the step, click on *Add*.

Web scenario step 2

We continue by logging in to the Zabbix frontend, and we do so by reusing the macros (variables) we defined on the scenario level - {user} and {password}.

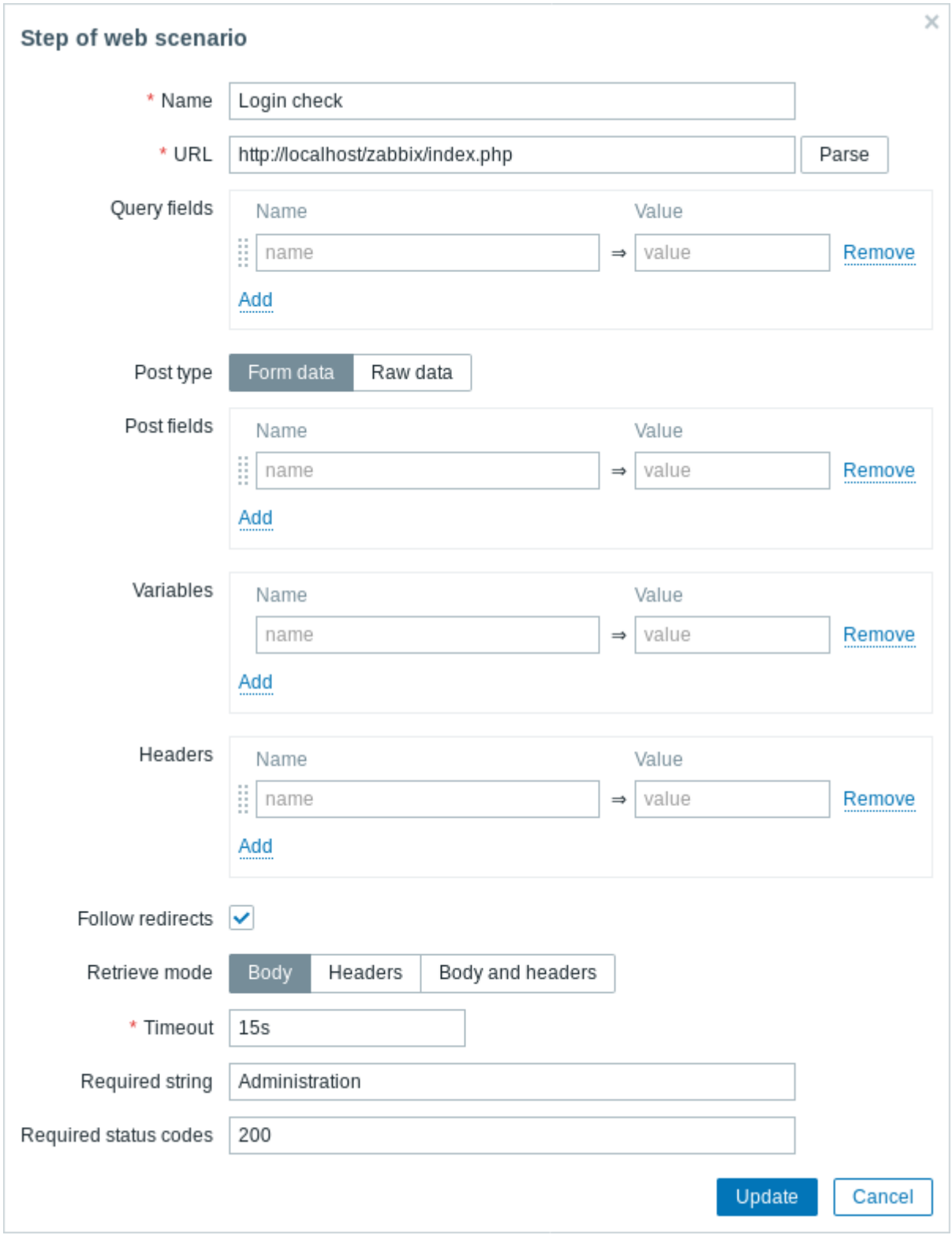


Attention:
Note that Zabbix frontend uses JavaScript redirect when logging in, thus first we must log in, and only in further steps we may check for logged-in features. Additionally, the login step must use full URL to **index.php** file.

Take note also of how we are getting the content of the `{sid}` variable (session ID) using a variable syntax with regular expression: `regex:name="csrf-token" content="([0-9a-z]{16})"`. This variable will be required in step 4.

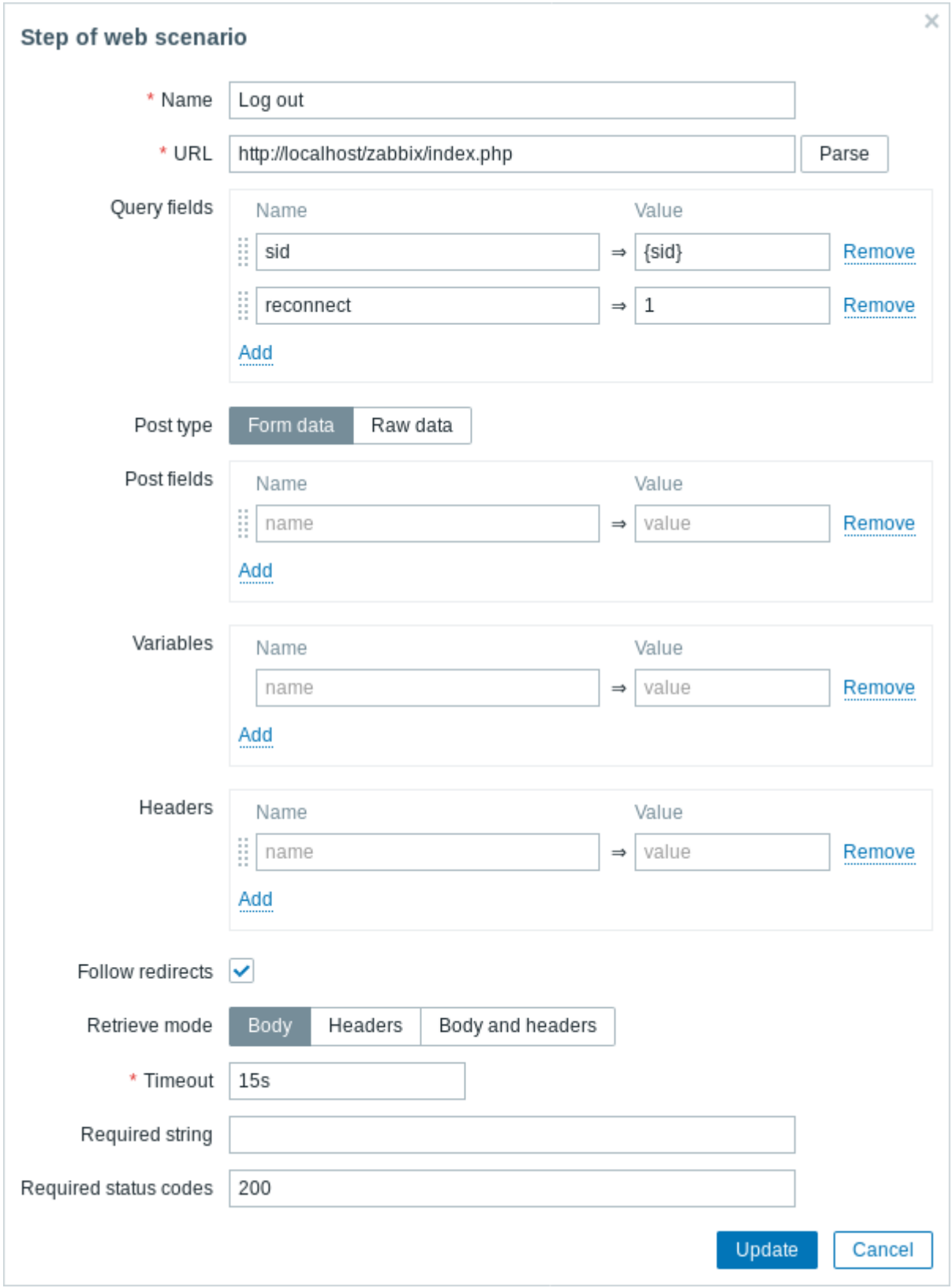
Web scenario step 3

Being logged in, we should now verify the fact. To do so, we check for a string that is only visible when logged in - for example, **Administration**.



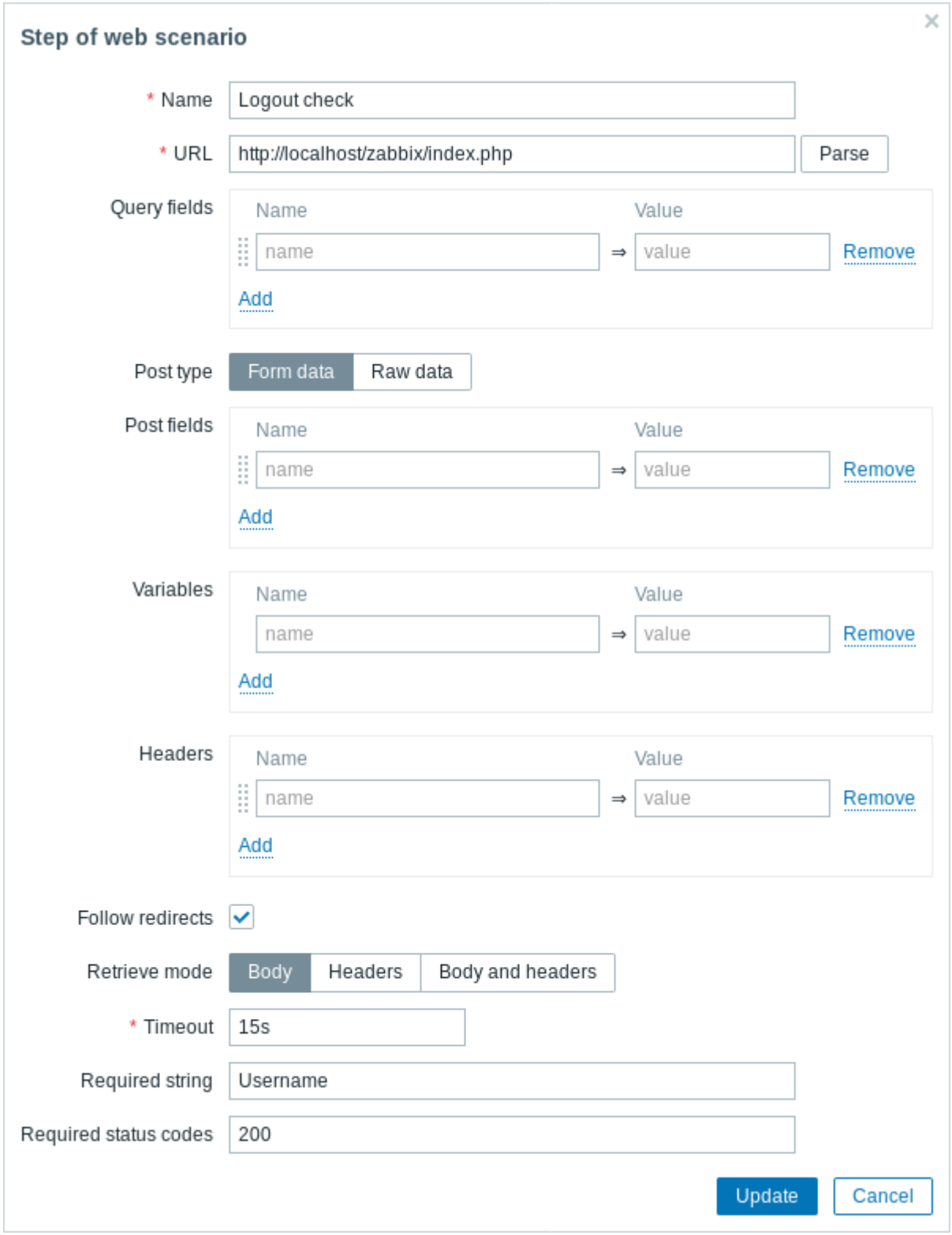
Web scenario step 4

Now that we have verified that frontend is accessible and we can log in and retrieve logged-in content, we should also log out - otherwise Zabbix database will become polluted with lots and lots of open session records.



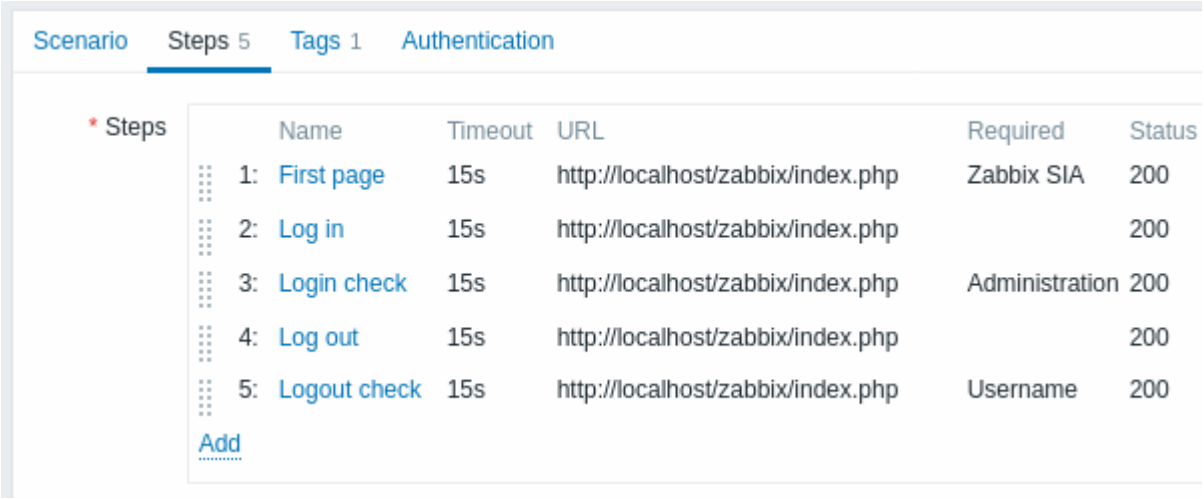
Web scenario step 5

We can also check that we have logged out by looking for the **Username** string.



Complete configuration of steps

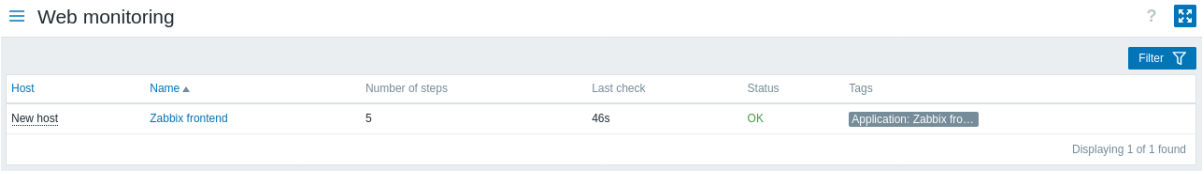
A complete configuration of web scenario steps should look like this:



Step 3

Save the finished web monitoring scenario.

The scenario will be added to a host. To view web scenario information go to *Monitoring* → *Hosts*, locate the host in the list and click on the Web hyperlink in the last column.



Click on the scenario name to see more detailed statistics:



10 Virtual machine monitoring

Overview Zabbix can use low-level discovery rules to automatically discover VMware hypervisors and virtual machines, and create hosts to monitor them based on pre-defined host prototypes.

Zabbix also includes templates for monitoring VMware vCenter or ESXi hypervisors.

The minimum required VMware vCenter or vSphere version is 5.1.

Data collection Virtual machine monitoring consists of two steps:

1. Zabbix *vmware collector* processes collect virtual machine data - the processes obtain necessary information from VMware web services over the SOAP protocol, pre-process it, and store it in Zabbix server shared memory.
2. Zabbix *poller* processes retrieve data using Zabbix simple check VMware monitoring item keys.

Zabbix divides collected data into VMware configuration data and VMware performance counter data. Both types of data are collected independently by the *vmware collector* processes.

The following statistics are available based on the VMware performance counter information:

- Datastore
- Disk device
- CPU
- Power
- Network interface
- Custom performance counter items

For the complete list of items that obtain data from VMware performance counters, see *VMware monitoring item keys*.

Configuration If Zabbix server is compiled from sources, it must be compiled with the `--with-libcurl --with-libxml2` configuration options to enable virtual machine monitoring. Zabbix packages are compiled with these options already enabled.

The following Zabbix server configuration file parameters can be modified for virtual machine monitoring:

- `StartVMwareCollectors`

Note:
It is recommended to enable more collectors than the number of monitored VMware services; otherwise, the retrieval of VMware performance counter statistics might be delayed by the retrieval of VMware configuration data (which takes a while for large installations).

 Generally, the value of `StartVMwareCollectors` should not dip below 2 and should not exceed twice the amount of monitored VMware services: Amount of services < `StartVMwareCollectors` < (Amount of services * 2). For example, when monitoring one VMware service, set `StartVMwareCollectors` to 2; when monitoring three services, set `StartVMwareCollectors` to 5.

 Note that the required number of collectors also depends on the scope of the VMware environment, and the `VMwareFrequency` and `VMwarePerfFrequency` configuration parameters.

- `VMwareCacheSize`
- `VMwareFrequency`
- `VMwarePerfFrequency`
- `VMwareTimeout`

Attention:
To support datastore capacity metrics, ensure that the value of the VMware `vpxd.stats.maxQueryMetrics` key is set to at least 64. For more information, see the *VMware Knowledge Base* article.

Discovery

Zabbix can use low-level discovery rules (for example, *vmware.hv.discovery[{$VMWARE.URL}]*) to automatically discover VMware hypervisors and virtual machines. Moreover, Zabbix can use host prototypes to automatically generate real hosts for the discovered entities. For more information, see *Host prototypes*.

Ready-to-use templates

Zabbix includes a range of ready-to-use templates designed for monitoring VMware vCenter or ESXi hypervisors. These templates contain pre-configured low-level discovery rules, along with various built-in checks for monitoring virtual installations.

The following templates can be used for monitoring VMware vCenter or ESXi hypervisors:

- *VMware* - uses UUID data for corresponding macros
- *VMware FQDN* - uses FQDN data for corresponding macros

Note:
For the correct functioning of the *VMware FQDN* template, each monitored virtual machine should have a unique OS name adhering to FQDN rules. Additionally, VMware Tools must be installed on every machine. If these prerequisites are met, using the *VMware FQDN* template is recommended. The *VMware FQDN* template has been available since Zabbix 5.2 with the introduction of the ability to create hosts with custom interfaces.

 A classic *VMware* template is also available and can be used if FQDN requirements are unmet. However, the *VMware* template has a known issue. Hosts for discovered virtual machines are created with names that are saved in vCenter (for example, "VM1", "VM2", etc.). If Zabbix agent is installed on these hosts, and active Zabbix agent autoregistration is enabled, the autoregistration process will read host names as they were registered during launch (for example, "vm1.example.com", "vm2.example.com", etc.). This can lead to the creation of new hosts for existing virtual machines (since no name matches have been found), resulting in duplicate hosts with different names.

The following templates are used for discovered entities and, typically, should not be manually linked to a host:

- *VMware Hypervisor*
- *VMware Guest*

Host macros configuration

To use VMware simple checks, the host must have the following user macros defined:

- {$VMWARE.URL} - VMware service (vCenter or ESXi hypervisor) SDK URL (https://servername/sdk)
- {$VMWARE.USERNAME} - VMware service user name
- {$VMWARE.PASSWORD} - VMware service {$VMWARE.USERNAME} user password

Configuration examples

For a basic example of how to set up Zabbix for monitoring VMware using the *VMware FQDN* template, see *Monitor VMware with Zabbix*.

For a more detailed example of how to create a host, a low-level discovery rule, and a host prototype for monitoring VMware, see *Setup example*.

Extended logging The data collected by the *vmware collector* processes can be logged for detailed debugging using debug level 5. The debug level can be configured in the server and proxy configuration files or using the runtime control option `-R log_level_increase="vmware collector,N"`, where "N" is the process number.

For example, to increase the debug level from 4 to 5 for all *vmware collector* processes, run the following command:

```
zabbix_server -R log_level_increase="vmware collector"
```

To increase the debug level from 4 to 5 for the second *vmware collector* process, run the following command:

```
zabbix_server -R log_level_increase="vmware collector,2"
```

When extended logging of VMware collector data is no longer required, it is recommended to decrease the debug level to default (3) by running the `-R log_level_decrease` command.

Troubleshooting

- In case of unavailable metrics, please ensure that they are not made unavailable or turned off by default in recent VMware vSphere versions, or if some limits are not placed on performance-metric database queries. For more information, see ZBX-12094.
- If `config.vpxd.stats.maxQueryMetrics` is invalid or exceeds the maximum number of characters permitted error, add a `config.vpxd.stats.maxQueryMetrics` parameter to the vCenter Server settings. The value of this parameter should be the same as the value of `maxQuerysize` in VMware's *web.xml* file. For more information, see *VMware Knowledge Base* article.

1 VMware monitoring item keys

Overview This page provides details on the simple checks that can be used to monitor VMware environments. The metrics are grouped by the monitoring target.

Supported item keys The item keys are listed without parameters and additional information. Click on the item key to see the full details.

Item key	Description	Item group
vmware.eventlog	The VMware event log.	General service
vmware.fullname	The VMware service full name.	
vmware.version	The VMware service version.	
vmware.cl.perfcounter	The VMware cluster performance counter metrics.	Cluster
vmware.cluster.alarms.get	The VMware cluster alarms data.	
vmware.cluster.discovery	The discovery of VMware clusters.	
vmware.cluster.property	The VMware cluster property.	
vmware.cluster.status	The VMware cluster status.	
vmware.cluster.tags.get	The VMware cluster tags array.	
vmware.datastore.alarms.get	The VMware datastore alarms data.	Datastore
vmware.datastore.discovery	The discovery of VMware datastores.	
vmware.datastore.hv.list	The list of datastore hypervisors.	
vmware.datastore.perfcounter	The VMware datastore performance counter value.	
vmware.datastore.property	The VMware datastore property.	

Item key	Description	Item group
vmware.datastore.read	The amount of time for a read operation from the datastore.	
vmware.datastore.size	The VMware datastore space in bytes or in percentage from total.	
vmware.datastore.tags.get	The VMware datastore tags array.	
vmware.datastore.write	The amount of time for a write operation to the datastore.	
vmware.dc.alarms.get	The VMware datacenter alarms data.	Datacenter
vmware.dc.discovery	The discovery of VMware datacenters.	
vmware.dc.tags.get	The VMware datacenter tags array.	
vmware.dvswitch.discovery	The discovery of VMware vSphere Distributed Switches.	vSphere Distributed Switch
vmware.dvswitch.fetchports.get	The VMware vSphere Distributed Switch ports data.	
vmware.hv.alarms.get	The VMware hypervisor alarms data.	Hypervisor
vmware.hv.cluster.name	The VMware hypervisor cluster name.	
vmware.hv.connectionstate	The VMware hypervisor connection state.	
vmware.hv.cpu.usage	The VMware hypervisor processor usage (Hz).	
vmware.hv.cpu.usage.perf	The VMware hypervisor processor usage as a percentage during the interval.	
vmware.hv.cpu.utilization	The VMware hypervisor processor usage as a percentage during the interval, depends on power management or HT.	
vmware.hv.datacenter.name	The VMware hypervisor datacenter name.	
vmware.hv.datastore.discovery	The discovery of VMware hypervisor datastores.	
vmware.hv.datastore.list	The list of VMware hypervisor datastores.	
vmware.hv.datastore.multipath	The number of available datastore paths.	
vmware.hv.datastore.read	The average amount of time for a read operation from the datastore.	
vmware.hv.datastore.size	The VMware datastore space in bytes or in percentage from total.	
vmware.hv.datastore.write	The average amount of time for a write operation to the datastore.	
vmware.hv.discovery	The discovery of VMware hypervisors.	
vmware.hv.diskinfo.get	The VMware hypervisor disk data.	
vmware.hv.fullname	The VMware hypervisor name.	
vmware.hv.hw.cpu.freq	The VMware hypervisor processor frequency.	
vmware.hv.hw.cpu.model	The VMware hypervisor processor model.	
vmware.hv.hw.cpu.num	The number of processor cores on VMware hypervisor.	
vmware.hv.hw.cpu.threads	The number of processor threads on VMware hypervisor.	
vmware.hv.hw.memory	The VMware hypervisor total memory size.	
vmware.hv.hw.model	The VMware hypervisor model.	
vmware.hv.hw.sensors.get	The VMware hypervisor hardware sensors value.	
vmware.hv.hw.serialnumber	The VMware hypervisor serial number.	
vmware.hv.hw.uuid	The VMware hypervisor BIOS UUID.	
vmware.hv.hw.vendor	The VMware hypervisor vendor name.	
vmware.hv.maintenance	The VMware hypervisor maintenance status.	
vmware.hv.memory.size.ballooned	The VMware hypervisor ballooned memory size.	
vmware.hv.memory.used	The VMware hypervisor used memory size.	
vmware.hv.net.if.discovery	The discovery of VMware hypervisor network interfaces.	
vmware.hv.network.in	The VMware hypervisor network input statistics.	
vmware.hv.network.linkspeed	The VMware hypervisor network interface speed.	
vmware.hv.network.out	The VMware hypervisor network output statistics.	
vmware.hv.perfcounter	The VMware hypervisor performance counter value.	
vmware.hv.property	The VMware hypervisor property.	
vmware.hv.power	The VMware hypervisor power usage.	
vmware.hv.sensor.health.state	The VMware hypervisor health state rollup sensor.	
vmware.hv.sensors.get	The VMware hypervisor HW vendor state sensors.	
vmware.hv.status	The VMware hypervisor status.	
vmware.hv.tags.get	The VMware hypervisor tags array.	
vmware.hv.uptime	The VMware hypervisor uptime.	
vmware.hv.version	The VMware hypervisor version.	
vmware.hv.vm.num	The number of virtual machines on the VMware hypervisor.	
vmware.rp.cpu.usage	The CPU usage in hertz during the interval on VMware Resource Pool.	Resource pool
vmware.rp.memory	The memory metrics of VMware resource pool.	
vmware.alarms.get	The VMware virtual center alarms data.	Virtual center
vmware.vm.alarms.get	The VMware virtual machine alarms data.	Virtual machine

Item key	Description	Item group
vmware.vm.attribute	The VMware virtual machine custom attribute value.	
vmware.vm.cluster.name	The VMware virtual machine name.	
vmware.vm.consolidationneeded	The VMware virtual machine disk requires consolidation.	
vmware.vm.cpu.latency	The percentage of time the virtual machine is unable to run because it is contending for access to the physical CPU(s).	
vmware.vm.cpu.num	The number of processors on VMware virtual machine.	
vmware.vm.cpu.readiness	The percentage of time that the virtual machine was ready, but could not get scheduled to run on the physical CPU.	
vmware.vm.cpu.ready	The time that the virtual machine was ready, but could not get scheduled to run on the physical CPU.	
vmware.vm.cpu.swapwait	The percentage of CPU time spent waiting for swap-in.	
vmware.vm.cpu.usage	The VMware virtual machine processor usage (Hz).	
vmware.vm.cpu.usage.perf	The VMware virtual machine processor usage as a percentage during the interval.	
vmware.vm.datacenter.name	The VMware virtual machine datacenter name.	
vmware.vm.discovery	The discovery of VMware virtual machines.	
vmware.vm.guest.memory.swapped	The amount of guest physical memory that is swapped out to the swap space.	
vmware.vm.guest.osuptime	The total time elapsed since the last operating system boot-up.	
vmware.vm.hv.name	The VMware virtual machine hypervisor name.	
vmware.vm.memory.size	The VMware virtual machine total memory size.	
vmware.vm.memory.size.ballooned	The VMware virtual machine ballooned memory size.	
vmware.vm.memory.size.compressed	The VMware virtual machine compressed memory size.	
vmware.vm.memory.size.consumed	The amount of host physical memory consumed for backing up guest physical memory pages.	
vmware.vm.memory.size.private	The VMware virtual machine private memory size.	
vmware.vm.memory.size.shared	The VMware virtual machine shared memory size.	
vmware.vm.memory.size.swapped	The VMware virtual machine swapped memory size.	
vmware.vm.memory.size.usage.guest	The VMware virtual machine guest memory usage.	
vmware.vm.memory.size.usage.host	The VMware virtual machine host memory usage.	
vmware.vm.memory.usage	The percentage of host physical memory that has been consumed.	
vmware.vm.net.if.discovery	The discovery of VMware virtual machine network interfaces.	
vmware.vm.net.if.in	The VMware virtual machine network interface input statistics.	
vmware.vm.net.if.out	The VMware virtual machine network interface output statistics.	
vmware.vm.net.if.usage	The VMware virtual machine network utilization during the interval.	
vmware.vm.perfcounter	The VMware virtual machine performance counter value.	
vmware.vm.powerstate	The VMware virtual machine power state.	
vmware.vm.property	The VMware virtual machine property.	
vmware.vm.snapshot.get	The VMware virtual machine snapshot state.	
vmware.vm.state	The VMware virtual machine state.	
vmware.vm.storage.committed	The VMware virtual machine committed storage space.	
vmware.vm.storage.readoio	The average number of outstanding read requests to the virtual disk during the collection interval.	
vmware.vm.storage.totalreadlatency	The average time a read from the virtual disk takes.	
vmware.vm.storage.totalwritelatency	The average time a write to the virtual disk takes.	
vmware.vm.storage.uncommitted	The VMware virtual machine uncommitted storage space.	
vmware.vm.storage.unshared	The VMware virtual machine unshared storage space.	
vmware.vm.storage.writeoio	The average number of outstanding write requests to the virtual disk during the collection interval.	
vmware.vm.tags.get	The VMware virtual machine tags array.	
vmware.vm.tools	The VMware virtual machine guest tools state.	
vmware.vm.uptime	The VMware virtual machine uptime.	
vmware.vm.vfs.dev.discovery	The discovery of VMware virtual machine disk devices.	
vmware.vm.vfs.dev.read	The VMware virtual machine disk device read statistics.	
vmware.vm.vfs.dev.write	The VMware virtual machine disk device write statistics.	
vmware.vm.vfs.fs.discovery	The discovery of VMware virtual machine file systems.	
vmware.vm.vfs.fs.size	The VMware virtual machine file system statistics.	

Item key details Parameters without angle brackets are mandatory. Parameters marked with angle brackets **< >** are optional.

vmware.eventlog[url,<mode>]

 The VMware event log.
 Return value: *Log*.

Parameters:

- **url** - the VMware service URL;
- **mode** - *all* (default) or *skip* - skip the processing of older data.

Comments:

- There must be only one `vmware.eventlog` item key per URL;
- See also example of filtering VMware event log records.

vmware.fullnameurl

 The VMware service full name.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL.

vmware.versionurl

 The VMware service version.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL.

vmware.cl.perfcounter[url,id,path,<instance>]

 The VMware cluster performance counter metrics.
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **id** - the VMware cluster ID. `id` can be received from `vmware.cluster.discovery[]` as {#CLUSTER.ID}.
- **path** - the performance counter path[1];
- **instance** - the performance counter instance.

vmware.cluster.alarms.get[url,id]

 The VMware cluster alarms data.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **id** - the VMware cluster ID.

vmware.cluster.discoveryurl

 The discovery of VMware clusters.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL.

vmware.cluster.property[url,id,prop]

 The VMware cluster property.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **id** - the VMware cluster ID;
- **prop** - the property path which is the name of a property of the VM object as defined in the VMware SDK.

Examples:

```
vmware.vm.property[{$VMWARE.URL},{$VMWARE.VM.UUID},overallStatus]
vmware.vm.property[{$VMWARE.URL},{$VMWARE.VM.UUID},runtime.powerState]
```

vmware.cluster.status[url,name]

 The VMware cluster status.
 Return value: *0* - gray; *1* - green; *2* - yellow; *3* - red.

Parameters:

- **url** - the VMware service URL;
- **name** - the VMware cluster name.

vmware.cluster.tags.get[url,id]

 The VMware cluster tags array.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **id** - the VMware cluster ID.

This item works with vSphere 6.5 and newer.

vmware.datastore.alarms.get[url,uuid]

 The VMware datastore alarms data.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware datastore global unique identifier.

vmware.datastore.discoveryurl

 The discovery of VMware datastores.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL.

vmware.datastore.hv.list[url,datastore]

 The list of datastore hypervisors.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **datastore** - the datastore name.

Output example:

```
esx7-01-host.zabbix.sandbox
esx7-02-host.zabbix.sandbox
```

vmware.datastore.perfcounter[url,uuid,path,<instance>]

 The VMware datastore performance counter value.
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware datastore global unique identifier;
- **path** - the performance counter path[1];
- **instance** - the performance counter instance. Use empty instance for aggregate values (default).

vmware.datastore.property[url,uuid,prop]

 The VMware datastore property.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware datastore global unique identifier;
- **prop** - the property path.

vmware.datastore.read[url,datastore,<mode>]

 The amount of time for a read operation from the datastore (milliseconds).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **datastore** - the datastore name;
- **mode** - *latency* (average value, default) or *maxlatency* (maximum value).

vmware.datastore.size[url,datastore,<mode>]

 The VMware datastore space in bytes or in percentage from total.
 Return value: *Integer* - for bytes; *Float* - for percentage.

Parameters:

- **url** - the VMware service URL;

- **datastore** - the datastore name;
- **mode** - possible values: *total* (default), *free*, *pfree* (free percentage), *uncommitted*.

vmware.datastore.tags.get[url,uuid]

 The VMware datastore tags array.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware datastore global unique identifier.

This item works with vSphere 6.5 and newer.

vmware.datastore.write[url,datastore,<mode>]

 The amount of time for a write operation to the datastore (milliseconds).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **datastore** - the datastore name;
- **mode** - *latency* (average value, default) or *maxlatency* (maximum value).

vmware.dc.alarms.get[url,id]

 The VMware datacenter alarms data.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **id** - the VMware datacenter ID.

vmware.dc.discoveryurl

 The discovery of VMware datacenters.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL.

vmware.dc.tags.get[url,id]

 The VMware datacenter tags array.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **id** - the VMware datacenter ID.

This item works with vSphere 6.5 and newer.

vmware.dvswitch.discoveryurl

 The discovery of VMware vSphere Distributed Switches.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL.

vmware.dvswitch.fetchports.get[url,uuid,<filter>,<mode>]

 The VMware vSphere Distributed Switch ports data.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware DVSwitch global unique identifier.
- **filter** - a single string with comma-separated criteria for selecting ports;
- **mode** - *state* (all XML without ”config” XML nodes, default) or *full*.

The **filter** parameter supports the criteria available in the VMware data object DistributedVirtualSwitchPortCriteria.

Example:

```
vmware.dvswitch.fetchports.get[{$VMWARE.URL},{$VMWARE.DVS.UUID},"active:true,connected:false,host:host-18,
```

vmware.hv.alarms.get[url,uuid]

 The VMware hypervisor alarms data.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.cluster.name[url,uuid]

 The VMware hypervisor cluster name.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.connectionstate[url,uuid]

 The VMware hypervisor connection state.
 Return value: *String*: *connected*, *disconnected*, or *notResponding*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.cpu.usage[url,uuid]

 The VMware hypervisor processor usage (Hz).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.cpu.usage.perf[url,uuid]

 The VMware hypervisor processor usage as a percentage during the interval.
 Return value: *Float*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.cpu.utilization[url,uuid]

 The VMware hypervisor processor usage as a percentage during the interval, depends on power management or HT.
 Return value: *Float*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.datacenter.name[url,uuid]

 The VMware hypervisor datacenter name.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.datastore.discovery[url,uuid]

 The discovery of VMware hypervisor datastores.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.datastore.list[url,uuid]

 The list of VMware hypervisor datastores.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;

- **uuid** - the VMware hypervisor global unique identifier.

Output example:

```
SSD-RAID1-VAULT1
SSD-RAID1-VAULT2
SSD-RAID10
```

vmware.hv.datastore.multipath[url,uuid,<datastore>,<partitionid>]

 The number of available datastore paths.
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier;
- **datastore** - the datastore UUID or name;
- **partitionid** - the internal ID of physical device from `vmware.hv.datastore.discovery`.

vmware.hv.datastore.read[url,uuid,datastore,<mode>]

 The average amount of time for a read operation from the datastore (milliseconds).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier;
- **datastore** - the datastore UUID or name;
- **mode** - *latency* (default).

vmware.hv.datastore.size[url,uuid,datastore,<mode>]

 The VMware datastore space in bytes or in percentage from total.
 Return value: *Integer* - for bytes; *Float* - for percentage.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier;
- **datastore** - the datastore UUID or name;
- **mode** - possible values: *total* (default), *free*, *pfree* (free percentage), *uncommitted*.

vmware.hv.datastore.write[url,uuid,datastore,<mode>]

 The average amount of time for a write operation to the datastore (milliseconds).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier;
- **datastore** - the datastore UUID or name;
- **mode** - *latency* (default).

vmware.hv.discoveryurl

 The discovery of VMware hypervisors.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL.

vmware.hv.diskinfo.get[url,uuid]

 The VMware hypervisor disk data.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.fullname[url,uuid]

 The VMware hypervisor name.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.hw.cpu.freq[url,uuid]

 The VMware hypervisor processor frequency (Hz).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.hw.cpu.model[url,uuid]

 The VMware hypervisor processor model.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.hw.cpu.num[url,uuid]

 The number of processor cores on VMware hypervisor.
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.hw.cpu.threads[url,uuid]

 The number of processor threads on VMware hypervisor.
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.hw.memory[url,uuid]

 The VMware hypervisor total memory size (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.hw.model[url,uuid]

 The VMware hypervisor model.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.hw.sensors.get[url,uuid]

 The VMware hypervisor hardware sensors value.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.hw.serialnumber[url,uuid]

 The VMware hypervisor serial number.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

This item works with vSphere API 6.7 and newer.

vmware.hv.hw.uuid[url,uuid]

 The VMware hypervisor BIOS UUID.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;

- **uuid** - the VMware hypervisor global unique identifier.

This item works with vSphere API 6.7 and newer.

vmware.hv.hw.vendor[url,uuid]

 The VMware hypervisor vendor name.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

This item works with vSphere API 6.7 and newer.

vmware.hv.maintenance[url,uuid]

 The VMware hypervisor maintenance status.
 Return value: *0* - not in maintenance; *1* - in maintenance.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.memory.size.ballooned[url,uuid]

 The VMware hypervisor ballooned memory size (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.memory.used[url,uuid]

 The VMware hypervisor used memory size (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.net.if.discovery[url,uuid]

 The discovery of VMware hypervisor network interfaces.
 Return value: *JSON object*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.network.in[url,uuid,<mode>]

 The VMware hypervisor network input statistics (bytes per second).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier;
- **mode** - *bps* (default), *packets*, *dropped*, *errors*, *broadcast*.

vmware.hv.network.linkspeed[url,uuid,ifname]

 The VMware hypervisor network interface speed.
 Return value: *Integer*. Returns *0*, if the network interface is down, otherwise the speed value of the interface.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier;
- **ifname** - the interface name.

vmware.hv.network.out[url,uuid,<mode>]

 The VMware hypervisor network output statistics (bytes per second).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier;

- **mode** - *bps* (default), *packets*, *dropped*, *errors*, *broadcast*.

vmware.hv.perfcounter[url,uuid,path,<instance>]

 The VMware hypervisor performance counter value.
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier;
- **path** - the performance counter path [1];
- **instance** - the performance counter instance. Use empty instance for aggregate values (default).

vmware.hv.property[url,uuid,prop]

 The VMware hypervisor property.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier;
- **prop** - the property path.

vmware.hv.power[url,uuid,<max>]

 The VMware hypervisor power usage (W).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier;
- **max** - the maximum allowed power usage.

vmware.hv.sensor.health.state[url,uuid]

 The VMware hypervisor health state rollup sensor.
 Return value: *Integer*: *0* - gray; *1* - green; *2* - yellow; *3* - red.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

Note that the item might not work in VMware vSphere 6.5 and newer, because VMware has deprecated the *VMware Rollup Health State* sensor.

vmware.hv.sensors.get[url,uuid]

 The VMware hypervisor HW vendor state sensors.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.status[url,uuid]

 The VMware hypervisor status.
 Return value: *Integer*: *0* - gray; *1* - green; *2* - yellow; *3* - red.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

This item uses the host system overall status property.

vmware.hv.tags.get[url,uuid]

 The VMware hypervisor tags array.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

This item works with vSphere 6.5 and newer.

vmware.hv.uptime[url,uuid]

 The VMware hypervisor uptime (seconds).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

This item uses the host system overall status property.

vmware.hv.version[url,uuid]

 The VMware hypervisor version.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.hv.vm.num[url,uuid]

 The number of virtual machines on the VMware hypervisor.
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware hypervisor global unique identifier.

vmware.rp.cpu.usage[url,rpid]

 The CPU usage in hertz during the interval on VMware Resource Pool.
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **rpid** - the VMware resource pool ID.

vmware.rp.memory[url,rpid,<mode>]

 The memory metrics of VMware resource pool.
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **rpid** - the VMware resource pool ID;
- **mode** - possible values:
consumed (default) - the amount of host physical memory consumed for backing up guest physical memory pages
 ballooned - the amount of guest physical memory reclaimed from the virtual machine by the balloon driver in the guest
overhead - the host physical memory consumed by ESXi data structures for running the virtual machines

vmware.alarms.geturl

 The VMware virtual center alarms data.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL.

vmware.vm.alarms.get[url,uuid]

 The VMware virtual machine alarms data.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.attribute[url,uuid,name]

 The VMware virtual machine custom attribute value.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **name** - the custom attribute name.

vmware.vm.cluster.name[url,uuid]

 The VMware virtual machine name.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **name** - the custom attribute name.

vmware.vm.consolidationneeded[url,uuid]

 The VMware virtual machine disk requires consolidation.
 Return value: *String*: *true* - consolidation is needed; *false* - consolidation is not needed.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.cpu.latency[url,uuid]

 The percentage of time the virtual machine is unable to run because it is contending for access to the physical CPU(s).
 Return value: *Float*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.cpu.num[url,uuid]

 The number of processors on VMware virtual machine.
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.cpu.readiness[url,uuid,<instance>]

 The percentage of time that the virtual machine was ready, but could not get scheduled to run on the physical CPU.
 Return value: *Float*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the CPU instance.

vmware.vm.cpu.ready[url,uuid]

 The time (in milliseconds) that the virtual machine was ready, but could not get scheduled to run on the physical CPU. CPU ready time is dependent on the number of virtual machines on the host and their CPU loads (%).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.cpu.swapwait[url,uuid,<instance>]

 The percentage of CPU time spent waiting for swap-in.
 Return value: *Float*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the CPU instance.

vmware.vm.cpu.usage[url,uuid]

 The VMware virtual machine processor usage (Hz).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.cpu.usage.perf[url,uuid]

 The VMware virtual machine processor usage as a percentage during the interval.
 Return value: *Float*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.datacenter.name[url,uuid]

 The VMware virtual machine datacenter name.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.discoveryurl

 The discovery of VMware virtual machines.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL.

vmware.vm.guest.memory.size.swapped[url,uuid]

 The amount of guest physical memory that is swapped out to the swap space (KB).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.guest.osuptime[url,uuid]

 The total time elapsed since the last operating system boot-up (in seconds).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.hv.name[url,uuid]

 The VMware virtual machine hypervisor name.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.memory.size[url,uuid]

 The VMware virtual machine total memory size (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.memory.size.ballooned[url,uuid]

 The VMware virtual machine ballooned memory size (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.memory.size.compressed[url,uuid]

 The VMware virtual machine compressed memory size (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.memory.size.consumed[url,uuid]

 The amount of host physical memory consumed for backing up guest physical memory pages (KB).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.memory.size.private[url,uuid]

 The VMware virtual machine private memory size (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.memory.size.shared[url,uuid]

 The VMware virtual machine shared memory size (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.memory.size.swapped[url,uuid]

 The VMware virtual machine swapped memory size (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.memory.size.usage.guest[url,uuid]

 The VMware virtual machine guest memory usage (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.memory.size.usage.host[url,uuid]

 The VMware virtual machine host memory usage (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.memory.usage[url,uuid]

 The percentage of host physical memory that has been consumed.
 Return value: *Float*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.net.if.discovery[url,uuid]

 The discovery of VMware virtual machine network interfaces.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.net.if.in[url,uuid,instance,<mode>]

 The VMware virtual machine network interface input statistics (bytes/packets per second).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the network interface instance;
- **mode** - *bps* (default) or *pps* - bytes or packets per second.

vmware.vm.net.if.out[url,uuid,instance,<mode>]

 The VMware virtual machine network interface output statistics (bytes/packets per second).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the network interface instance;
- **mode** - *bps* (default) or *pps* - bytes or packets per second.

vmware.vm.net.if.usage[url,uuid,<instance>]

 The VMware virtual machine network utilization (combined transmit-rates and receive-rates) during the interval (KBps).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the network interface instance.

vmware.vm.perfcounter[url,uuid,path,<instance>]

 The VMware virtual machine performance counter value.
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **path** - the performance counter path [1];
- **instance** - the performance counter instance. Use empty instance for aggregate values (default).

vmware.vm.powerstate[url,uuid]

 The VMware virtual machine power state.
 Return value: *0* - poweredOff; *1* - poweredOn; *2* - suspended.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.property[url,uuid,prop]

 The VMware virtual machine property.
 Return value: *String*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **prop** - the property path.

vmware.vm.snapshot.get[url,uuid]

 The VMware virtual machine snapshot state.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.state[url,uuid]

 The VMware virtual machine state.
 Return value: *String*: *notRunning*, *resetting*, *running*, *shuttingDown*, *standby*, or *unknown*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.storage.committed[url,uuid]

 The VMware virtual machine committed storage space (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;

- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.storage.readoio[url,uuid,instance]

 The average number of outstanding read requests to the virtual disk during the collection interval.
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the disk device instance.

vmware.vm.storage.totalreadlatency[url,uuid,instance]

 The average time a read from the virtual disk takes (milliseconds).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the disk device instance.

vmware.vm.storage.totalwritelatency[url,uuid,instance]

 The average time a write to the virtual disk takes (milliseconds).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the disk device instance.

vmware.vm.storage.uncommitted[url,uuid]

 The VMware virtual machine uncommitted storage space (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.storage.unshared[url,uuid]

 The VMware virtual machine unshared storage space (bytes).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.storage.writeoio[url,uuid,instance]

 The average number of outstanding write requests to the virtual disk during the collection interval.
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the disk device instance.

vmware.vm.tags.get[url,uuid]

 The VMware virtual machine tags array.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

This item works with vSphere 6.5 and newer.

vmware.vm.tools[url,uuid,mode]

 The VMware virtual machine guest tools state.
 Return value: *String*: *guestToolsExecutingScripts* - VMware Tools is starting; *guestToolsNotRunning* - VMware Tools is not running; *guestToolsRunning* - VMware Tools is running.

Parameters:

- **url** - the VMware service URL;

- **uuid** - the VMware virtual machine global unique identifier;
- **mode** - *version*, *status*.

vmware.vm.uptime[url,uuid]

 The VMware virtual machine uptime (seconds).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.vfs.dev.discovery[url,uuid]

 The discovery of VMware virtual machine disk devices.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

vmware.vm.vfs.dev.read[url,uuid,instance,<mode>]

 The VMware virtual machine disk device read statistics (bytes/operations per second).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the disk device instance;
- **mode** - *bps* (default) or *ops* - bytes or operations per second.

vmware.vm.vfs.dev.write[url,uuid,instance,<mode>]

 The VMware virtual machine disk device write statistics (bytes/operations per second).
 Return value: *Integer* [2].

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **instance** - the disk device instance;
- **mode** - *bps* (default) or *ops* - bytes or operations per second.

vmware.vm.vfs.fs.discovery[url,uuid]

 The discovery of VMware virtual machine file systems.
 Return value: JSON object.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier.

VMware Tools must be installed on the guest virtual machine for this item to work.

vmware.vm.vfs.fs.size[url,uuid,fsname,<mode>]

 The VMware virtual machine file system statistics (bytes/percentages).
 Return value: *Integer*.

Parameters:

- **url** - the VMware service URL;
- **uuid** - the VMware virtual machine global unique identifier;
- **fsname** - the file system name;
- **mode** - *total*, *free*, *used*, *pfree*, or *pused*.

VMware Tools must be installed on the guest virtual machine for this item to work.

Footnotes

[1] See Creating custom performance counter names for VMware.

[2] The value of these items is obtained from VMware performance counters and the VMwarePerfFrequency parameter is used to refresh their data in Zabbix VMware cache:

- vmware.cl.perfcounter
- vmware.hv.datastore.read
- vmware.hv.datastore.write
- vmware.hv.network.in

- vmware.hv.network.out
- vmware.hv.perfcounter
- vmware.vm.cpu.ready
- vmware.vm.net.if.in
- vmware.vm.net.if.out
- vmware.vm.perfcounter
- vmware.vm.vfs.dev.read
- vmware.vm.vfs.dev.write

More info

See Virtual machine monitoring for detailed information how to configure Zabbix to monitor VMware environments.

2 Virtual machine discovery key fields

The following table lists fields returned by virtual machine related discovery keys.

Item key		
Description	**Field**	**Retrieved content**
vmware.cluster.discovery		
Performs cluster discovery.	{#CLUSTER.ID}	Cluster identifier.
	{#CLUSTER.NAME}	Cluster name.
	”resource_pool”	An array containing resource pool data, including resource group ID, tags array, resource pool path, number of virtual machines: [{”rpid”:”resource group id”, ”tags”:[{}], ”rpath”:”resource group path”, ”vm_count”:0}]
	”tags”	For tags array structure, see ”tags” field. An array containing tags with tag name, description and category: [{”tag”:”tag name”, ”tag_description”:”tag description”, ”category”:”tag category”}]
vmware.datastore.discovery		
Performs datastore discovery.	{#DATASTORE}	Datastore name.
	{#DATASTORE.EXTENT}	An array containing datastore extent partition ID and instance name: [{”partitionid”:1, ”instance”:”name”}]
	{#DATASTORE.TYPE}	Datastore type. Value examples: `VMFS`, `NFS`, `vsan`, etc.
	{#DATASTORE.UUID}	Datastore identifier.
	”tags”	An array containing tags with tag name, description and category: [{”tag”:”tag name”, ”tag_description”:”tag description”, ”category”:”tag category”}]
vmware.dc.discovery		
Performs datacenter discovery.	{#DATACENTER}	Datacenter name.
	{#DATACENTERID}	Datacenter identifier.
	”tags”	An array containing tags with tag name, description and category: [{”tag”:”tag name”, ”tag_description”:”tag description”, ”category”:”tag category”}]
vmware.dvswitch.discovery		

Item key		
Performs vSphere distributed switches discovery.	{#DVS.NAME}	Switch name.
	{#DVS.UUID}	Switch identifier.
vmware.hv.discovery		
Performs hypervisor discovery.	{#HV.UUID}	Unique hypervisor identifier.
	{#HV.ID}	Hypervisor identifier (HostSystem managed object name).
	{#HV.NAME}	Hypervisor name.
	{#HV.NETNAME}	Hypervisor network host name.
	{#HV.IP}	Hypervisor IP address, might be empty. In case of an HA configuration with multiple net interfaces, the following selection priority for interface is observed: - prefer the IP which shares the IP-subnet with the vCenter IP - prefer the IP from IP-subnet with default gateway - prefer the IP from interface with the lowest ID This field is supported since Zabbix 5.2.2.
	{#CLUSTER.NAME}	Cluster name, might be empty.
	{#DATACENTER.NAME}	Datacenter name.
	{#PARENT.NAME}	Name of container that stores the hypervisor. Supported since Zabbix 4.0.3.
	{#PARENT.TYPE}	Type of container in which the hypervisor is stored. The values could be `Datacenter`, `Folder`, `ClusterComputeResource`, `VMware`, where ”VMware” stands for unknown container type. Supported since Zabbix 4.0.3.
	”resource_pool”	An array containing resource pool data, including resource group ID, tags array, resource pool path, number of virtual machines: [{”rpid”:”resource group id”, ”tags”:[{}], ”rpath”:”resource group path”, ”vm_count”:0}] For tags array structure, see ”tags” field.
	”tags”	An array containing tags with tag name, description and category: [{”tag”:”tag name”, ”tag_description”:”tag description”, ”category”:”tag category”}]
vmware.hv.datastore.discovery		
Performs hypervisor datastore discovery. Note that multiple hypervisors can use the same datastore.	{#DATASTORE}	Datastore name.
	{#DATASTORE.TYPE}	Datastore type. Value examples: `VMFS`, `NFS`, `vsan`, etc.
	{#DATASTORE.UUID}	Datastore identifier.
	{#MULTIPATH.COUNT}	Registered number of datastore paths.
	{#MULTIPATH.PARTITION.COUNT}	Number of available disk partitions.
	”datastore_extent”	An array containing datastore extent instance name and partition ID: [{”partitionid”:1, ”instance”:”name”}]
	”tags”	An array containing tags with tag name, description and category: [{”tag”:”tag name”, ”tag_description”:”tag description”, ”category”:”tag category”}]
vmware.hv.net.if.discovery		
Performs hypervisor network interfaces discovery.	{#IFNAME}	Interface name.
	{#IFDRIVER}	Interface driver.

Item key		
	{#IFDUPLEX}	Interface duplex settings.
	{#IFSPEED}	Interface speed.
	{#IFMAC}	Interface mac address.
vmware.vm.discovery		
Performs virtual machine discovery.	{#VM.UUID}	Unique virtual machine identifier.
	{#VM.ID}	Virtual machine identifier (VirtualMachine managed object name).
	{#VM.NAME}	Virtual machine name.
	{#HV.NAME}	Hypervisor name.
	{#HV.UUID}	Unique hypervisor identifier.
	{#HV.ID}	Hypervisor identifier (HostSystem managed object name).
	{#CLUSTER.NAME}	Cluster name, might be empty.
	{#DATACENTER.NAME}	Datacenter name.
	{#DATASTORE.NAME}	Datastore name.
	{#DATASTORE.UUID}	Datastore identifier.
	{#VM.IP}	Virtual machine IP address, might be empty.
	{#VM.DNS}	Virtual machine DNS name, might be empty.
	{#VM.GUESTFAMILY}	Guest virtual machine OS family, might be empty.
	{#VM.GUESTFULLNAME}	Full guest virtual machine OS name, might be empty.
	{#VM.FOLDER}	The chain of virtual machine parent folders, can be used as value for nested groups; folder names are combined with "/". Might be empty.
	{#VM.TOOLS.STATUS}	VMware virtual machine tools state.
	{#VM.POWERSTATE}	VMware virtual machine power state (`poweredOff`, `poweredOn`, or `suspended`).
	{#VM.RPOOL.ID}	Resource pool identifier.
	{#VM.RPOOL.PATH}	Full resource pool path excluding the "root" name "Resources". Folder names are combined with "/".
	{#VM.SNAPSHOT.COUNT}	Number of VM snapshots.
	"tags"	An array containing tags with tag name, description and category: [{"tag":"tag name", "tag_description":"tag description", "category":"tag category"}]
	"vm_customattribute"	An array of virtual machine custom attributes (if defined): [{"name":"custom field name", "value":"custom field value"}]
vmware.vm.net.if.discovery		
Performs virtual machine network interface discovery.	{#IFNAME}	Network interface name.
	{#IFDESC}	Interface description.
	{#IFMAC}	Interface mac address.
	{#IFCONNECTED}	Interface connection status (`false` - disconnected; `true` - connected).
	{#IFTYPE}	Interface type.
	{#IFBACKINGDEVICE}	Name of the backing device.
	{#IFDVSWITCH.UUID}	Unique vSphere Distributed Switch identifier.
	{#IFDVSWITCH.PORTGROUP}	Distributed port group.
	{#IFDVSWITCH.PORT}	vSphere Distributed Switch port.
vmware.vm.vfs.dev.discovery		
Performs virtual machine disk device discovery.	{#DISKNAME}	Disk device name.
vmware.vm.vfs.fs.discovery		
Performs virtual machine file system discovery.	{#FSNAME}	File system name.

3 JSON examples for VMware items

Overview This section provides additional information about JSON objects returned by various VMware items.

vmware.*.alarms.get The items **vmware.alarms.get[]**, **vmware.cluster.alarms.get[]**, **vmware.datastore.alarms.get[]**, **vmware.dc.alarms.get[]**, **vmware.hv.alarms.get[]**, **vmware.vm.alarms.get[]** return JSON objects with the following structure (values are provided as an example):

```
{
    "alarms": [
        {
            "name": "Host connection and power state",
            "system_name": "alarm.HostConnectionStateAlarm",
            "description": "Default alarm to monitor host connection and power state",
            "enabled": true,
            "key": "alarm-1.host-2013",
            "time": "2022-06-27T05:27:38.759976Z",
            "overall_status": "red",
            "acknowledged": false
        },
        {
            "name": "Host memory usage",
            "system_name": "alarm.HostMemoryUsageAlarm",
            "description": "Default alarm to monitor host memory usage",
            "enabled": true,
            "key": "alarm-4.host-1004",
            "time": "2022-05-16T13:32:42.47863Z",
            "overall_status": "yellow",
            "acknowledged": false
        },
        {
            // other alarms
        }
    ]
}
```

vmware.*.tags.get The items **vmware.cluster.tags.get[]**, **vmware.datastore.tags.get[]**, **vmware.dc.tags.get[]**, **vmware.hv.tags.get[]**, **vmware.vm.tags.get[]** return JSON objects with the following structure (values are provided as an example):

```
{
    "tags": [
        {
            "name": "Windows",
        "description": "tag for cat OS type",
        "category": "OS type"
    },
    {
            "name": "SQL Server",
        "description": "tag for cat application name",
        "category": "application name"
        },
    {
            // other tags
        }
    ]
}
```

vmware.hv.diskinfo.get The item **vmware.hv.diskinfo.get[]** returns JSON objects with the following structure (values are provided as an example):

```
[
  {
    "instance": "mpx.vmhba32:C0:T0:L0",
    "hv_uuid": "8002299e-d7b9-8728-d224-76004bbb6100",
    "datastore_uuid": "",
    "operational_state": [
      "ok"
    ],
    "lun_type": "disk",
    "queue_depth": 1,
    "model": "USB DISK",
    "vendor": "SMI Corp",
    "revision": "1100",
    "serial_number": "CCYYMMDDHHmmSS9S62CK",
    "vsan": {}
  },
  {
    // other instances
  }
]
```


vmware.dvswitch.fetchports.get The item **vmware.dvswitch.fetchports.get[]** returns JSON objects with the following structure (values are provided as an example):


```
{
    "FetchDVPortsResponse":
    {
        "returnval": [
            {
                "key": "0",
                "dvsUuid": "50 36 6a 24 25 c0 10 9e-05 4a f6 ea 4e 3d 09 88",
                "portgroupKey": "dvportgroup-2023",
                "proxyHost":
                {
                    "@type": "HostSystem",
                    "#text": "host-2021"
                },
                "connectee":
                {
                    "connectedEntity":
                    {
                        "@type": "HostSystem",
                        "#text": "host-2021"
                    },
                    "nicKey": "vmnic0",
                    "type": "pnic"
                },
                "conflict": "false",
                "state":
                {
                    "runtimeInfo":
                    {
                        "linkUp": "true",
                        "blocked": "false",
                        "vlanIds":
                        {
                            "start": "0",
                            "end": "4094"
                        },
                        "trunkingMode": "true",
                        "linkPeer": "vmnic0",
                        "macAddress": "00:00:00:00:00:00",
                        "statusDetail": null,
```

```
                        "vmDirectPathGen2Active": "false",
                        "vmDirectPathGen2InactiveReasonOther": "portNptIncompatibleConnectee"
                    },
                    "stats":
                    {
                        "packetsInMulticast": "2385470",
                        "packetsOutMulticast": "45",
                        "bytesInMulticast": "309250248",
                        "bytesOutMulticast": "5890",
                        "packetsInUnicast": "155601537",
                        "packetsOutUnicast": "113008658",
                        "bytesInUnicast": "121609489384",
                        "bytesOutUnicast": "47240279759",
                        "packetsInBroadcast": "1040420",
                        "packetsOutBroadcast": "7051",
                        "bytesInBroadcast": "77339771",
                        "bytesOutBroadcast": "430392",
                        "packetsInDropped": "0",
                        "packetsOutDropped": "0",
                        "packetsInException": "0",
                        "packetsOutException": "0"
                    }
                },
                "connectionCookie": "1702765133",
                "lastStatusChange": "2022-03-25T14:01:11Z",
                "hostLocalPort": "false"
            },
            {
                //other keys
            }
        ]
    }
}
```

vmware.hv.hw.sensors.get The item **vmware.hv.hw.sensors.get[]** returns JSON objects with the following structure (values are provided as an example):

```
{
    "val":
        {
            "@type": "HostHardwareStatusInfo",
            "storageStatusInfo": [
                {
                    "name": "Intel Corporation HD Graphics 630 #2",
                "status":
                        {
                            "label": "Unknown",
                    "summary": "Cannot report on the current status of the physical element",
                    "key": "Unknown"
                }
            },
            {
                    "name": "Intel Corporation 200 Series/Z370 Chipset Family USB 3.0 xHCI Controller #20"
                "status":
                        {
                     "label": "Unknown",
                     "summary": "Cannot report on the current status of the physical element",
                     "key": "Unknown"
                 }
            },
            {
                    // other hv hw sensors
                }
```

```
        ]
        }
}
```

vmware.hv.sensors.get The item **vmware.hv.sensors.get[]** returns JSON objects with the following structure (values are provided as an example):

```
{
    "val":
        {
            "@type": "ArrayOfHostNumericSensorInfo", "HostNumericSensorInfo": [
                {
                    "@type": "HostNumericSensorInfo",
                    "name": "System Board 1 PwrMeter Output --- Normal",
                    "healthState":
                        {
                            "label": "Green",
                            "summary": "Sensor is operating under normal conditions",
                    "key": "green"
                },
                    "currentReading": "10500",
                "unitModifier": "-2",
                "baseUnits": "Watts",
                "sensorType": "other"
            },
                {
                    "@type": "HostNumericSensorInfo",
                    "name": "Power Supply 1 PS 1 Output --- Normal",
                    "healthState":
                        {
                            "label": "Green",
                    "summary": "Sensor is operating under normal conditions",
                    "key": "green"
                },
                    "currentReading": "10000",
            "unitModifier": "-2",
            "baseUnits": "Watts",
            "sensorType": "power"
        },
            {
                    // other hv sensors
                }
        ]
        }
}
```

vmware.vm.snapshot.get If any snapshots exist, the item **vmware.snapshot.get[]** returns a JSON object with the following structure (values are provided as an example):

```
{
  "snapshot": [
    {
      "name": "VM Snapshot 4%2f1%2f2022, 9:16:39 AM",
      "description": "Descr 1",
      "createtime": "2022-04-01T06:16:51.761Z",
      "size": 5755795171,
      "uniquesize": 5755795171
    },
    {
      "name": "VM Snapshot 4%2f1%2f2022, 9:18:21 AM",
      "description": "Descr 2",
      "createtime": "2022-04-01T06:18:29.164999Z",
      "size": 118650595,
```

```
      "uniquesize": 118650595
    },
    {
      "name": "VM Snapshot 4%2f1%2f2022, 9:37:29 AM",
      "description": "Descr 3",
      "createtime": "2022-04-01T06:37:53.534999Z",
      "size": 62935016,
      "uniquesize": 62935016
    }
  ],
  "count": 3,
  "latestdate": "2022-04-01T06:37:53.534999Z",
  "latestage": 22729203,
  "oldestdate": "2022-04-01T06:16:51.761Z",
  "oldestage": 22730465,
  "size": 5937380782,
  "uniquesize": 5937380782
}
```


If no snapshot exists, the item **vmware.snapshot.get[]** returns a JSON object with empty values:


```
{
  "snapshot": [],
  "count": 0,
  "latestdate": null,
  "latestage": 0,
  "oldestdate": null,
  "oldestage": 0,
  "size": 0,
  "uniquesize": 0
}
```


4 VMware monitoring setup example

Overview

The following example describes how to set up Zabbix for monitoring VMware virtual machines. This involves:

- creating a host that represents your VMware environment;
- creating a low-level discovery rule that discovers virtual machines in your VMware environment;
- creating a host prototype, based on which Zabbix will generate real hosts for virtual machines discovered by the low-level discovery rule.

Prerequisites

> **Note:**
> This example does not cover the configuration of VMware. It is assumed that VMware is already configured.

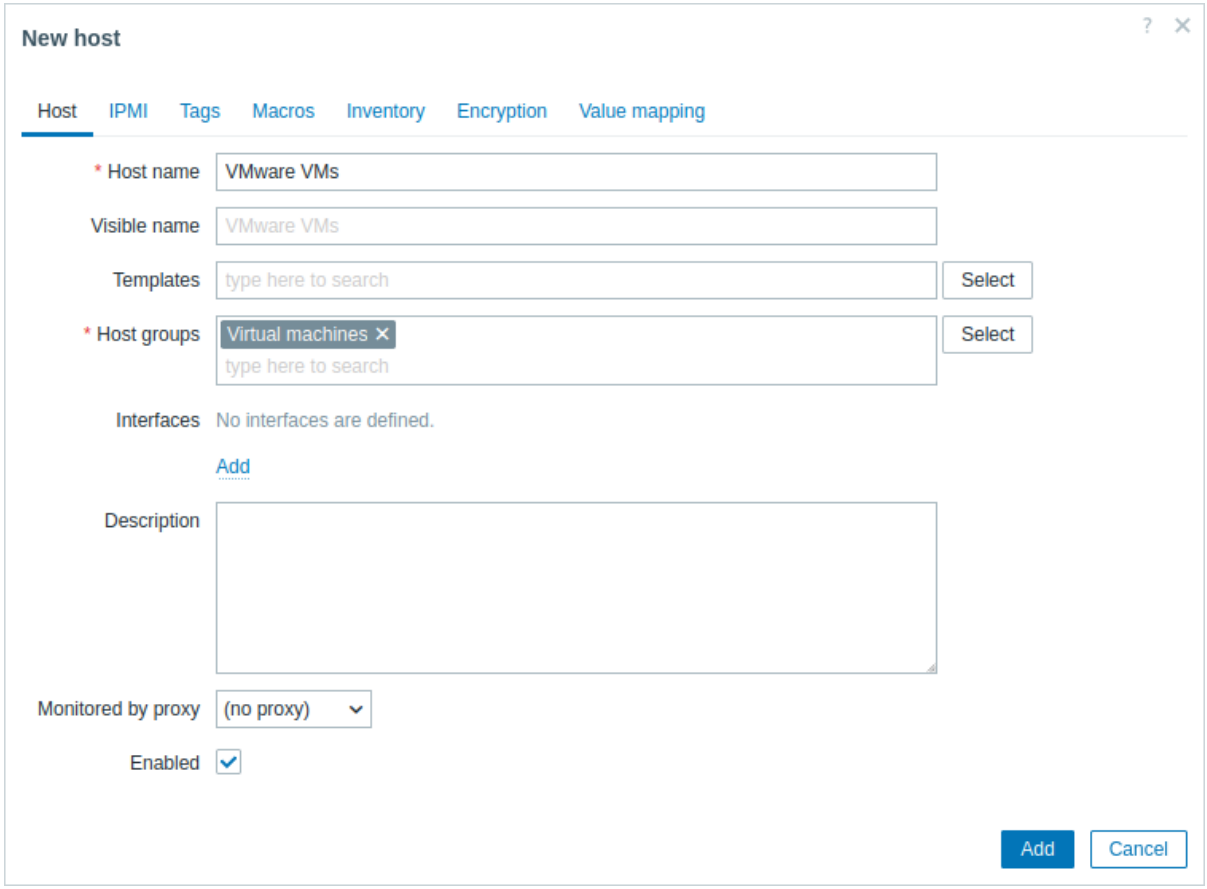
Before proceeding, set the `StartVMwareCollectors` parameter in Zabbix server configuration file to 2 or more (the default value is 0).

Create a host

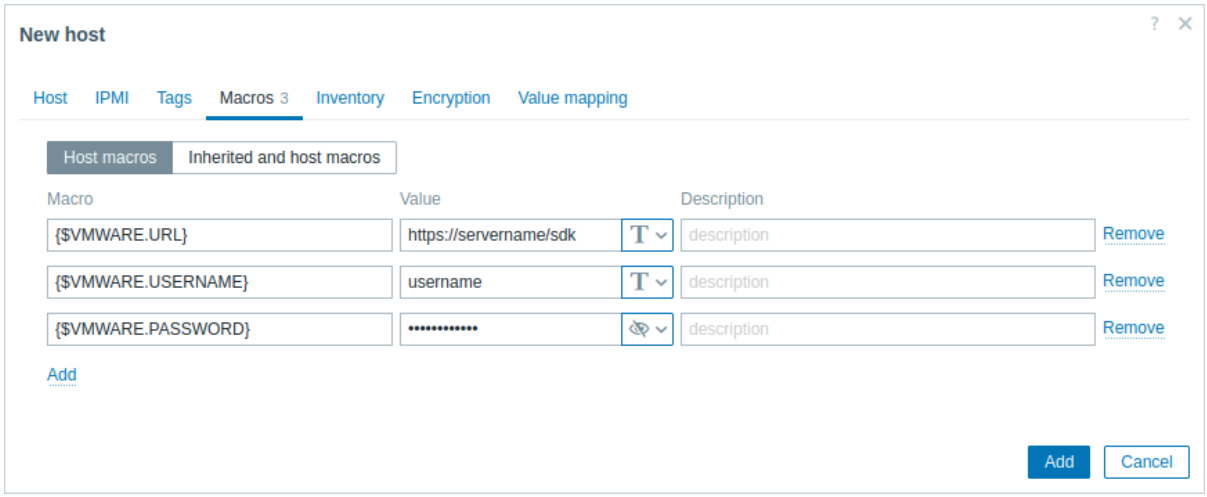
1. Go to *Data collection* → *Hosts*.

2. Create a host:

- In the *Host name* field, enter a host name (for example, "VMware VMs").
- In the *Host groups* field, type or select a host group (for example, "Virtual machines").



- In the *Macros* tab, set the following host macros:
  - {$VMWARE.URL} - VMware service (ESXi hypervisor) SDK URL (https://servername/sdk)
  - {$VMWARE.USERNAME} - VMware service user name
  - {$VMWARE.PASSWORD} - VMware service {$VMWARE.USERNAME} user password



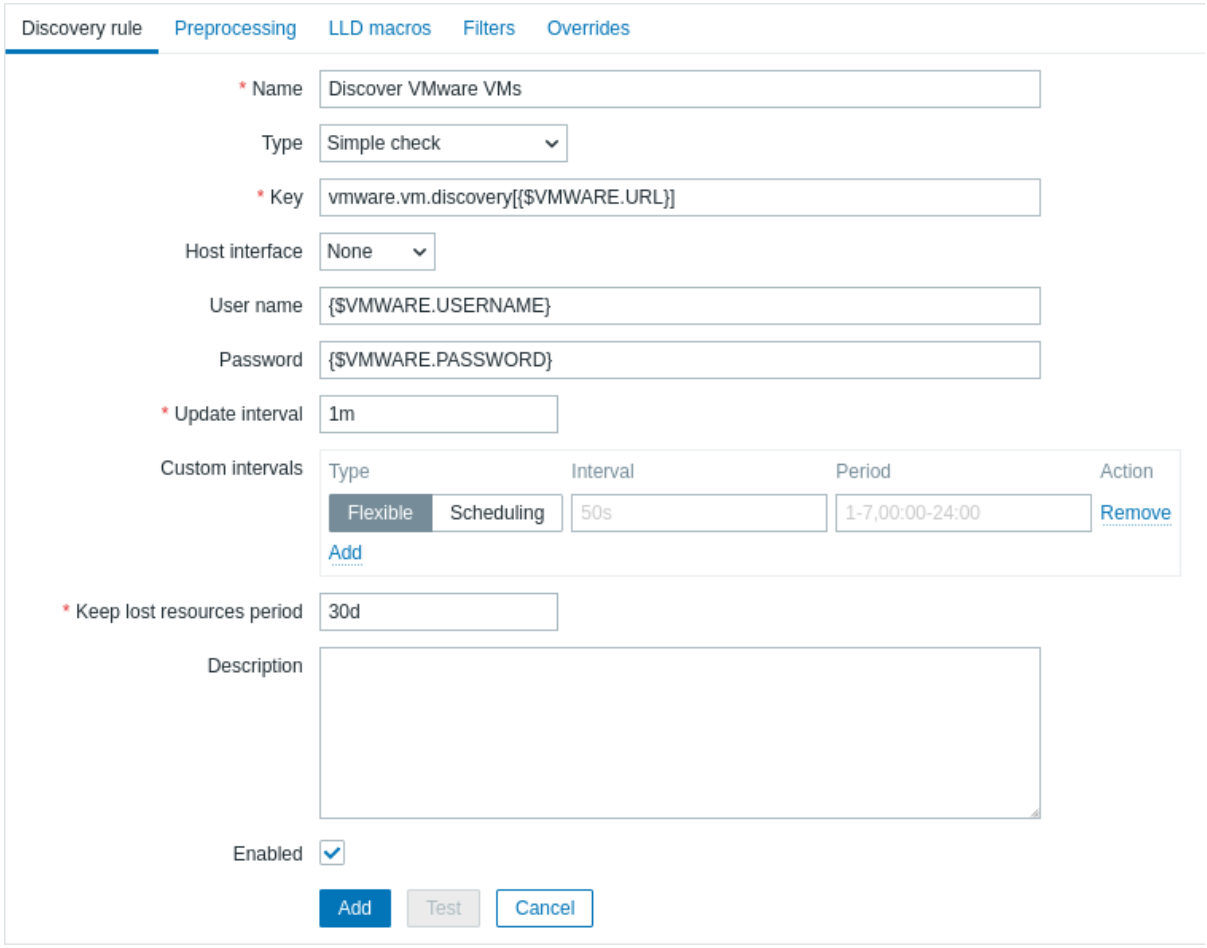
3. Click the *Add* button to create the host. This host will represent your VMware environment.

Create a low-level discovery rule

1. Click *Discovery* for the created host to go to the list of low-level discovery rules for that host.

2. Create a low-level discovery rule:

- In the *Name* field, enter a low-level discovery rule name (for example, "Discover VMware VMs").
- In the *Type* field, select "Simple check".
- In the *Key* field, enter the built-in item key for discovering VMware virtual machines: `vmware.vm.discovery[{$VMWARE.URL}]`
- In the *User name* and *Password* fields, enter the corresponding macros previously configured on the host.



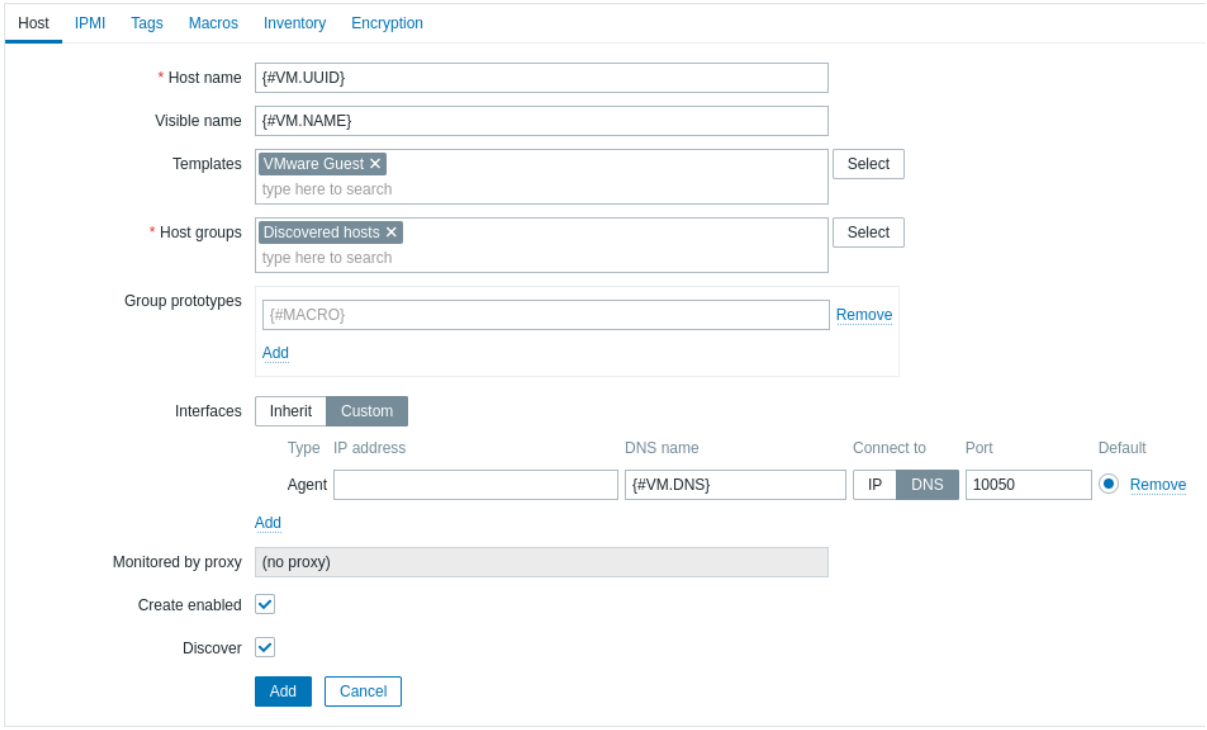
3. Click the *Add* button to create the low-level discovery rule. This discovery rule will discover virtual machines in your VMware environment.

Create a host prototype

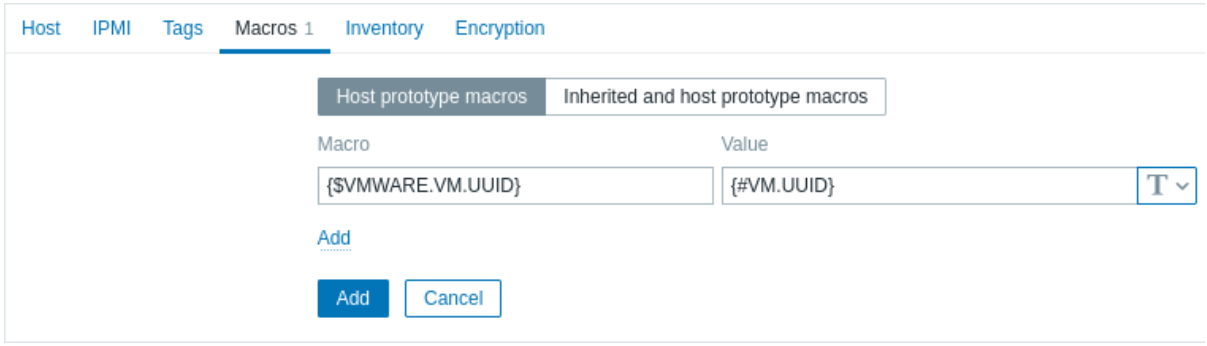
1. In the list of low-level discovery rules, click *Host prototypes* for the previously created low-level discovery rule.

2. Create a host prototype. Since host prototypes are blueprints for creating hosts through low-level discovery rules, most fields will contain low-level discovery macros. This ensures that the hosts are created with properties based on the content retrieved by the previously created low-level discovery rule.

- In the *Host name* field, enter the macro {#VM.UUID}.
- In the *Visible name* field, enter the macro {#VM.NAME}.
- In the *Templates* field, enter or select the "VMware Guest" template. This template contains VMware items and discovery rules for monitoring the power state of a virtual machine, CPU usage, memory usage, network devices, etc.
- In the *Host groups* field, enter or select a host group (for example, "Discovered hosts").
- In the *Interfaces* field, add a custom host interface and, in the *DNS name* field, enter the macro {#VM.DNS}. Alternatively (based on the configuration of your VMware environment), in the *IP address* field, enter the macro {#VM.IP}. This is necessary for the correct functioning of the *VMware Guest* template.



- In the *Macros* tab, set the {$VMWARE.VM.UUID} macro with the value {#VM.UUID}. This is necessary for the correct functioning of the *VMware Guest* template that uses this macro as a host-level user macro in item parameters (for example, *vmware.vm.net.if.discovery[{$VMWARE.URL}, {$VMWARE.VM.UUID}]*).

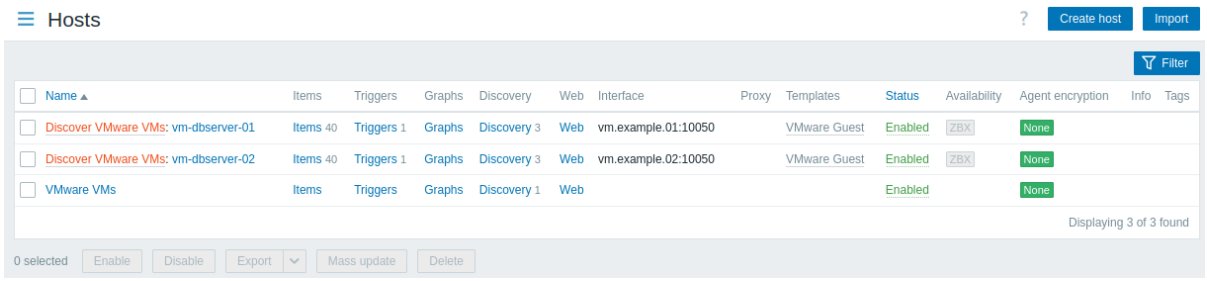


3. Click the *Add* button to create the host prototype. This host prototype will be used to create hosts for virtual machines discovered by the previously created low-level discovery rule.

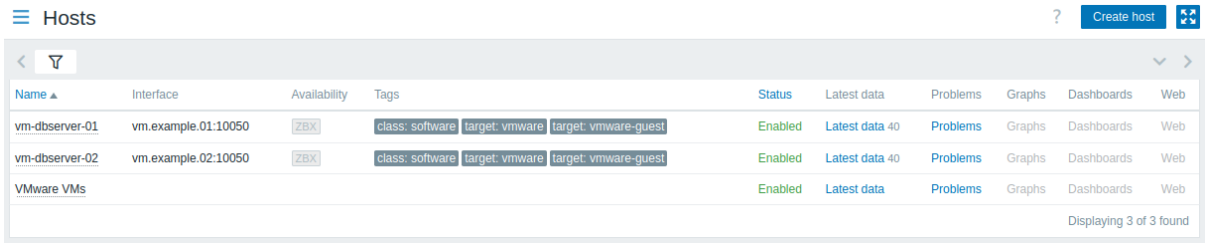
View hosts and metrics

After the host prototype has been created, the low-level discovery rule will create hosts for discovered VMware virtual machines, and Zabbix will start to monitor them. Note that the discovery and creation of hosts can also be executed manually, if necessary.

To view the created hosts, navigate to the *Data collection → Hosts* menu section.



To view collected metrics, navigate to the *Monitoring → Hosts* menu section and click *Latest data* for one of the hosts.



11 Maintenance

Overview You can define maintenance periods for hosts and host groups in Zabbix.

Furthermore, it is possible to define maintenance only for a single trigger (or subset of triggers) by specifying trigger tags. In this case maintenance will be activated only for those triggers; all other triggers of the host or host group will not be in maintenance.

There are two maintenance types - with data collection and with no data collection.

During a maintenance "with data collection" triggers are processed as usual and events are created when required. However, problem escalations are paused for hosts/triggers in maintenance, if the *Pause operations for suppressed problems* option is checked in action configuration. In this case, escalation steps that may include sending notifications or remote commands will be ignored for as long as the maintenance period lasts. Note that problem recovery and update operations are not suppressed during maintenance, only escalations.

For example, if escalation steps are scheduled at 0, 30 and 60 minutes after a problem start, and there is a half-hour long maintenance lasting from 10 minutes to 40 minutes after a real problem arises, steps two and three will be executed a half-hour later, or at 60 minutes and 90 minutes (providing the problem still exists). Similarly, if a problem arises during the maintenance, the escalation will start after the maintenance.

To receive problem notifications during the maintenance normally (without delay), you have to uncheck the *Pause operations for suppressed problems* option in action configuration.

> **Note:**
> If at least one host (used in the trigger expression) is not in maintenance mode, Zabbix will send a problem notification.

Zabbix server must be running during maintenance. Timer processes are responsible for switching host status to/from maintenance at 0 seconds of every minute. Note that when a host enters maintenance, Zabbix server timer processes will read all open problems to check if it is required to suppress those. This may have a performance impact if there are many open problems. Zabbix server will also read all open problems upon startup, even if there are no maintenances configured at the time.

Note that the Zabbix server (or proxy) always collects data regardless of the maintenance type (including "no data" maintenance). The data is later ignored by the server if 'no data collection' is set.

When "no data" maintenance ends, triggers using nodata() function will not fire before the next check during the period they are checking.

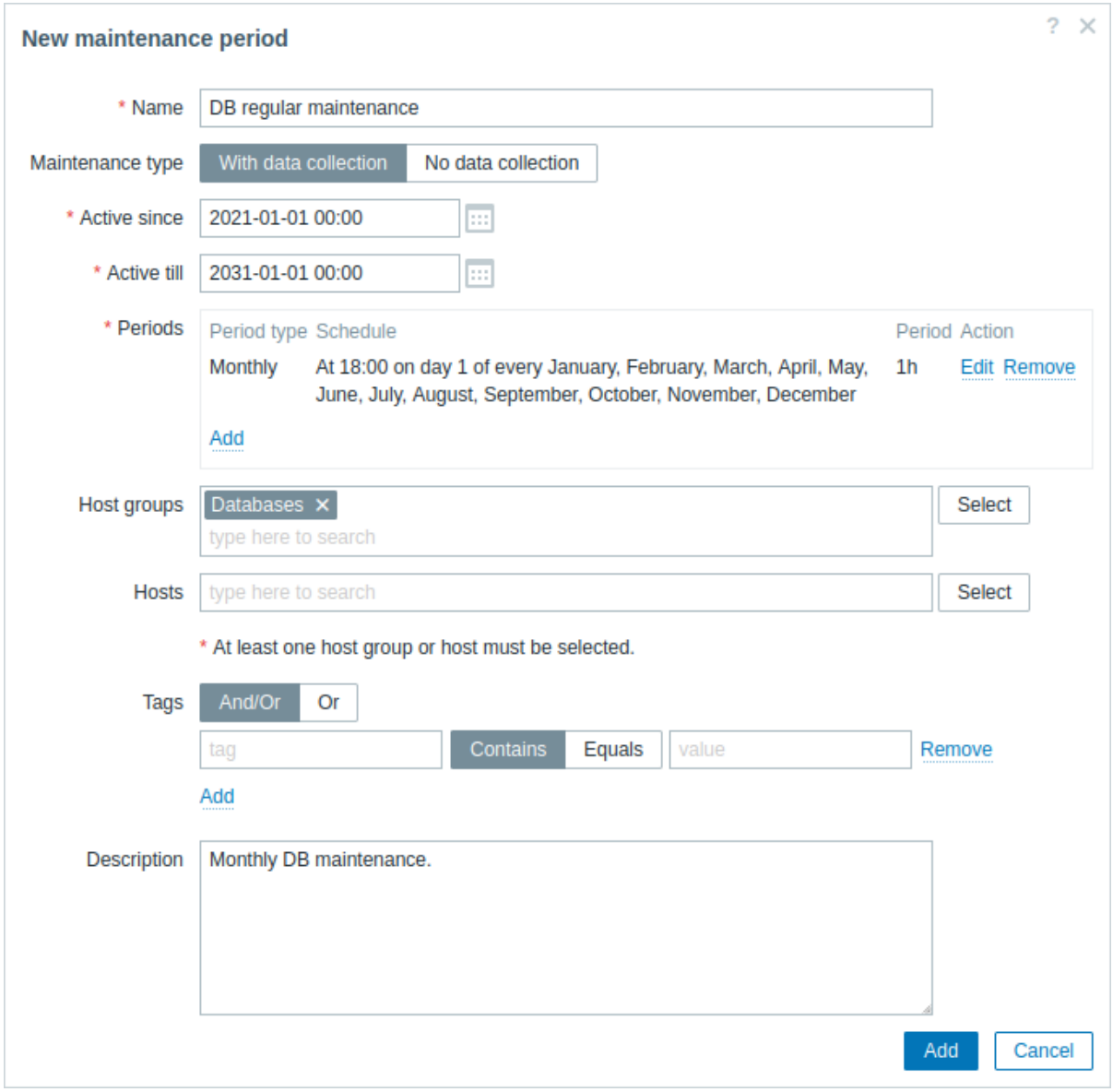
If a log item is added while a host is in maintenance and the maintenance ends, only new logfile entries since the end of the maintenance will be gathered.

If a timestamped value is sent for a host that is in a "no data" maintenance type (e.g. using Zabbix sender) then this value will be dropped however it is possible to send a timestamped value in for an expired maintenance period and it will be accepted.

If maintenance period, hosts, groups or tags are changed by the user, the changes will only take effect after configuration cache synchronization.

Configuration To configure a maintenance period:

1. Go to: *Data collection → Maintenance*.
2. Click on *Create maintenance period* (or on the name of an existing maintenance period).
3. Enter maintenance parameters in the form.



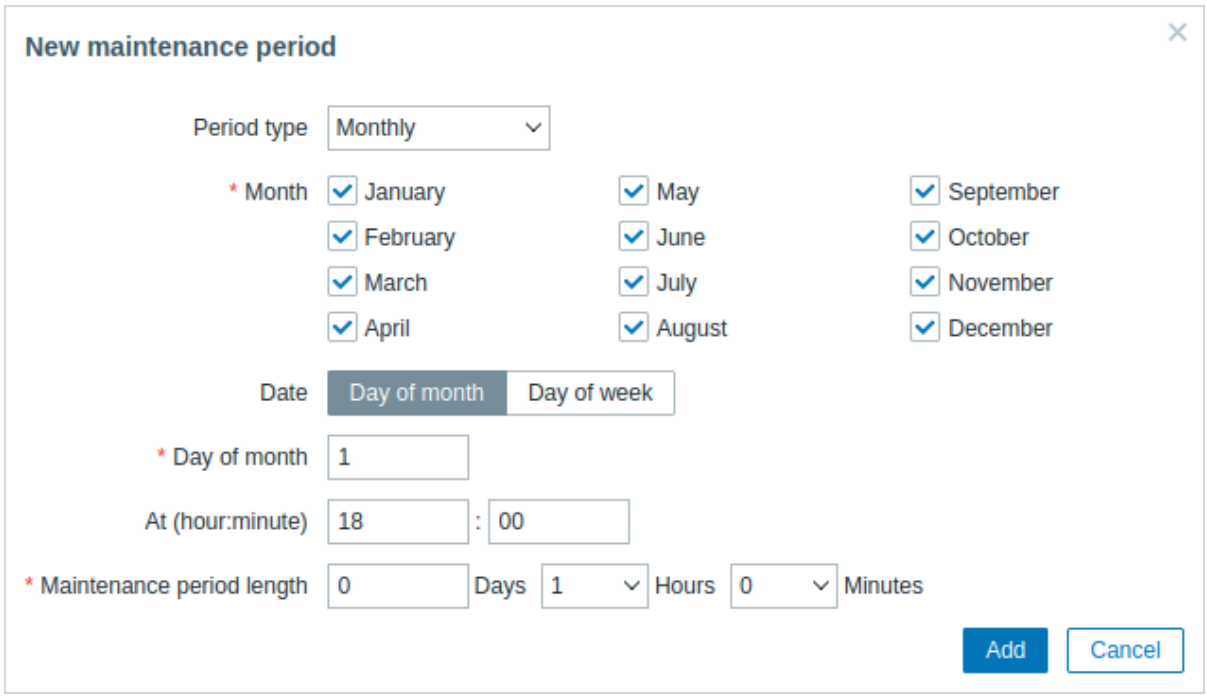
All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Name of the maintenance period.
Maintenance type	Two types of maintenance can be set: **With data collection** - data will be collected by the server during maintenance, triggers will be processed; **No data collection** - data will not be collected by the server during maintenance.
Active since	The date and time when executing maintenance periods becomes active. *Note:* Setting this time alone does not activate a maintenance period; maintenance periods must be configured in *Periods* (see below).
Active till	The date and time when executing maintenance periods stops being active.
Periods	This block allows you to define the exact days and hours when the maintenance takes place. Clicking on Add opens a popup window with a flexible *Maintenance period* form where you can define maintenance schedule. See Maintenance periods for a detailed description.
Host groups	Select host groups that the maintenance will be activated for. The maintenance will be activated for all hosts from the specified host group(s). This field is auto-complete, so starting to type in it will display a dropdown of all available host groups. Specifying a parent host group implicitly selects all nested host groups. Thus the maintenance will also be activated on hosts from nested groups.
Hosts	Select hosts that the maintenance will be activated for. This field is auto-complete, so starting to type in it will display a dropdown of all available hosts.

Parameter	Description
Tags	If maintenance tags are specified, maintenance for the selected hosts will be activated, but only problems with matching tags will be suppressed (that is, no actions will be taken). In case of multiple tags, they are calculated as follows: **And/Or** - all tags must correspond; however tags with the same tag name are calculated by the Or condition; **Or** - enough if one tag corresponds. There are two ways of matching the tag value: **Contains** - case-sensitive substring match (tag value contains the entered string); **Equals** - case-sensitive string match (tag value equals the entered string). Tags can be specified only if *With data collection* mode is selected.
Description	Description of maintenance period.

Maintenance periods

The maintenance period window is for scheduling time for a recurring or a one-time maintenance. The form is dynamic with available fields changing based on the *Period type* selected.



Period type	Description
One time only	Configure a one time only maintenance period: *Date* - date and time of the maintenance period; *Maintenance period length* - for how long the maintenance will be active.
Daily	Configure a daily maintenance period: *Every day(s)* - maintenance frequency (1 - *(default)* every day, 2 - every two days, etc.); *At (hour:minute)* - time of the day when maintenance starts; *Maintenance period length* - for how long the maintenance will be active. When *Every day(s)* parameter is greater than ”1”, the starting day is the day that the *Active since* time falls on. Examples: - if *Active since* is set to ”2021-01-01 12:00”, *Every day(s)* is set to ”2”, and *At (hour:minute)* is set to ”23:00”, then the first maintenance period will start on January 1 at 23:00, while the second maintenance period will start on January 3 at 23:00; - if *Active since* is set to ”2021-01-01 12:00”, *Every day(s)* is set to ”2”, and *At (hour:minute)* is set to ”01:00”, then the first maintenance period will start on January 3 at 01:00, while the second maintenance period will start on January 5 at 01:00.

Period type	Description
Weekly	Configure a weekly maintenance period: *Every week(s)* - maintenance frequency (1 - *(default)* every week, 2 - every two weeks, etc.); *Day of week* - on which day the maintenance should take place; *At (hour:minute)* - time of the day when maintenance starts; *Maintenance period length* - for how long the maintenance will be active. When *Every week(s)* parameter is greater than "1", the starting week is the week that the *Active since* time falls on. For examples, see parameter *Daily* description above.
Monthly	Configure a monthly maintenance period: *Month* - select all months during which the regular maintenance is carried out; *Date:* ***Day of month*** - select this option if the maintenance should take place on the same date each month (for example, every 1st day of the month), and then select the required day in the field *Day of month* that appears; *Date:* ***Day of week*** - select this option if the maintenance should take place only on certain days (for example, every first Monday of the month), then select (in the drop-down) the required week of the month (first, second, third, fourth, or last), and then mark the checkboxes for maintenance day(s); *At (hour:minute)* - time of the day when maintenance starts; *Maintenance period length* - for how long the maintenance will be active.

Attention:
When creating a maintenance period, the time zone of the user who creates it is used. However, when recurring maintenance periods (*Daily*, *Weekly*, *Monthly*) are scheduled, the time zone of the Zabbix server is used. To ensure predictable behavior of recurring maintenance periods, it is required to use a common time zone for all parts of Zabbix.

When done, press *Add* to add the maintenance period to the *Periods* block.

Note that Daylight Saving Time (DST) changes do not affect how long the maintenance will be. For example, let's say that we have a two-hour maintenance configured that usually starts at 01:00 and finishes at 03:00:

- if after one hour of maintenance (at 02:00) a DST change happens and current time changes from 02:00 to 03:00, the maintenance will continue for one more hour (till 04:00);
- if after two hours of maintenance (at 03:00) a DST change happens and current time changes from 03:00 to 02:00, the maintenance will stop, because two hours have passed;
- if a maintenance period starts during the hour that is skipped by a DST change, then the maintenance will not start.

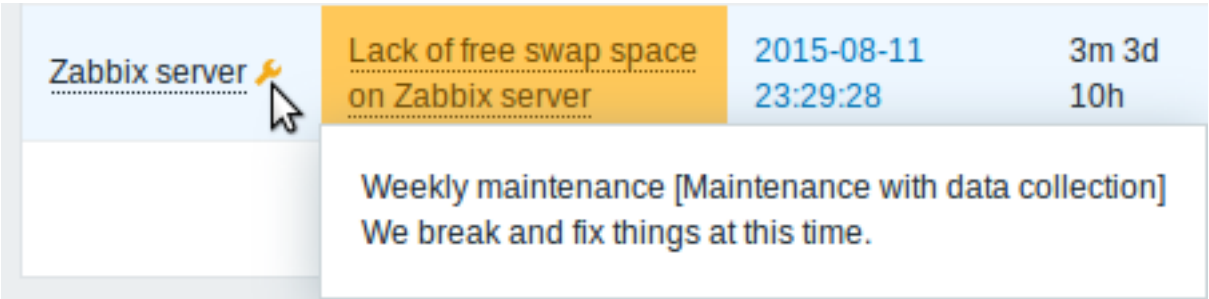
If a maintenance period is set to "1 day" (the actual period of the maintenance is 24 hours, since Zabbix calculates days in hours), starts at 00:00 and finishes at 00:00 the next day:

- the maintenance will stop at 01:00 the next day if current time changes forward one hour;
- the maintenance will stop at 23:00 that day if current time changes back one hour.

Display Displaying hosts in maintenance

An orange wrench icon next to the host name indicates that this host is in maintenance in:

- *Dashboards*
- *Monitoring → Problems*
- *Inventory → Hosts → Host inventory details*
- *Data collection → Hosts* (See 'Status' column)



Maintenance details are displayed when the mouse pointer is positioned over the icon.

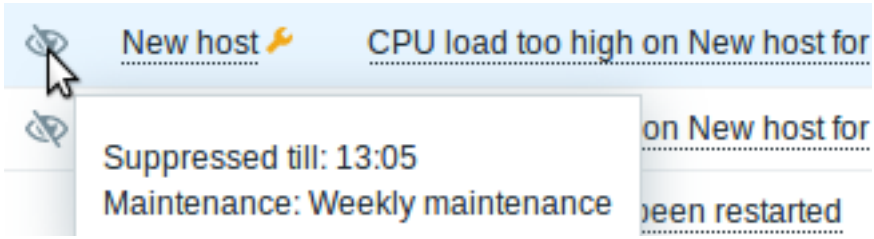
Additionally, hosts in maintenance get an orange background in *Monitoring* → *Maps*.

Displaying suppressed problems

Normally problems for hosts in maintenance are suppressed, i.e. not displayed in the frontend. However, it is also possible to configure that suppressed problems are shown, by selecting the *Show suppressed problems* option in these locations:

- *Dashboards* (in *Problem hosts*, *Problems*, *Problems by severity*, *Trigger overview* widget configuration)
- *Monitoring* → *Problems* (in the filter)
- *Monitoring* → *Maps* (in map configuration)
- Global notifications (in user profile configuration)

When suppressed problems are displayed, the following icon is displayed: . Rolling a mouse over the icon displays more details.



12 Regular expressions

Overview Perl Compatible Regular Expressions (PCRE, PCRE2) are supported in Zabbix.

There are two ways of using regular expressions in Zabbix:

- manually entering a regular expression
- using a global regular expression created in Zabbix

Regular expressions You may manually enter a regular expression in supported places. Note that the expression may not start with @ because that symbol is used in Zabbix for referencing global regular expressions.

> **Warning:**
> It's possible to run out of stack when using regular expressions. See the pcrestack man page for more information.

Note that in multiline matching, the ^ and $ anchors match at the beginning/end of each line respectively, instead of the beginning/end of the entire string.

See also examples for correct escaping in various contexts.

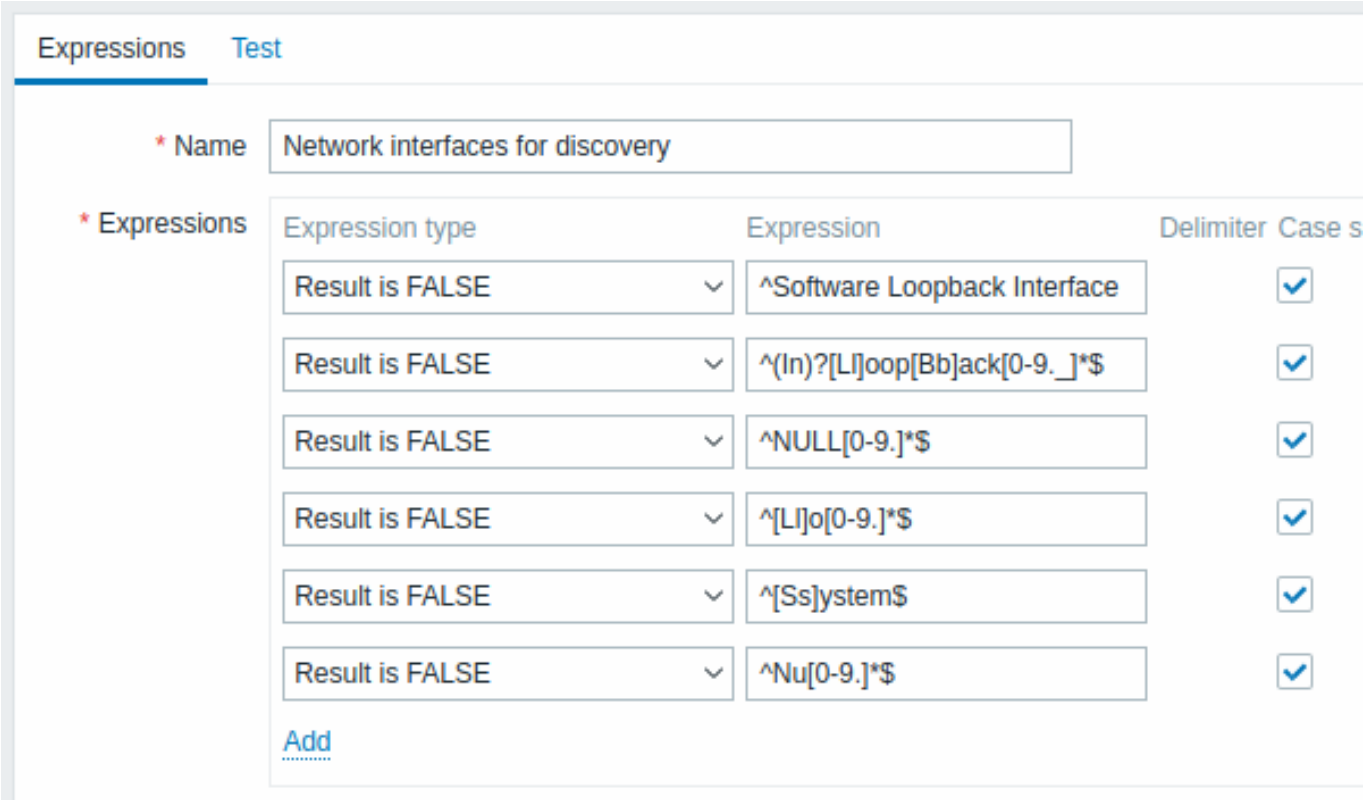
Global regular expressions There is an advanced editor for creating and testing complex regular expressions in Zabbix frontend.

Once a regular expression has been created this way, it can be used in several places in the frontend by referring to its name, prefixed with @, for example, *@mycustomregexp*.

To create a global regular expression:

- Go to: *Administration* → *General*
- Select *Regular expressions* from the dropdown
- Click on *New regular expression*

The **Expressions** tab allows to set the regular expression name and add subexpressions.



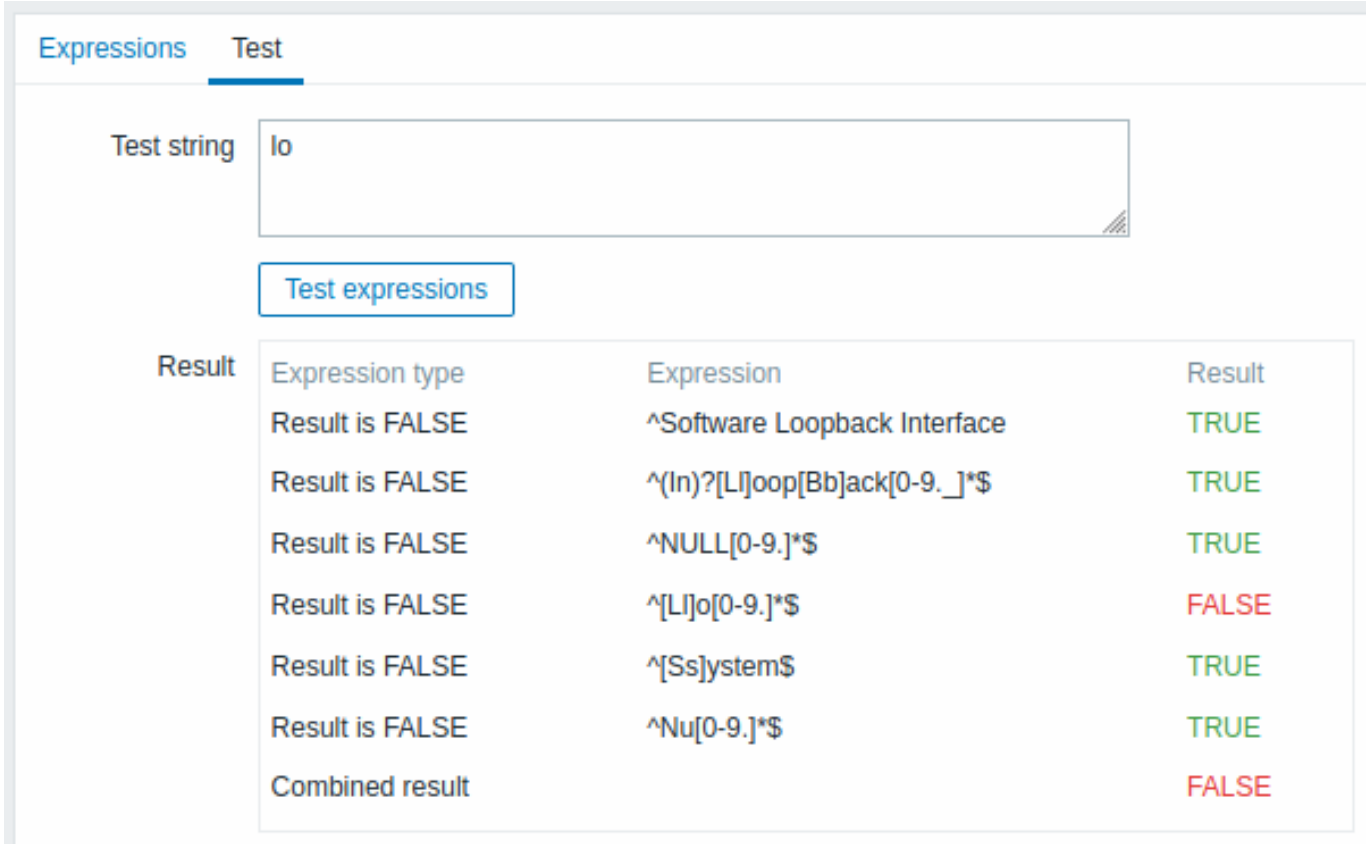
All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Set the regular expression name. Any Unicode characters are allowed.
Expressions	Click on *Add* in the Expressions block to add a new subexpression.
Expression type	Select expression type: **Character string included** - match the substring **Any character string included** - match any substring from a delimited list. The delimited list includes a comma (,), a dot (.) or a forward slash (/). **Character string not included** - match any string except the substring **Result is TRUE** - match the regular expression **Result is FALSE** - do not match the regular expression
Expression	Enter substring/regular expression.
Delimiter	A comma (,), a dot (.) or a forward slash (/) to separate text strings in a regular expression. This parameter is active only when *"Any character string included"* expression type is selected.
Case sensitive	A checkbox to specify whether a regular expression is sensitive to capitalization of letters.

A forward slash (/) in the expression is treated literally, rather than a delimiter. This way it is possible to save expressions containing a slash, without errors.

Attention:
A custom regular expression name in Zabbix may contain commas, spaces, etc. In those cases where that may lead to misinterpretation when referencing (for example, a comma in the parameter of an item key) the whole reference may be put in quotes like this: *"@My custom regexp for purpose1, purpose2"*.
Regular expression names must not be quoted in other locations (for example, in LLD rule properties).

In the **Test** tab the regular expression and its subexpressions can be tested by providing a test string.



Results show the status of each subexpression and total custom expression status.

Total custom expression status is defined as *Combined result*. If several sub expressions are defined Zabbix uses AND logical operator to calculate *Combined result*. It means that if at least one Result is False *Combined result* has also False status.

Default global regular expressions Zabbix comes with several global regular expression in its default dataset.

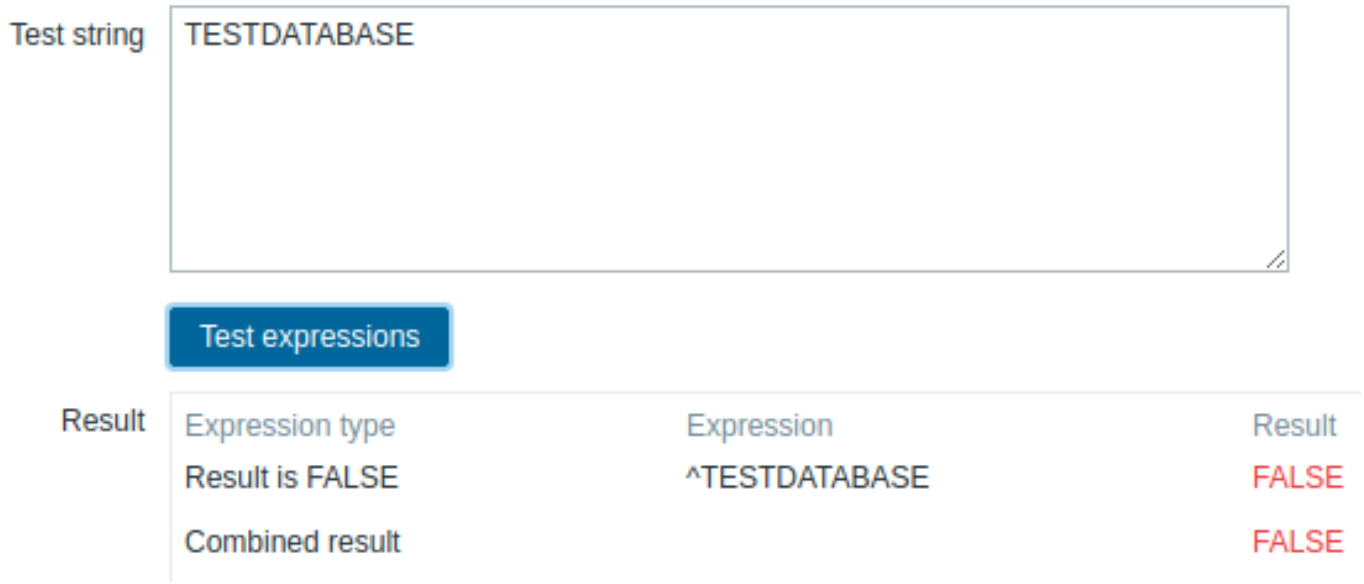
Name	Expression	Matches
File systems for discovery	`^(btrfs|ext2|ext3|ext4|jfs|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|refs|apfs|ntfs|fat32|zfs)`	”btrfs” or ”ext2” or ”ext3” or ”ext4” or ”jfs” or ”reiser” or ”xfs” or ”ffs” or ”ufs” or ”jfs” or ”jfs2” or ”vxfs” or ”hfs” or ”refs” or ”apfs” or ”ntfs” or ”fat32” or ”zfs”
Network interfaces for discovery	`^Software Loopback Interface`	Strings starting with ”Software Loopback Interface”.
	`^lo$`	”lo”
	`^(In)?[Ll]oop[Bb]ack[0-9._]*$`	Strings that optionally start with ”In”, then have ”L” or ”l”, then ”oop”, then ”B” or ”b”, then ”ack”, which can be optionally followed by any number of digits, dots or underscores.
	`^NULL[0-9.]*$`	Strings starting with ”NULL” optionally followed by any number of digits or dots.
	`^[Ll]o[0-9.]*$`	Strings starting with ”Lo” or ”lo” and optionally followed by any number of digits or dots.
	`^[Ss]ystem$`	”System” or ”system”
	`^Nu[0-9.]*$`	Strings starting with ”Nu” optionally followed by any number of digits or dots.
Storage devices for SNMP discovery	`^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$`	”Physical memory” or ”Virtual memory” or ”Memory buffers” or ”Cached memory” or ”Swap space”
Windows service names for discovery	`^(MMCSS|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_optimization_v4.0.30319_32)$`	”MMCSS” or ”gupdate” or ”SysmonLog” or strings like ”clr_optimization_v2.0.50727_32” and ”clr_optimization_v4.0.30319_32” where instead of dots you can put any character except newline.

Name	Expression	Matches
Windows service startup states for discovery	`^(automatic|automatic delayed)$`	”automatic” or ”automatic delayed”

Examples Example 1

Use of the following expression in low-level discovery to discover databases except a database with a specific name:

`^TESTDATABASE$`

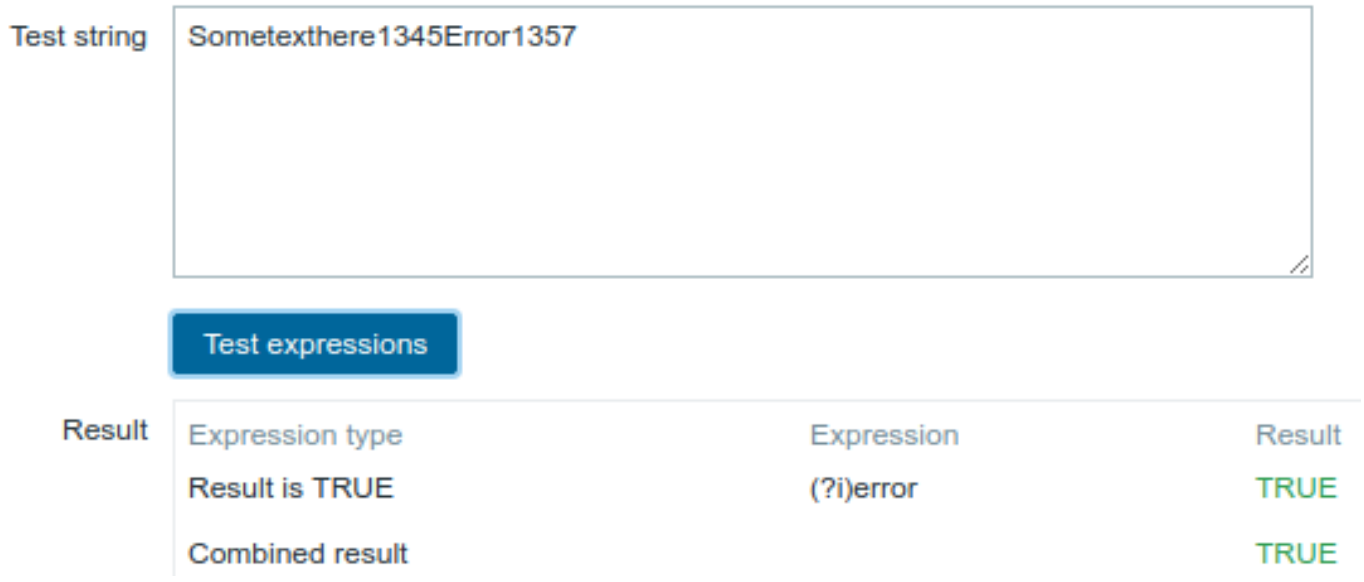


Chosen *Expression type*: ”Result is FALSE”. Doesn’t match name, containing string ”*TESTDATABASE*”.

Example with an inline regex modifier

Use of the following regular expression including an inline modifier (?i) to match the characters ”error”:

`(?i)error`



Chosen *Expression type*: ”Result is TRUE”. Characters ”error” are matched.

Another example with an inline regex modifier

Use of the following regular expression including multiple inline modifiers to match the characters after a specific line:

`(?<=match (?i)everything(?-i) after this line\n)(?sx).*# we add s modifier to allow . match newline charac`



Chosen Expression type: "Result is TRUE". Characters after a specific line are matched.

Attention:
g modifier can't be specified in line. The list of available modifiers can be found in pcresyntax man page. For more information about PCRE syntax please refer to PCRE HTML documentation.

Regular expression support by location

Location	Regular expression	Global regular expression	Multiline matching	Comments
Agent items				
eventlog[]	Yes	Yes	Yes	`regexp`, `severity`, `source`, `eventid` parameters
log[]				`regexp` parameter
log.count[]				
logrt[]		Yes/No		`regexp` parameter supports both, `file_regexp` parameter supports non-global expressions only
logrt.count[]				
proc.cpu.util[]		No	No	`cmdline` parameter
proc.get[]				
proc.mem[]				
proc.num[]				
sensor[]				`device` and `sensor` parameters on Linux 2.4
system.hw.macaddr[]				`interface` parameter
system.sw.packages[]				`regexp` parameter
system.sw.packages.get[]				`regexp` parameter
vfs.dir.count[]				`regex_incl`, `regex_excl`, `regex_excl_dir` parameters
vfs.dir.get[]				`regex_incl`, `regex_excl`, `regex_excl_dir` parameters
vfs.dir.size[]				`regex_incl`, `regex_excl`, `regex_excl_dir` parameters
vfs.file.regexp[]			Yes	`regexp` parameter
vfs.file.regmatch[]				
web.page.regexp[]				
SNMP traps				
snmptrap[]	Yes	Yes	No	`regexp` parameter
Item value pre-processing	Yes	No	No	`pattern` parameter

Location	Regular expression	Global regular expression	Multiline matching	Comments
Functions for trig-gers/calculated items				
count()	Yes	Yes	Yes	`pattern` parameter if `operator` parameter is *regexp* or *iregexp*
countunique()	Yes	Yes		
find()	Yes	Yes		
logeventid()	Yes	Yes	No	`pattern` parameter
logsource()				
Low-level dis-cov-ery				
Filters	Yes	Yes	No	*Regular expression* field
Overrides	Yes	No		In *matches*, *does not match* options for *Operation* conditions
Action con-di-tions	Yes	No	No	In *matches*, *does not match* options for *Host name* and *Host metadata* autoregistration conditions
Web mon-i-tor-ing	Yes	No	Yes	*Variables* with a **regex:** prefix *Required string* field
User macro con-text	Yes	No	No	In macro context with a **regex:** prefix
Macro func-tions				
regsub()	Yes	No	No	`pattern` parameter
iregsub()				
Icon map-ping	Yes	Yes	No	*Expression* field
Value map-ping	Yes	No	No	*Value* field if mapping type is `regexp`

13 Problem acknowledgment

Overview Problem events in Zabbix can be acknowledged by users.

If a user gets notified about a problem event, they can go to Zabbix frontend, open the problem update popup window of that problem using one of the ways listed below and acknowledge the problem. When acknowledging, they can enter their comment for it, saying that they are working on it or whatever else they may feel like saying about it.

This way, if another system user spots the same problem, they immediately see if it has been acknowledged and the comments so far.

This way the workflow of resolving problems with more than one system user can take place in a coordinated way.

Acknowledgment status is also used when defining action operations. You can define, for example, that a notification is sent to a higher level manager only if an event is not acknowledged for some time.

To acknowledge events and comment on them, a user must have at least read permissions to the corresponding triggers. To change problem severity or close problem, a user must have read-write permissions to the corresponding triggers.

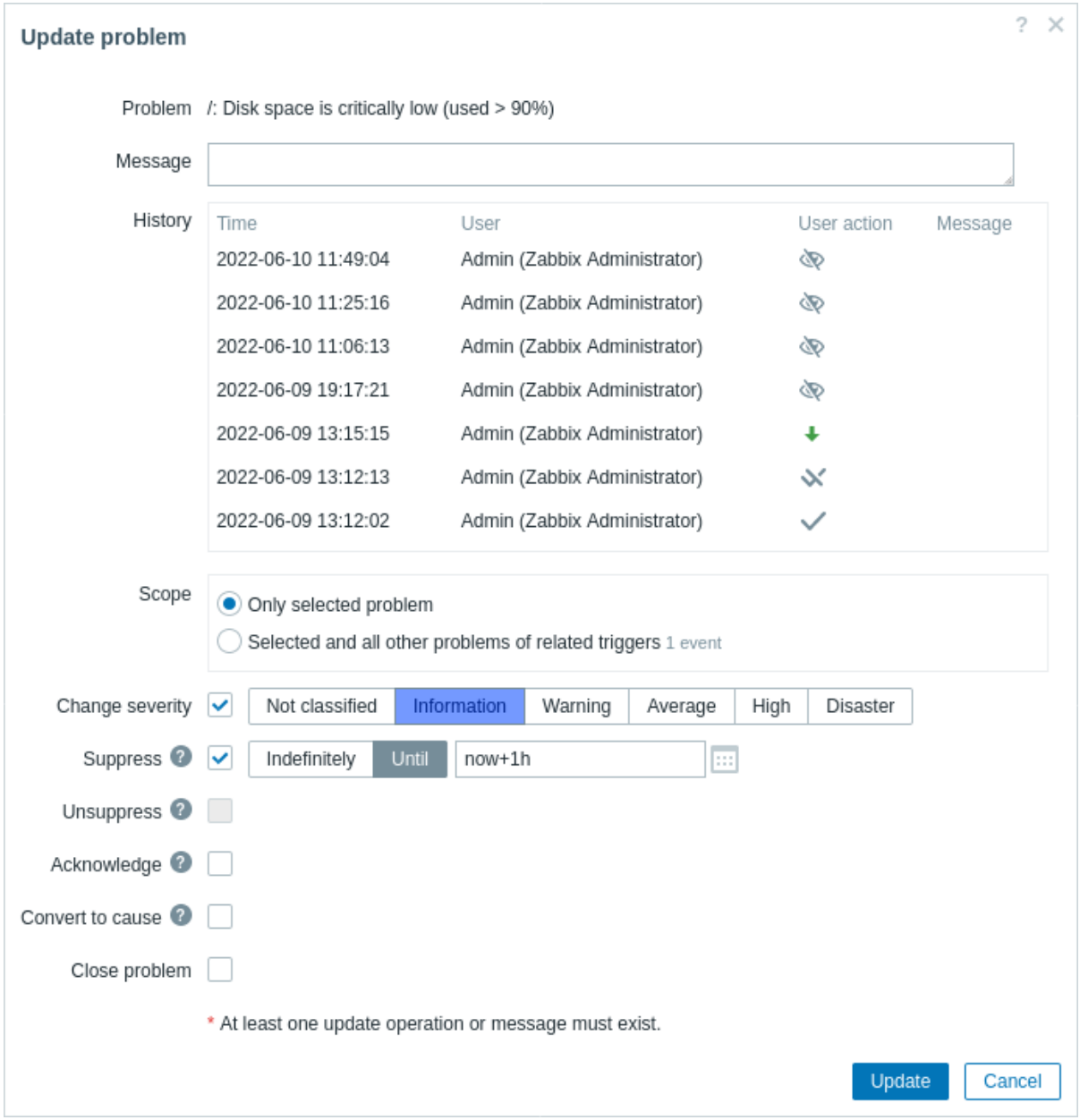
There are **several** ways to access the problem update popup window, which allows acknowledging a problem.

- You may select problems in *Monitoring* → *Problems* and then click on *Mass update* below the list
- You can click on *Update* in the *Update* column of a problem in:
  - *Dashboards* (*Problems* and *Problems by severity* widgets)
  - *Monitoring* → *Problems*
  - *Monitoring* → *Problems* → *Event details*
- You can click on an unresolved problem cell in:
  - *Dashboards* (*Trigger overview* widget)

The popup menu contains an *Update* option that will take you to the problem update window.

Updating problems The problem update popup allows to:

- comment on the problem
- view comments and actions so far
- change problem severity
- suppress/unsuppress problem
- acknowledge/unacknowledge problem
- change symptom problem to cause problem
- manually close problem



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Problem	If only one problem is selected, the problem name is displayed. If several problems are selected, *N problems selected* is displayed.
Message	Enter text to comment on the problem (maximum 2048 characters).
History	Previous activities and comments on the problem are listed, along with the time and user details. For the meaning of icons used to denote user actions see the event detail page. Note that history is displayed if only one problem is selected for the update.
Scope	Define the scope of such actions as changing severity, acknowledging or manually closing problems: **Only selected problem** - will affect this event only **Selected and all other problems of related triggers** - in case of acknowledgment/closing problem, will affect this event and all other problems that are not acknowledged/closed so far. If the scope contains problems already acknowledged or closed, these problems will not be acknowledged/closed repeatedly. On the other hand, the number of message and severity change operations are not limited.
Change severity	Mark the checkbox and click on the severity button to update problem severity. The checkbox for changing severity is available if read-write permissions exist for at least one of the selected problems. Only those problems that are read-writable will be updated when clicking on *Update*. If read-write permissions exist for none of the selected triggers, the checkbox is disabled.
Suppress	Mark the checkbox to suppress the problem: **Indefinitely** - suppress indefinitely **Until** - suppress until a given time. Both absolute and relative time formats are supported, for example: `now+1d` - for one day from now (default) `now/w` - until the end of the current week `2022-05-28 12:00:00` - until absolute date/time Note that a simple period (e. g., `1d`, `1w`) is not supported. Availability of this option depends on the "Suppress problems" user role settings. See also: Problem suppression
Unsuppress	Mark the checkbox to unsuppress the problem. This checkbox is active only if at least one of the selected problems is suppressed. Availability of this option depends on the "Suppress problems" user role settings.
Acknowledge	Mark the checkbox to acknowledge the problem. This checkbox is available if there is at least one unacknowledged problem among the selected. It is not possible to add another acknowledgment for an already acknowledged problem (it is possible to add another comment though).
Unacknowledge	Mark the checkbox to unacknowledge the problem. This checkbox is available if there is at least one acknowledged problem among the selected.
Convert to cause	Mark the checkbox to convert the symptom problem(s) to cause problem(s).
Close problem	Mark the checkbox to manually close the selected problem(s). The checkbox for closing a problem is available if the *Allow manual close* option is checked in trigger configuration for at least one of the selected problems. Only those problems will be closed that are allowed to be closed when clicking on *Update*. If no problem is manually closeable, the checkbox is disabled. Already closed problems will not be closed repeatedly.

Display Based on acknowledgment information it is possible to configure how the problem count is displayed in the dashboard or maps. To do that, you have to make selections in the *Problem display* option, available in both map configuration and the *Problems by severity* dashboard widget. It is possible to display all problem count, unacknowledged problem count as separated from the total or unacknowledged problem count only.

Based on problem update information (acknowledgment, etc.), it is possible to configure update operations - send a message or execute remote commands.

1 Problem suppression

Overview

Problem suppression offers a way of temporarily hiding a problem that can be dealt with later. This is useful for cleaning up the

problem list in order to give the highest priority to the most urgent issues. For example, sometimes an issue may arise on the weekend that is not urgent enough to be dealt with immediately, so it can be "snoozed" until Monday morning.

Problem suppression allows to hide a *single* problem, in contrast to problem suppression through host maintenance when all problems of the maintenance host are hidden.

Operations for trigger actions will be paused for suppressed problems the same way as it is done with host maintenance.

Configuration

A problem can be suppressed through the **problem update** window, where suppression is one of the problem update options along with commenting, changing severity, acknowledging, etc.

A problem may also be unsuppressed through the same problem update window.

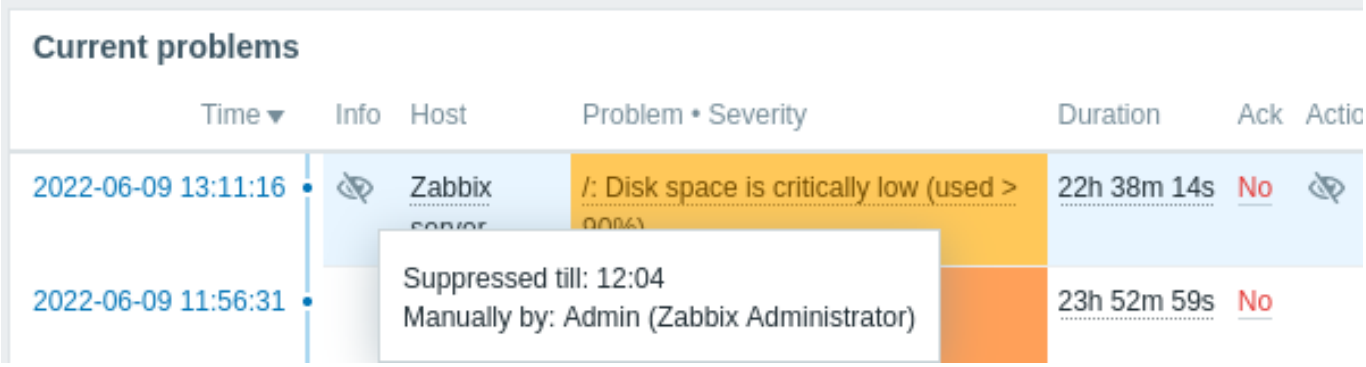
Display

Once suppressed the problem is marked by a blinking suppression icon in the *Info* column, before being hidden.

The suppression icon is blinking while the suppression task is in the waiting list. Once the task manager has suppressed the problem, the icon will stop blinking. If the suppression icon keeps blinking for a long time, this may indicate a server problem, for example, if the server is down and the task manager cannot complete the task. The same logic applies to unsuppression. In the short period after the task is submitted and the server has not completed it, the unsuppression icon is blinking.

A suppressed problem may be both hidden or shown, depending on the problem filter/widget settings.

When shown in the problem list, a suppressed problem is marked by the suppression icon and suppression details are shown on mouseover:



Suppression details are also displayed in a popup when positioning the mouse on the suppression icon in the *Actions* column.

14 Configuration export/import

Overview Zabbix export/import functionality makes it possible to exchange various configuration entities between one Zabbix system and another.

Typical use cases for this functionality:

- share templates or network maps - Zabbix users may share their configuration parameters
- upload a template to Zabbix Community templates. Then others can download the template and import the file into Zabbix.
- integrate with third-party tools - universal YAML, XML and JSON formats make integration and data import/export possible with third-party tools and applications

What can be exported/imported

Objects that can be exported/imported are:

- Host groups (*through Zabbix API only*)
- Template groups (*through Zabbix API only*)
- Templates
- Hosts
- Network maps
- Media types
- Images

Export format

Data can be exported using the Zabbix web frontend or Zabbix API. Supported export formats are YAML, XML and JSON.

Details about export

- All supported elements are exported in one file.
- Host and template entities (items, triggers, graphs, discovery rules) that are inherited from linked templates are not exported. Any changes made to those entities on a host level (such as changed item interval, modified regular expression or added prototypes to the low-level discovery rule) will be lost when exporting; when importing, all entities from linked templates are re-created as on the original linked template.
- Entities created by low-level discovery and any entities depending on them are not exported. For example, a trigger created for an LLD-rule generated item will not be exported.

Details about import

- Import stops at the first error.
- When updating existing images during image import, "imagetype" field is ignored, i.e. it is impossible to change image type via import.
- When importing hosts/templates using the "Delete missing" option, host/template macros not present in the import file will be deleted too.
- Empty tags for items, triggers, graphs, discoveryRules, itemPrototypes, triggerPrototypes, graphPrototypes are meaningless i.e. it's the same as if it was missing.
- Import supports YAML, XML and JSON, the import file must have a correct file extension: .yaml and .yml for YAML, .xml for XML and .json for JSON. See compatibility information about supported XML versions.
- Import supports configuration files only in UTF-8 encoding (with or without BOM); other encodings (UTF16LE, UTF16BE, UTF32LE, UTF32BE, etc.) will result in an import conversion error.

YAML base format The YAML export format contains the following nodes:

- Root node for Zabbix YAML export
- Export version

```
zabbix_export:
  version: '6.4'
```

Other nodes are dependent on exported objects.

XML format The XML export format contains the following tags:

- Default header for XML documents
- Root tag for Zabbix XML export
- Export version

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>6.4</version>
</zabbix_export>
```

Other tags are dependent on exported objects.

JSON format The JSON export format contains the following objects:

- Root object for Zabbix JSON export
- Export version

```
{
    "zabbix_export": {
        "version": "6.4"
    }
}
```

Other objects are dependent on exported objects.

1 Template groups

In the frontend template groups can be exported only with template export. When a template is exported all groups it belongs to are exported with it automatically.

API allows to export template groups independently from templates.

Export format

```
template_groups:
  - uuid: 36bff6c29af64692839d077febfc7079
    name: 'Network devices'
```

Element tags

Parameter	Type	Description
uuid	*string*	Unique identifier for this template group.
name	*string*	Group name.

2 Host groups

In the frontend host groups can be exported only with host export. When a host is exported all groups it belongs to are exported with it automatically.

API allows to export host groups independently from hosts.

Export format

```
host_groups:
  - uuid: 6f6799aa69e844b4b3918f779f2abf08
    name: 'Zabbix servers'
```

Element tags

Parameter	Type	Description
uuid	*string*	Unique identifier for this host group.
name	*string*	Group name.

3 Templates

Overview

Templates are exported with many related objects and object relations.

Template export contains:

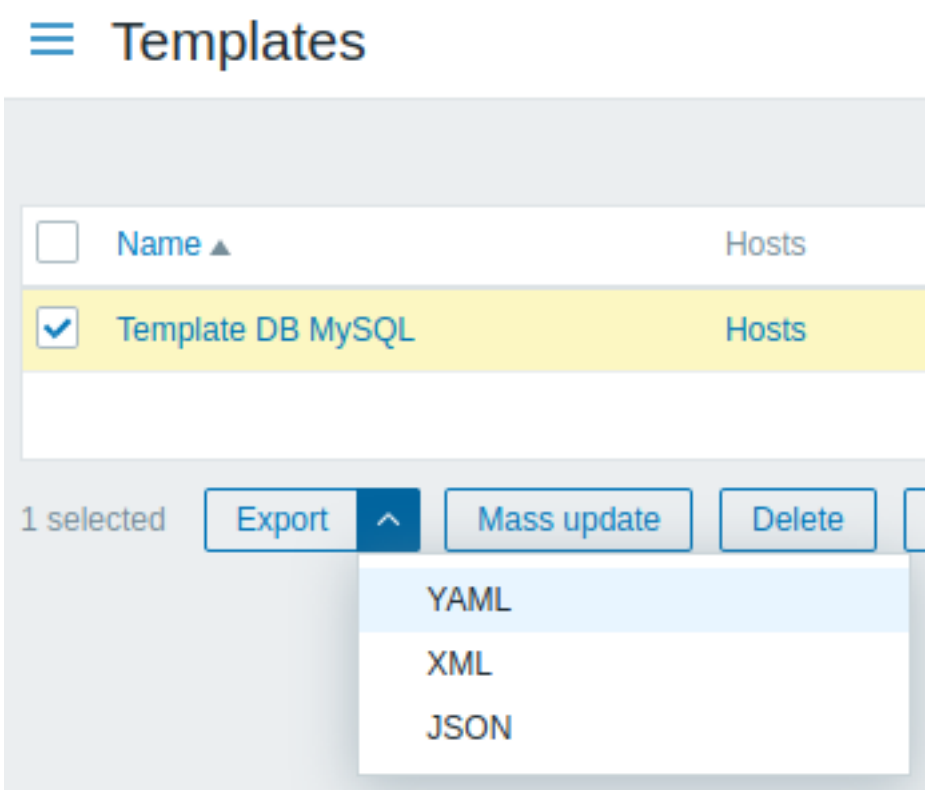
- Linked template groups
- Linked host groups (if used in host prototype configuration)
- Template data
- Linkage to other templates
- Linkage to template groups
- Directly linked items
- Directly linked triggers
- Directly linked graphs
- Directly linked dashboards
- Directly linked discovery rules with all prototypes
- Directly linked web scenarios
- Value maps

Exporting

To export templates, do the following:

1. Go to *Data collection* → *Templates*.
2. Mark the checkboxes of the templates to export.

3. Click on *Export* below the list.



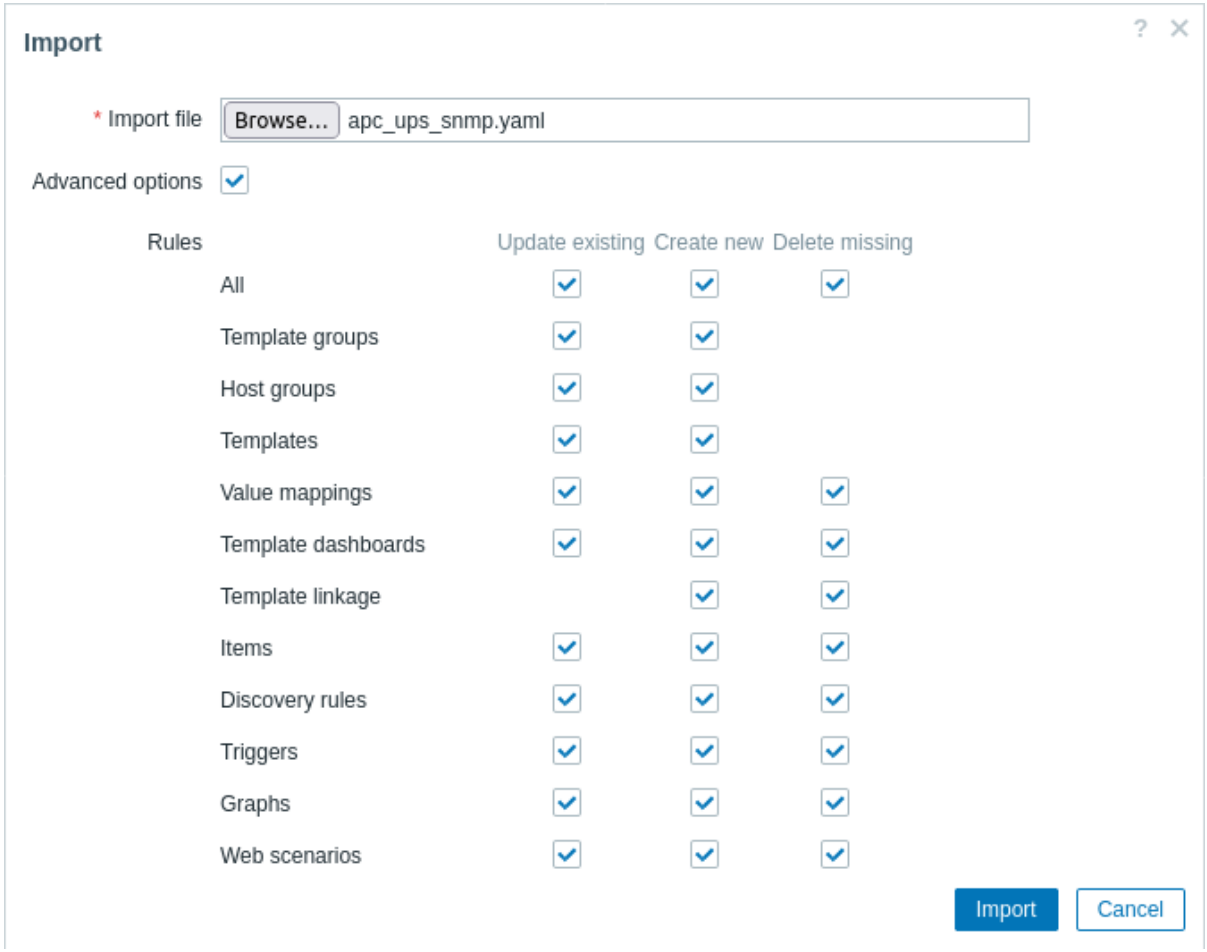
Depending on the selected format, templates are exported to a local file with a default name:

- *zabbix_export_templates.yaml* - in YAML export (default option for export);
- *zabbix_export_templates.xml* - in XML export;
- *zabbix_export_templates.json* - in JSON export.

Importing

To import templates, do the following:

1. Go to: *Data collection* → *Templates*.
2. Click on *Import* to the right.
3. Select the import file.
4. Click on *Import*.



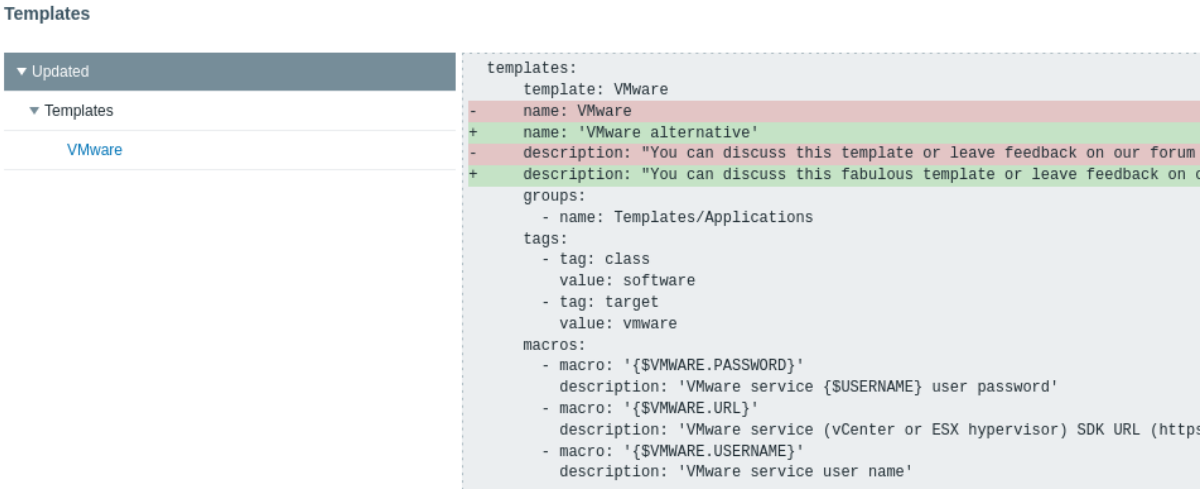
If you mark the *Advanced options* checkbox, a detailed list of all importable elements will be displayed - mark or unmark each import rule as required.

If you click the checkbox in the *All* row, all elements below it become marked/unmarked.

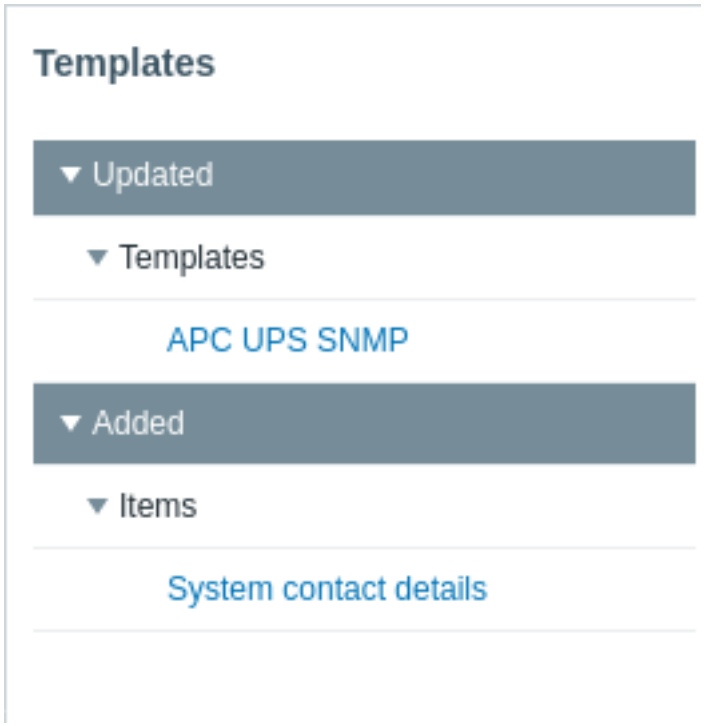
Import rules:

Rule	Description
Update existing	Existing elements will be updated with data taken from the import file. Otherwise, they will not be updated.
Create new	The import will add new elements using data from the import file. Otherwise, it will not add them.
Delete missing	The import will remove existing elements not present in the import file. Otherwise, it will not remove them. If *Delete missing* is marked for *Template linkage*, current template linkage not present in the import file will be unlinked. Entities (items, triggers, graphs, etc.) inherited from the unlinked templates will not be removed (unless the *Delete missing* option is selected for each entity as well).

On the next screen, you will be able to view the content of a template being imported. If this is a new template all elements will be listed in green. If updating an existing template, new template elements are highlighted in green; removed template elements are highlighted in red; elements that have not changed are listed on a gray background.



The menu on the left can be used to navigate through the list of changes. Section *Updated* highlights all changes made to existing template elements. Section *Added* lists new template elements. The elements in each section are grouped by element type; press on the gray arrow down to expand or collapse the group of elements.



Review template changes, then press *Import* to perform template import. A success or failure message of the import will be displayed in the frontend.

Export format

Export format in YAML:

```
zabbix_export:
  version: '6.4'
  template_groups:
    - uuid: a571c0d144b14fd4a87a9d9b2aa9fcd6
      name: Templates/Applications
  templates:
    - uuid: 56079badd056419383cc26e6a4fcc7e0
      template: VMware
      name: VMware
      description: |
        You can discuss this template or leave feedback on our forum https://www.zabbix.com/forum/zabbix-s

        Template tooling version used: 0.38
      vendor:
        name: Zabbix
```

```
  version: 6.4-0
templates:
  - name: 'VMware macros'
groups:
  - name: Templates/Applications
items:
  - uuid: 5ce209f4d94f460488a74a92a52d92b1
    name: 'VMware: Event log'
    type: SIMPLE
    key: 'vmware.eventlog[{$VMWARE.URL},skip]'
    history: 7d
    trends: '0'
    value_type: LOG
    username: '{$VMWARE.USERNAME}'
    password: '{$VMWARE.PASSWORD}'
    description: 'Collect VMware event log. See also: https://www.zabbix.com/documentation/6.2/manua
    tags:
      - tag: Application
        value: VMware
  - uuid: ee2edadb8ce943ef81d25dbbba8667a4
    name: 'VMware: Full name'
    type: SIMPLE
    key: 'vmware.fullname[{$VMWARE.URL}]'
    delay: 1h
    history: 7d
    trends: '0'
    value_type: CHAR
    username: '{$VMWARE.USERNAME}'
    password: '{$VMWARE.PASSWORD}'
    description: 'VMware service full name.'
    preprocessing:
      - type: DISCARD_UNCHANGED_HEARTBEAT
        parameters:
          - 1d
    tags:
      - tag: Application
        value: VMware
  - uuid: a0ec9145f2234fbea79a28c57ebdb44d
    name: 'VMware: Version'
    type: SIMPLE
    key: 'vmware.version[{$VMWARE.URL}]'
    delay: 1h
    history: 7d
    trends: '0'
    value_type: CHAR
    username: '{$VMWARE.USERNAME}'
    password: '{$VMWARE.PASSWORD}'
    description: 'VMware service version.'
    preprocessing:
      - type: DISCARD_UNCHANGED_HEARTBEAT
        parameters:
          - 1d
    tags:
      - tag: Application
        value: VMware
discovery_rules:
  - uuid: 16ffc933cce74cf28a6edf306aa99782
    name: 'Discover VMware clusters'
    type: SIMPLE
    key: 'vmware.cluster.discovery[{$VMWARE.URL}]'
    delay: 1h
    username: '{$VMWARE.USERNAME}'
```

```
      password: '{$VMWARE.PASSWORD}'
      description: 'Discovery of clusters'
      item_prototypes:
        - uuid: 46111f91dd564a459dbc1d396e2e6c76
          name: 'VMware: Status of "{#CLUSTER.NAME}" cluster'
          type: SIMPLE
          key: 'vmware.cluster.status[{$VMWARE.URL},{#CLUSTER.NAME}]'
          history: 7d
          username: '{$VMWARE.USERNAME}'
          password: '{$VMWARE.PASSWORD}'
          description: 'VMware cluster status.'
          valuemap:
            name: 'VMware status'
          tags:
            - tag: Application
              value: VMware
    - uuid: 8fb6a45cbe074b0cb6df53758e2c6623
      name: 'Discover VMware datastores'
      type: SIMPLE
      key: 'vmware.datastore.discovery[{$VMWARE.URL}]'
      delay: 1h
      username: '{$VMWARE.USERNAME}'
      password: '{$VMWARE.PASSWORD}'
      item_prototypes:
        - uuid: 4b61838ba4c34e709b25081ae5b059b5
          name: 'VMware: Average read latency of the datastore {#DATASTORE}'
          type: SIMPLE
          key: 'vmware.datastore.read[{$VMWARE.URL},{#DATASTORE},latency]'
          history: 7d
          username: '{$VMWARE.USERNAME}'
          password: '{$VMWARE.PASSWORD}'
          description: 'Amount of time for a read operation from the datastore (milliseconds).'
          tags:
            - tag: Application
              value: VMware
        - uuid: 5355c401dc244bc588ccd18767577c93
          name: 'VMware: Free space on datastore {#DATASTORE} (percentage)'
          type: SIMPLE
          key: 'vmware.datastore.size[{$VMWARE.URL},{#DATASTORE},pfree]'
          delay: 5m
          history: 7d
          value_type: FLOAT
          units: '%'
          username: '{$VMWARE.USERNAME}'
          password: '{$VMWARE.PASSWORD}'
          description: 'VMware datastore space in percentage from total.'
          tags:
            - tag: Application
              value: VMware
        - uuid: 84f13c4fde2d4a17baaf0c8c1eb4f2c0
          name: 'VMware: Total size of datastore {#DATASTORE}'
          type: SIMPLE
          key: 'vmware.datastore.size[{$VMWARE.URL},{#DATASTORE}]'
          delay: 5m
          history: 7d
          units: B
          username: '{$VMWARE.USERNAME}'
          password: '{$VMWARE.PASSWORD}'
          description: 'VMware datastore space in bytes.'
          tags:
            - tag: Application
              value: VMware
```

```
      - uuid: 540cd0fbc56c4b8ea19f2ff5839ce00d
        name: 'VMware: Average write latency of the datastore {#DATASTORE}'
        type: SIMPLE
        key: 'vmware.datastore.write[{$VMWARE.URL},{#DATASTORE},latency]'
        history: 7d
        username: '{$VMWARE.USERNAME}'
        password: '{$VMWARE.PASSWORD}'
        description: 'Amount of time for a write operation to the datastore (milliseconds).'
        tags:
          - tag: Application
            value: VMware
    - uuid: a5bc075e89f248e7b411d8f960897a08
      name: 'Discover VMware hypervisors'
      type: SIMPLE
      key: 'vmware.hv.discovery[{$VMWARE.URL}]'
      delay: 1h
      username: '{$VMWARE.USERNAME}'
      password: '{$VMWARE.PASSWORD}'
      description: 'Discovery of hypervisors.'
      host_prototypes:
        - uuid: 051a1469d4d045cbbf818fcc843a352e
          host: '{#HV.UUID}'
          name: '{#HV.NAME}'
          group_links:
            -   group:
                name: Templates/Applications
          group_prototypes:
            - name: '{#CLUSTER.NAME}'
            - name: '{#DATACENTER.NAME}'
          templates:
            - name: 'VMware Hypervisor'
          macros:
            - macro: '{$VMWARE.HV.UUID}'
              value: '{#HV.UUID}'
              description: 'UUID of hypervisor.'
          custom_interfaces: 'YES'
          interfaces:
            - ip: '{#HV.IP}'
    - uuid: 9fd559f4e88c4677a1b874634dd686f5
      name: 'Discover VMware VMs'
      type: SIMPLE
      key: 'vmware.vm.discovery[{$VMWARE.URL}]'
      delay: 1h
      username: '{$VMWARE.USERNAME}'
      password: '{$VMWARE.PASSWORD}'
      description: 'Discovery of guest virtual machines.'
      host_prototypes:
        - uuid: 23b9ae9d6f33414880db1cb107115810
          host: '{#VM.UUID}'
          name: '{#VM.NAME}'
          group_links:
            -   group:
                name: Templates/Applications
          group_prototypes:
            - name: '{#CLUSTER.NAME} (vm)'
            - name: '{#DATACENTER.NAME}/{#VM.FOLDER} (vm)'
            - name: '{#HV.NAME}'
          templates:
            - name: 'VMware Guest'
          macros:
            - macro: '{$VMWARE.VM.UUID}'
              value: '{#VM.UUID}'
```

```
                description: 'UUID of guest virtual machine.'
              custom_interfaces: 'YES'
              interfaces:
                - ip: '{#VM.IP}'
      valuemaps:
        - uuid: 3c59c22905054d42ac4ee8b72fe5f270
          name: 'VMware status'
          mappings:
            - value: '0'
              newvalue: gray
            - value: '1'
              newvalue: green
            - value: '2'
              newvalue: yellow
            - value: '3'
              newvalue: red
```

Element tags

Element tag values are explained in the table below.

Template tags

Element	Element property	Required	Type	Range	Description
template_groups		x			Root element for template groups.
	uuid	x	string		Unique identifier for this template group.
	name	x	string		Template group name.
host_groups		x			Root element for host groups that are used by host prototypes.
	uuid	x	string		Unique identifier for this host group.
	name	x	string		Host group name.
templates		-			Root element for templates.
	uuid	x	string		Unique identifier for this template.
	template	x	string		Unique template name.
	name	-	string		Visible template name.
	description	-	text		Template description.
vendor		-			Root element for template vendor. Element added only if the exported template contains vendor data.
	name	-	string		Template vendor name.
	version	-	string		Template version. For out-of-the-box templates, version is displayed as follows: major version of Zabbix, delimiter ("-"), revision number (increased with each new version of the template, and reset with each major version of Zabbix). For example, 6.4-0, 6.4-3, 7.0-0, 7.0-3.
templates		-			Root element for linked templates.
	name	x	string		Template name.
groups		-			Root element for template groups.
	name	x	string		Template group name.
tags		-			Root element for template tags.
	tag	x	string		Tag name.
	value	-	string		Tag value.
macros		-			Root element for template user macros.
	macro	x	string		User macro name.
	type	-	string	0 - TEXT (default) 1 - SECRET_TEXT 2 - VAULT	Type of the macro.
	value	-	string		User macro value.

Element	Element property	Required	Type	Range	Description
	description	-	`string`		User macro description.
valuemaps		-			Root element for template value maps.
	uuid	x	`string`		Unique identifier for this value map.
	name	x	`string`		Value map name.
	mapping	-			Root element for mappings.
	value	x	`string`		Value of a mapping.
	newvalue	x	`string`		New value of a mapping.

Template item tags

Element	Element property	Required	Type	Range[1]	Description
items		-			Root element for items.
	uuid	x	`string`		Unique identifier for the item.
	name	x	`string`		Item name.
	type	-	`string`	0 - ZABBIX_PASSIVE (default) 2 - TRAP 3 - SIMPLE 5 - INTERNAL 7 - ZABBIX_ACTIVE 10 - EXTERNAL 11 - ODBC 12 - IPMI 13 - SSH 14 - TELNET 15 - CALCULATED 16 - JMX 17 - SNMP_TRAP 18 - DEPENDENT 19 - HTTP_AGENT 20 - SNMP_AGENT 21 - ITEM_TYPE_SCRIPT	Item type.
	snmp_oid	-	`string`		SNMP object ID. Required by SNMP items.
	key	x	`string`		Item key.
	delay	-	`string`	Default: 1m	Update interval of the item. Accepts seconds or a time unit with suffix (30s, 1m, 2h, 1d). Optionally one or more custom intervals can be specified either as flexible intervals or scheduling. Multiple intervals are separated by a semicolon. User macros may be used. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Flexible intervals may be written as two macros separated by a forward slash (e.g. `{$FLEX_INTERVAL}/{$FLEX_PERIOD}`).
	history	-	`string`	Default: 90d	Time period of how long the history data should be stored. A time period using the time suffix, a user macro or LLD macro.
	trends	-	`string`	Default: 365d	Time period of how long the trends data should be stored. A time period using the time suffix, a user macro or LLD macro.

Element	Element property	Required	Type	Range[1]	Description
	status	-	string	0 - ENABLED (default) 1 - DISABLED	Item status.
	value_type	-	string	0 - FLOAT 1 - CHAR 2 - LOG 3 - UNSIGNED (default) 4 - TEXT	Received value type.
	allowed_hosts	-	string		List of IP addresses (comma delimited) of hosts allowed sending data for the item. Used by trapper and HTTP agent items.
	units	-	string		Units of returned values (bps, B, etc).
	params	-	text		Additional parameters depending on the type of the item: - executed script for Script, SSH and Telnet items; - SQL query for database monitor items; - formula for calculated items.
	ipmi_sensor	-	string		IPMI sensor. Used only by IPMI items.
	authtype	-	string	Authentication type for SSH agent items: 0 - PASSWORD (default) 1 - PUBLIC_KEY Authentication type for HTTP agent items: 0 - NONE (default) 1 - BASIC 2 - NTLM	Authentication type. Used only by SSH and HTTP agent items.
	username	-	string		Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. Required by SSH and Telnet items. When used by JMX agent, password should also be specified together with the username or both properties should be left blank.
	password	-	string		Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. When used by JMX agent, username should also be specified together with the password or both properties should be left blank.
	publickey	-	string		Name of the public key file. Required for SSH agent items.
	privatekey	-	string		Name of the private key file. Required for SSH agent items.
	port	-	string		Custom port monitored by the item. Can contain user macros. Used only by SNMP items.
	description	-	text		Item description.

Element	Element property	Required	Type	Range[1]	Description
	inventory_link	-	string	0 - NONE Capitalized host inventory field name. For example: 4 - ALIAS 6 - OS_FULL 14 - HARDWARE etc.	Host inventory field that is populated by the item. Refer to the host inventory page for a list of supported host inventory fields and their IDs.
	logtimefmt	-	string		Format of the time in log entries. Used only by log items.
	jmx_endpoint	-	string		JMX endpoint. Used only by JMX agent items.
	url	-	string		URL string. Required only for HTTP agent items.
	allow_traps	-	string	0 - NO (default) 1 - YES	Allow to populate value as in a trapper item. Used only by HTTP agent items.
	follow_redirects	-	string	0 - NO 1 - YES (default)	Follow HTTP response redirects while polling data. Used only by HTTP agent items.
headers		-			Root element for HTTP(S) request headers, where header name is used as key and header value as value. Used only by HTTP agent items.
	name	x	string		Header name.
	value	x	string		Header value.
	http_proxy	-	string		HTTP(S) proxy connection string. Used only by HTTP agent items.
	output_format	-	string	0 - RAW (default) 1 - JSON	How to process response. Used only by HTTP agent items.
	post_type	-	string	0 - RAW (default) 2 - JSON 3 - XML	Type of post data body. Used only by HTTP agent items.
	posts	-	string		HTTP(S) request body data. Used only by HTTP agent items.
query_fields		-			Root element for query parameters. Used only by HTTP agent items.
	name	x	string		Parameter name.
	value	-	string		Parameter value.
	request_method	-	string	0 - GET (default) 1 - POST 2 - PUT 3 - HEAD	Request method. Used only by HTTP agent items.
	retrieve_mode	-	string	0 - BODY (default) 1 - HEADERS 2 - BOTH	What part of response should be stored. Used only by HTTP agent items.
	ssl_cert_file	-	string		Public SSL Key file path. Used only by HTTP agent items.
	ssl_key_file	-	string		Private SSL Key file path. Used only by HTTP agent items.

Element	Element property	Required	Type	Range[1]	Description
	ssl_key_password		string		Password for SSL Key file.
	status_codes	-	string		Used only by HTTP agent items. Ranges of required HTTP status codes separated by commas. Supports user macros. Example: 200,200-{$M},{$M},200-400
	timeout	-	string		Used only by HTTP agent items. Item data polling request timeout. Supports user macros.
	verify_host	-	string	0 - NO (default) 1 - YES	Used by HTTP agent and Script items. Whether to validate that the host name for the connection matches the one in the host's certificate.
	verify_peer	-	string	0 - NO (default) 1 - YES	Used only by HTTP agent items. Whether to validate that the host's certificate is authentic.
parameters		-			Used only by HTTP agent items. Root element for user-defined parameters.
	name	x	string		Used only by Script items. Parameter name.
	value	-	string		Used only by Script items. Parameter value.
value map		-			Used only by Script items. Value map.
	name	x	string		Name of the value map to use for the item.
preprocessing		-			Root element for item value preprocessing.
step		-			Individual item value preprocessing step.

Element	Element property	Required	Type	Range[1]	Description
	type	x	`string`	1 - MULTIPLIER 2 - RTRIM 3 - LTRIM 4 - TRIM 5 - REGEX 6 - BOOL_TO_DECIMAL 7 - OCTAL_TO_DECIMAL 8 - HEX_TO_DECIMAL 9 - SIMPLE_CHANGE (calculated as (received value-previous value)) 10 - CHANGE_PER_SECOND (calculated as (received value-previous value)/(time now-time of last check)) 11 - XMLPATH 12 - JSONPATH 13 - IN_RANGE 14 - MATCHES_REGEX 15 - NOT_MATCHES_REGEX 16 - CHECK_JSON_ERROR 17 - CHECK_XML_ERROR 18 - CHECK_REGEX_ERROR 19 - DISCARD_UNCHANGED 20 - DIS-CARD_UNCHANGED_HEARTBEAT 21 - JAVASCRIPT 22 - PROMETHEUS_PATTERN 23 - PROMETHEUS_TO_JSON 24 - CSV_TO_JSON 25 - STR_REPLACE 26 - CHECK_NOT_SUPPORTED	Type of the item value preprocessing step.
	parameters	-			Root element for parameters of the item value preprocessing step.
	parameter	x	`string`		Individual parameter of the item value preprocessing step.
	error_handler	-	`string`	0 - ORIGINAL_ERROR (default) 1 - DISCARD_VALUE 2 - CUSTOM_VALUE 3 - CUSTOM_ERROR	Action type used in case of preprocessing step failure.
	error_handler_params		`string`		Error handler parameters used with 'error_handler'.
master_item		-			Individual item master item. Required by dependent items.

Element	Element property	Required	Type	Range[1]	Description
	key	x	`string`		Dependent item master item key value. Recursion up to 3 dependent items and maximum count of dependent items equal to 29999 are allowed.
triggers		-			Root element for simple triggers.
	For trigger element tag values, see template trigger tags.				
tags		-			Root element for item tags.
	tag	x	`string`		Tag name.
	value	-	`string`		Tag value.

Template low-level discovery rule tags

Element	Element property	Required	Type	Range	Description
discovery_rules		-			Root element for low-level discovery rules.
	For most of the element tag values, see element tag values for a regular item. Only the tags that are specific to low-level discovery rules, are described below.				
	type	-	`string`	0 - ZABBIX_PASSIVE (default) 2 - TRAP 3 - SIMPLE 5 - INTERNAL 7 - ZABBIX_ACTIVE 10 - EXTERNAL 11 - ODBC 12 - IPMI 13 - SSH 14 - TELNET 16 - JMX 18 - DEPENDENT 19 - HTTP_AGENT 20 - SNMP_AGENT	Item type.
	lifetime	-	`string`	Default: 30d	Time period after which items that are no longer discovered will be deleted. Seconds, time unit with suffix or user macro.
filter					Individual filter.

Element	Element property	Required	Type	Range	Description
	evaltype	-	`string`	0 - AND_OR (default) 1 - AND 2 - OR 3 - FORMULA	Logic to use for checking low-level discovery rule filter conditions.
	formula	-	`string`		Custom calculation formula for filter conditions.
conditions		-			Root element for filter conditions.
	macro	x	`string`		Low-level discovery macro name.
	value	-	`string`		Filter value: regular expression or global regular expression.
	operator	-	`string`	8 - MATCHES_REGEX (default) 9 - NOT_MATCHES_REGEX	Condition operator.
	formulaid	x	`character`		Arbitrary unique ID that is used to reference a condition from the custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
lld_macro_paths		-			Root element for LLD macro paths.
	lld_macro	x	`string`		Low-level discovery macro name.
	path	x	`string`		Selector for value which will be assigned to the corresponding macro.
preprocessing		-			LLD rule value preprocessing.
step		-			Individual LLD rule value preprocessing step.
	For most of the element tag values, see element tag values for a template item value preprocessing. Only the tags that are specific to template low-level discovery value preprocessing, are described below.				

Element	Element property	Required	Type	Range	Description
	type	x	`string`	5 - REGEX 11 - XMLPATH 12 - JSONPATH 15 - NOT_MATCHES_REGEX 16 - CHECK_JSON_ERROR 17 - CHECK_XML_ERROR 20 - DISCARD_UNCHANGED_HEARTBEAT 21 - JAVASCRIPT 23 - PROMETHEUS_TO_JSON 24 - CSV_TO_JSON 25 - STR_REPLACE	Type of the item value preprocessing step.
trigger_prototypes *For trigger prototype element tag values, see regular template trigger tags.*		-			Root element for trigger prototypes.
graph_prototypes *For graph prototype element tag values, see regular template graph tags.*		-			Root element for graph prototypes.
host_prototypes *For host prototype element tag values, see regular host tags.*		-			Root element for host prototypes.
item_prototypes *For item prototype element tag values, see regular template item tags.*		-			Root element for item prototypes.
master_item		-			Individual item prototype master item/item prototype data.
	key	x	`string`		Dependent item prototype master item/item prototype key value. Required for a dependent item.

Template trigger tags

Element	Element property	Required	Type	Range[1]	Description
triggers		-			Root element for triggers.
	uuid	x	string		Unique identifier for this trigger.
	expression	x	string		Trigger expression.
	recovery_mode	-	string	0 - EXPRESSION (default) 1 - RECOVERY_EXPRESSION 2 - NONE	Basis for generating OK events.
	recovery_expression	-	string		Trigger recovery expression.
	correlation_mode	-	string	0 - DISABLED (default) 1 - TAG_VALUE	Correlation mode (no event correlation or event correlation by tag).
	correlation_tag	-	string		The tag name to be used for event correlation.
	name	x	string		Trigger name.
	event_name	-	string		Event name.
	opdata	-	string		Operational data.
	url_name	-	string		Label for the URL associated with the trigger.
	url	-	string		URL associated with the trigger.
	status	-	string	0 - ENABLED (default) 1 - DISABLED	Trigger status.
	priority	-	string	0 - NOT_CLASSIFIED (default) 1 - INFO 2 - WARNING 3 - AVERAGE 4 - HIGH 5 - DISASTER	Trigger severity.
	description	-	text		Trigger description.
	type	-	string	0 - SINGLE (default) 1 - MULTIPLE	Event generation type (single problem event or multiple problem events).
	manual_close	-	string	0 - NO (default) 1 - YES	Manual closing of problem events.
dependencies		-			Root element for dependencies.
	name	x	string		Dependency trigger name.
	expression	x	string		Dependency trigger expression.
	recovery_expression		string		Dependency trigger recovery expression.
tags		-			Root element for trigger tags.
	tag	x	string		Tag name.
	value	-	string		Tag value.

Template graph tags

Element	Element property	Required	Type	Range[1]	Description
graphs		-			Root element for graphs.
	uuid	x	string		Unique identifier for this graph.
	name	x	string		Graph name.
	width	-	integer	20-65535 (default: 900)	Graph width, in pixels. Used for preview and for pie/exploded graphs.
	height	-	integer	20-65535 (default: 200)	Graph height, in pixels. Used for preview and for pie/exploded graphs.
	yaxismin	-	double	Default: 0	Value of Y axis minimum. Used if 'ymin_type_1' is FIXED.
	yaxismax	-	double	Default: 0	Value of Y axis maximum. Used if 'ymax_type_1' is FIXED.

Element	Element property	Required	Type	Range[1]	Description
	show_work_period	-	string	0 - NO 1 - YES (default)	Highlight non-working hours. Used by normal and stacked graphs.
	show_triggers	-	string	0 - NO 1 - YES (default)	Display simple trigger values as a line. Used by normal and stacked graphs.
	type	-	string	0 - NORMAL (default) 1 - STACKED 2 - PIE 3 - EXPLODED	Graph type.
	show_legend	-	string	0 - NO 1 - YES (default)	Display graph legend.
	show_3d	-	string	0 - NO (default) 1 - YES	Enable 3D style. Used by pie and exploded pie graphs.
	percent_left	-	double	Default:0	Show the percentile line for left axis. Used only for normal graphs.
	percent_right	-	double	Default:0	Show the percentile line for right axis. Used only for normal graphs.
	ymin_type_1	-	string	0 - CALCULATED (default) 1 - FIXED 2 - ITEM	Minimum value of Y axis. Used by normal and stacked graphs.
	ymax_type_1	-	string	0 - CALCULATED (default) 1 - FIXED 2 - ITEM	Maximum value of Y axis. Used by normal and stacked graphs.
ymin_item_1		-			Individual item details. Required if 'ymin_type_1' is ITEM.
	host	x	string		Item host.
	key	x	string		Item key.
ymax_item_1		-			Individual item details. Required if 'ymax_type_1' is ITEM.
	host	x	string		Item host.
	key	x	string		Item key.
graph_items		x			Root element for graph items.
	sortorder	-	integer		Draw order. The smaller value is drawn first. Can be used to draw lines or regions behind (or in front of) another.
	drawtype	-	string	0 - SINGLE_LINE (default) 1 - FILLED_REGION 2 - BOLD_LINE 3 - DOTTED_LINE 4 - DASHED_LINE 5 - GRADIENT_LINE	Draw style of the graph item. Used only by normal graphs.
	color	-	string		Element color (6 symbols, hex).
	yaxisside	-	string	0 - LEFT (default) 1 - RIGHT	Side of the graph where the graph item's Y scale will be drawn. Used by normal and stacked graphs.

Element	Element property	Required	Type	Range[1]	Description
	calc_fnc	-	string	1 - MIN 2 - AVG (default) 4 - MAX 7 - ALL (minimum, average and maximum; used only by simple graphs) 9 - LAST (used only by pie and exploded pie graphs)	Data to draw if more than one value exists for an item.
	type	-	string	0 - SIMPLE (default) 2 - GRAPH_SUM (value of the item represents the whole pie; used only by pie and exploded pie graphs)	Graph item type.
item		x			Individual item.
	host	x	string		Item host.
	key	x	string		Item key.

Template web scenario tags

Element	Element property	Required	Type	Range[1]	Description
httptests		-			Root element for web scenarios.
	uuid	x	string		Unique identifier for this web scenario.
	name	x	string		Web scenario name.
	delay	-	string	Default: 1m	Frequency of executing the web scenario. Seconds, time unit with suffix or user macro.
	attempts	-	integer	1-10 (default: 1)	The number of attempts for executing web scenario steps.
	agent	-	string	Default: Zabbix	Client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers.
	http_proxy	-	string		Specify an HTTP proxy to use, using the format: `http://[username[:password]@]proxy.example.co`
variables		-			Root element for scenario-level variables (macros) that may be used in scenario steps.
	name	x	string		Variable name.
	value	x	text		Variable value.
headers		-			Root element for HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol.
	name	x	string		Header name.
	value	x	text		Header value.
	status	-	string	0 - ENABLED (default) 1 - DISABLED	Web scenario status.
	authentication	-	string	0 - NONE (default) 1 - BASIC 2 - NTLM	Authentication method.
	http_user	-	string		User name used for basic, HTTP or NTLM authentication.
	http_password	-	string		Password used for basic, HTTP or NTLM authentication.
	verify_peer	-	string	0 - NO (default) 1 - YES	Whether to validate that the host's certificate is authentic.

Element	Element property	Required	Type	Range[1]	Description
	verify_host	-	string	0 - NO (default) 1 - YES	Whether to validate that the host name for the connection matches the one in the host's certificate.
	ssl_cert_file	-	string		Name of the SSL certificate file used for client authentication (must be in PEM format).
	ssl_key_file	-	string		Name of the SSL private key file used for client authentication (must be in PEM format).
	ssl_key_password	-	string		SSL private key file password.
steps		x			Root element for web scenario steps.
	name	x	string		Web scenario step name.
	url	x	string		URL for monitoring.
query_fields		-			Root element for query fields - an array of HTTP fields that will be added to the URL when performing a request.
	name	x	string		Query field name.
	value	-	string		Query field value.
posts		-			HTTP POST variables as a string (raw post data) or as an array of HTTP fields (form field data).
	name	x	string		Post field name.
	value	x	string		Post field value.
variables		-			Root element of step-level variables (macros) that should be applied after this step. If the variable value has a 'regex:' prefix, then its value is extracted from the data returned by this step according to the regular expression pattern following the 'regex:' prefix
	name	x	string		Variable name.
	value	x	text		Variable value.
headers		-			Root element for HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol.
	name	x	string		Header name.
	value	x	text		Header value.
	follow_redirects	-	string	0 - NO 1 - YES (default)	Follow HTTP redirects.
	retrieve_mode	-	string	0 - BODY (default) 1 - HEADERS 2 - BOTH	HTTP response retrieve mode.
	timeout	-	string	Default: 15s	Timeout of step execution. Seconds, time unit with suffix or user macro.
	required	-	string		Text that must be present in the response. Ignored if empty.
	status_codes	-	string		A comma delimited list of accepted HTTP status codes. Ignored if empty. For example: 200-201,210-299
tags		-			Root element for web scenario tags.
	tag	x	string		Tag name.
	value	-	string		Tag value.

Template dashboard tags

Element	Element property	Required	Type	Range[1]	Description
dashboards		-			Root element for template dashboards.

Element	Element property	Required	Type	Range[1]	Description
	uuid	x	`string`		Unique identifier for this dashboard.
	name	x	`string`		Template dashboard name.
	display period	-	`integer`		Display period of dashboard pages.
	auto_start	-	`string`	0 - no 1 - yes	Slideshow auto start.
pages		-			Root element for template dashboard pages.
	name	-	`string`		Page name.
	display period	-	`integer`		Page display period.
	sortorder	-	`integer`		Page sorting order.
widgets		-			Root element for template dashboard widgets.
	type	x	`string`		Widget type.
	name	-	`string`		Widget name.
	x	-	`integer`	0-23	Horizontal position from the left side of the template dashboard.
	y	-	`integer`	0-62	Vertical position from the top of the template dashboard.
	width	-	`integer`	1-24	Widget width.
	height	-	`integer`	2-32	Widget height.
	hide_header	-	`string`	0 - no 1 - yes	Hide widget header.
fields		-			Root element for the template dashboard widget fields.
	type	x	`string`	0 - INTEGER 1 - STRING 3 - HOST 4 - ITEM 5 - ITEM_PROTOTYPE 6 - GRAPH 7 - GRAPH_PROTOTYPE	Widget field type.
	name	x	`string`		Widget field name.
	value	x	mixed		Widget field value, depending on the field type.

Footnotes

[1] For string values, only the string will be exported (e.g. "ZABBIX_ACTIVE") without the numbering used in this table. The numbers for range values (corresponding to the API values) in this table is used for ordering only.

4 Hosts

Overview

Hosts are exported with many related objects and object relations.

Host export contains:

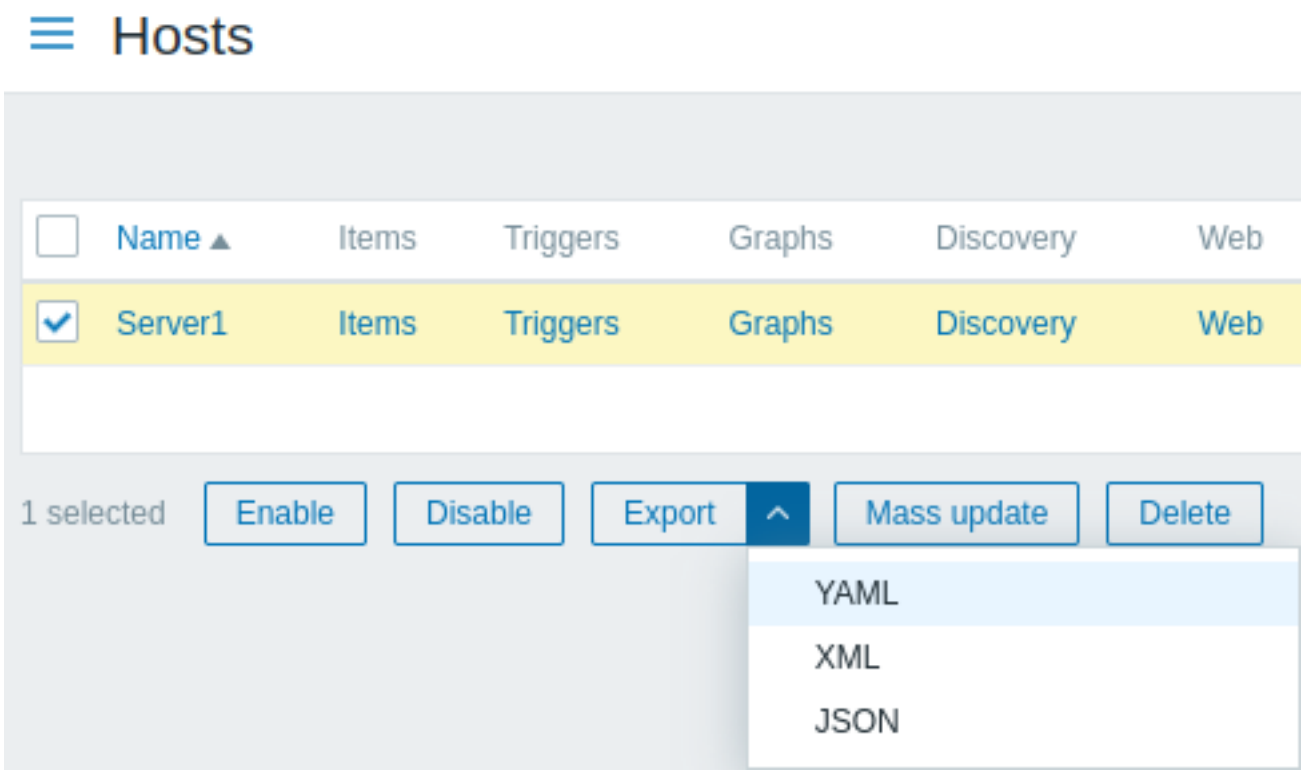
- Linked host groups
- Host data
- Template linkage
- Host group linkage
- Host interfaces
- Directly linked items
- Directly linked triggers
- Directly linked graphs
- Directly linked discovery rules with all prototypes
- Directly linked web scenarios
- Host macros

- Host inventory data
- Value maps

Exporting

To export hosts, do the following:

1. Go to *Data collection* → *Hosts*.
2. Mark the checkboxes of the hosts to export.
3. Click on *Export* below the list.



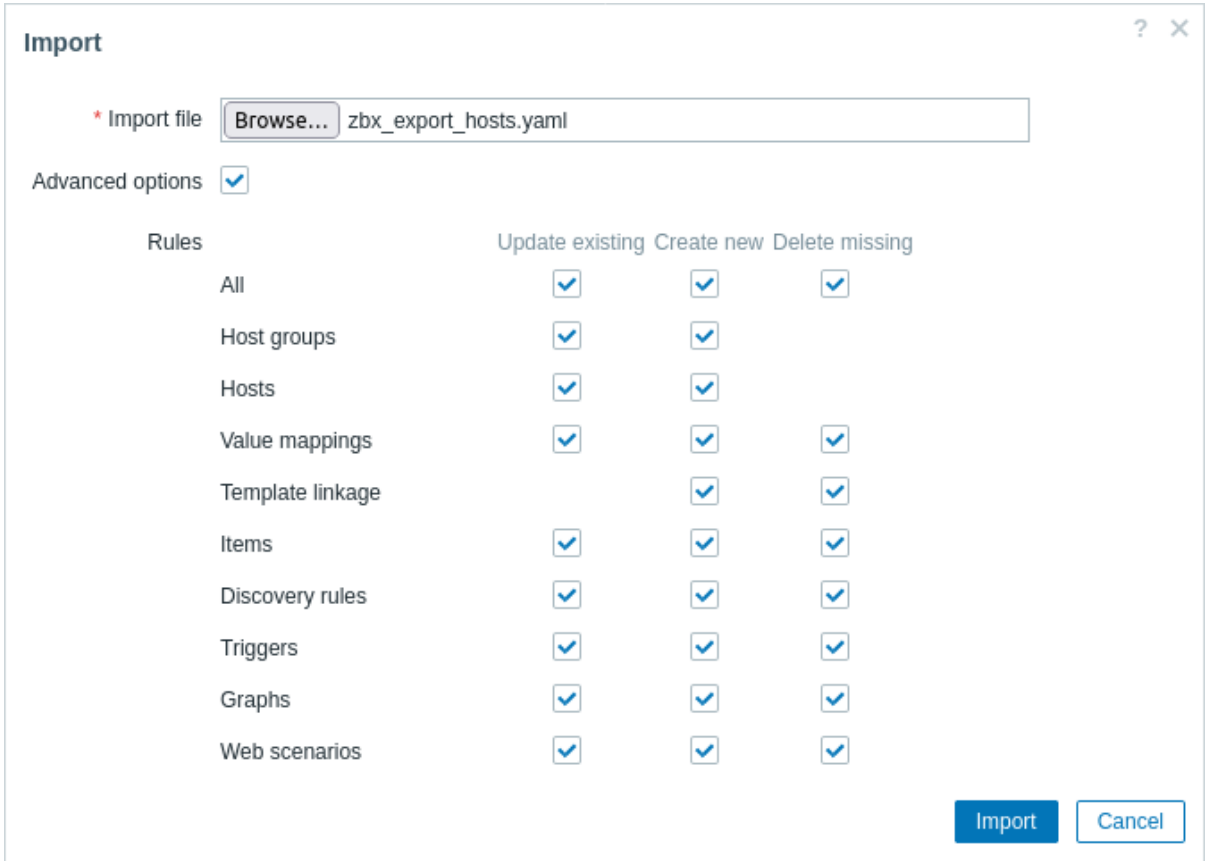
Depending on the selected format, hosts are exported to a local file with a default name:

- *zabbix_export_hosts.yaml* - in YAML export (default option for export);
- *zabbix_export_hosts.xml* - in XML export;
- *zabbix_export_hosts.json* - in JSON export.

Importing

To import hosts, do the following:

1. Go to *Data collection* → *Hosts*.
2. Click on *Import* to the right.
3. Select the import file.
4. Click on *Import*.



If you mark the *Advanced options* checkbox, a detailed list of all importable elements will be displayed - mark or unmark each import rule as required.

If you click the checkbox in the *All* row, all elements below it become marked/unmarked.

Import rules:

Rule	Description
Update existing	Existing elements will be updated with data taken from the import file. Otherwise they will not be updated.
Create new	The import will add new elements using data from the import file. Otherwise it will not add them.
Delete missing	The import will remove existing elements not present in the import file. Otherwise it will not remove them. If *Delete missing* is marked for template linkage, existing template linkage not present in the import file will be unlinked. Entities (items, triggers, graphs, etc.) inherited from the unlinked templates will not be removed (unless the *Delete missing* option is selected for each entity as well).

A success or failure message of the import will be displayed in the frontend.

Export format

Export format in YAML:

```
zabbix_export:
  version: '6.4'
  host_groups:
    - uuid: f2481361f99448eea617b7b1d4765566
      name: 'Discovered hosts'
    - uuid: 6f6799aa69e844b4b3918f779f2abf08
      name: 'Zabbix servers'
  hosts:
    - host: 'Zabbix server 1'
      name: 'Main Zabbix server'
      templates:
        - name: 'Linux by Zabbix agent'
```

```
      - name: 'Zabbix server health'
    groups:
      - name: 'Discovered hosts'
      - name: 'Zabbix servers'
    interfaces:
      - ip: 192.168.1.1
        interface_ref: if1
    items:
      - name: 'Zabbix trap'
        type: TRAP
        key: trap
        delay: '0'
        history: 1w
        preprocessing:
          - type: MULTIPLIER
            parameters:
              - '8'
        tags:
          - tag: Application
            value: 'Zabbix server'
        triggers:
          - expression: 'last(/Zabbix server 1/trap)=0'
            name: 'Last value is zero'
            priority: WARNING
            tags:
              - tag: Process
                value: 'Internal test'
    tags:
      - tag: Process
        value: Zabbix
    macros:
      - macro: '{$HOST.MACRO}'
        value: '123'
      - macro: '{$PASSWORD1}'
        type: SECRET_TEXT
    inventory:
      type: 'Zabbix server'
      name: yyyyyy-HP-Pro-3010-Small-Form-Factor-PC
      os: 'Linux yyyyyy-HP-Pro-3010-Small-Form-Factor-PC 4.4.0-165-generic #193-Ubuntu SMP Tue Sep 17 17
    inventory_mode: AUTOMATIC
graphs:
  - name: 'CPU utilization server'
    show_work_period: 'NO'
    show_triggers: 'NO'
    graph_items:
      - drawtype: FILLED_REGION
        color: FF5555
        item:
          host: 'Zabbix server 1'
          key: 'system.cpu.util[,steal]'
      - sortorder: '1'
        drawtype: FILLED_REGION
        color: 55FF55
        item:
          host: 'Zabbix server 1'
          key: 'system.cpu.util[,softirq]'
      - sortorder: '2'
        drawtype: FILLED_REGION
        color: '009999'
        item:
          host: 'Zabbix server 1'
          key: 'system.cpu.util[,interrupt]'
```

```
      - sortorder: '3'
        drawtype: FILLED_REGION
        color: '990099'
        item:
          host: 'Zabbix server 1'
          key: 'system.cpu.util[,nice]'
      - sortorder: '4'
        drawtype: FILLED_REGION
        color: '999900'
        item:
          host: 'Zabbix server 1'
          key: 'system.cpu.util[,iowait]'
      - sortorder: '5'
        drawtype: FILLED_REGION
        color: '990000'
        item:
          host: 'Zabbix server 1'
          key: 'system.cpu.util[,system]'
      - sortorder: '6'
        drawtype: FILLED_REGION
        color: '000099'
        calc_fnc: MIN
        item:
          host: 'Zabbix server 1'
          key: 'system.cpu.util[,user]'
      - sortorder: '7'
        drawtype: FILLED_REGION
        color: '009900'
        item:
          host: 'Zabbix server 1'
          key: 'system.cpu.util[,idle]'
```

Element tags

Element tag values are explained in the table below.

Host tags

Element	Element property	Required	Type	Range[1]	Description
host_groups		x			Root element for host groups.
	uuid	x	string		Unique identifier for this host group.
	name	x	string		Host group name.
hosts		-			Root element for hosts.
	host	x	string		Unique host name.
	name	-	string		Visible host name.
	description	-	text		Host description.
	status	-	string	0 - ENABLED (default) 1 - DISABLED	Host status.
	ipmi_authtype	-	string	-1 - DEFAULT (default) 0 - NONE 1 - MD2 2 - MD5 4 - STRAIGHT 5 - OEM 6 - RMCP_PLUS	IPMI session authentication type.
	ipmi_privilege	-	string	1 - CALLBACK 2 - USER (default) 3 - OPERATOR 4 - ADMIN 5 - OEM	IPMI session privilege level.
	ipmi_username		string		Username for IPMI checks.
	ipmi_password	-	string		Password for IPMI checks.

Element	Element property	Required	Type	Range[1]	Description
proxy		-			Proxy.
	name	x	string		Name of the proxy (if any) that monitors the host.
templates		-			Root element for linked templates.
	name	x	string		Template name.
interfaces		-			Root element for host interfaces.
	default	-	string	0 - NO 1 - YES (default)	Whether this is the primary host interface. There can be only one primary interface of one type on a host.
	type	-	string	1 - ZABBIX (default) 2 - SNMP 3 - IPMI 4 - JMX	Interface type.
	useip	-	string	0 - NO 1 - YES (default)	Whether to use IP as the interface for connecting to the host (if not, DNS will be used).
	ip	-	string		IP address, can be either IPv4 or IPv6. Required if the connection is made via IP.
	dns	-	string		DNS name. Required if the connection is made via DNS.
	port	-	string		Port number. Supports user macros.
	interface_ref	x	string	Format: `if<N>`	Interface reference name to be used in items.
details		-			Root element for interface details.
	version	-	string	1 - SNMPV1 2 - SNMP_V2C (default) 3 - SNMP_V3	Use this SNMP version.
	community	-	string		SNMP community. Required by SNMPv1 and SNMPv2 items.
	contextname	-	string		SNMPv3 context name. Used only by SNMPv3 items.
	securityname	-	string		SNMPv3 security name. Used only by SNMPv3 items.
	securitylevel	-	string	0 - NOAUTHNOPRIV (default) 1 - AUTHNOPRIV 2 - AUTHPRIV	SNMPv3 security level. Used only by SNMPv3 items.
	authprotocol	-	string	0 - MD5 (default) 1 - SHA1 2 - SHA224 3 - SHA256 4 - SHA384 5 - SHA512	SNMPv3 authentication protocol. Used only by SNMPv3 items.
	authpassphrase	-	string		SNMPv3 authentication passphrase. Used only by SNMPv3 items.
	privprotocol	-	string	0 - DES (default) 1 - AES128 2 - AES192 3 - AES256 4 - AES192C 5 - AES256C	SNMPv3 privacy protocol. Used only by SNMPv3 items.
	privpassphrase		string		SNMPv3 privacy passphrase. Used only by SNMPv3 items.
	bulk	-	string	0 - NO 1 - YES (default)	Use bulk requests for SNMP.

Element	Element property	Required	Type	Range[1]	Description
items		-			Root element for items.
	For item element tag values, see host item tags.				
tags		-			Root element for host tags.
	tag	x	`string`		Tag name.
	value	-	`string`		Tag value.
macros		-			Root element for macros.
	macro	x			User macro name.
	type	-	`string`	0 - TEXT (default) 1 - SECRET_TEXT 2 - VAULT	Type of the macro.
	value	-	`string`		User macro value.
	description	-	`string`		User macro description.
inventory		-			Root element for host inventory.
	<inventory_property>				Individual inventory property. All available inventory properties are listed under the respective tags, e.g. <type>, <name>, <os> (see example above).
inventory_mode		-	`string`	-1 - DISABLED 0 - MANUAL (default) 1 - AUTOMATIC	Inventory mode.
valuemaps		-			Root element for host value maps.
	name	x	`string`		Value map name.
	mapping	-			Root element for mappings.
	value	x	`string`		Value of a mapping.
	newvalue	x	`string`		New value of a mapping.

Host item tags

Element	Element property	Required	Type	Range[1]	Description
items		-			Root element for items.
	name	x	`string`		Item name.
	type	-	`string`	0 - ZABBIX_PASSIVE (default) 2 - TRAP 3 - SIMPLE 5 - INTERNAL 7 - ZABBIX_ACTIVE 10 - EXTERNAL 11 - ODBC 12 - IPMI 13 - SSH 14 - TELNET 15 - CALCULATED 16 - JMX 17 - SNMP_TRAP 18 - DEPENDENT 19 - HTTP_AGENT 20 - SNMP_AGENT 21 - ITEM_TYPE_SCRIPT	Item type.
	snmp_oid	-	`string`		SNMP object ID. Required by SNMP items.
	key	x	`string`		Item key.

Element	Element property	Required	Type	Range[1]	Description
	delay	-	string	Default: 1m	Update interval of the item. Note that `delay` will be always '0' for trapper items. Accepts seconds or a time unit with suffix (30s, 1m, 2h, 1d). Optionally one or more custom intervals can be specified either as flexible intervals or scheduling. Multiple intervals are separated by a semicolon. User macros may be used. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Flexible intervals may be written as two macros separated by a forward slash (e.g. `{$FLEX_INTERVAL}/{$FLEX_PERIOD}`).
	history	-	string	Default: 90d	Time period of how long the history data should be stored. A time period using the time suffix, a user macro or LLD macro.
	trends	-	string	Default: 365d	Time period of how long the trends data should be stored. A time period using the time suffix, a user macro or LLD macro.
	status	-	string	0 - ENABLED (default) 1 - DISABLED	Item status.
	value_type	-	string	0 - FLOAT 1 - CHAR 2 - LOG 3 - UNSIGNED (default) 4 - TEXT	Received value type.
	allowed_hosts	-	string		List of IP addresses (comma delimited) of hosts allowed sending data for the item. Used by trapper and HTTP agent items.
	units	-	string		Units of returned values (bps, B, etc).
	params	-	text		Additional parameters depending on the type of the item: - executed script for Script, SSH and Telnet items; - SQL query for database monitor items; - formula for calculated items.
	ipmi_sensor	-	string		IPMI sensor. Used only by IPMI items.
	authtype	-	string	Authentication type for SSH agent items: 0 - PASSWORD (default) 1 - PUBLIC_KEY Authentication type for HTTP agent items: 0 - NONE (default) 1 - BASIC 2 - NTLM	Authentication type. Used only by SSH and HTTP agent items.

Element	Element property	Required	Type	Range[1]	Description
	username	-	string		Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. Required by SSH and Telnet items. When used by JMX agent, password should also be specified together with the username or both properties should be left blank.
	password	-	string		Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. When used by JMX agent, username should also be specified together with the password or both properties should be left blank.
	publickey	-	string		Name of the public key file. Required for SSH agent items.
	privatekey	-	string		Name of the private key file. Required for SSH agent items.
	description	-	text		Item description.
	inventory_link	-	string	0 - NONE Capitalized host inventory field name. For example: 4 - ALIAS 6 - OS_FULL 14 - HARDWARE etc.	Host inventory field that is populated by the item. Refer to the host inventory page for a list of supported host inventory fields and their IDs.
	logtimefmt	-	string		Format of the time in log entries. Used only by log items.
	interface_ref	-	string	Format: `if<N>`	Reference to the host interface.
	jmx_endpoint	-	string		JMX endpoint. Used only by JMX agent items.
	url	-	string		URL string. Required only for HTTP agent items.
	allow_traps	-	string	0 - NO (default) 1 - YES	Allow to populate value as in a trapper item. Used only by HTTP agent items.
	follow_redirects	-	string	0 - NO 1 - YES (default)	Follow HTTP response redirects while polling data. Used only by HTTP agent items.
headers		-			Root element for HTTP(S) request headers, where header name is used as key and header value as value. Used only by HTTP agent items.
	name	x	string		Header name.
	value	x	string		Header value.
	http_proxy	-	string		HTTP(S) proxy connection string. Used only by HTTP agent items.
	output_format	-	string	0 - RAW (default) 1 - JSON	How to process response. Used only by HTTP agent items.

Element	Element property	Required	Type	Range[1]	Description
	post_type	-	string	0 - RAW (default) 2 - JSON 3 - XML	Type of post data body. Used only by HTTP agent items.
	posts	-	string		HTTP(S) request body data. Used only by HTTP agent items.
query_fields		-			Root element for query parameters. Used only by HTTP agent items.
	name	x	string		Parameter name.
	value	-	string		Parameter value.
	request_method		string	0 - GET (default) 1 - POST 2 - PUT 3 - HEAD	Request method. Used only by HTTP agent items.
	retrieve_mode	-	string	0 - BODY (default) 1 - HEADERS 2 - BOTH	What part of response should be stored. Used only by HTTP agent items.
	ssl_cert_file	-	string		Public SSL Key file path. Used only by HTTP agent items.
	ssl_key_file	-	string		Private SSL Key file path. Used only by HTTP agent items.
	ssl_key_password		string		Password for SSL Key file. Used only by HTTP agent items.
	status_codes	-	string		Ranges of required HTTP status codes separated by commas. Supports user macros. Example: 200,200-{$M},{$M},200-400 Used only by HTTP agent items.
	timeout	-	string		Item data polling request timeout. Supports user macros. Used by HTTP agent and Script items.
	verify_host	-	string	0 - NO (default) 1 - YES	Whether to validate that the host name for the connection matches the one in the host's certificate. Used only by HTTP agent items.
	verify_peer	-	string	0 - NO (default) 1 - YES	Whether to validate that the host's certificate is authentic. Used only by HTTP agent items.
parameters		-			Root element for user-defined parameters. Used only by Script items.
	name	x	string		Parameter name. Used only by Script items.
	value	-	string		Parameter value. Used only by Script items.
value map		-			Value map.
	name	x	string		Name of the value map to use for the item.
preprocessing		-			Root element for item value preprocessing.
step		-			Individual item value preprocessing step.

Element	Element property	Required	Type	Range[1]	Description
	type	x	string	1 - MULTIPLIER 2 - RTRIM 3 - LTRIM 4 - TRIM 5 - REGEX 6 - BOOL_TO_DECIMAL 7 - OCTAL_TO_DECIMAL 8 - HEX_TO_DECIMAL 9 - SIMPLE_CHANGE (calculated as (received value-previous value)) 10 - CHANGE_PER_SECOND (calculated as (received value-previous value)/(time now-time of last check)) 11 - XMLPATH 12 - JSONPATH 13 - IN_RANGE 14 - MATCHES_REGEX 15 - NOT_MATCHES_REGEX 16 - CHECK_JSON_ERROR 17 - CHECK_XML_ERROR 18 - CHECK_REGEX_ERROR 19 - DISCARD_UNCHANGED 20 - DIS-CARD_UNCHANGED_HEARTBEAT 21 - JAVASCRIPT 22 - PROMETHEUS_PATTERN 23 - PROMETHEUS_TO_JSON 24 - CSV_TO_JSON 25 - STR_REPLACE 26 - CHECK_NOT_SUPPORTED 27 - XML_TO_JSON	Type of the item value preprocessing step.
	parameters	-			Root element for parameters of the item value preprocessing step.
	parameter	x	string		Individual parameter of the item value preprocessing step.
	error_handler	-	string	0 - ORIGINAL_ERROR (default) 1 - DISCARD_VALUE 2 - CUSTOM_VALUE 3 - CUSTOM_ERROR	Action type used in case of preprocessing step failure.
	error_handler_params		string		Error handler parameters used with 'error_handler'.
master_item		-			Individual item master item. Required by dependent items.

Element	Element property	Required	Type	Range[1]	Description
	key	x	`string`		Dependent item master item key value. Recursion up to 3 dependent items and maximum count of dependent items equal to 29999 are allowed.
triggers		-			Root element for simple triggers.
	For trigger element tag values, see host trigger tags.				
tags		-			Root element for item tags.
	tag	x	`string`		Tag name.
	value	-	`string`		Tag value.

Host low-level discovery rule tags

Element	Element property	Required	Type	Range[1]	Description
discovery_rules		-			Root element for low-level discovery rules.
	For most of the element tag values, see element tag values for a regular item. Only the tags that are specific to low-level discovery rules, are described below.				
	type	-	`string`	0 - ZABBIX_PASSIVE (default) 2 - TRAP 3 - SIMPLE 5 - INTERNAL 7 - ZABBIX_ACTIVE 10 - EXTERNAL 11 - ODBC 12 - IPMI 13 - SSH 14 - TELNET 16 - JMX 18 - DEPENDENT 19 - HTTP_AGENT 20 - SNMP_AGENT	Item type.
	lifetime	-	`string`	Default: 30d	Time period after which items that are no longer discovered will be deleted. Seconds, time unit with suffix or user macro.
filter					Individual filter.

Element	Element property	Required	Type	Range[1]	Description
	evaltype	-	`string`	0 - AND_OR (default) 1 - AND 2 - OR 3 - FORMULA	Logic to use for checking low-level discovery rule filter conditions.
	formula	-	`string`		Custom calculation formula for filter conditions.
conditions		-			Root element for filter conditions.
	macro	x	`string`		Low-level discovery macro name.
	value	-	`string`		Filter value: regular expression or global regular expression.
	operator	-	`string`	8 - MATCHES_REGEX (default) 9 - NOT_MATCHES_REGEX	Condition operator.
	formulaid	x	`character`		Arbitrary unique ID that is used to reference a condition from the custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
lld_macro_paths		-			Root element for LLD macro paths.
	lld_macro	x	`string`		Low-level discovery macro name.
	path	x	`string`		Selector for value which will be assigned to the corresponding macro.
preprocessing step		-			LLD rule value preprocessing. Individual LLD rule value preprocessing step.
	For most of the element tag values, see element tag values for a host item value preprocessing. Only the tags that are specific to low-level discovery value preprocessing, are described below.				

Element	Element property	Required	Type	Range[1]	Description
	type	x	`string`	5 - REGEX 11 - XMLPATH 12 - JSONPATH 15 - NOT_MATCHES_REGEX 16 - CHECK_JSON_ERROR 17 - CHECK_XML_ERROR 20 - DISCARD_UNCHANGED_HEARTBEAT 21 - JAVASCRIPT 23 - PROMETHEUS_TO_JSON 24 - CSV_TO_JSON 25 - STR_REPLACE 27 - XML_TO_JSON	Type of the item value preprocessing step.
trigger_prototypes *For trigger prototype element tag values, see regular host trigger tags.*		-			Root element for trigger prototypes.
graph_prototypes *For graph prototype element tag values, see regular host graph tags.*		-			Root element for graph prototypes.
host_prototypes *For host prototype element tag values, see regular host tags.*		-			Root element for host prototypes.
item_prototypes *For item prototype element tag values, see regular host item tags.*		-			Root element for item prototypes.
master_item		-			Individual item prototype master item/item prototype data.
	key	x	`string`		Dependent item prototype master item/item prototype key value. Required for a dependent item.

Host trigger tags

Element	Element property	Required	Type	Range[1]	Description
triggers		-			Root element for triggers.
	expression	x	string		Trigger expression.
	recovery_mode	-	string	0 - EXPRESSION (default) 1 - RECOVERY_EXPRESSION 2 - NONE	Basis for generating OK events.
	recovery_expression	-	string		Trigger recovery expression.
	correlation_mode	-	string	0 - DISABLED (default) 1 - TAG_VALUE	Correlation mode (no event correlation or event correlation by tag).
	correlation_tag	-	string		The tag name to be used for event correlation.
	name	x	string		Trigger name.
	event_name	-	string		Event name.
	opdata	-	string		Operational data.
	url_name	-	string		Label for the URL associated with the trigger.
	url	-	string		URL associated with the trigger.
	status	-	string	0 - ENABLED (default) 1 - DISABLED	Trigger status.
	priority	-	string	0 - NOT_CLASSIFIED (default) 1 - INFO 2 - WARNING 3 - AVERAGE 4 - HIGH 5 - DISASTER	Trigger severity.
	description	-	text		Trigger description.
	type	-	string	0 - SINGLE (default) 1 - MULTIPLE	Event generation type (single problem event or multiple problem events).
	manual_close	-	string	0 - NO (default) 1 - YES	Manual closing of problem events.
dependencies		-			Root element for dependencies.
	name	x	string		Dependency trigger name.
	expression	x	string		Dependency trigger expression.
	recovery_expression		string		Dependency trigger recovery expression.
tags		-			Root element for event tags.
	tag	x	string		Tag name.
	value	-	string		Tag value.

Host graph tags

Element	Element property	Required	Type	Range[1]	Description
graphs		-			Root element for graphs.
	name	x	string		Graph name.
	width	-	integer	20-65535 (default: 900)	Graph width, in pixels. Used for preview and for pie/exploded graphs.
	height	-	integer	20-65535 (default: 200)	Graph height, in pixels. Used for preview and for pie/exploded graphs.
	yaxismin	-	double	Default: 0	Value of Y axis minimum. Used if 'ymin_type_1' is FIXED.
	yaxismax	-	double	Default: 0	Value of Y axis maximum. Used if 'ymax_type_1' is FIXED.
	show_work_period	-	string	0 - NO 1 - YES (default)	Highlight non-working hours. Used by normal and stacked graphs.

Element	Element property	Required	Type	Range[1]	Description
	show_triggers	-	string	0 - NO 1 - YES (default)	Display simple trigger values as a line. Used by normal and stacked graphs.
	type	-	string	0 - NORMAL (default) 1 - STACKED 2 - PIE 3 - EXPLODED	Graph type.
	show_legend	-	string	0 - NO 1 - YES (default)	Display graph legend.
	show_3d	-	string	0 - NO (default) 1 - YES	Enable 3D style. Used by pie and exploded pie graphs.
	percent_left	-	double	Default:0	Show the percentile line for left axis. Used only for normal graphs.
	percent_right	-	double	Default:0	Show the percentile line for right axis. Used only for normal graphs.
	ymin_type_1	-	string	0 - CALCULATED (default) 1 - FIXED 2 - ITEM	Minimum value of Y axis. Used by normal and stacked graphs.
	ymax_type_1	-	string	0 - CALCULATED (default) 1 - FIXED 2 - ITEM	Maximum value of Y axis. Used by normal and stacked graphs.
ymin_item_1		-			Individual item details. Required if 'ymin_type_1' is ITEM.
	host	x	string		Item host.
	key	x	string		Item key.
ymax_item_1		-			Individual item details. Required if 'ymax_type_1' is ITEM.
	host	x	string		Item host.
	key	x	string		Item key.
graph_items		x			Root element for graph items.
	sortorder	-	integer		Draw order. The smaller value is drawn first. Can be used to draw lines or regions behind (or in front of) another.
	drawtype	-	string	0 - SINGLE_LINE (default) 1 - FILLED_REGION 2 - BOLD_LINE 3 - DOTTED_LINE 4 - DASHED_LINE 5 - GRADIENT_LINE	Draw style of the graph item. Used only by normal graphs.
	color	-	string		Element color (6 symbols, hex).
	yaxisside	-	string	0 - LEFT (default) 1 - RIGHT	Side of the graph where the graph item's Y scale will be drawn. Used by normal and stacked graphs.

Element	Element property	Required	Type	Range[1]	Description
	calc_fnc	-	string	1 - MIN 2 - AVG (default) 4 - MAX 7 - ALL (minimum, average and maximum; used only by simple graphs) 9 - LAST (used only by pie and exploded pie graphs)	Data to draw if more than one value exists for an item.
	type	-	string	0 - SIMPLE (default) 2 - GRAPH_SUM (value of the item represents the whole pie; used only by pie and exploded pie graphs)	Graph item type.
item		x			Individual item.
	host	x	string		Item host.
	key	x	string		Item key.

Host web scenario tags

Element	Element property	Required	Type	Range[1]	Description
httptests		-			Root element for web scenarios.
	name	x	string		Web scenario name.
	delay	-	string	Default: 1m	Frequency of executing the web scenario. Seconds, time unit with suffix or user macro.
	attempts	-	integer	1-10 (default: 1)	The number of attempts for executing web scenario steps.
	agent	-	string	Default: Zabbix	Client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers.
	http_proxy	-	string		Specify an HTTP proxy to use, using the format: `http://[username[:password]@]proxy.example.co`
variables		-			Root element for scenario-level variables (macros) that may be used in scenario steps.
	name	x	text		Variable name.
	value	x	text		Variable value.
headers		-			Root element for HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol.
	name	x	text		Header name.
	value	x	text		Header value.
	status	-	string	0 - ENABLED (default) 1 - DISABLED	Web scenario status.
	authentication	-	string	0 - NONE (default) 1 - BASIC 2 - NTLM	Authentication method.
	http_user	-	string		User name used for basic, HTTP or NTLM authentication.
	http_password	-	string		Password used for basic, HTTP or NTLM authentication.
	verify_peer	-	string	0 - NO (default) 1 - YES	Whether to validate that the host's certificate is authentic.

Element	Element property	Required	Type	Range[1]	Description
	verify_host	-	string	0 - NO (default) 1 - YES	Whether to validate that the host name for the connection matches the one in the host's certificate.
	ssl_cert_file	-	string		Name of the SSL certificate file used for client authentication (must be in PEM format).
	ssl_key_file	-	string		Name of the SSL private key file used for client authentication (must be in PEM format).
	ssl_key_password	-	string		SSL private key file password.
steps		x			Root element for web scenario steps.
	name	x	string		Web scenario step name.
	url	x	string		URL for monitoring.
query_fields		-			Root element for query fields - an array of HTTP fields that will be added to the URL when performing a request.
	name	x	string		Query field name.
	value	-	string		Query field value.
posts		-			HTTP POST variables as a string (raw post data) or as an array of HTTP fields (form field data).
	name	x	string		Post field name.
	value	x	string		Post field value.
variables		-			Root element of step-level variables (macros) that should be applied after this step. If the variable value has a 'regex:' prefix, then its value is extracted from the data returned by this step according to the regular expression pattern following the 'regex:' prefix
	name	x	string		Variable name.
	value	x	string		Variable value.
headers		-			Root element for HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol.
	name	x	string		Header name.
	value	x	string		Header value.
	follow_redirects	-	string	0 - NO 1 - YES (default)	Follow HTTP redirects.
	retrieve_mode	-	string	0 - BODY (default) 1 - HEADERS 2 - BOTH	HTTP response retrieve mode.
	timeout	-	string	Default: 15s	Timeout of step execution. Seconds, time unit with suffix or user macro.
	required	-	string		Text that must be present in the response. Ignored if empty.
	status_codes	-	string		A comma delimited list of accepted HTTP status codes. Ignored if empty. For example: 200-201,210-299
tags		-			Root element for web scenario tags.
	tag	x	string		Tag name.
	value	-	string		Tag value.

Footnotes

[1] For string values, only the string will be exported (e.g. "ZABBIX_ACTIVE") without the numbering used in this table. The numbers for range values (corresponding to the API values) in this table is used for ordering only.

5 Network maps

Overview

Network map export contains:

- All related images
- Map structure - all map settings, all contained elements with their settings, map links and map link status indicators

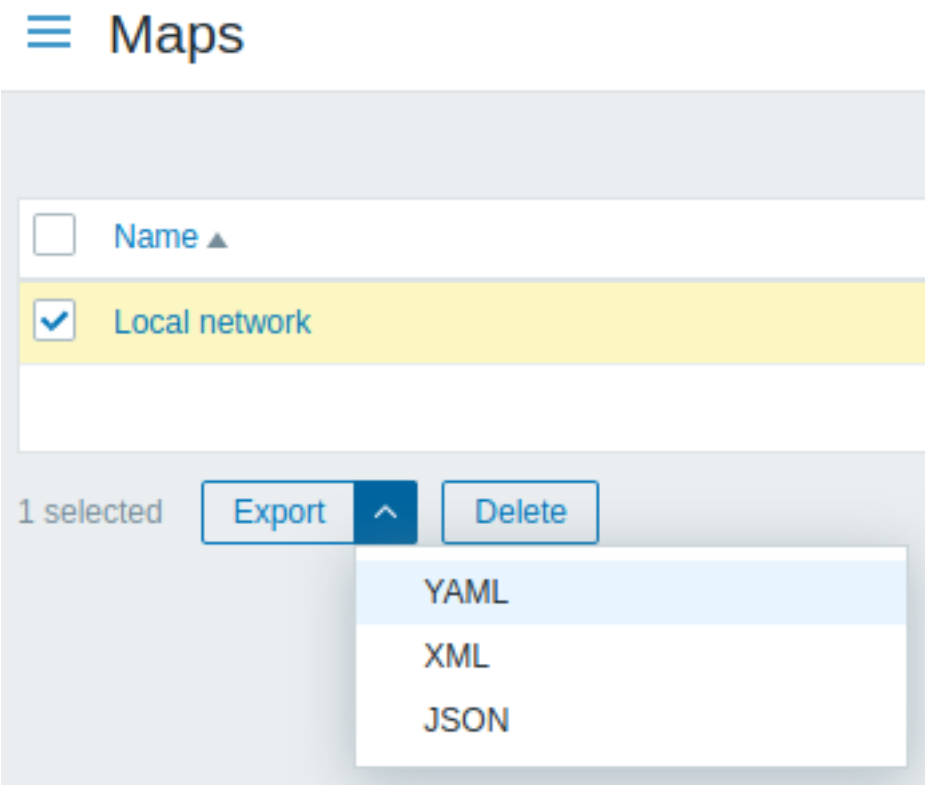
Warning:
Any host groups, hosts, triggers, other maps or other elements that may be related to the exported map are not exported. Thus, if at least one of the elements the map refers to is missing, importing it will fail.

Network map export/import is supported since Zabbix 1.8.2.

Exporting

To export network maps, do the following:

1. Go to *Monitoring* → *Maps*.
2. Mark the checkboxes of the network maps to export.
3. Click on *Export* below the list.



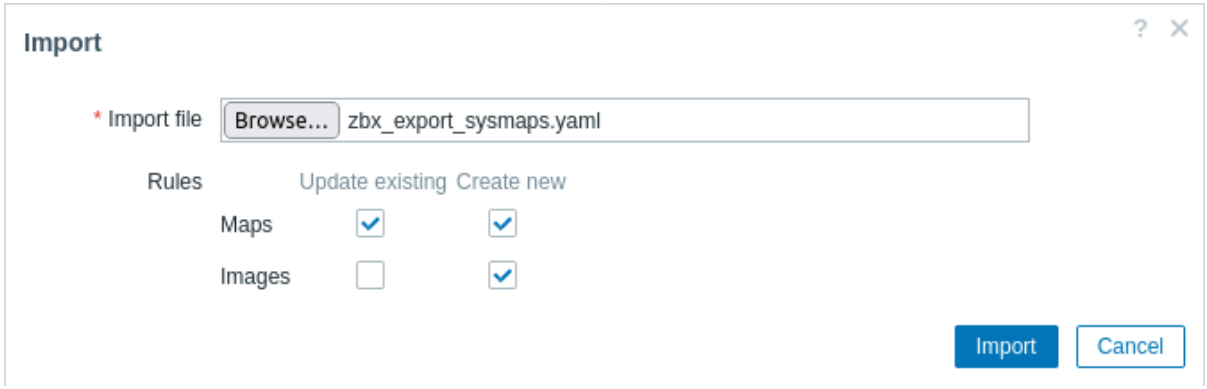
Depending on the selected format, maps are exported to a local file with a default name:

- *zabbix_export_maps.yaml* - in YAML export (default option for export);
- *zabbix_export_maps.xml* - in XML export;
- *zabbix_export_maps.json* - in JSON export.

Importing

To import network maps, do the following:

1. Go to *Monitoring* → *Maps*.
2. Click on *Import* to the right.
3. Select the import file.
4. Mark the required options in import rules.
5. Click on *Import*.



Import rules:

Rule	Description
Update existing	Existing maps will be updated with data taken from the import file. Otherwise they will not be updated.
Create new	The import will add new maps using data from the import file. Otherwise it will not add them.

If you uncheck both map options and check the respective options for images, images only will be imported. Image importing is only available to Super Admin users.

A success or failure message of the import will be displayed in the frontend.

Warning:
If replacing an existing image, it will affect all maps that are using this image.

Export format

Export to YAML:

```yaml
zabbix_export:
  version: '6.4'
  images:
    - name: Zabbix_server_3D_(128)
      imagetype: '1'
      encodedImage: iVBOR...5CYII=
  maps:
    - name: 'Local network'
      width: '680'
      height: '200'
      label_type: '0'
      label_location: '0'
      highlight: '1'
      expandproblem: '1'
      markelements: '1'
      show_unack: '0'
      severity_min: '0'
      show_suppressed: '0'
      grid_size: '50'
      grid_show: '1'
      grid_align: '1'
      label_format: '0'
      label_type_host: '2'
      label_type_hostgroup: '2'
      label_type_trigger: '2'
      label_type_map: '2'
      label_type_image: '2'
      label_string_host: ''
      label_string_hostgroup: ''
      label_string_trigger: ''
```

```
    label_string_map: ''
    label_string_image: ''
    expand_macros: '1'
    background: {  }
    iconmap: {  }
    urls: {  }
    selements:
      - elementtype: '0'
        elements:
          - host: 'Zabbix server'
        label: |
          {HOST.NAME}
          {HOST.CONN}
        label_location: '0'
        x: '111'
        'y': '61'
        elementsubtype: '0'
        areatype: '0'
        width: '200'
        height: '200'
        viewtype: '0'
        use_iconmap: '0'
        selementid: '1'
        icon_off:
          name: Zabbix_server_3D_(128)
        icon_on: {  }
        icon_disabled: {  }
        icon_maintenance: {  }
        urls: {  }
        evaltype: '0'
    shapes:
      - type: '0'
        x: '0'
        'y': '0'
        width: '680'
        height: '15'
        text: '{MAP.NAME}'
        font: '9'
        font_size: '11'
        font_color: '000000'
        text_halign: '0'
        text_valign: '0'
        border_type: '0'
        border_width: '0'
        border_color: '000000'
        background_color: ''
        zindex: '0'
    lines: {  }
    links: {  }
```

Element tags

Element tag values are explained in the table below.

Element	Element property	Type	Range	Description
images				Root element for images.
	name	string		Unique image name.
	imagetype	integer	1 - image 2 - background	Image type.
	encodedImage			Base64 encoded image.
maps				Root element for maps.
	name	string		Unique map name.

Element	Element property	Type	Range	Description
	width	integer		Map width, in pixels.
	height	integer		Map height, in pixels.
	label_type	integer	0 - label 1 - host IP address 2 - element name 3 - status only 4 - nothing	Map element label type.
	label_location	integer	0 - bottom 1 - left 2 - right 3 - top	Map element label location by default.
	highlight	integer	0 - no 1 - yes	Enable icon highlighting for active triggers and host statuses.
	expandproblem	integer	0 - no 1 - yes	Display problem trigger for elements with a single problem.
	markelements	integer	0 - no 1 - yes	Highlight map elements that have recently changed their status.
	show_unack	integer	0 - count of all problems 1 - count of unacknowledged problems 2 - count of acknowledged and unacknowledged problems separately	Problem display.
	severity_min	integer	0 - not classified 1 - information 2 - warning 3 - average 4 - high 5 - disaster	Minimum trigger severity to show on the map by default.
	show_suppressed	integer	0 - no 1 - yes	Display problems which would otherwise be suppressed (not shown) because of host maintenance.
	grid_size	integer	20, 40, 50, 75 or 100	Cell size of a map grid in pixels, if ”grid_show=1”
	grid_show	integer	0 - yes 1 - no	Display a grid in map configuration.
	grid_align	integer	0 - yes 1 - no	Automatically align icons in map configuration.
	label_format	integer	0 - no 1 - yes	Use advanced label configuration.
	label_type_host	integer	0 - label 1 - host IP address 2 - element name 3 - status only 4 - nothing 5 - custom label	Display as host label, if ”label_format=1”
	label_type_hostgroup	integer	0 - label 2 - element name 3 - status only 4 - nothing 5 - custom label	Display as host group label, if ”label_format=1”
	label_type_trigger	integer	0 - label 2 - element name 3 - status only 4 - nothing 5 - custom label	Display as trigger label, if ”label_format=1”

Element	Element property	Type	Range	Description
	label_type_map	integer	0 - label 2 - element name 3 - status only 4 - nothing 5 - custom label	Display as map label, if ”label_format=1”
	label_type_image	integer	0 - label 2 - element name 4 - nothing 5 - custom label	Display as image label, if ”label_format=1”
	label_string_host	string		Custom label for host elements, if ”label_type_host=5”
	label_string_hostgroup	string		Custom label for host group elements, if ”label_type_hostgroup=5”
	label_string_trigger	string		Custom label for trigger elements, if ”label_type_trigger=5”
	label_string_map	string		Custom label for map elements, if ”label_type_map=5”
	label_string_image	string		Custom label for image elements, if ”label_type_image=5”
	expand_macros	integer	0 - no 1 - yes	Expand macros in labels in map configuration.
	background	id		ID of the background image (if any), if ”imagetype=2”
	iconmap	id		ID of the icon mapping (if any).
urls				Used by maps or each map element.
	name	string		Link name.
	url	string		Link URL.
	elementtype	integer	0 - host 1 - map 2 - trigger 3 - host group 4 - image	Map item type the link belongs to.
selements				
	elementtype	integer	0 - host 1 - map 2 - trigger 3 - host group 4 - image	Map element type.
	label	string		Icon label.
	label_location	integer	-1 - use map default 0 - bottom 1 - left 2 - right 3 - top	
	x	integer		Location on the X axis.
	y	integer		Location on the Y axis.
	elementsubtype	integer	0 - single host group 1 - all host groups	Element subtype, if ”elementtype=3”
	areatype	integer	0 - same as whole map 1 - custom size	Area size, if ”elementsubtype=1”
	width	integer		Width of area, if ”areatype=1”
	height	integer		Height of area, if ”areatype=1”
	viewtype	integer	0 - place evenly in the area	Area placement algorithm, if ”elementsubtype=1”
	use_iconmap	integer	0 - no 1 - yes	Use icon mapping for this element. Relevant only if iconmapping is activated on map level.
	selementid	id		Unique element record ID.
	evaltype	integer		Evaluation type for tags.
tags				Problem tags (for host and host group elements). If tags are given, only problems with these tags will be displayed on the map.

Element	Element property	Type	Range	Description
	tag	`string`		Tag name.
	value	`string`		Tag value.
	operator	`integer`		Operator.
elements				Zabbix entities that are represented on the map (host, host group, map etc).
	host			
icon_off				Image to use when element is in 'OK' status.
icon_on				Image to use when element is in 'Problem' status.
icon_disabled				Image to use when element is disabled.
icon_maintenance				Image to use when element is in maintenance.
	name	`string`		Unique image name.
shapes				
	type	`integer`	0 - rectangle 1 - ellipse	Shape type.
	x	`integer`		X coordinates of the shape in pixels.
	y	`integer`		Y coordinates of the shape in pixels.
	width	`integer`		Shape width.
	height	`integer`		Shape height.
	border_type	`integer`	0 - none 1 - bold line 2 - dotted line 3 - dashed line	Type of the border for the shape.
	border_width	`integer`		Width of the border in pixels.
	border_color	`string`		Border color represented in hexadecimal code.
	text	`string`		Text inside of shape.
	font	`integer`	0 - Georgia, serif 1 - "Palatino Linotype", "Book Antiqua", Palatino, serif 2 - "Times New Roman", Times, serif 3 - Arial, Helvetica, sans-serif 4 - "Arial Black", Gadget, sans-serif 5 - "Comic Sans MS", cursive, sans-serif 6 - Impact, Charcoal, sans-serif 7 - "Lucida Sans Unicode", "Lucida Grande", sans-serif 8 - Tahoma, Geneva, sans-serif 9 - "Trebuchet MS", Helvetica, sans-serif 10 - Verdana, Geneva, sans-serif 11 - "Courier New", Courier, monospace 12 - "Lucida Console", Monaco, monospace	Text font style.
	font_size	`integer`		Font size in pixels.
	font_color	`string`		Font color represented in hexadecimal code.
	text_halign	`integer`	0 - center 1 - left 2 - right	Horizontal alignment of text.
	text_valign	`integer`	0 - middle 1 - top 2 - bottom	Vertical alignment of text.

Element	Element property	Type	Range	Description
	background_color	string		Background (fill) color represented in hexadecimal code.
	zindex	integer		Value used to order all shapes and lines (z-index).
lines				
	x1	integer		X coordinates of the line point 1 in pixels.
	y1	integer		Y coordinates of the line point 1 in pixels.
	x2	integer		X coordinates of the line point 2 in pixels.
	y2	integer		Y coordinates of the line point 2 in pixels.
	line_type	integer	0 - none 1 - bold line 2 - dotted line 3 - dashed line	Line type.
	line_width	integer		Line width in pixels.
	line_color	string		Line color represented in hexadecimal code.
	zindex	integer		Value used to order all shapes and lines (z-index).
links				Links between map elements.
	drawtype	integer	0 - line 2 - bold line 3 - dotted line 4 - dashed line	Link style.
	color	string		Link color (6 symbols, hex).
	label	string		Link label.
	selementid1	id		ID of one element to connect.
	selementid2	id		ID of the other element to connect.
linktriggers				Link status indicators.
	drawtype	integer	0 - line 2 - bold line 3 - dotted line 4 - dashed line	Link style when trigger is in the 'Problem' state.
	color	string		Link color (6 symbols, hex) when trigger is in the 'Problem' state.
trigger				Trigger used for indicating link status.
	description	string		Trigger name.
	expression	string		Trigger expression.
	recovery_expression	string		Trigger recovery expression.

6 Media types

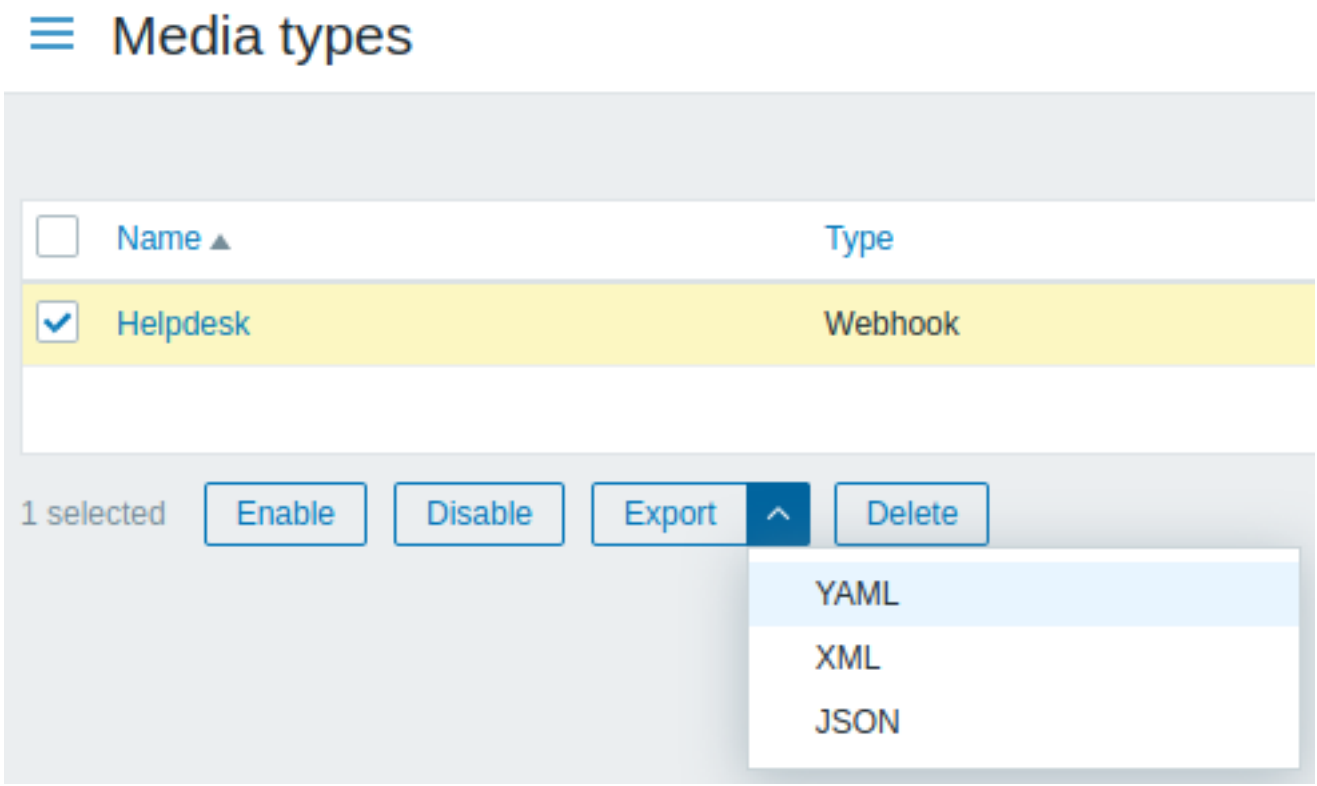
Overview

Media types are exported with all related objects and object relations.

Exporting

To export media types, do the following:

1. Go to *Alerts* → *Media types*.
2. Mark the checkboxes of the media types to export.
3. Click on *Export* below the list.



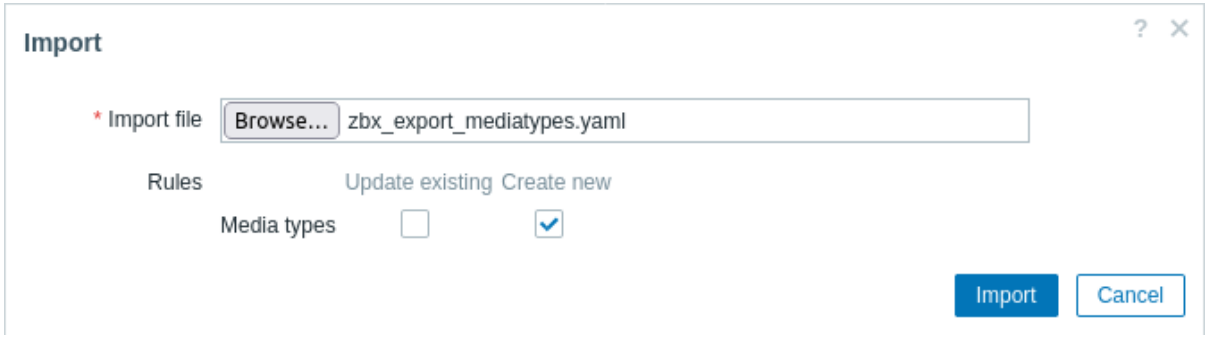
Depending on the selected format, media types are exported to a local file with a default name:

- *zabbix_export_mediatypes.yaml* - in YAML export (default option for export);
- *zabbix_export_mediatypes.xml* - in XML export;
- *zabbix_export_mediatypes.json* - in JSON export.

Importing

To import media types, do the following:

1. Go to *Alerts* → *Media types*.
2. Click on *Import* to the right.
3. Select the import file.
4. Mark the required options in import rules.
5. Click on *Import*.



Import rules:

Rule	Description
Update existing	Existing elements will be updated with data taken from the import file. Otherwise they will not be updated.
Create new	The import will add new elements using data from the import file. Otherwise it will not add them.

A success or failure message of the import will be displayed in the frontend.

Export format

Export to YAML:

```
zabbix_export:
  version: '6.4'
  media_types:
    -
      name: Pushover
      type: WEBHOOK
      parameters:
        - name: endpoint
          value: 'https://api.pushover.net/1/messages.json'
        - name: eventid
          value: '{EVENT.ID}'
        - name: event_nseverity
          value: '{EVENT.NSEVERITY}'
        - name: event_source
          value: '{EVENT.SOURCE}'
        - name: event_value
          value: '{EVENT.VALUE}'
        - name: expire
          value: '1200'
        - name: message
          value: '{ALERT.MESSAGE}'
        - name: priority_average
          value: '0'
        - name: priority_default
          value: '0'
        - name: priority_disaster
          value: '0'
        - name: priority_high
          value: '0'
        - name: priority_information
          value: '0'
        - name: priority_not_classified
          value: '0'
        - name: priority_warning
          value: '0'
        - name: retry
          value: '60'
        - name: title
          value: '{ALERT.SUBJECT}'
        - name: token
          value: '<PUSHOVER TOKEN HERE>'
        - name: triggerid
          value: '{TRIGGER.ID}'
        - name: url
          value: '{$ZABBIX.URL}'
        - name: url_title
          value: Zabbix
        - name: user
          value: '{ALERT.SENDTO}'
      max_sessions: '0'
      script: |
        try {
            var params = JSON.parse(value),
                request = new HttpRequest(),
                data,
                response,
                severities = [
                    {name: 'not_classified', color: '#97AAB3'},
                    {name: 'information', color: '#7499FF'},
                    {name: 'warning', color: '#FFC859'},
                    {name: 'average', color: '#FFA059'},
                    {name: 'high', color: '#E97659'},
```

```
            {name: 'disaster', color: '#E45959'},
            {name: 'resolved', color: '#009900'},
            {name: 'default', color: '#000000'}
        ],
        priority;

    if (typeof params.HTTPProxy === 'string' && params.HTTPProxy.trim() !== '') {
        request.setProxy(params.HTTPProxy);
    }

    if ([0, 1, 2, 3].indexOf(parseInt(params.event_source)) === -1) {
        throw 'Incorrect "event_source" parameter given: "' + params.event_source + '".\nMust be 0
    }

    if (params.event_value !== '0' && params.event_value !== '1'
        && (params.event_source === '0' || params.event_source === '3')) {
        throw 'Incorrect "event_value" parameter given: ' + params.event_value + '\nMust be 0 or 1
    }

    if ([0, 1, 2, 3, 4, 5].indexOf(parseInt(params.event_nseverity)) === -1) {
        params.event_nseverity = '7';
    }

    if (params.event_value === '0') {
        params.event_nseverity = '6';
    }

    priority = params['priority_' + severities[params.event_nseverity].name] || params.priority_de

    if (isNaN(priority) || priority < -2 || priority > 2) {
        throw '"priority" should be -2..2';
    }

    if (params.event_source === '0' && isNaN(params.triggerid)) {
        throw 'field "triggerid" is not a number';
    }

    if (isNaN(params.eventid)) {
        throw 'field "eventid" is not a number';
    }

    if (typeof params.message !== 'string' || params.message.trim() === '') {
        throw 'field "message" cannot be empty';
    }

    data = {
        token: params.token,
        user: params.user,
        title: params.title,
        message: params.message,
        url: (params.event_source === '0')
            ? params.url + '/tr_events.php?triggerid=' + params.triggerid + '&eventid=' + params.e
            : params.url,
        url_title: params.url_title,
        priority: priority
    };

    if (priority == 2) {
        if (isNaN(params.retry) || params.retry < 30) {
            throw 'field "retry" should be a number with value of at least 30 if "priority" is set
        }
```

```
            if (isNaN(params.expire) || params.expire > 10800) {
                throw 'field "expire" should be a number with value of at most 10800 if "priority" is
            }

            data.retry = params.retry;
            data.expire = params.expire;
        }

        data = JSON.stringify(data);
        Zabbix.log(4, '[ Pushover Webhook ] Sending request: ' + params.endpoint + '\n' + data);

        request.addHeader('Content-Type: application/json');
        response = request.post(params.endpoint, data);

        Zabbix.log(4, '[ Pushover Webhook ] Received response with status code ' + request.getStatus()

        if (response !== null) {
            try {
                response = JSON.parse(response);
            }
            catch (error) {
                Zabbix.log(4, '[ Pushover Webhook ] Failed to parse response received from Pushover');
                response = null;
            }
        }

        if (request.getStatus() != 200 || response === null || typeof response !== 'object' || respons
            if (response !== null && typeof response === 'object' && typeof response.errors === 'objec
                    && typeof response.errors[0] === 'string') {
                throw response.errors[0];
            }
            else {
                throw 'Unknown error. Check debug log for more information.';
            }
        }

        return 'OK';
    }
    catch (error) {
        Zabbix.log(4, '[ Pushover Webhook ] Pushover notification failed: ' + error);
        throw 'Pushover notification failed: ' + error;
    }
description: |
    Please refer to setup guide here: https://git.zabbix.com/projects/ZBX/repos/zabbix/browse/template

    Set token parameter with to your Pushover application key.
    When assigning Pushover media to the Zabbix user - add user key into send to field.
message_templates:
    - event_source: TRIGGERS
      operation_mode: PROBLEM
      subject: 'Problem: {EVENT.NAME}'
      message: |
        Problem started at {EVENT.TIME} on {EVENT.DATE}
        Problem name: {EVENT.NAME}
        Host: {HOST.NAME}
        Severity: {EVENT.SEVERITY}
        Operational data: {EVENT.OPDATA}
        Original problem ID: {EVENT.ID}
        {TRIGGER.URL}
    - event_source: TRIGGERS
      operation_mode: RECOVERY
      subject: 'Resolved in {EVENT.DURATION}: {EVENT.NAME}'
```

```
      message: |
        Problem has been resolved at {EVENT.RECOVERY.TIME} on {EVENT.RECOVERY.DATE}
        Problem name: {EVENT.NAME}
        Problem duration: {EVENT.DURATION}
        Host: {HOST.NAME}
        Severity: {EVENT.SEVERITY}
        Original problem ID: {EVENT.ID}
        {TRIGGER.URL}
    - event_source: TRIGGERS
      operation_mode: UPDATE
      subject: 'Updated problem in {EVENT.AGE}: {EVENT.NAME}'
      message: |
        {USER.FULLNAME} {EVENT.UPDATE.ACTION} problem at {EVENT.UPDATE.DATE} {EVENT.UPDATE.TIME}.
        {EVENT.UPDATE.MESSAGE}

        Current problem status is {EVENT.STATUS}, age is {EVENT.AGE}, acknowledged: {EVENT.ACK.STATUS}
    - event_source: DISCOVERY
      operation_mode: PROBLEM
      subject: 'Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}'
      message: |
        Discovery rule: {DISCOVERY.RULE.NAME}

        Device IP: {DISCOVERY.DEVICE.IPADDRESS}
        Device DNS: {DISCOVERY.DEVICE.DNS}
        Device status: {DISCOVERY.DEVICE.STATUS}
        Device uptime: {DISCOVERY.DEVICE.UPTIME}

        Device service name: {DISCOVERY.SERVICE.NAME}
        Device service port: {DISCOVERY.SERVICE.PORT}
        Device service status: {DISCOVERY.SERVICE.STATUS}
        Device service uptime: {DISCOVERY.SERVICE.UPTIME}
    - event_source: AUTOREGISTRATION
      operation_mode: PROBLEM
      subject: 'Autoregistration: {HOST.HOST}'
      message: |
        Host name: {HOST.HOST}
        Host IP: {HOST.IP}
        Agent port: {HOST.PORT}
```

Element tags

Element tag values are explained in the table below.

Element	Element property	Required	Type	Range[1]	Description
media_types		-			Root element for media_types.
	name	x	string		Media type name.
	type	x	string	0 - EMAIL 1 - SMS 2 - SCRIPT 4 - WEBHOOK	Transport used by the media type.
	status	-	string	0 - ENABLED (default) 1 - DISABLED	Whether the media type is enabled.
	max_sessions	-	integer	Possible values for SMS: 1 - (default) Possible values for other media types: 0-100, 0 - unlimited	The maximum number of alerts that can be processed in parallel.
	attempts	-	integer	1-10 (default: 3)	The maximum number of attempts to send an alert.

Element	Element property	Required	Type	Range[1]	Description
	attempt_interval	-	string	0-60s (default: 10s)	The interval between retry attempts. Accepts seconds and time unit with suffix.
	description	-	string		Media type description.
message_templates		-			Root element for media type message templates.
	event_source	x	string	0 - TRIGGERS 1 - DISCOVERY 2 - AUTOREGISTRATION 3 - INTERNAL 4 - SERVICE	Event source.
	operation_mode	x	string	0 - PROBLEM 1 - RECOVERY 2 - UPDATE	Operation mode.
	subject	-	string		Message subject.
	message	-	string		Message body.
Used only by email media type					
	smtp_server	x	string		SMTP server.
	smtp_port	-	integer	Default: 25	SMTP server port to connect to.
	smtp_helo	x	string		SMTP helo.
	smtp_email	x	string		Email address from which notifications will be sent.
	smtp_security	-	string	0 - NONE (default) 1 - STARTTLS 2 - SSL_OR_TLS	SMTP connection security level to use.
	smtp_verify_host	-	string	0 - NO (default) 1 - YES	SSL verify host for SMTP. Optional if smtp_security is STARTTLS or SSL_OR_TLS.
	smtp_verify_peer	-	string	0 - NO (default) 1 - YES	SSL verify peer for SMTP. Optional if smtp_security is STARTTLS or SSL_OR_TLS.
	smtp_authentication	-	string	0 - NONE (default) 1 - PASSWORD	SMTP authentication method to use.
	username	-	string		Username.
	password	-	string		Authentication password.
	content_type	-	string	0 - TEXT 1 - HTML (default)	Message format.
Used only by SMS media type					
	gsm_modem	x	string		Serial device name of the GSM modem.
Used only by script media type					
	script name	x	string		Script name.
parameters		-			Root element for script parameters.
Used only by webhook media type					
	script	x	string		Script.
	timeout	-	string	1-60s (default: 30s)	Javascript script HTTP request timeout interval.
	process_tags	-	string	0 - NO (default) 1 - YES	Whether to process returned tags.
	show_event_menu	-	string	0 - NO (default) 1 - YES	If {EVENT.TAGS.*} were successfully resolved in event_menu_url and event_menu_name fields, this field indicates presence of entry in the event menu.

Element	Element property	Required	Type	Range[1]	Description
	event_menu_url		string		URL of the event menu entry. Supports {EVENT.TAGS.*} macro.
	event_menu_name		string		Name of the event menu entry. Supports {EVENT.TAGS.*} macro.
parameters		-			Root element for webhook media type parameters.
	name	x	string		Webhook parameter name.
	value	-	string		Webhook parameter value.

Footnotes

[1] For string values, only the string will be exported (e.g. "EMAIL") without the numbering used in this table. The numbers for range values (corresponding to the API values) in this table is used for ordering only.

15 Discovery

Please use the sidebar to access content in the Discovery section.

1 Network discovery

Overview

Zabbix offers automatic network discovery functionality that is effective and very flexible.

With network discovery properly set up you can:

- speed up Zabbix deployment
- simplify administration
- use Zabbix in rapidly changing environments without excessive administration

Zabbix network discovery is based on the following information:

- IP ranges
- Availability of external services (FTP, SSH, WEB, POP3, IMAP, TCP, etc)
- Information received from Zabbix agent (only unencrypted mode is supported)
- Information received from SNMP agent

It does NOT provide:

- Discovery of network topology

Network discovery basically consists of two phases: discovery and actions.

Discovery

Zabbix periodically scans the IP ranges defined in network discovery rules. The frequency of the check is configurable for each rule individually.

Note that one discovery rule will always be processed by a single discoverer process. The IP range will not be split between multiple discoverer processes.

Each rule has a set of service checks defined to be performed for the IP range.

Note:
Discovery checks are processed independently from the other checks. If any checks do not find a service (or fail), other checks will still be processed.

Every check of a service and a host (IP) performed by the network discovery module generates a discovery event.

Event	Check of service result
Service Discovered	The service is 'up' after it was 'down' or when discovered for the first time.
Service Up	The service is 'up', after it was already 'up'.
Service Lost	The service is 'down' after it was 'up'.
Service Down	The service is 'down', after it was already 'down'.
Host Discovered	At least one service of a host is 'up' after all services of that host were 'down' or a service is discovered which belongs to a not registered host.
Host Up	At least one service of a host is 'up', after at least one service was already 'up'.
Host Lost	All services of a host are 'down' after at least one was 'up'.
Host Down	All services of a host are 'down', after they were already 'down'.

Actions

Discovery events can be the basis of relevant actions, such as:

- Sending notifications
- Adding/removing hosts
- Enabling/disabling hosts
- Adding hosts to a group
- Removing hosts from a group
- Linking hosts to/unlinking from a template
- Executing remote scripts

These actions can be configured with respect to the device type, IP, status, uptime/downtime, etc. For full details on configuring actions for network-discovery based events, see action operation and conditions pages.

Since network discovery actions are event-based, they will be triggered both when a discovered host is online and when it is offline. It is highly recommended to add an action condition *Discovery status: up* to avoid such actions as *Add host* being triggered upon *Service Lost*/*Service Down* events. Otherwise, if a discovered host is manually removed, it will still generate *Service Lost*/*Service Down* events and will be recreated during the next discovery cycle.

Note:
Linking a discovered host to templates will fail collectively if any of the linkable templates has a unique entity (e.g. item key) that is the same as a unique entity (e.g. item key) already existing on the host or on another of the linkable templates.

Host creation

A host is added if the *Add host* operation is selected. A host is also added, even if the *Add host* operation is missing, if you select operations resulting in actions on a host. Such operations are:

- enable host
- disable host
- add host to a host group
- link template to a host

Created hosts are added to the *Discovered hosts* group (by default, configurable in *Administration* → *General* → *Other*). If you wish hosts to be added to another group, add a *Remove from host groups* operation (specifying "Discovered hosts") and also add an *Add to host groups* operation (specifying another host group), because a host must belong to a host group.

The IP address of the discovered device is the criterion for finding a host in the system. If a host with that IP address and interface type already exists, that host will be the target for performing operations.

If the IP address of the discovered host is changed or the interface is deleted, a new host will be created upon the next discovery.

Host naming

When adding hosts, a host name is the result of reverse DNS lookup or IP address if reverse lookup fails. Lookup is performed from the Zabbix server or Zabbix proxy, depending on which is doing the discovery. If lookup fails on the proxy, it is not retried on the server. If the host with such a name already exists, the next host would get **_2** appended to the name, then **_3** and so on.

It is also possible to override DNS/IP lookup and instead use an item value for host name, for example:

- You may discover multiple servers with Zabbix agent running using a Zabbix agent item for discovery and assign proper names to them automatically, based on the string value returned by this item
- You may discover multiple SNMP network devices using an SNMP agent item for discovery and assign proper names to them automatically, based on the string value returned by this item

If the host name has been set using an item value, it is not updated during the following discovery checks. If it is not possible to set host name using an item value, default value (DNS name) is used.

If a host already exists with the discovered IP address, a new host is not created. However, if the discovery action contains operations (link template, add to host group, etc), they are performed on the existing host.

Host removal

Hosts discovered by a network discovery rule are removed automatically from *Monitoring* → *Discovery* if a discovered entity is not in the rule's IP range any more. Hosts are removed immediately.

Interface creation when adding hosts

When hosts are added as a result of network discovery, they get interfaces created according to these rules:

- the services detected - for example, if an SNMP check succeeded, an SNMP interface will be created
- if a host responded both to Zabbix agent and SNMP requests, both types of interfaces will be created
- if uniqueness criteria are Zabbix agent or SNMP-returned data, the first interface found for a host will be created as the default one. Other IP addresses will be added as additional interfaces. Action's conditions (such as Host IP) do not impact adding interfaces. *Note* that this will work if all interfaces are discovered by the same discovery rule. If a different discovery rule discovers a different interface of the same host, an additional host will be added.
- if a host responded to agent checks only, it will be created with an agent interface only. If it would start responding to SNMP later, additional SNMP interfaces would be added.
- if 3 separate hosts were initially created, having been discovered by the "IP" uniqueness criteria, and then the discovery rule is modified so that hosts A, B and C have identical uniqueness criteria result, B and C are created as additional interfaces for A, the first host. The individual hosts B and C remain. In *Monitoring* → *Discovery* the added interfaces will be displayed in the "Discovered device" column, in black font and indented, but the "Monitored host" column will only display A, the first created host. "Uptime/Downtime" is not measured for IPs that are considered to be additional interfaces.

Changing proxy setting

The hosts discovered by different proxies are always treated as different hosts. While this allows to perform discovery on matching IP ranges used by different subnets, changing proxy for an already monitored subnet is complicated because the proxy changes must be also applied to all discovered hosts.

For example the steps to replace proxy in a discovery rule:

1. disable discovery rule
2. sync proxy configuration
3. replace the proxy in the discovery rule
4. replace the proxy for all hosts discovered by this rule
5. enable discovery rule

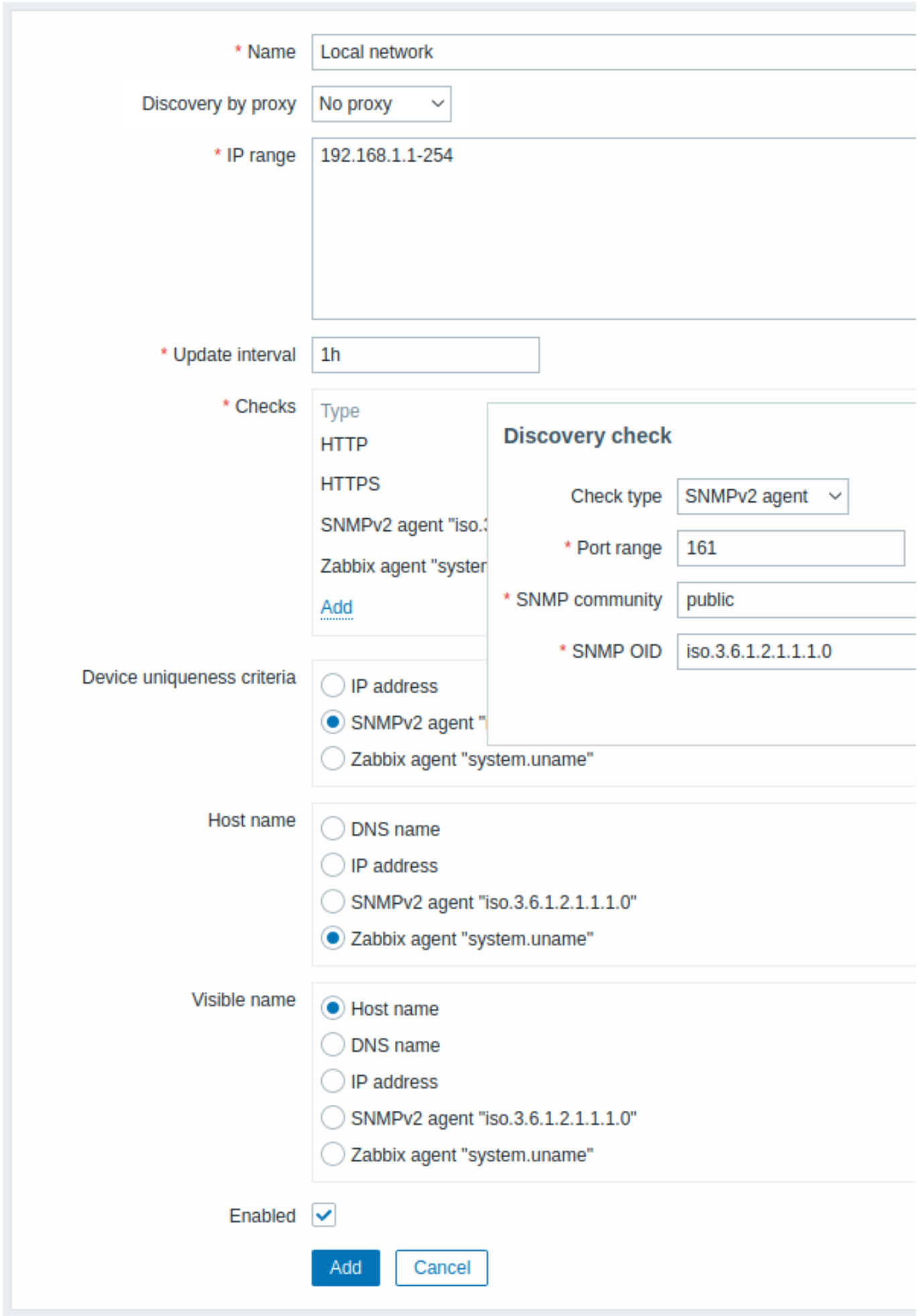
1 Configuring a network discovery rule

Overview

To configure a network discovery rule used by Zabbix to discover hosts and services:

- Go to *Data collection* → *Discovery*
- Click on *Create rule* (or on the rule name to edit an existing one)
- Edit the discovery rule attributes

Rule attributes



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Unique name of the rule. For example, ”Local network”.
Discovery by proxy	What performs discovery: **no proxy** - Zabbix server is doing discovery **<proxy name>** - this proxy performs discovery
IP range	The range of IP addresses for discovery. It may have the following formats: Single IP: 192.168.1.33 Range of IP addresses: 192.168.1-10.1-255. The range is limited by the total number of covered addresses (less than 64K). IP mask: 192.168.4.0/24 supported IP masks: /16 - /30 for IPv4 addresses /112 - /128 for IPv6 addresses List: 192.168.1.1-255, 192.168.2.1-100, 192.168.2.200, 192.168.4.0/24 Since Zabbix 3.0.0 this field supports spaces, tabulation and multiple lines.
Update interval	This parameter defines how often Zabbix will execute the rule. The interval is measured after the execution of previous discovery instance ends so there is no overlap. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. *Note* that if a user macro is used and its value is changed (e.g. 1w → 1h), the next check will be executed according to the previous value (far in the future with the example values).
Checks	Zabbix will use this list of checks for discovery. Click on Add to configure a new check in a popup window. Supported checks: SSH, LDAP, SMTP, FTP, HTTP, HTTPS, POP, NNTP, IMAP, TCP, Telnet, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping. A protocol-based discovery uses the **net.tcp.service[]** functionality to test each host, except for SNMP which queries an SNMP OID. Zabbix agent is tested by querying an item in unencrypted mode. Please see agent items for more details. The ‘Ports’ parameter may be one of following: Single port: 22 Range of ports: 22-45 List: 22-45,55,60-70
Device uniqueness criteria	Uniqueness criteria may be: **IP address** - do not process multiple single-IP devices. If a device with the same IP already exists it will be considered already discovered and a new host will not be added. **<discovery check>** - either Zabbix agent or SNMP agent check. Note that uniqueness criteria used during discovery is not the same as host identification in the system when performing actions. Uniqueness criteria during discovery define whether two or more discovered devices are the same (or different), whereas only the IP address is the criterion for host identification in Zabbix (see Host creation).
Host name	Set the technical host name of a created host using: **DNS name** - DNS name (default) **IP address** - IP address **<discovery check>** - received string value of the discovery check (e.g. Zabbix agent, SNMP agent check) See also: Host naming. This option is supported since 4.2.0.
Visible name	Set the visible host name of a created host using: **Host name** - technical host name (default) **DNS name** - DNS name **IP address** - IP address **<discovery check>** - received string value of the discovery check (e.g. Zabbix agent, SNMP agent check) See also: Host naming. This option is supported since 4.2.0.
Enabled	With the check-box marked the rule is active and will be executed by Zabbix server. If unmarked, the rule is not active. It won’t be executed.

A real life scenario

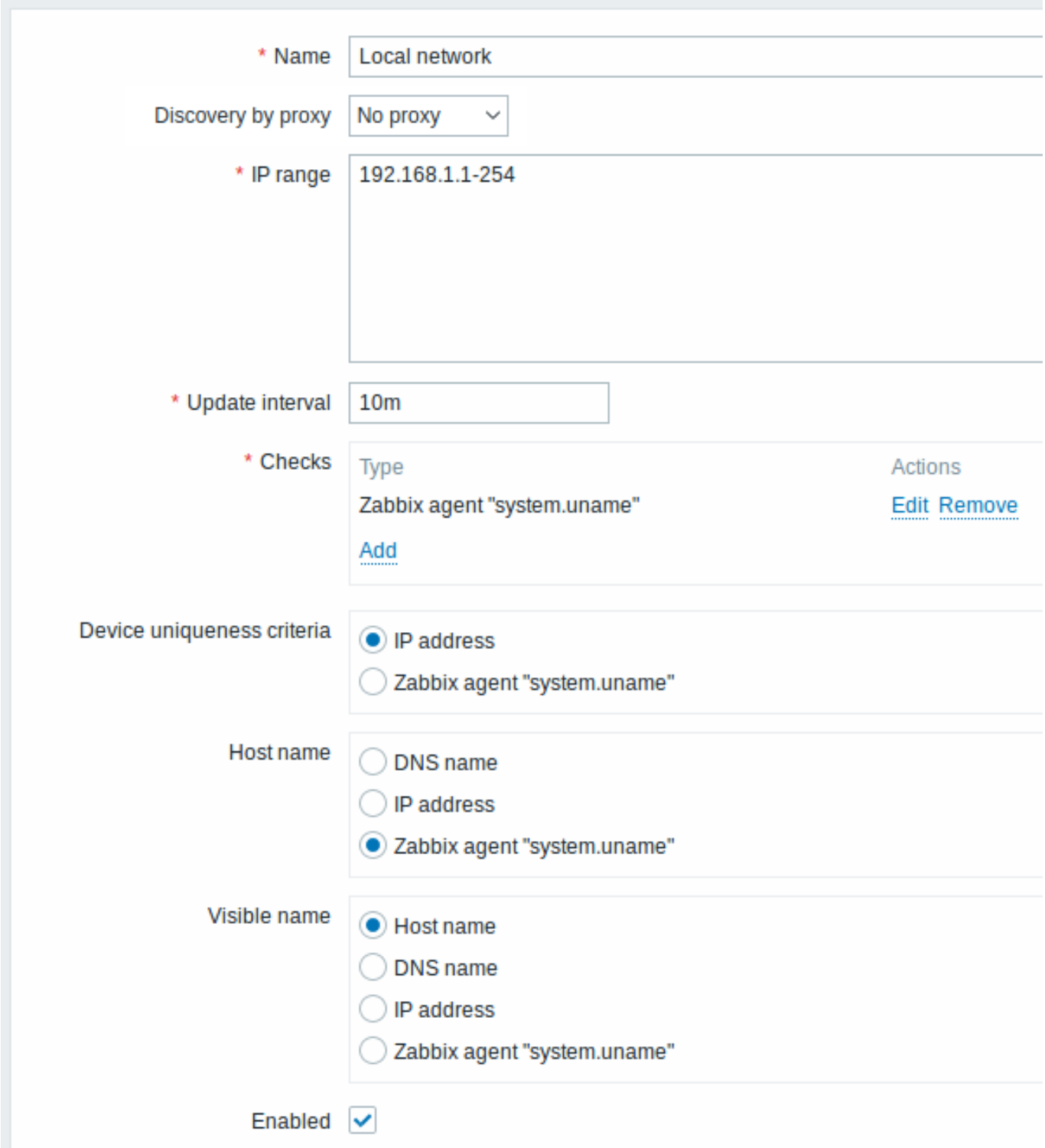
In this example, we would like to set up network discovery for the local network having an IP range of 192.168.1.1-192.168.1.254.

In our scenario we want to:

- discover those hosts that have Zabbix agent running
- run discovery every 10 minutes
- add a host to monitoring if the host uptime is more than 1 hour
- remove hosts if the host downtime is more than 24 hours
- add Linux hosts to the ”Linux servers” group
- add Windows hosts to the ”Windows servers” group
- use the template *Linux* for Linux hosts
- use the template *Windows* for Windows hosts

Step 1

Defining a network discovery rule for our IP range.



Zabbix will try to discover hosts in the IP range of 192.168.1.1-192.168.1.254 by connecting to Zabbix agents and getting the value from the **system.uname** key. The value received from the agent can be used to name the hosts and also to apply different actions

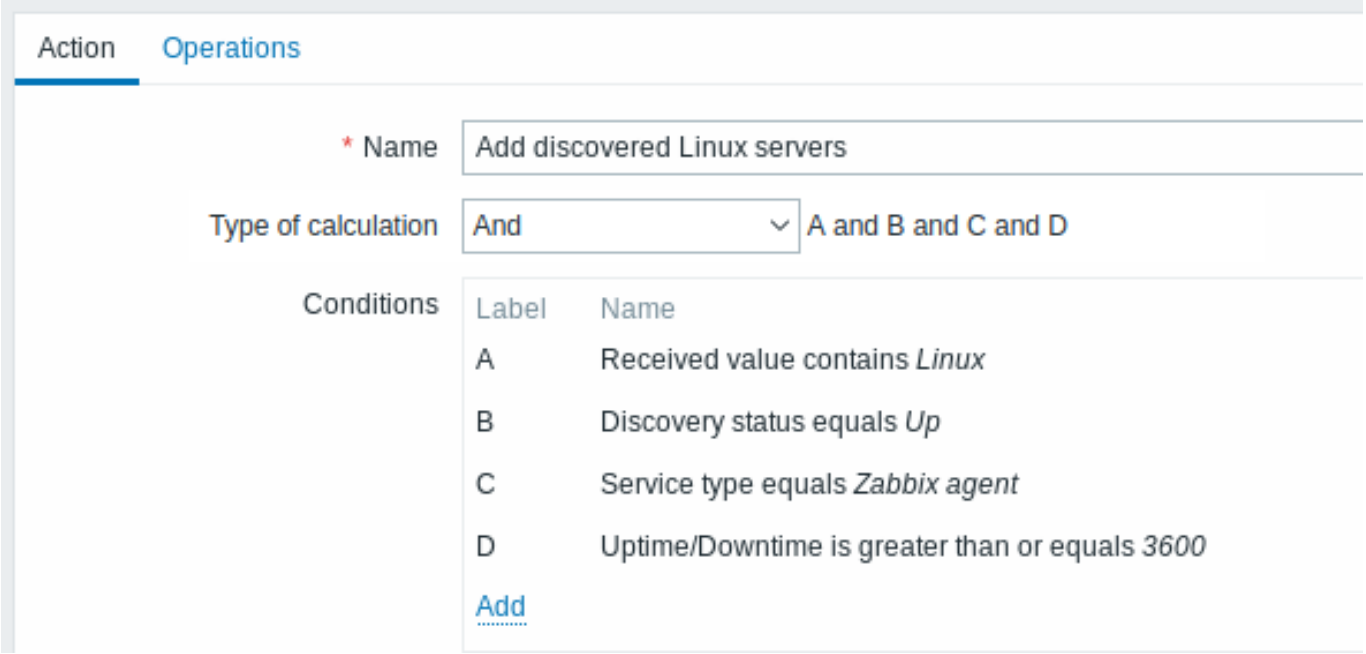
for different operating systems. For example, link Windows servers to the template *Windows*, Linux servers to the template *Linux*.

The rule will be executed every 10 minutes.

When this rule is added, Zabbix will automatically start the discovery and generation of the discovery-based events for further processing.

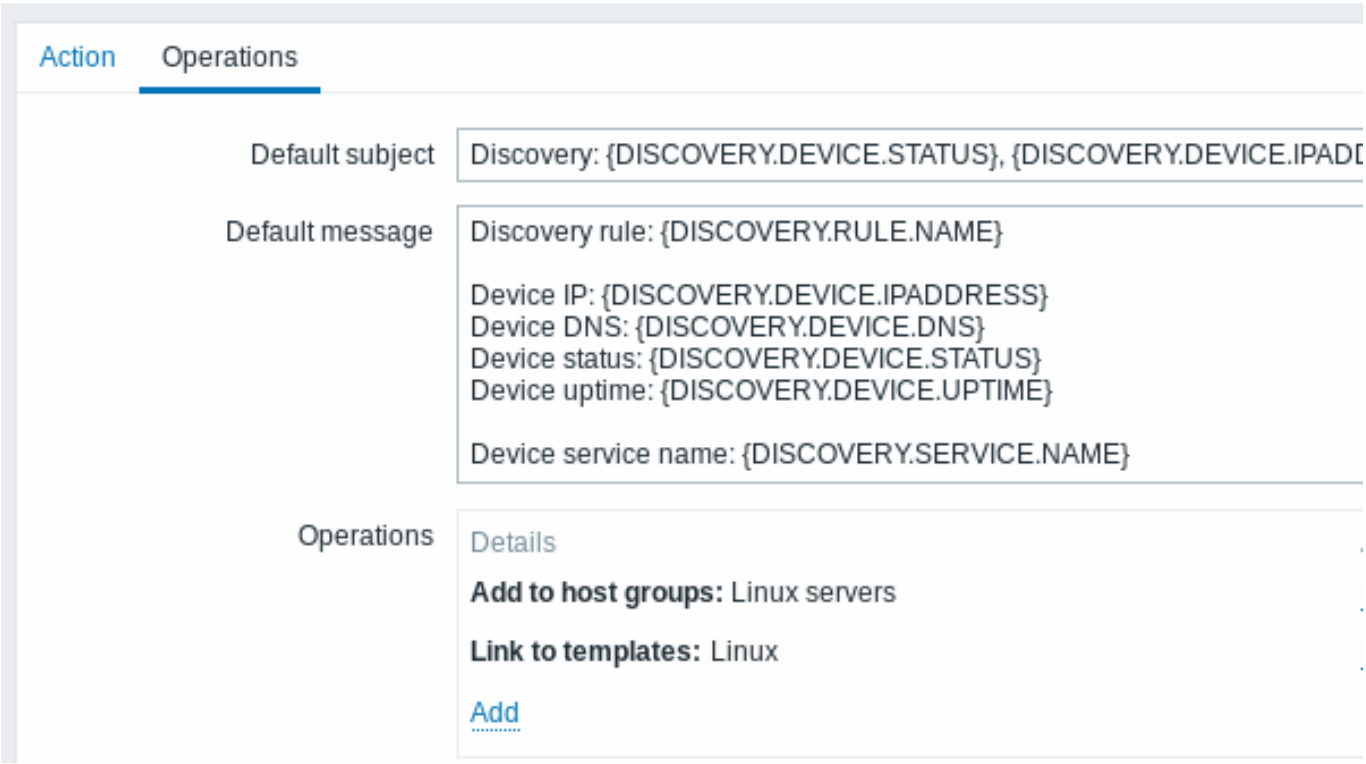
Step 2

Defining a discovery action for adding the discovered Linux servers to the respective group/template.



The action will be activated if:

- the ”Zabbix agent” service is ”up”
- the value of system.uname (the Zabbix agent key we used in rule definition) contains ”Linux”
- Uptime is 1 hour (3600 seconds) or more



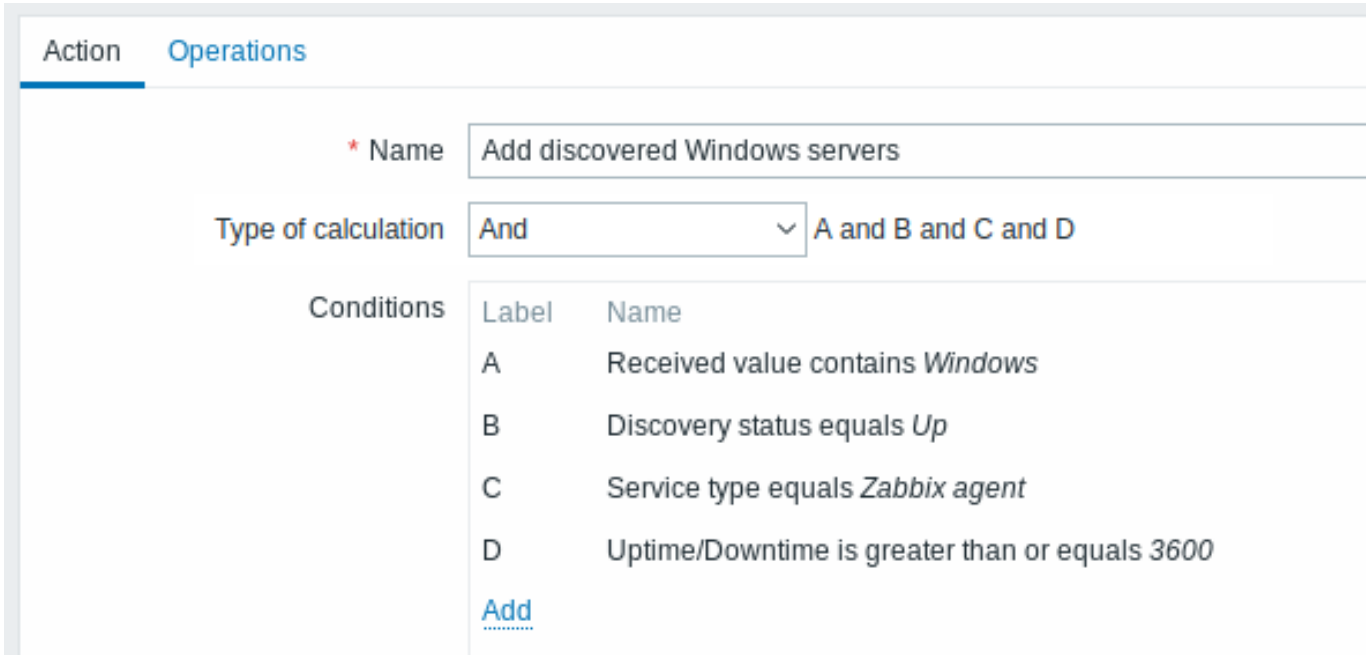
The action will execute the following operations:

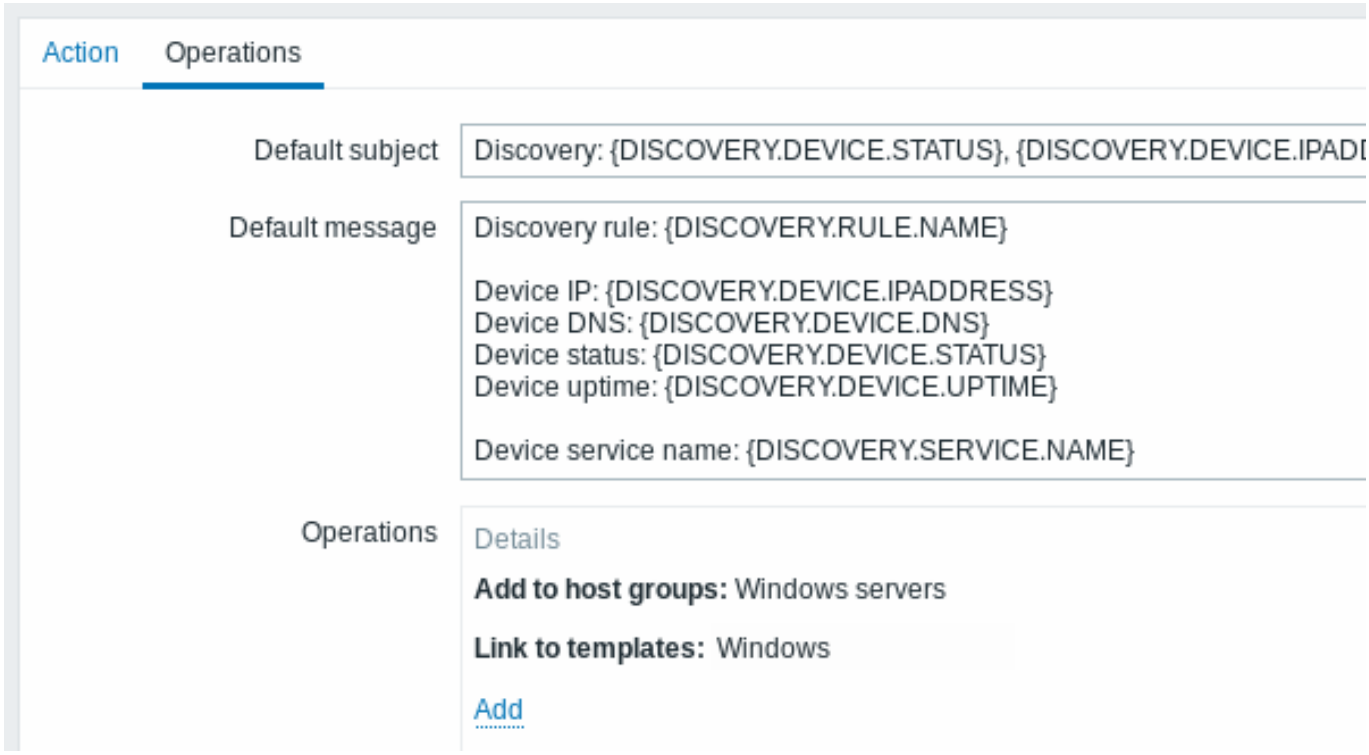
- add the discovered host to the ”Linux servers” group (and also add host if it wasn’t added previously)

- link host to the *Linux* template. Zabbix will automatically start monitoring the host using items and triggers from the "Linux" template.

Step 3

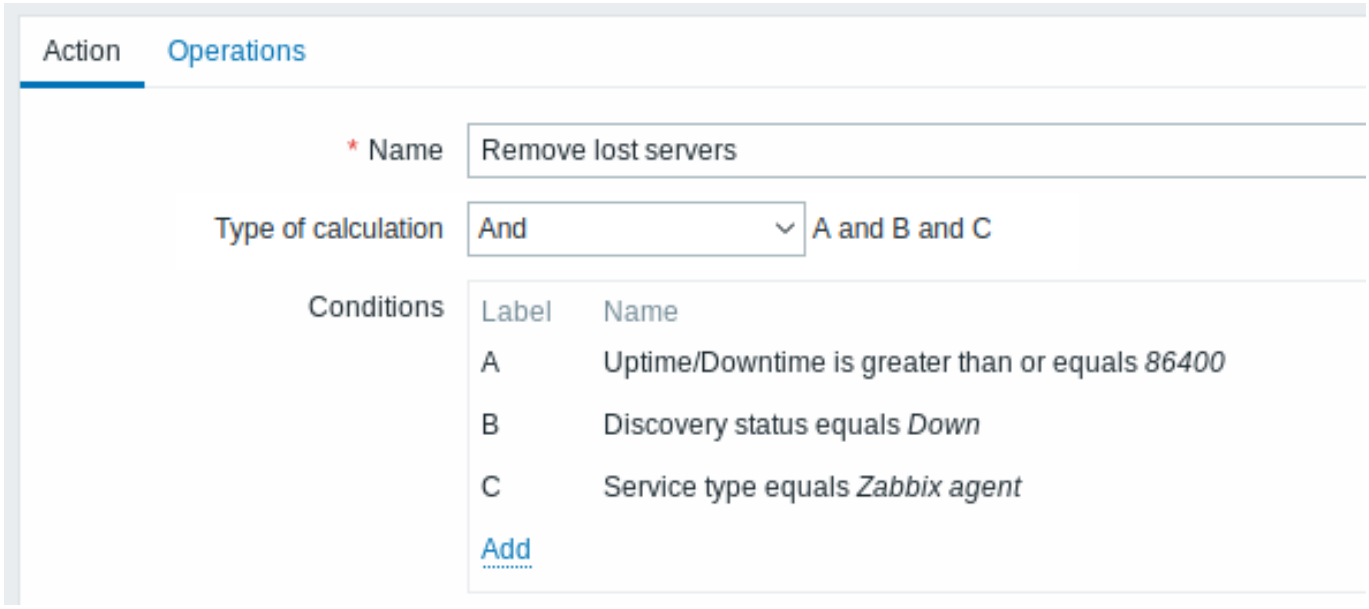
Defining a discovery action for adding the discovered Windows servers to the respective group/template.

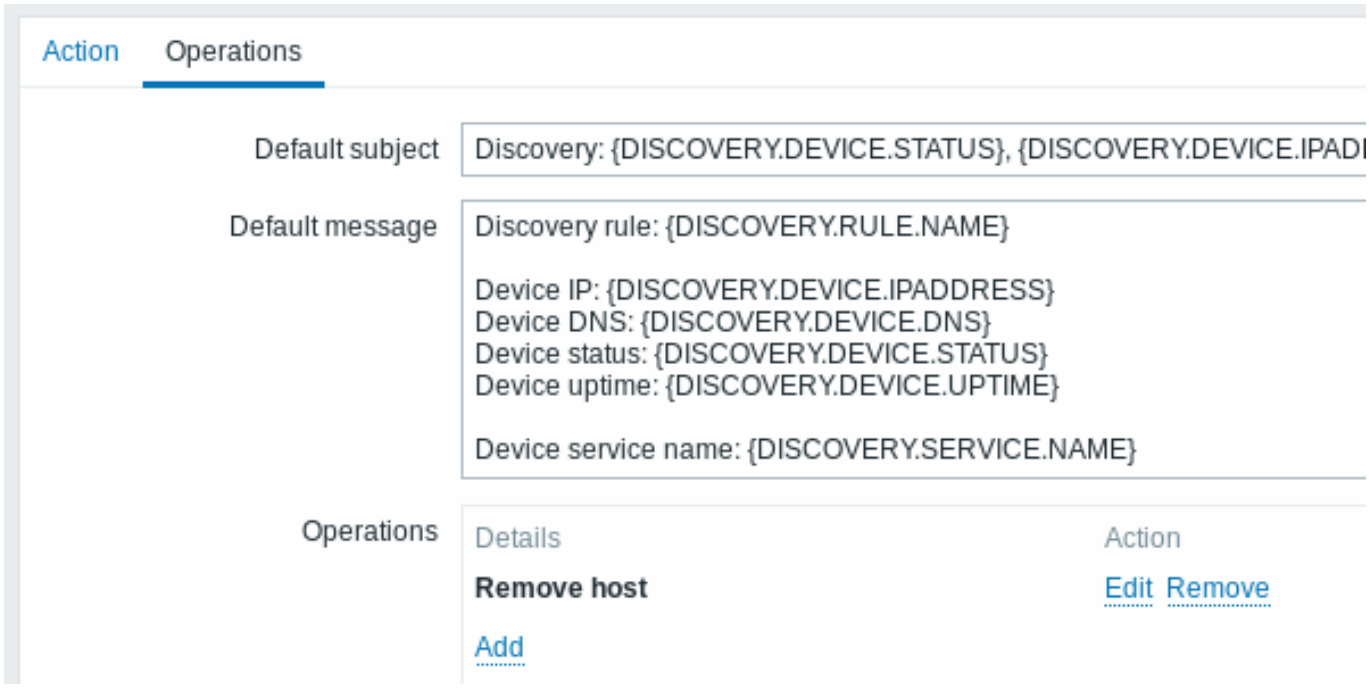




Step 4

Defining a discovery action for removing lost servers.





A server will be removed if ”Zabbix agent” service is ’down’ for more than 24 hours (86400 seconds).

2 Active agent autoregistration

Overview

It is possible to allow active Zabbix agent autoregistration, after which the server can start monitoring them. This way new hosts can be added for monitoring without configuring them manually on the server.

Autoregistration can happen when a previously unknown active agent asks for checks.

The feature might be very handy for automatic monitoring of new Cloud nodes. As soon as you have a new node in the Cloud Zabbix will automatically start the collection of performance and availability data of the host.

Active agent autoregistration also supports the monitoring of added hosts with passive checks. When the active agent asks for checks, providing it has the ’ListenIP’ or ’ListenPort’ configuration parameters defined in the configuration file, these are sent along to the server. (If multiple IP addresses are specified, the first one is sent to the server.)

Server, when adding the new autoregistered host, uses the received IP address and port to configure the agent. If no IP address value is received, the one used for the incoming connection is used. If no port value is received, 10050 is used.

It is possible to specify that the host should be autoregistered with a DNS name as the default agent interface.

Autoregistration is rerun:

- if host metadata information changes:
  - due to HostMetadata changed and agent restarted
  - due to value returned by HostMetadataItem changed
- for manually created hosts with metadata missing
- if a host is manually changed to be monitored by another Zabbix proxy
- if autoregistration for the same host comes from a new Zabbix proxy

The active agent autoregistration heartbeat for Zabbix server and Zabbix proxy is 120 seconds. So in case a discovered host is deleted, the autoregistration will be rerun in 120 seconds.

Configuration

Specify server

Make sure you have the Zabbix server identified in the agent configuration file - zabbix_agentd.conf

```
ServerActive=10.0.0.1
```

Unless you specifically define a *Hostname* in zabbix_agentd.conf, the system hostname of agent location will be used by server for naming the host. The system hostname in Linux can be obtained by running the 'hostname' command.

If *Hostname* is defined in Zabbix agent configuration as a comma-delimited list of hosts, hosts will be created for all listed hostnames.

Restart the agent after making any changes to the configuration file.

Action for active agent autoregistration

When server receives an autoregistration request from an agent it calls an action. An action of event source "Autoregistration" must be configured for agent autoregistration.

Note:
Setting up network discovery is not required to have active agents autoregister.

In the Zabbix frontend, go to *Alerts* → *Actions*, select *Autoregistration actions* and click on *Create action*:

- In the Action tab, give your action a name
- Optionally specify conditions. You can do a substring match or regular expression match in the conditions for host name/host metadata. If you are going to use the "Host metadata" condition, see the next section.
- In the Operations tab, add relevant operations, such as - 'Add host', 'Add to host group' (for example, *Discovered hosts*), 'Link to templates', etc.

Note:
If the hosts that will be autoregistering are likely to be supported for active monitoring only (such as hosts that are firewalled from your Zabbix server) then you might want to create a specific template like *Template_Linux-active* to link to.

Created hosts are added to the *Discovered hosts* group (by default, configurable in *Administration* → *General* → *Other*). If you wish hosts to be added to another group, add a *Remove from host group* operation (specifying "Discovered hosts") and also add an *Add to host group* operation (specifying another host group), because a host must belong to a host group.

Secure autoregistration

A secure way of autoregistration is possible by configuring PSK-based authentication with encrypted connections.

The level of encryption is configured globally in *Administration* → *General* → *Autoregistration*. It is possible to select no encryption, TLS encryption with PSK authentication or both (so that some hosts may register without encryption while others through encryption).

Authentication by PSK is verified by Zabbix server before adding a host. If successful, the host is added and *Connections from/to host* are set to 'PSK' only with identity/pre-shared key the same as in the global autoregistration setting.

Attention:
To ensure security of autoregistration on installations using proxies, encryption between Zabbix server and proxy should be enabled.

Using DNS as default interface

HostInterface and HostInterfaceItem configuration parameters allow to specify a custom value for the host interface during autoregistration.

More specifically, they are useful if the host should be autoregistered with a DNS name as the default agent interface rather than its IP address. In that case the DNS name should be specified or returned as the value of either HostInterface or HostInterfaceItem parameters. Note that if the value of one of the two parameters changes, the autoregistered host interface is updated. So it is possible to update the default interface to another DNS name or update it to an IP address. For the changes to take effect though, the agent has to be restarted.

Note:
If HostInterface or HostInterfaceItem parameters are not configured, the listen_dns parameter is resolved from the IP address. If such resolving is configured incorrectly, it may break autoregistration because of invalid hostname.

Using host metadata

When agent is sending an autoregistration request to the server it sends its hostname. In some cases (for example, Amazon cloud nodes) a hostname is not enough for Zabbix server to differentiate discovered hosts. Host metadata can be optionally used to send other information from an agent to the server.

Host metadata is configured in the agent configuration file - zabbix_agentd.conf. There are 2 ways of specifying host metadata in the configuration file:

`HostMetadata`
`HostMetadataItem`

See the description of the options in the link above.

The HostMetadataItem parameter may return up to 65535 UTF-8 code points. A longer value will be truncated.

Note that on MySQL, the effective maximum length in characters will be less if the returned value contains multibyte characters. For example, a value containing 3-byte characters only will be limited to 21844 characters in total, while a value containing 4-byte characters only will be limited to 16383 symbols.

Attention:
An autoregistration attempt happens every time an active agent sends a request to refresh active checks to the server. The delay between requests is specified in the RefreshActiveChecks parameter of the agent. The first request is sent immediately after the agent is restarted.

Example 1

Using host metadata to distinguish between Linux and Windows hosts.

Say you would like the hosts to be autoregistered by the Zabbix server. You have active Zabbix agents (see "Configuration" section above) on your network. There are Windows hosts and Linux hosts on your network and you have "Linux by Zabbix agent" and "Windows by Zabbix agent" templates available in your Zabbix frontend. So at host registration, you would like the appropriate Linux/Windows template to be applied to the host being registered. By default, only the hostname is sent to the server at autoregistration, which might not be enough. In order to make sure the proper template is applied to the host you should use host metadata.

Frontend configuration

The first thing to do is to configure the frontend. Create 2 actions. The first action:

- Name: Linux host autoregistration
- Conditions: Host metadata contains *Linux*
- Operations: Link templates: Linux

Note:
You can skip an "Add host" operation in this case. Linking a template to a host requires adding the host first so the server will do that automatically.

The second action:

- Name: Windows host autoregistration
- Conditions: Host metadata contains *Windows*
- Operations: Link templates: Windows

Agent configuration

Now you need to configure the agents. Add the next line to the agent configuration files:

```
HostMetadataItem=system.uname
```

This way you make sure host metadata will contain ”Linux” or ”Windows” depending on the host an agent is running on. An example of host metadata in this case:

```
Linux: Linux server3 3.2.0-4-686-pae #1 SMP Debian 3.2.41-2 i686 GNU/Linux
Windows: Windows WIN-0PXGGSTYNHO 6.0.6001 Windows Server 2008 Service Pack 1 Intel IA-32
```

Do not forget to restart the agent after making any changes to the configuration file.

Example 2

Step 1

Using host metadata to allow some basic protection against unwanted hosts registering.

Frontend configuration

Create an action in the frontend, using some hard-to-guess secret code to disallow unwanted hosts:

- Name: Autoregistration action Linux
- Conditions:
  - Type of calculation: AND
  - Condition (A): Host metadata contains //Linux//
  - Condition (B): Host metadata contains //21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae//
- Operations:
  - Send message to users: Admin via all media
  - Add to host groups: Linux servers
  - Link templates: Linux

Please note that this method alone does not provide strong protection because data is transmitted in plain text. Configuration cache reload is required for changes to have an immediate effect.

Agent configuration

Add the next line to the agent configuration file:

```
HostMetadata=Linux      21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

where ”Linux” is a platform, and the rest of the string is the hard-to-guess secret text.

Do not forget to restart the agent after making any changes to the configuration file.

Step 2

It is possible to add additional monitoring for an already registered host.

Frontend configuration

Update the action in the frontend:

- Name: Autoregistration action Linux
- Conditions:
  - Type of calculation: AND
  - Condition (A): Host metadata contains Linux
  - Condition (B): Host metadata contains 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
- Operations:
  - Send message to users: Admin via all media
  - Add to host groups: Linux servers
  - Link templates: Linux
  - Link templates: MySQL by Zabbix Agent

Agent configuration

Update the next line in the agent configuration file:

```
HostMetadata=MySQL on Linux 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
```

Do not forget to restart the agent after making any changes to the configuration file.

3 Low-level discovery

Overview Low-level discovery provides a way to automatically create items, triggers, and graphs for different entities on a computer. For instance, Zabbix can automatically start monitoring file systems or network interfaces on your machine, without the need to create items for each file system or network interface manually. Additionally, it is possible to configure Zabbix to remove unneeded entities automatically based on the actual results of periodically performed discovery.

A user can define their own types of discovery, provided they follow a particular JSON protocol.

The general architecture of the discovery process is as follows.

First, a user creates a discovery rule in *Data collection* → *Templates*, in the *Discovery* column. A discovery rule consists of (1) an item that discovers the necessary entities (for instance, file systems or network interfaces) and (2) prototypes of items, triggers, and graphs that should be created based on the value of that item.

An item that discovers the necessary entities is like a regular item seen elsewhere: the server asks a Zabbix agent (or whatever the type of the item is set to) for a value of that item, the agent responds with a textual value. The difference is that the value the agent responds with should contain a list of discovered entities in a JSON format. While the details of this format are only important for implementers of custom discovery checks, it is necessary to know that the returned value contains a list of macro → value pairs. For instance, item "net.if.discovery" might return two pairs: "{#IFNAME}" → "lo" and "{#IFNAME}" → "eth0".

These macros are used in names, keys and other prototype fields where they are then substituted with the received values for creating real items, triggers, graphs or even hosts for each discovered entity. See the full list of options for using LLD macros.

When the server receives a value for a discovery item, it looks at the macro → value pairs and for each pair generates real items, triggers, and graphs, based on their prototypes. In the example with "net.if.discovery" above, the server would generate one set of items, triggers, and graphs for the loopback interface "lo", and another set for interface "eth0".

Note that since **Zabbix 4.2**, the format of the JSON returned by low-level discovery rules has been changed. It is no longer expected that the JSON will contain the "data" object. Low-level discovery will now accept a normal JSON containing an array, in order to support new features such as the item value preprocessing and custom paths to low-level discovery macro values in a JSON document.

Built-in discovery keys have been updated to return an array of LLD rows at the root of JSON document. Zabbix will automatically extract a macro and value if an array field uses the {#MACRO} syntax as a key. Any new native discovery checks will use the new syntax without the "data" elements. When processing a low-level discovery value first the root is located (array at `$.` or `$.data`).

While the "data" element has been removed from all native items related to discovery, for backward compatibility Zabbix will still accept the JSON notation with a "data" element, though its use is discouraged. If the JSON contains an object with only one "data" array element, then it will automatically extract the content of the element using JSONPath `$.data`. Low-level discovery now accepts optional user-defined LLD macros with a custom path specified in JSONPath syntax.

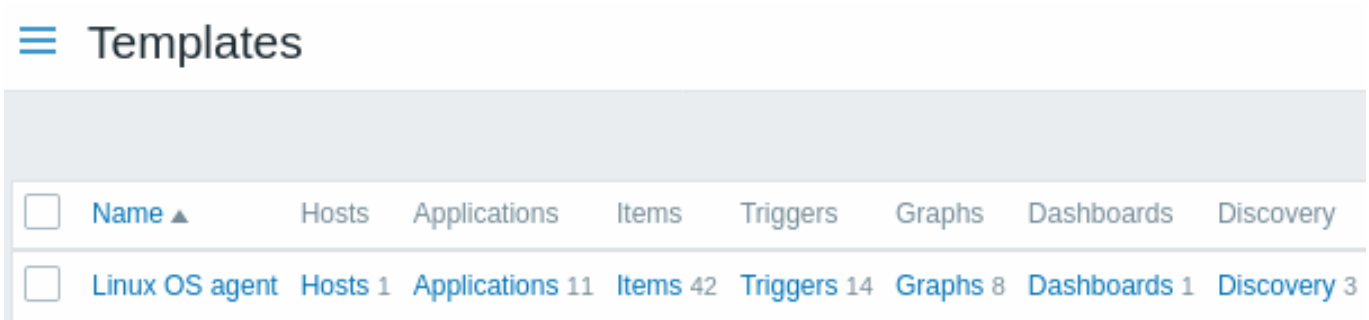
Warning:
As a result of the changes above, newer agents no longer will be able to work with an older Zabbix server.

See also: Discovered entities

Configuring low-level discovery We will illustrate low-level discovery based on an example of file system discovery.

To configure the discovery, do the following:

- Go to: *Data collection* → *Templates* or *Hosts*
- Click on *Discovery* in the row of an appropriate template/host



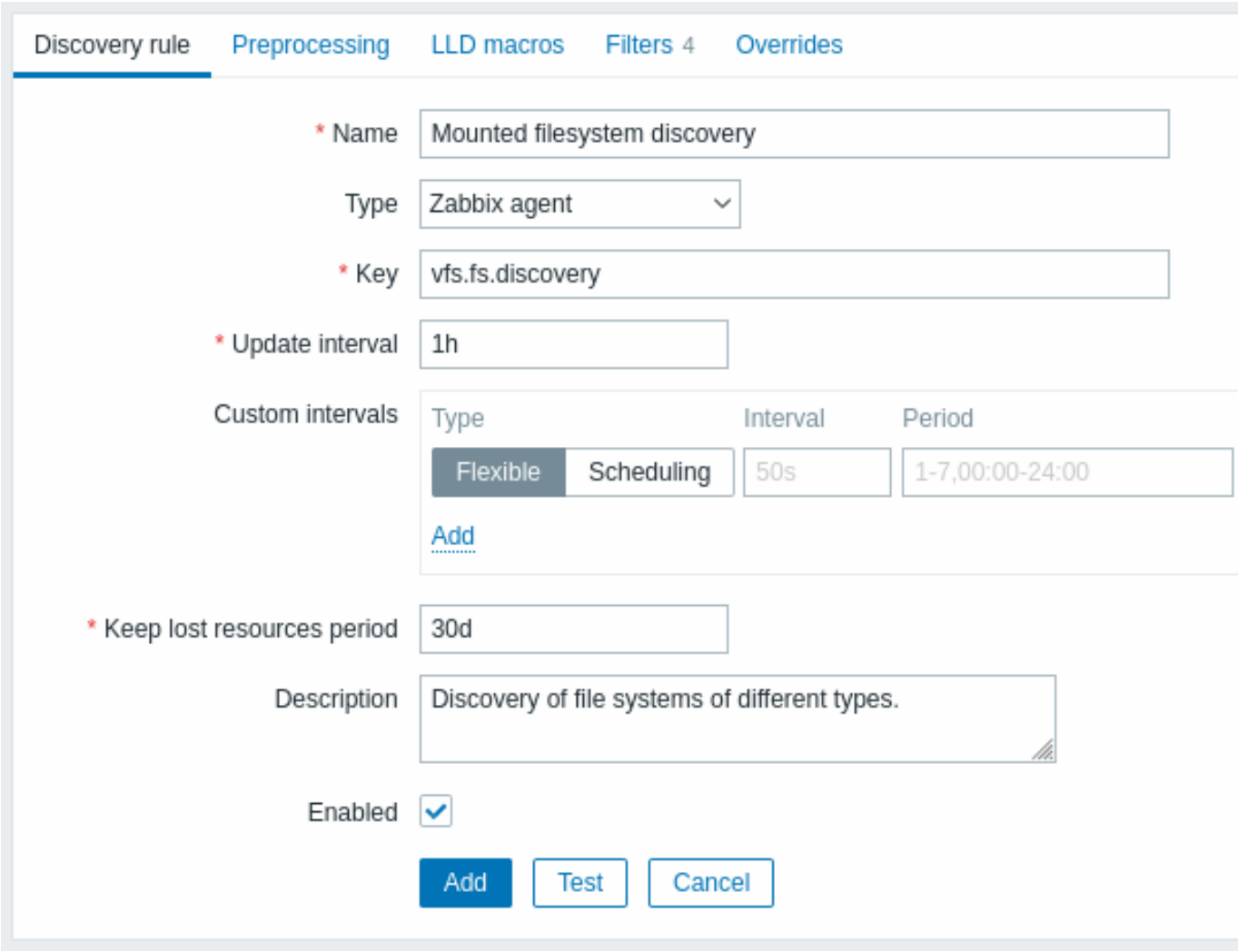
- Click on *Create discovery rule* in the upper right corner of the screen
- Fill in the discovery rule form with the required details

Discovery rule

The discovery rule form contains five tabs, representing, from left to right, the data flow during discovery:

- *Discovery rule* - specifies, most importantly, the built-in item or custom script to retrieve discovery data
- *Preprocessing* - applies some preprocessing to the discovered data
- *LLD macros* - allows to extract some macro values to use in discovered items, triggers, etc
- *Filters* - allows to filter the discovered values
- *Overrides* - allows to modify items, triggers, graphs or host prototypes when applying to specific discovered objects

The **Discovery rule** tab contains the item key to use for discovery (as well as some general discovery rule attributes):



All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Name of discovery rule.
Type	The type of check to perform discovery. In this example we are using a *Zabbix agent* item type. The discovery rule can also be a dependent item, depending on a regular item. It cannot depend on another discovery rule. For a dependent item, select the respective type (*Dependent item*) and specify the master item in the 'Master item' field. The master item must exist.
Key	Enter the discovery item key (up to 2048 characters). For example, you may use the built-in "vfs.fs.discovery" item key to return a JSON with the list of file systems present on the computer, their types and mount options. Note that another option for filesystem discovery is using discovery results by the "vfs.fs.get" agent key, supported since Zabbix 4.4.5 (see example).

Parameter	Description
Update interval	This field specifies how often Zabbix performs discovery. In the beginning, when you are just setting up file system discovery, you might wish to set it to a small interval, but once you know it works you can set it to 30 minutes or more, because file systems usually do not change very often. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. *Note*: The update interval can only be set to '0' if custom intervals exist with a non-zero value. If set to '0', and a custom interval (flexible or scheduled) exists with a non-zero value, the item will be polled during the custom interval duration. New discovery rules will be checked within 60 seconds of their creation, unless they have Scheduling or Flexible update interval and the *Update interval* is set to 0. *Note* that for an existing discovery rule the discovery can be performed immediately by pushing the *Execute now* button.
Custom intervals	You can create custom rules for checking the item: **Flexible** - create an exception to the *Update interval* (interval with different frequency) **Scheduling** - create a custom polling schedule. For detailed information see Custom intervals. Scheduling is supported since Zabbix 3.0.0.
Keep lost resources period	This field allows you to specify the duration for how long the discovered entity will be retained (won't be deleted) once its discovery status becomes "Not discovered anymore" (between 1 hour to 25 years; or "0"). Time suffixes are supported, e.g. 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. *Note:* If set to "0", entities will be deleted immediately. Using "0" is not recommended, since just wrongly editing the filter may end up in the entity being deleted with all the historical data.
Description	Enter a description.
Enabled	If checked, the rule will be processed.

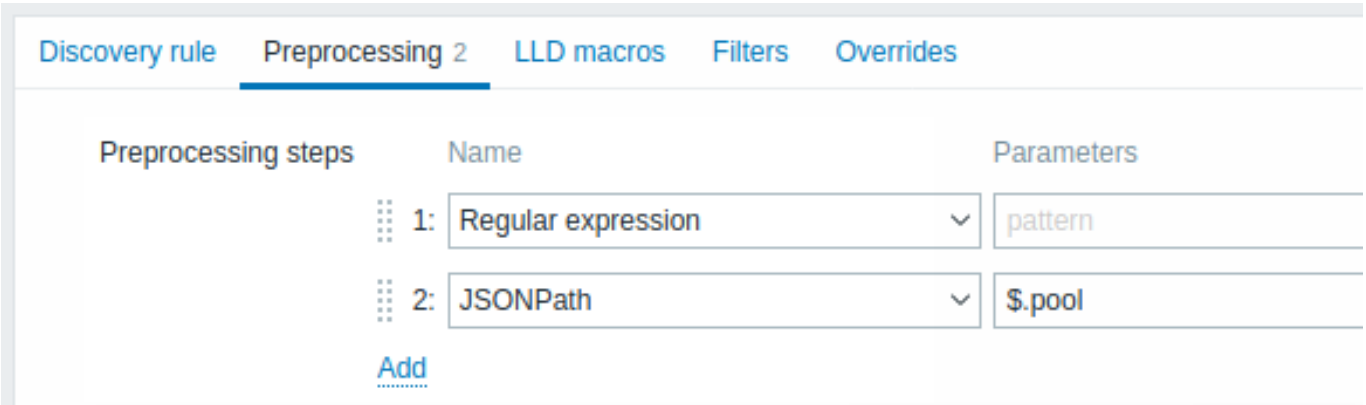
Note:
Discovery rule history is not preserved.

Preprocessing

The **Preprocessing** tab allows to define transformation rules to apply to the result of discovery. One or several transformations are possible in this step. Transformations are executed in the order in which they are defined. All preprocessing is done by Zabbix server.

See also:

- Preprocessing details
- Preprocessing testing



Type		
	Transformation	Description
Text		

Type	
Regular expression	Match the received value to the <pattern> regular expression and replace value with the extracted <output>. The regular expression supports extraction of maximum 10 captured groups with the \N sequence. Parameters: **pattern** - regular expression **output** - output formatting template. An \N (where N=1…9) escape sequence is replaced with the Nth matched group. A \0 escape sequence is replaced with the matched text. If you mark the *Custom on fail* checkbox, it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.
Replace	Find the search string and replace it with another (or nothing). All occurrences of the search string will be replaced. Parameters: **search string** - the string to find and replace, case-sensitive (required) **replacement** - the string to replace the search string with. The replacement string may also be empty effectively allowing to delete the search string when found. It is possible to use escape sequences to search for or replace line breaks, carriage return, tabs and spaces "\n \r \t \s"; backslash can be escaped as "\\" and escape sequences can be escaped as "\\n". Escaping of line breaks, carriage return, tabs is automatically done during low-level discovery. Supported since 5.0.0.
Structured data	
JSONPath	Extract value or fragment from JSON data using JSONPath functionality. If you mark the Custom on fail checkbox, it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.

XML XPath	Extract value or fragment from XML data using XPath functionality.
	For this option to work, Zabbix server must be compiled with libxml support.
	Examples:
	`number(/document/item/value)` will extract 10 from `<document><item><value>10</value></item></document>`
	`number(/document/item/@attribute)` will extract 10 from `<document><item attribute="10"></item></document>`
	`/document/item` will extract `<item><value>10</value></item>` from `<document><item><value>10</value></item></document>`
	Note that namespaces are not supported.
	Supported since 4.4.0.
	If you mark the *Custom on fail* checkbox, it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.

CSV to JSON	Convert CSV file data into JSON format. For more information, see: CSV to JSON preprocessing. Supported since 4.4.0.
XML to JSON	Convert data in XML format to JSON. For more information, see: Serialization rules. If you mark the *Custom on fail* checkbox, it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.
SNMP	
SNMP walk value	Extract value by the specified OID/MIB name and apply formatting options: **Unchanged** - return Hex-STRING as unescaped hex string (*note* that display hints are still applied); **UTF-8 from Hex-STRING** - convert Hex-STRING to UTF-8 string; **MAC from Hex-STRING** - convert Hex-STRING to MAC address string (which will have `' '` replaced by `':'`); **Integer from BITS** - convert the first 8 bytes of a bit string expressed as a sequence of hex characters (e.g. "1A 2B 3C 4D") into a 64-bit unsigned integer. In bit strings longer than 8 bytes, consequent bytes will be ignored. If you mark the Custom on fail checkbox, it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.

Type	
SNMP walk to JSON	Convert SNMP values to JSON. Specify a field name in the JSON and the corresponding SNMP OID path. Field values will be populated by values in the specified SNMP OID path. You may use this preprocessing step for SNMP OID discovery. Similar value formatting options as in the *SNMP walk value* step are available. If you mark the Custom on fail checkbox, it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.
Custom scripts	
JavaScript	Enter JavaScript code in the block that appears when clicking in the parameter field or on the pencil icon. Note that available JavaScript length depends on the database used. For more information, see: Javascript preprocessing
Validation	
Does not match regular expression	Specify a regular expression that a value must not match. E.g. `Error:(.*?)\.` If you mark the *Custom on fail* checkbox, it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.
Check for error in JSON	Check for an application-level error message located at JSONPath. Stop processing if succeeded and message is not empty; otherwise continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to user as is, without adding preprocessing step information. E.g. `$.errors`. If a JSON like `{"errors":"e1"}` is received, the next preprocessing step will not be executed. If you mark the *Custom on fail* checkbox, it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.
Check for error in XML	Check for an application-level error message located at xpath. Stop processing if succeeded and message is not empty; otherwise continue processing with the value that was before this preprocessing step. Note that these external service errors are reported to user as is, without adding preprocessing step information. No error will be reported in case of failing to parse invalid XML. Supported since 4.4.0. If you mark the *Custom on fail* checkbox, it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.
Throttling	
Discard unchanged with heartbeat	Discard a value if it has not changed within the defined time period (in seconds). Positive integer values are supported to specify the seconds (minimum - 1 second). Time suffixes can be used in this field (e.g. 30s, 1m, 2h, 1d). User macros and low-level discovery macros can be used in this field. Only one throttling option can be specified for a discovery item. E.g. `1m`. If identical text is passed into this rule twice within 60 seconds, it will be discarded. *Note*: Changing item prototypes does not reset throttling. Throttling is reset only when preprocessing steps are changed.
Prometheus	
Prometheus to JSON	Convert required Prometheus metrics to JSON. See Prometheus checks for more details.

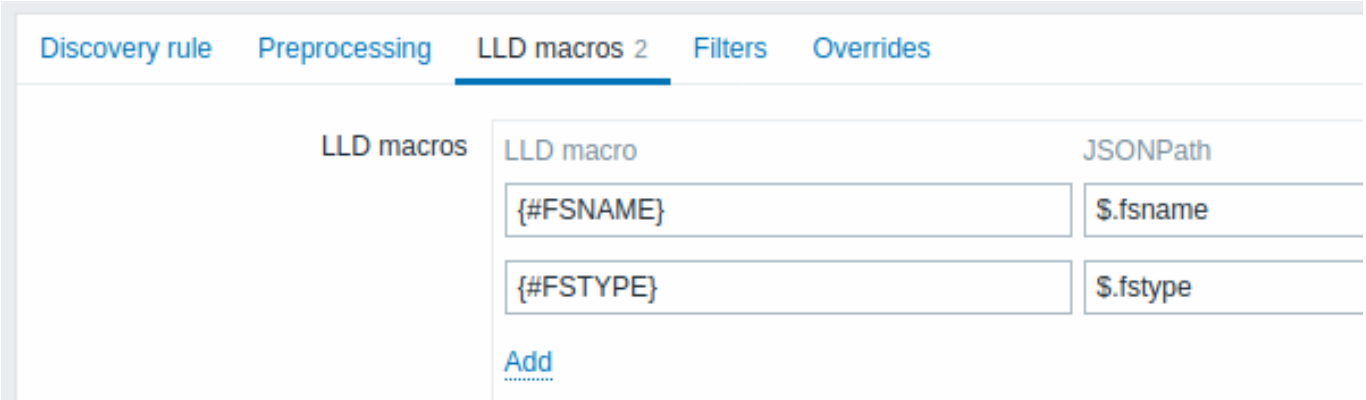
Note that if the discovery rule has been applied to the host via template then the content of this tab is read-only.

Custom macros

The **LLD macros** tab allows to specify custom low-level discovery macros.

Custom macros are useful in cases when the returned JSON does not have the required macros already defined. So, for example:

- The native `vfs.fs.discovery` key for filesystem discovery returns a JSON with some pre-defined LLD macros such as {#FSNAME}, {#FSTYPE}. These macros can be used in item, trigger prototypes (see subsequent sections of the page) directly; defining custom macros is not needed;
- The `vfs.fs.get` agent item also returns a JSON with filesystem data, but without any pre-defined LLD macros. In this case you may define the macros yourself, and map them to the values in the JSON using JSONPath:

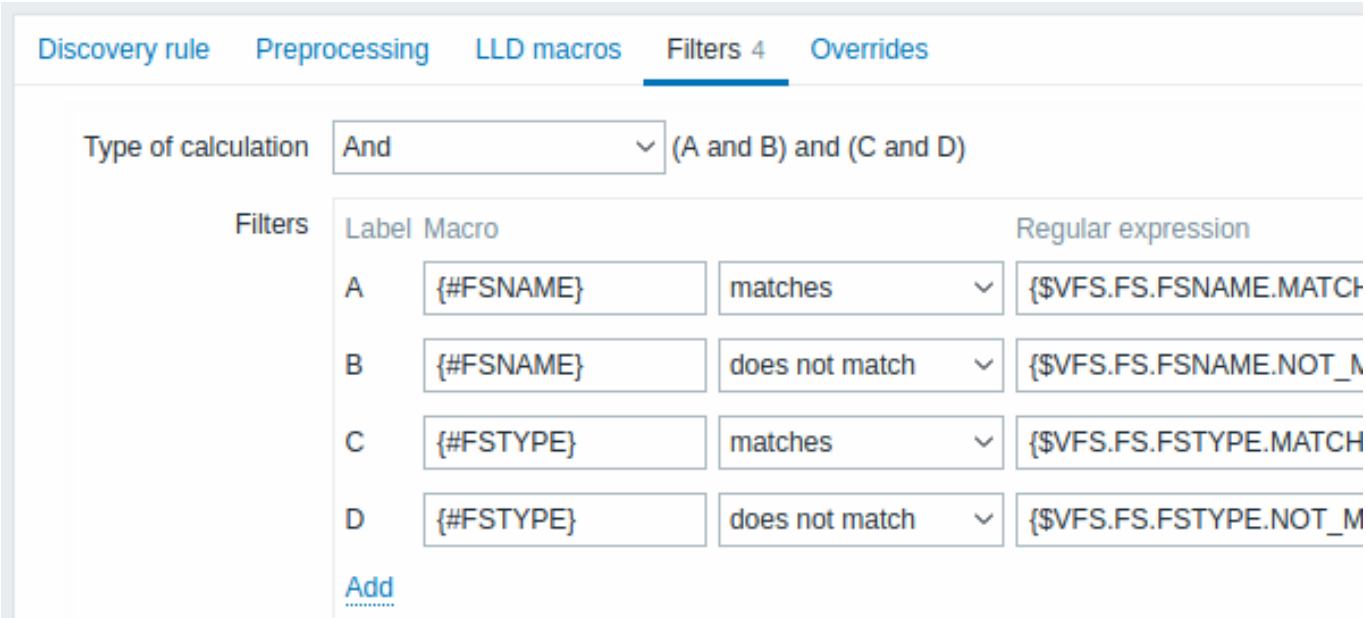


The extracted values can be used in discovered items, triggers, etc. Note that values will be extracted from the result of discovery and any preprocessing steps so far.

Parameter	Description
LLD macro *JSONPath*	Name of the low-level discovery macro, using the following syntax: `{#MACRO}`. Path that is used to extract LLD macro value from an LLD row, using JSONPath syntax. The values extracted from the returned JSON are used to replace the LLD macros in item, trigger, etc. prototype fields. JSONPath can be specified using the dot notation or the bracket notation. Bracket notation should be used in case of any special characters and Unicode, like `$['unicode + special chars #1']['unicode + special chars #2']`. For example, `$.foo` will extract "bar" and "baz" from this JSON: `[{"foo":"bar"}, {"foo":"baz"}]` Note that `$.foo` will extract "bar" and "baz" also from this JSON: `{"data":[{"foo":"bar"}, {"foo":"baz"}]}` because a single "data" object is processed automatically (for backwards compatibility with the low-level discovery implementation in Zabbix versions before 4.2).

Filter

A filter can be used to generate real items, triggers, and graphs only for entities that match the criteria. The **Filters** tab contains discovery rule filter definitions allowing to filter discovery values:



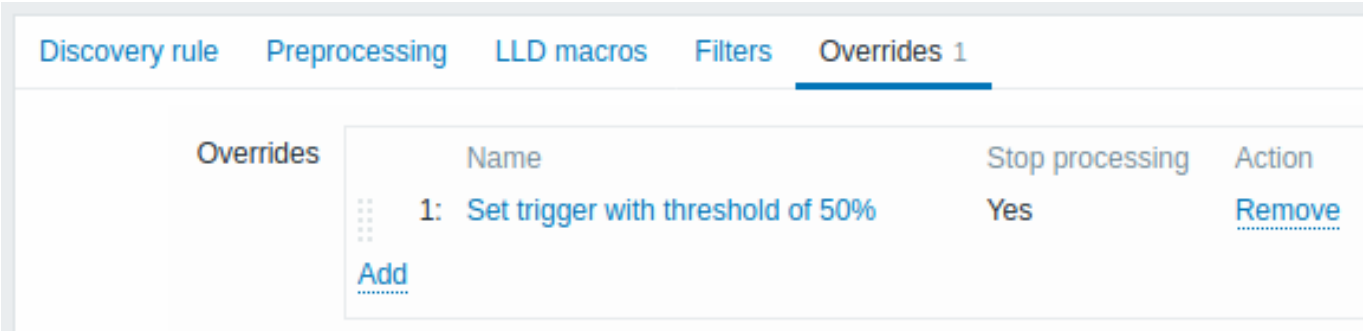
Parameter	Description
Type of calculation	The following options for calculating filters are available: **And** - all filters must be passed; **Or** - enough if one filter is passed; **And/Or** - uses *And* with different macro names and *Or* with the same macro name; **Custom expression** - offers the possibility to define a custom calculation of filters. The formula must include all filters in the list. Limited to 255 symbols.
Filters	The following filter condition operators are available: *matches*, *does not match*, *exists*, *does not exist*. *Matches* and *does not match* operators expect a Perl Compatible Regular Expression (PCRE). For instance, if you are only interested in C:, D:, and E: file systems, you could put {#FSNAME} into "Macro" and "^C|^D|^E" regular expression into "Regular expression" text fields. Filtering is also possible by file system types using {#FSTYPE} macro (e.g. "^ext|^reiserfs") and by drive types (supported only by Windows agent) using {#FSDRIVETYPE} macro (e.g., "fixed"). You can enter a regular expression or reference a global regular expression in "Regular expression" field. In order to test a regular expression you can use "grep -E", for example: `for f in ext2 nfs reiserfs smbfs; do echo $f | grep -E '^ext|^reiserfs' || echo "SKIP: $f"; done` *Exists* and *does not exist* operators allow to filter entities based on the presence or absence of the specified LLD macro in the response. Note that if a macro from the filter is missing in the response, the found entity will be ignored, unless a "does not exist" condition is specified for this macro. A warning will be displayed, if the absence of a macro affects the expression result. For example, if {#B} is missing in: {#A} matches 1 and {#B} matches 2 - will give a warning {#A} matches 1 or {#B} matches 2 - no warning

Warning:
A mistake or a typo in the regular expression used in the LLD rule (for example, an incorrect "File systems for discovery" regular expression) may cause deletion of thousands of configuration elements, historical values, and events for many hosts.

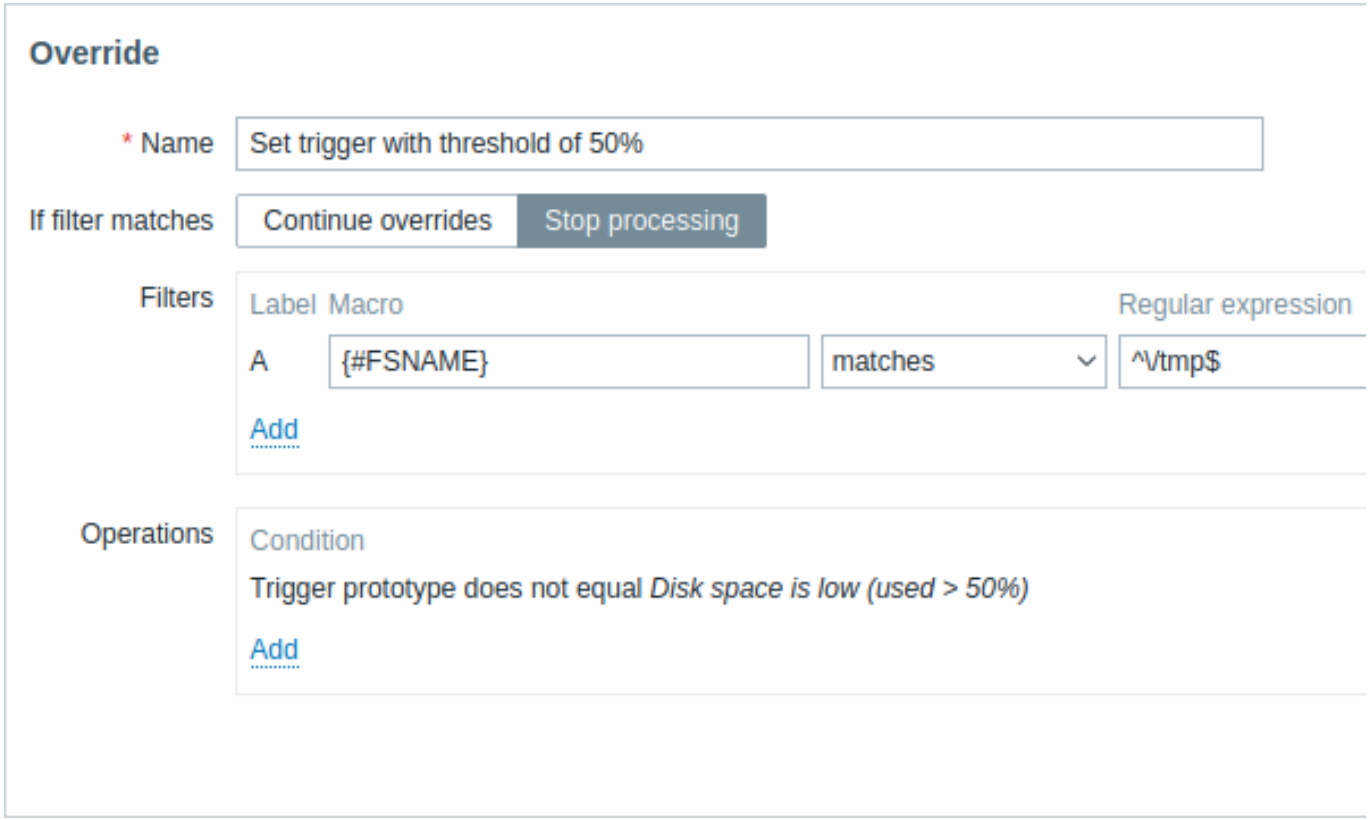
Attention:
Zabbix database in MySQL must be created as case-sensitive if file system names that differ only by case are to be discovered correctly.

Override

The **Override** tab allows setting rules to modify the list of item, trigger, graph and host prototypes or their attributes for discovered objects that meet given criteria.



Overrides (if any) are displayed in a reorderable drag-and-drop list and executed in the order in which they are defined. To configure details of a new override, click on Add in the *Overrides* block. To edit an existing override, click on the override name. A popup window will open allowing to edit the override rule details.

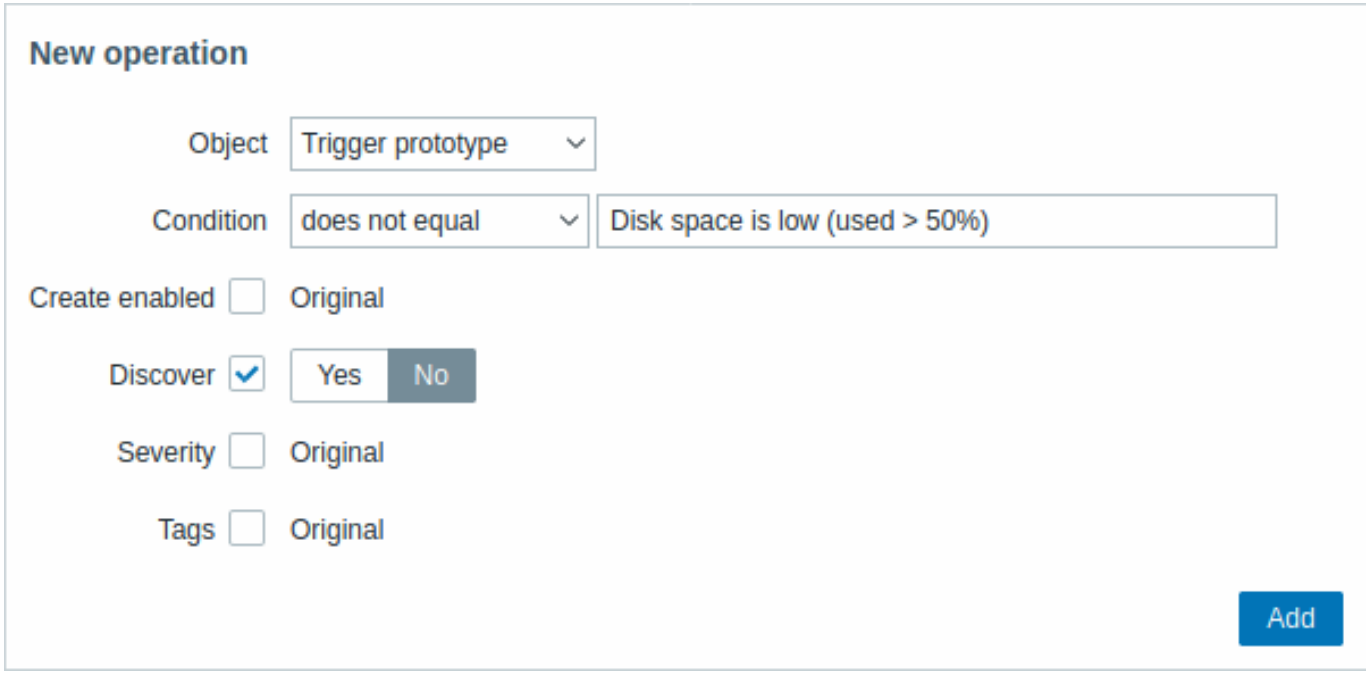


All mandatory parameters are marked with red asterisks.

Parameter	Description
Name	A unique (per LLD rule) override name.
If filter matches	Defines whether next overrides should be processed when filter conditions are met: **Continue overrides** - subsequent overrides will be processed. **Stop processing** - operations from preceding (if any) and this override will be executed, subsequent overrides will be ignored for matched LLD rows.
Filters	Determines to which discovered entities the override should be applied. Override filters are processed after discovery rule filters and have the same functionality.
Operations	Override operations are displayed with these details: **Condition** - an object type (item prototype/trigger prototype/graph prototype/host prototype) and a condition to be met (equals/does not equal/contains/does not contain/matches/does not match) **Action** - links for editing and removing an operation are displayed.

Configuring an operation

To configure details of a new operation, click on Add in the Operations block. To edit an existing operation, click on Edit next to the operation. A popup window where you can edit the operation details will open.



Parameter	Description
Object	Four types of objects are available: Item prototype Trigger prototype Graph prototype Host prototype
Condition	Allows filtering entities to which the operation should be applied.
Operator	Supported operators: **equals** - apply to this prototype **does not equal** - apply to all prototypes, except this **contains** - apply, if prototype name contains this string **does not contain** - apply, if prototype name does not contain this string **matches** - apply, if prototype name matches regular expression **does not match** - apply, if prototype name does not match regular expression
Pattern	A regular expression or a string to search for.
Object: *Item prototype*	
Create enabled	When the checkbox is marked, the buttons will appear, allowing to override original item prototype settings: *Yes* - the item will be added in an enabled state. *No* - the item will be added to a discovered entity but in a disabled state.
Discover	When the checkbox is marked, the buttons will appear, allowing to override original item prototype settings: *Yes* - the item will be added. *No* - the item will not be added.
Update interval	When the checkbox is marked, two options will appear, allowing to set different interval for the item: *Delay* - Item update interval. User macros and time suffixes (e.g. 30s, 1m, 2h, 1d) are supported. Should be set to 0 if *Custom interval* is used. *Custom interval* - click Add to specify flexible/scheduling intervals. For detailed information see Custom intervals.
History storage period	When the checkbox is marked, the buttons will appear, allowing to set different history storage period for the item: *Do not keep history* - if selected, the history will not be stored. *Storage period* - if selected, an input field for specifying storage period will appear to the right. User macros and LLD macros are supported.

Parameter		Description
	Trend storage period	When the checkbox is marked, the buttons will appear, allowing to set different trend storage period for the item: *Do not keep trends* - if selected, the trends will not be stored. *Storage period* - if selected, an input field for specifying storage period will appear to the right. User macros and LLD macros are supported.
	Tags	When the checkbox is marked, a new block will appear, allowing to specify tag-value pairs. These tags will be appended to the tags specified in the item prototype, even if the tag names match.
Object: *Trigger prototype*		
	Create enabled	When the checkbox is marked, the buttons will appear, allowing to override original trigger prototype settings: *Yes* - the trigger will be added in an enabled state. *No* - the trigger will be added to a discovered entity, but in a disabled state.
	Discover	When the checkbox is marked, the buttons will appear, allowing to override original trigger prototype settings: *Yes* - the trigger will be added. *No* - the trigger will not be added.
	Severity	When the checkbox is marked, trigger severity buttons will appear, allowing to modify trigger severity.
	Tags	When the checkbox is marked, a new block will appear, allowing to specify tag-value pairs. These tags will be appended to the tags specified in the trigger prototype, even if the tag names match.
Object: *Graph prototype*		
	Discover	When the checkbox is marked, the buttons will appear, allowing to override original graph prototype settings: *Yes* - the graph will be added. *No* - the graph will not be added.
Object: *Host prototype*		
	Create enabled	When the checkbox is marked, the buttons will appear, allowing to override original host prototype settings: *Yes* - the host will be created in an enabled state. *No* - the host will be created in a disabled state.
	Discover	When the checkbox is marked, the buttons will appear, allowing to override original host prototype settings: *Yes* - the host will be discovered. *No* - the host will not be discovered.
	Link templates	When the checkbox is marked, an input field for specifying templates will appear. Start typing the template name or click on *Select* next to the field and select templates from the list in a popup window. All templates linked to a host prototype will be replaced by templates from this override.
	Tags	When the checkbox is marked, a new block will appear, allowing to specify tag-value pairs. These tags will be appended to the tags specified in the host prototype, even if the tag names match.

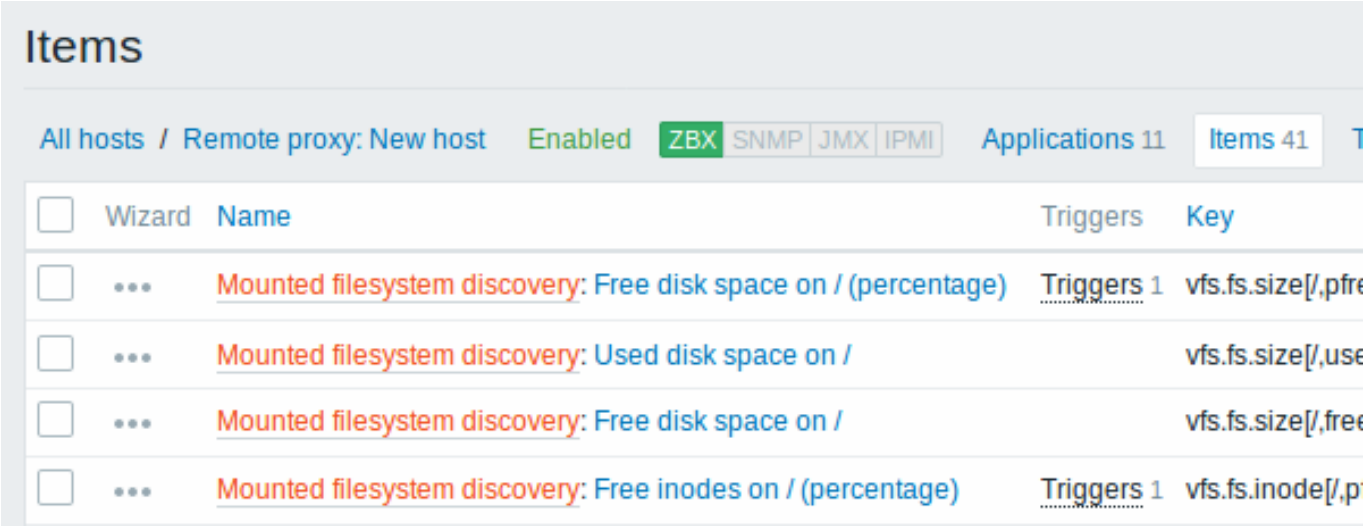
Parameter	Description
Host inventory	When the checkbox is marked, the buttons will appear, allowing to select different inventory mode for the host prototype: *Disabled* - do not populate host inventory *Manual* - provide details manually *Automated* - auto-fill host inventory data based on collected metrics.

Form buttons

Buttons at the bottom of the form allow to perform several operations.

Add	Add a discovery rule. This button is only available for new discovery rules.
Update	Update the properties of a discovery rule. This button is only available for existing discovery rules.
Clone	Create another discovery rule based on the properties of the current discovery rule.
Check now	Perform discovery based on the discovery rule immediately. The discovery rule must already exist. See more details. *Note* that when performing discovery immediately, configuration cache is not updated, thus the result will not reflect very recent changes to discovery rule configuration.
Delete	Delete the discovery rule.
Cancel	Cancel the editing of discovery rule properties.

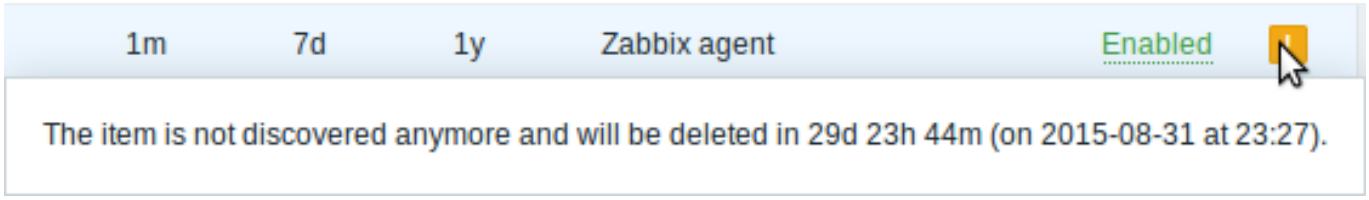
Discovered entities The screenshots below illustrate how discovered items, triggers, and graphs look like in the host's configuration. Discovered entities are prefixed with an orange link to a discovery rule they come from.



Note that discovered entities will not be created in case there are already existing entities with the same uniqueness criteria, for example, an item with the same key or graph with the same name. An error message is displayed in this case in the frontend that the low-level discovery rule could not create certain entities. The discovery rule itself, however, will not turn unsupported because some entity could not be created and had to be skipped. The discovery rule will go on creating/updating other entities.

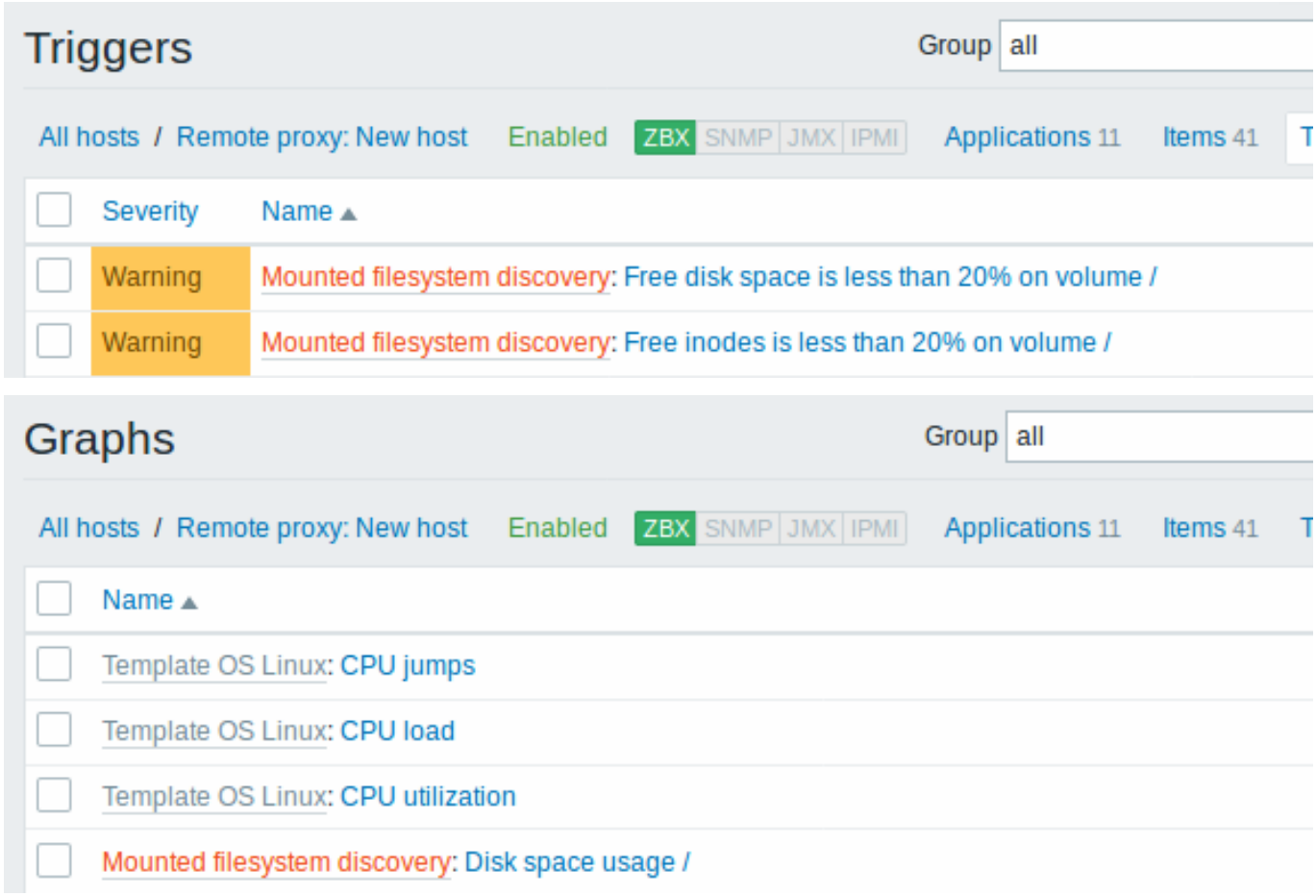
Items (similarly, triggers and graphs) created by a low-level discovery rule will be deleted automatically if a discovered entity (file system, interface, etc) stops being discovered (or does not pass the filter anymore). In this case the items, triggers and graphs will be deleted after the days defined in the *Keep lost resources period* field pass.

When discovered entities become 'Not discovered anymore', a lifetime indicator is displayed in the item list. Move your mouse pointer over it and a message will be displayed indicating how many days are left until the item is deleted.



If entities were marked for deletion, but were not deleted at the expected time (disabled discovery rule or item host), they will be deleted the next time the discovery rule is processed.

Entities containing other entities, which are marked for deletion, will not update if changed on the discovery rule level. For example, LLD-based triggers will not update if they contain items that are marked for deletion.



Other types of discovery More detail and how-tos on other types of out-of-the-box discovery is available in the following sections:

- discovery of network interfaces;
- discovery of CPUs and CPU cores;
- discovery of SNMP OIDs;
- discovery of JMX objects;
- discovery using ODBC SQL queries;
- discovery of Windows services;
- discovery of host interfaces in Zabbix.

For more detail on the JSON format for discovery items and an example of how to implement your own file system discoverer as a Perl script, see creating custom LLD rules.

Creating custom LLD rules It is also possible to create a completely custom LLD rule, discovering any type of entities - for example, databases on a database server.

To do so, a custom item should be created that returns JSON, specifying found objects and optionally - some properties of them. The amount of macros per entity is not limited - while the built-in discovery rules return either one or two macros (for example, two for filesystem discovery), it is possible to return more.

The required JSON format is best illustrated with an example. Suppose we are running an old Zabbix 1.8 agent (one that does not support "vfs.fs.discovery"), but we still need to discover file systems. Here is a simple Perl script for Linux that discovers mounted

file systems and outputs JSON, which includes both file system name and type. One way to use it would be as a UserParameter with key "vfs.fs.discovery_perl":

```
###!/usr/bin/perl

$first = 1;

print "[\n";

for (`cat /proc/mounts`)
{
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t{\n";
    print "\t\t\"{#FSNAME}\":\"$fsname\",\n";
    print "\t\t\"{#FSTYPE}\":\"$fstype\"\n";
    print "\t}\n";
}

print "]\n";
```

Attention:
Allowed symbols for LLD macro names are **0-9** , **A-Z** , _ , **.**

Lowercase letters are not supported in the names.

An example of its output (reformatted for clarity) is shown below. JSON for custom discovery checks has to follow the same format.

```
[
    { "{#FSNAME}":"/",                          "{#FSTYPE}":"rootfs"   },
    { "{#FSNAME}":"/sys",                       "{#FSTYPE}":"sysfs"    },
    { "{#FSNAME}":"/proc",                      "{#FSTYPE}":"proc"     },
    { "{#FSNAME}":"/dev",                       "{#FSTYPE}":"devtmpfs" },
    { "{#FSNAME}":"/dev/pts",                   "{#FSTYPE}":"devpts"   },
    { "{#FSNAME}":"/lib/init/rw",               "{#FSTYPE}":"tmpfs"    },
    { "{#FSNAME}":"/dev/shm",                   "{#FSTYPE}":"tmpfs"    },
    { "{#FSNAME}":"/home",                      "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/tmp",                       "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/usr",                       "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/var",                       "{#FSTYPE}":"ext3"     },
    { "{#FSNAME}":"/sys/fs/fuse/connections",   "{#FSTYPE}":"fusectl"  }
]
```

In previous example it is required that the keys match the LLD macro names used in prototypes, the alternative is to extract LLD macro values using JSONPath {#FSNAME} → $.fsname and {#FSTYPE} → $.fstype, thus making such script possible:

```
###!/usr/bin/perl

$first = 1;

print "[\n";

for (`cat /proc/mounts`)
{
    ($fsname, $fstype) = m/\S+ (\S+) (\S+)/;

    print "\t,\n" if not $first;
    $first = 0;

    print "\t{\n";
    print "\t\t\"fsname\":\"$fsname\",\n";
```

```
    print "\t\t\"fstype\":\"$fstype\"\n";
    print "\t}\n";
}

print "]\n";
```

An example of its output (reformatted for clarity) is shown below. JSON for custom discovery checks has to follow the same format.

```
[
    { "fsname":"/",                          "fstype":"rootfs"   },
    { "fsname":"/sys",                       "fstype":"sysfs"    },
    { "fsname":"/proc",                      "fstype":"proc"     },
    { "fsname":"/dev",                       "fstype":"devtmpfs" },
    { "fsname":"/dev/pts",                   "fstype":"devpts"   },
    { "fsname":"/lib/init/rw",               "fstype":"tmpfs"    },
    { "fsname":"/dev/shm",                   "fstype":"tmpfs"    },
    { "fsname":"/home",                      "fstype":"ext3"     },
    { "fsname":"/tmp",                       "fstype":"ext3"     },
    { "fsname":"/usr",                       "fstype":"ext3"     },
    { "fsname":"/var",                       "fstype":"ext3"     },
    { "fsname":"/sys/fs/fuse/connections",   "fstype":"fusectl"  }
]
```

Then, in the discovery rule's "Filter" field, we could specify "{#FSTYPE}" as a macro and "rootfs|ext3" as a regular expression.

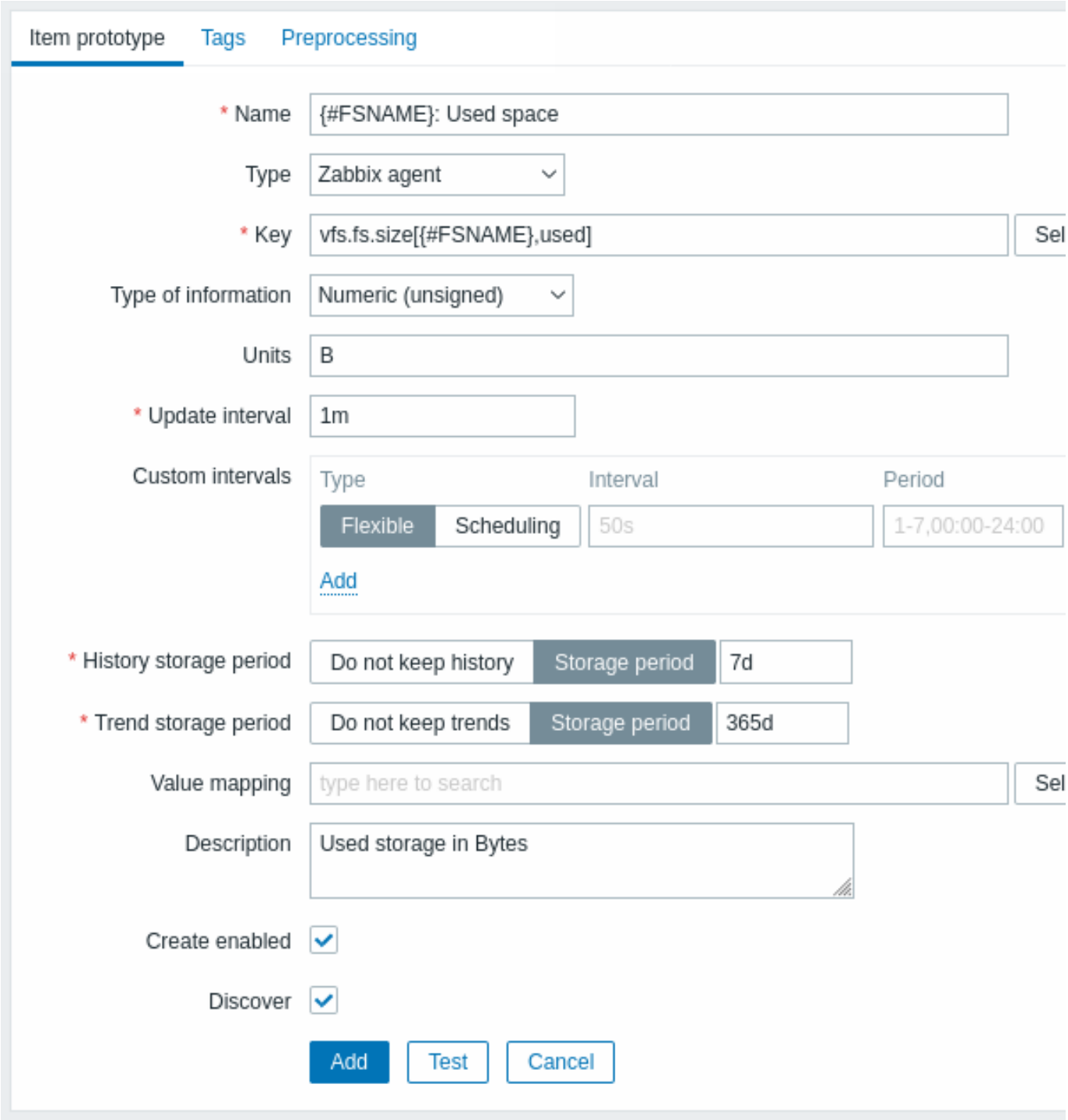
Note:
You don't have to use macro names FSNAME/FSTYPE with custom LLD rules, you are free to use whatever names you like. In case JSONPath is used then LLD row will be an array element that can be an object, but it can be also another array or a value.

Note that, if using a user parameter, the return value is limited to 16MB. For more details, see data limits for LLD return values.

1 Item prototypes

Once a rule is created, go to the items for that rule and press "Create item prototype" to create an item prototype.

Note how the {#FSNAME} macro is used where a file system name is required. The use of a low-level discovery macro is mandatory in the item key to make sure that the discovery is processed correctly. When the discovery rule is processed, this macro will be substituted with the discovered file system.



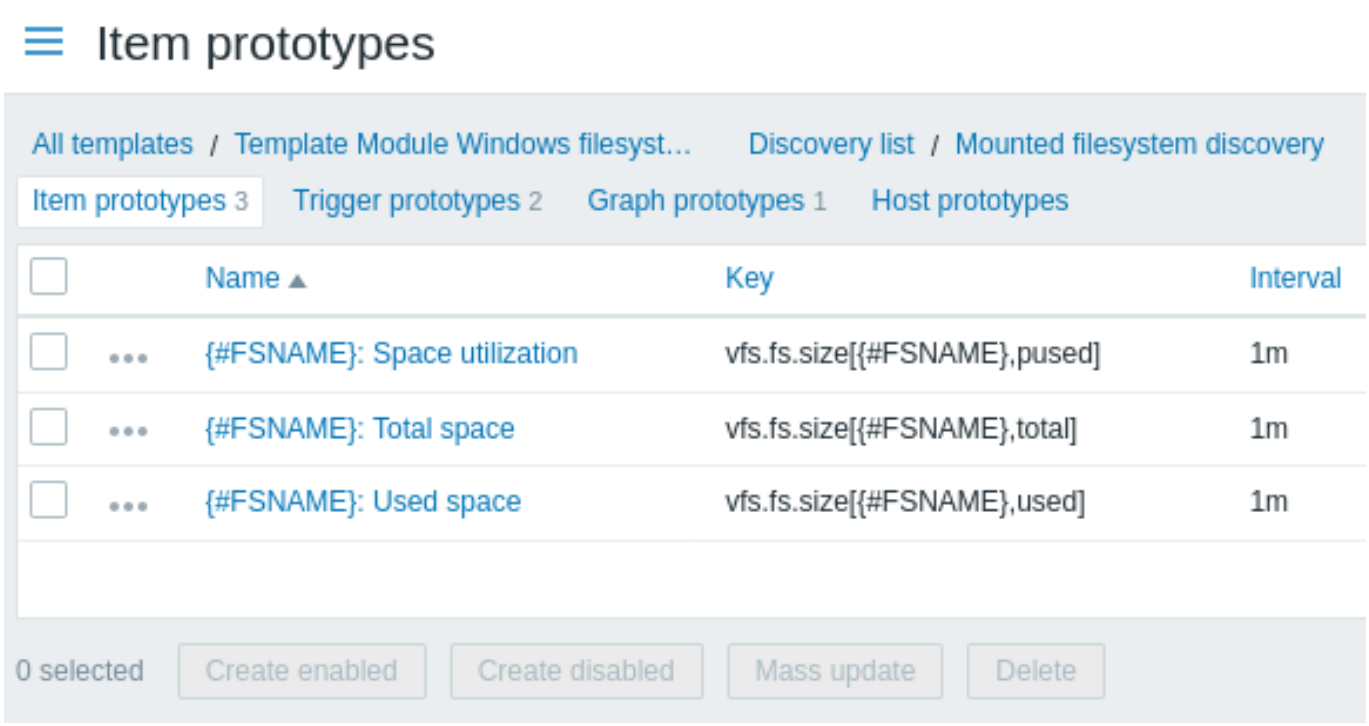
Low-level discovery macros and user macros are supported in item prototype configuration and item value preprocessing parameters. Note that when used in update intervals, a single macro has to fill the whole field. Multiple macros in one field or macros mixed with text are not supported.

Note:
Context-specific escaping of low-level discovery macros is performed for safe use in regular expression and XPath preprocessing parameters.

Attributes that are specific for item prototypes:

Parameter	Description
Create enabled	If checked the item will be added in an enabled state. If unchecked, the item will be added to a discovered entity, but in a disabled state.
Discover	If checked (default) the item will be added to a discovered entity. If unchecked, the item will not be added to a discovered entity, unless this setting is overridden in the discovery rule.

We can create several item prototypes for each file system metric we are interested in:



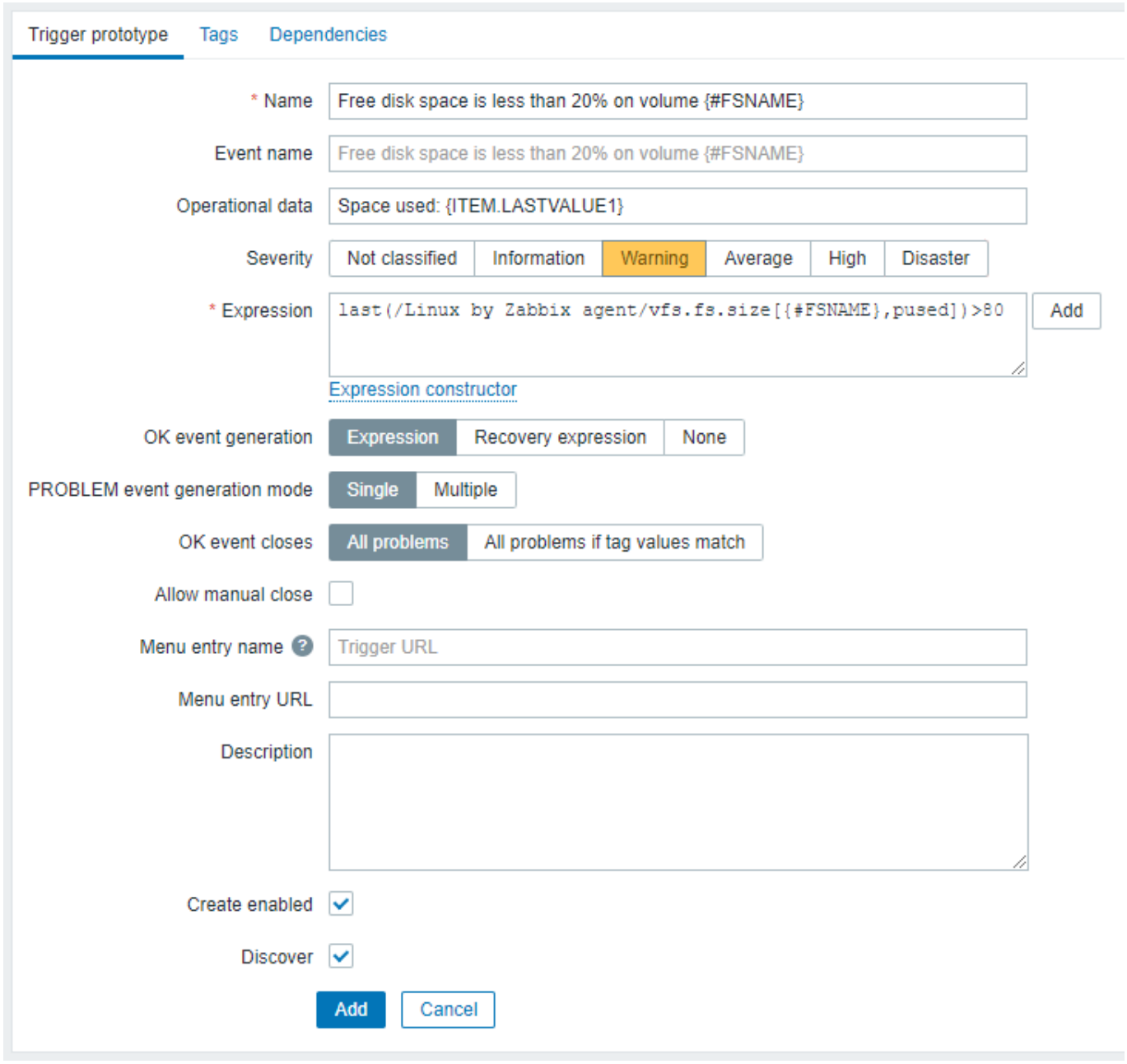
Click on the three-dot icon to open the menu for the specific item prototype with these options:

- *Create trigger prototype* - create a trigger prototype based on this item prototype
- *Trigger prototypes* - click to see a list with links to already-configured trigger prototypes of this item prototype
- *Create dependent item* - create a dependent item for this item prototype

Mass update option is available if you want to update properties of several item prototypes at once.

2 Trigger prototypes

We create trigger prototypes in a similar way as item prototypes:

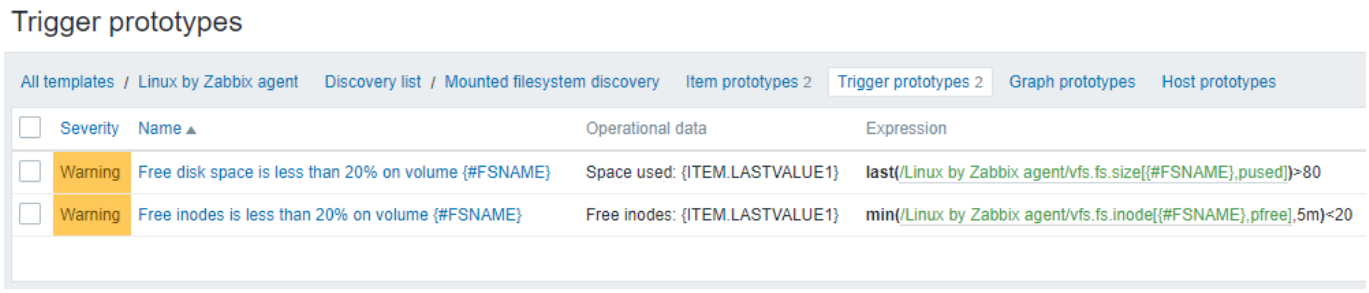


Attributes that are specific for trigger prototypes:

Parameter	Description
Create enabled	If checked the trigger will be added in an enabled state. If unchecked, the trigger will be added to a discovered entity, but in a disabled state.
Discover	If checked (default) the trigger will be added to a discovered entity. If unchecked, the trigger will not be added to a discovered entity, unless this setting is overridden in the discovery rule.

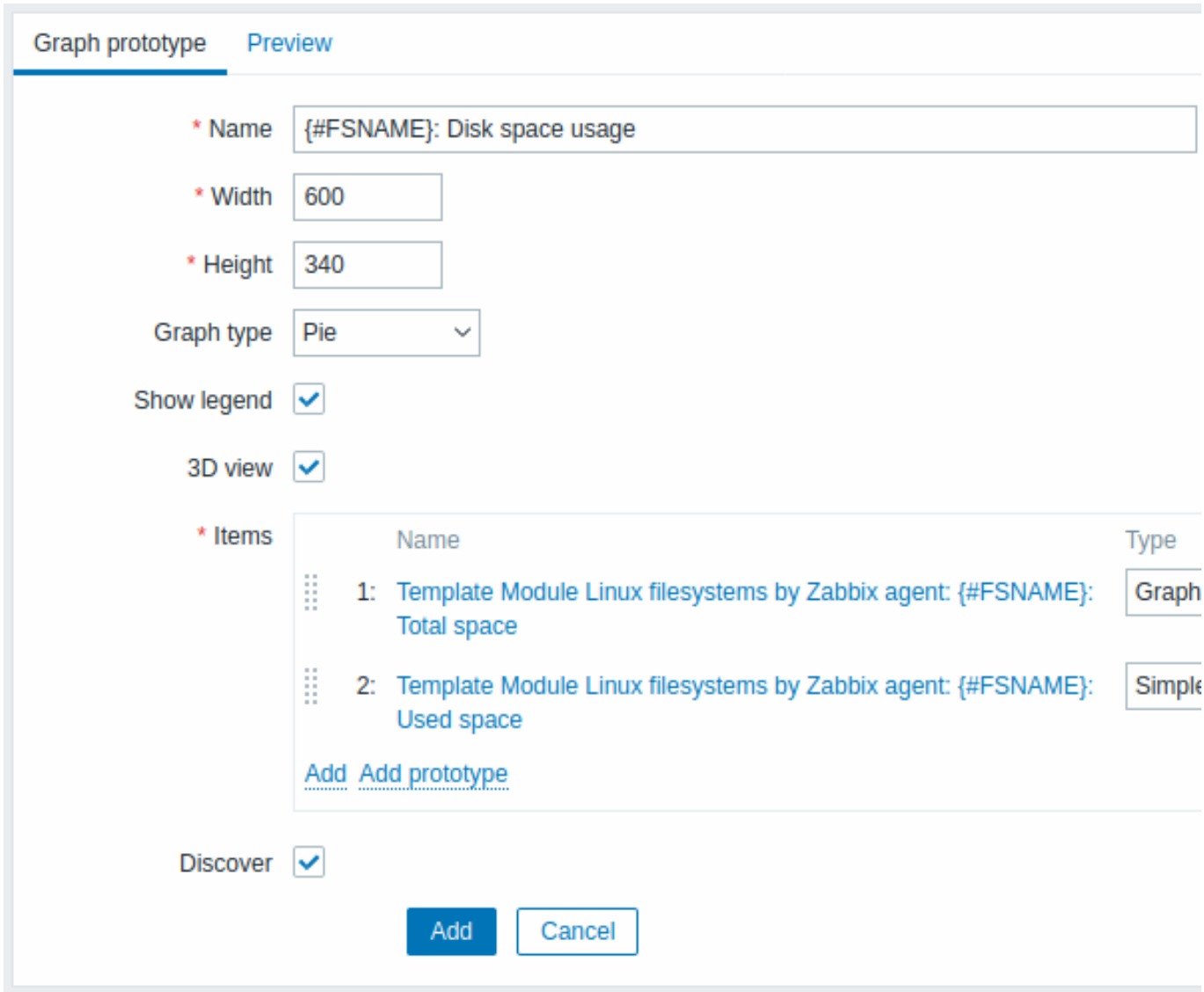
When real triggers are created from the prototypes, there may be a need to be flexible as to what constant ('20' in our example) is used for comparison in the expression. See how user macros with context can be useful to accomplish such flexibility.

You can define dependencies between trigger prototypes as well (supported since Zabbix 3.0). To do that, go to the *Dependencies* tab. A trigger prototype may depend on another trigger prototype from the same low-level discovery (LLD) rule or on a regular trigger. A trigger prototype may not depend on a trigger prototype from a different LLD rule or on a trigger created from trigger prototype. Host trigger prototype cannot depend on a trigger from a template.



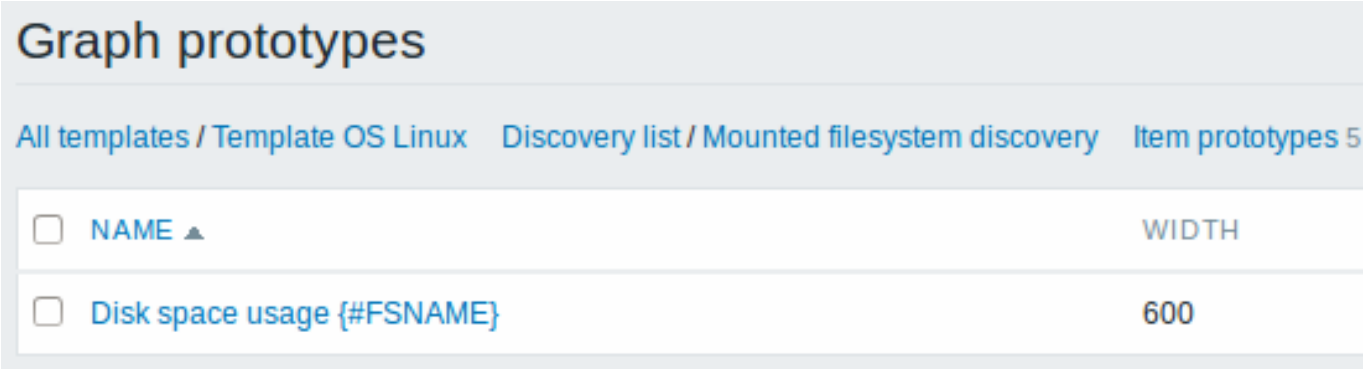
3 Graph prototypes

We can create graph prototypes, too:

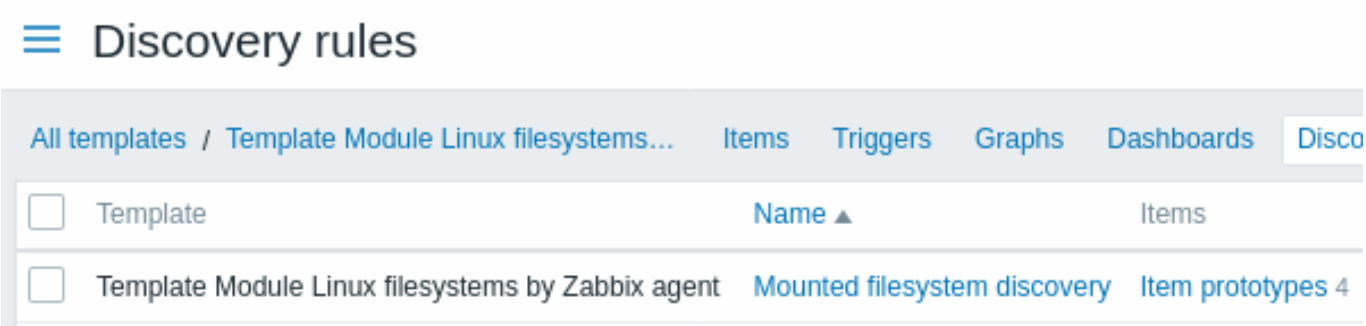


Attributes that are specific for graph prototypes:

Parameter	Description
Discover	If checked (default) the graph will be added to a discovered entity. If unchecked, the graph will not be added to a discovered entity, unless this setting is overridden in the discovery rule.



Finally, we have created a discovery rule that looks as shown below. It has five item prototypes, two trigger prototypes, and one graph prototype.



4 Host prototypes

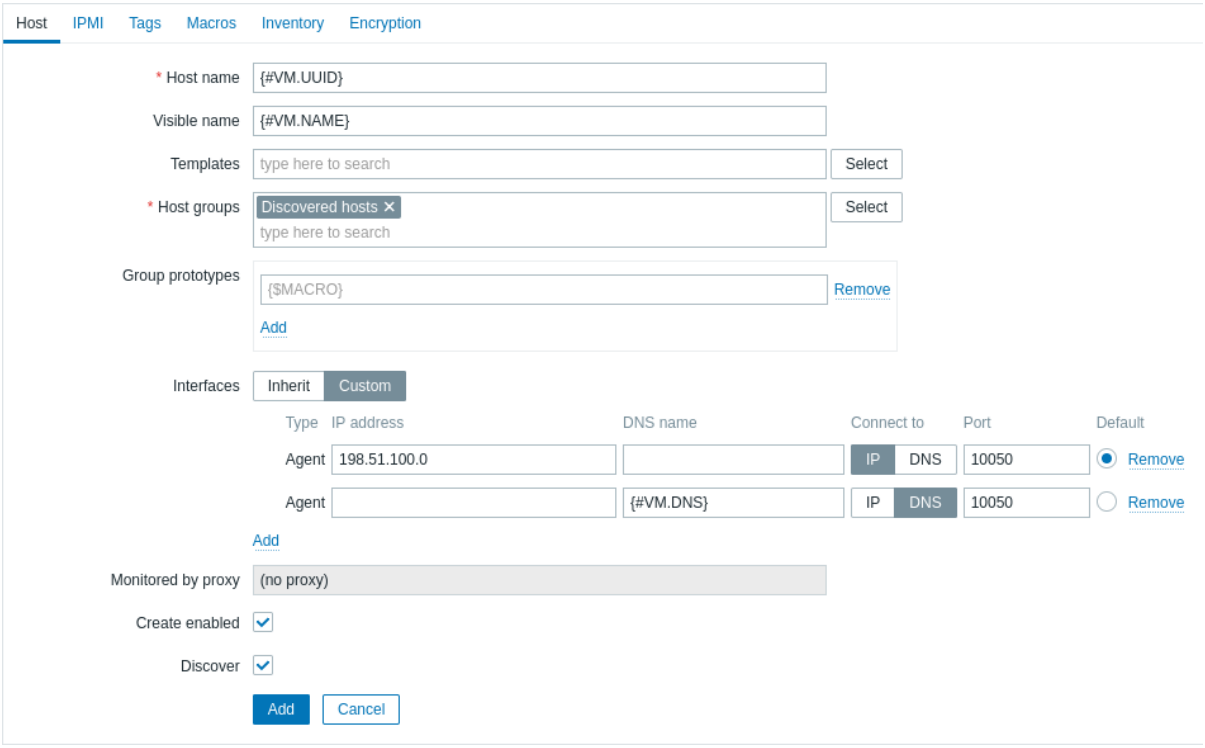
Host prototypes are blueprints for creating hosts through low-level discovery rules. Before being discovered as hosts, these prototypes cannot have items and triggers, except those linked from templates.

Configuration

Host prototypes are configured under low-level discovery rules.

To create a host prototype:

1. Go to *Data collection → Hosts*.
2. Click *Discovery* for the required host to navigate to the list of low-level discovery rules configured for that host.
3. Click *Host prototypes* for the required discovery rule.
4. Click the *Create host prototype* button in the upper right corner.



Host prototypes have the same parameters as regular hosts; however, the following parameters support different or additional configuration:

Parameter	Description
Host name	This parameter must contain at least one low-level discovery macro to ensure unique host names for created hosts.
Visible name	Low-level discovery macros are supported.
Group prototypes	Allows specifying host group prototypes by using low-level discovery macros. The specified group prototypes are created as host groups and assigned to the created hosts.
Interfaces	Set whether discovered hosts inherit the IP from the host that the discovery rule belongs to (default), or get custom interfaces. Low-level discovery macros and user macros are supported.
Create enabled	Set the status of discovered hosts; if unchecked, hosts will be created as disabled.
Discover	Set whether hosts will be created from the host prototype; if unchecked, hosts will not be created from the host prototype (unless this setting is overridden in the low-level discovery rule).

Note:
Low-level discovery macros are also supported for tag values and host prototype user macro values.
 Value maps are not supported for host prototypes.

For an example on how to configure a host prototype, see *Virtual machine monitoring*.

Host interfaces

To add custom interfaces, switch the *Interface* selector from ”Inherit” to ”Custom”. Click Add and select the interface type - Zabbix agent, SNMP, JMX, IPMI.

Note:
If *Custom* is selected, but no interfaces have been set, the hosts will be created without interfaces.
 If *Inherit* is selected and the host prototype belongs to a template, all discovered hosts will inherit the host interface from the host to which the template is linked.

If multiple custom interfaces are specified, the primary interface can be set in the *Default* column.

Warning:
A host will only be created if a host interface contains correct data.

Discovered hosts

In the host list, discovered hosts are prefixed with the name of the discovery rule that created them.

Discovered hosts inherit most parameters from host prototypes as *read-only*. Only the following discovered host parameters can be configured:

- *Templates* - link additional templates or unlink manually added ones. Templates inherited from a host prototype cannot be unlinked.
- *Status* - manually enable/disable a host.
- *Tags* - manually add tags alongside tags inherited from the host prototype. Manual or inherited tags cannot have duplicates (tags with the same name and value). If an inherited tag has the same name and value as a manual tag, it will replace the manual tag during discovery.
- *Macros* - manually add host macros alongside macros inherited from the host prototype; change macro values and types on the host level.
- *Description*.

Discovered hosts can be deleted manually. Note, however, that they will be discovered again if discovery is enabled for them.

Hosts that are no longer discovered will be automatically deleted based on the *Keep lost resources period (in days)* value of the discovery rule.

Note:
Zabbix does not support nested host prototypes, that is, host prototypes on hosts discovered by low-level discovery rules.

5 Notes on low-level discovery

Using LLD macros in user macro contexts

LLD macros may be used inside user macro context, for example, in trigger prototypes.

Multiple LLD rules for the same item

Since Zabbix agent version 3.2 it is possible to define several low-level discovery rules with the same discovery item.

To do that you need to define the Alias agent parameter, allowing to use altered discovery item keys in different discovery rules, for example `vfs.fs.discovery[foo]`, `vfs.fs.discovery[bar]`, etc.

Data limits for return values

There is no limit for low-level discovery rule JSON data if it is received directly by Zabbix server. This is because the return values are processed without being stored in a database.

There is also no limit for custom low-level discovery rules. However, if custom low-level discovery rule data is retrieved using a user parameter, the user parameter return value limit applies.

If data has to go through Zabbix proxy, it has to store this data in the database. In such a case, database limits apply.

6 Discovery rules

Please use the sidebar to see discovery rule configuration examples for various cases.

1 Discovery of mounted filesystems

Overview

It is possible to discover mounted filesystems and their properties:

- mountpoint name
- filesystem type
- filesystem size
- inode statistics
- mount options

To do that, you may use a combination of:

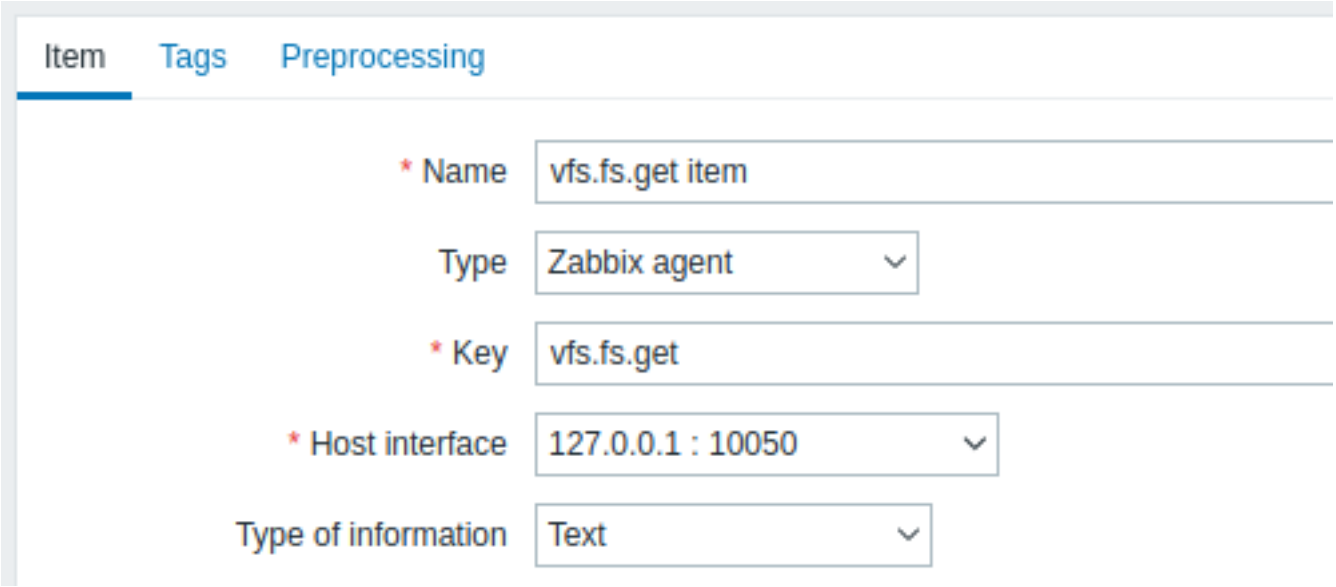
- the `vfs.fs.get` agent item as the master item
- dependent low-level discovery rule and item prototypes

Configuration

Master item

Create a Zabbix agent item using the following key:

`vfs.fs.get`



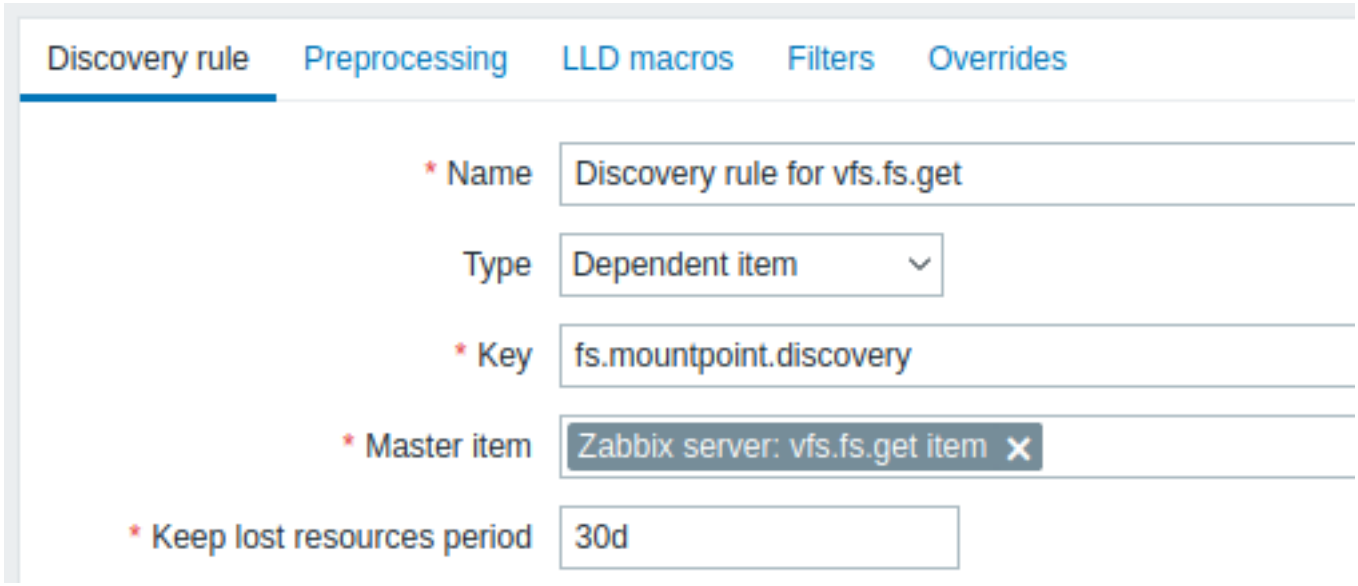
Set the type of information to ”Text” for possibly big JSON data.

The data returned by this item will contain something like the following for a mounted filesystem:

```
[
  {
    "fsname": "/",
    "fstype": "ext4",
    "bytes": {
      "total": 249405239296,
      "free": 24069537792,
      "used": 212595294208,
      "pfree": 10.170306,
      "pused": 89.829694
    },
    "inodes": {
      "total": 15532032,
      "free": 12656665,
      "used": 2875367,
      "pfree": 81.487503,
      "pused": 18.512497
    },
    "options": "rw,noatime,errors=remount-ro"
  }
]
```

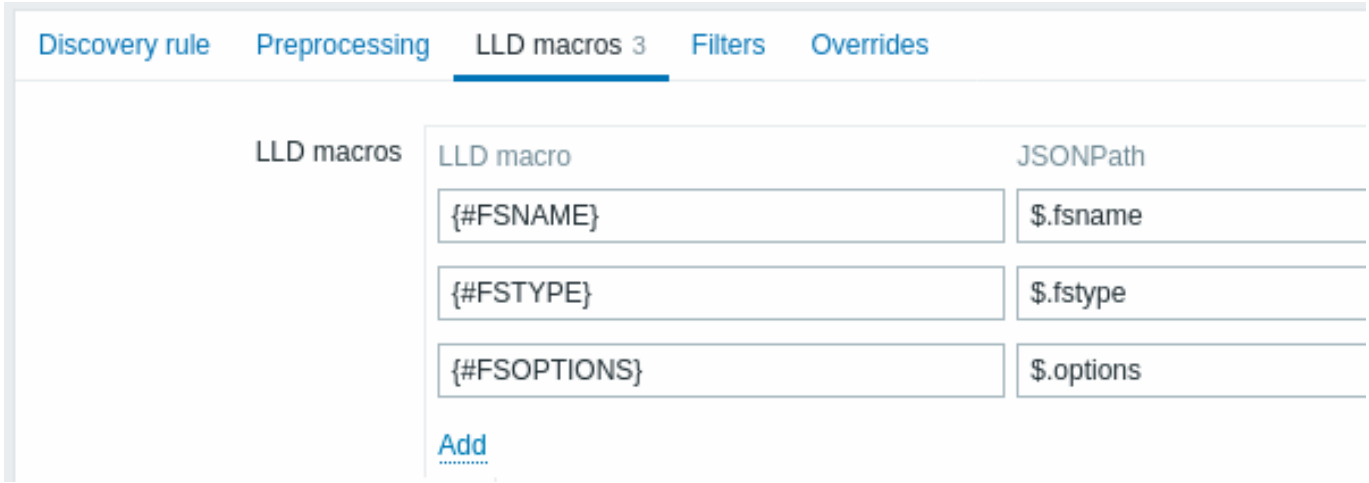
Dependent LLD rule

Create a low-level discovery rule as ”Dependent item” type:

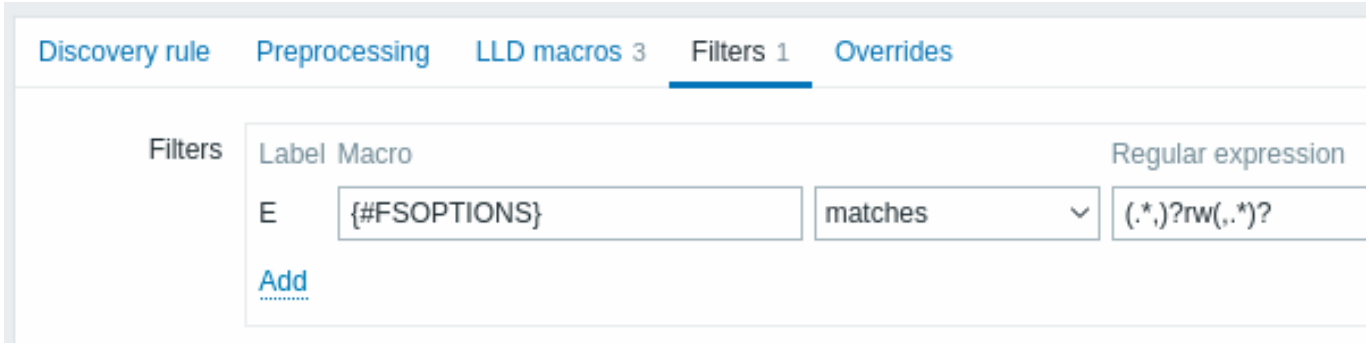


As master item select the `vfs.fs.get` item we created.

In the ”LLD macros” tab define custom macros with the corresponding JSONPath:

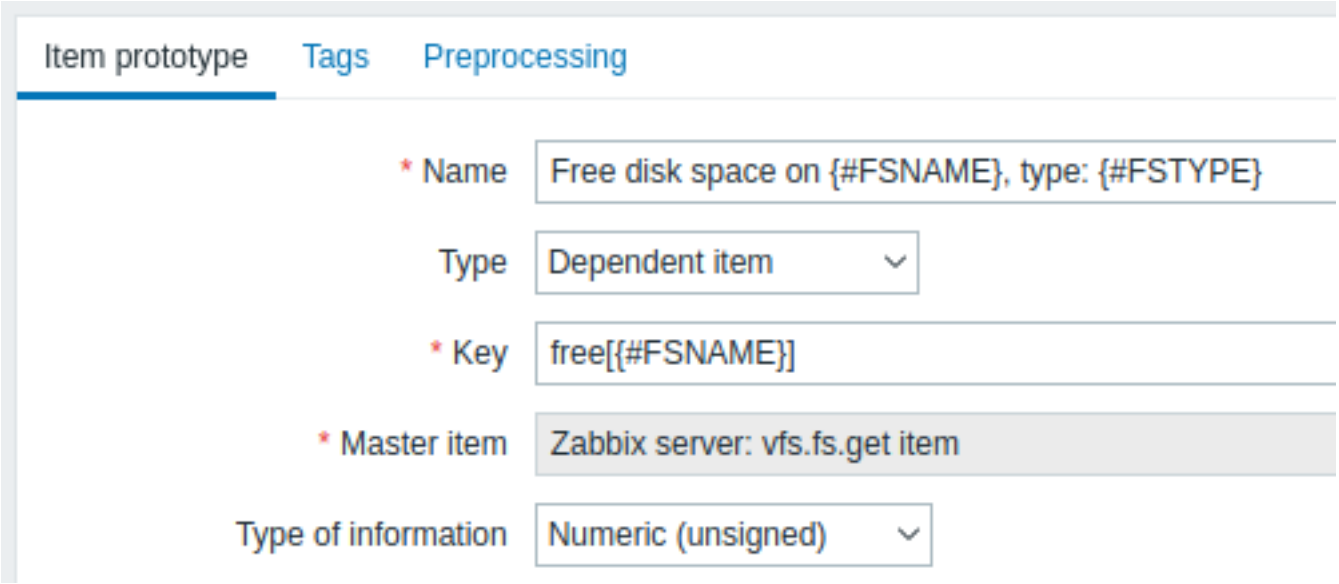


In the ”Filters” tab you may add a regular expression that filters only **read-write** filesystems:



Dependent item prototype

Create an item prototype with ”Dependent item” type in this LLD rule. As master item for this prototype select the `vfs.fs.get` item we created.



Note the use of custom macros in the item prototype name and key:

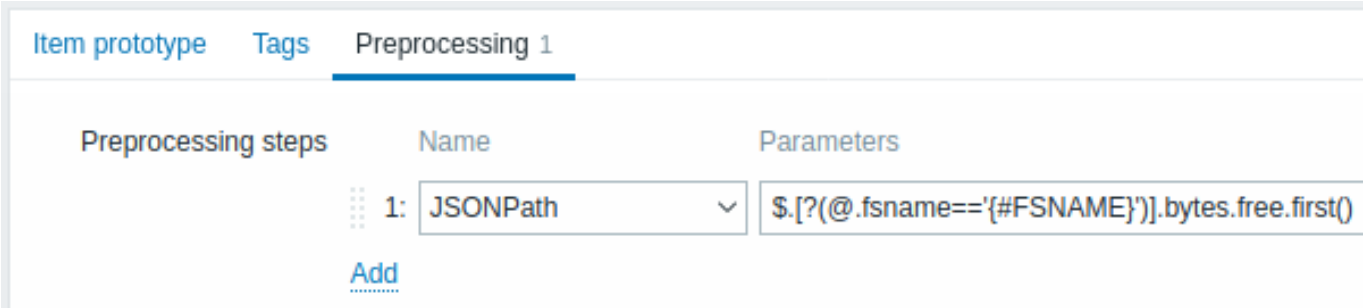
- *Name*: Free disk space on {#FSNAME}, type: {#FSTYPE}
- *Key*: Free[{#FSNAME}]

As type of information, use:

- *Numeric (unsigned)* for metrics like 'free', 'total', 'used'
- *Numeric (float)* for metrics like 'pfree', 'pused' (percentage)

In the item prototype "Preprocessing" tab select JSONPath and use the following JSONPath expression as parameter:

```
$.[?(@.fsname=='{#FSNAME}')].bytes.free.first()
```



When discovery starts, one item per each mountpoint will be created. This item will return the number of free bytes for the given mountpoint.

2 Discovery of network interfaces

In a similar way as file systems are discovered, it is possible to also discover network interfaces.

Item key

The item key to use in the discovery rule is

```
net.if.discovery
```

This item is supported since Zabbix agent 2.0.

Supported macros

You may use the {#IFNAME} macro in the discovery rule filter and prototypes of items, triggers and graphs.

Examples of item prototypes that you might wish to create based on "net.if.discovery":

- "net.if.in[{#IFNAME},bytes]",
- "net.if.out[{#IFNAME},bytes]".

Note that on Windows {#IFGUID} is also returned.

3 Discovery of CPUs and CPU cores

In a similar way as file systems are discovered, it is possible to also discover CPUs and CPU cores.

Item key

The item key to use in the discovery rule is

`system.cpu.discovery`

This item is supported since Zabbix agent 2.4.

Supported macros

This discovery key returns two macros - {#CPU.NUMBER} and {#CPU.STATUS} identifying the CPU order number and status respectively. Note that a clear distinction cannot be made between actual, physical processors, cores and hyperthreads. {#CPU.STATUS} on Linux, UNIX and BSD systems returns the status of the processor, which can be either "online" or "offline". On Windows systems, this same macro may represent a third value - "unknown" - which indicates that a processor has been detected, but no information has been collected for it yet.

CPU discovery relies on the agent's collector process to remain consistent with the data provided by the collector and save resources on obtaining the data. This has the effect of this item key not working with the test (-t) command line flag of the agent binary, which will return a NOT_SUPPORTED status and an accompanying message indicating that the collector process has not been started.

Item prototypes that can be created based on CPU discovery include, for example:

- `system.cpu.util[{#CPU.NUMBER},<type>,<mode>]`
- `system.hw.cpu[{#CPU.NUMBER},<info>]`

For detailed item key description, see Zabbix agent item keys.

4 Discovery of SNMP OIDs

Overview

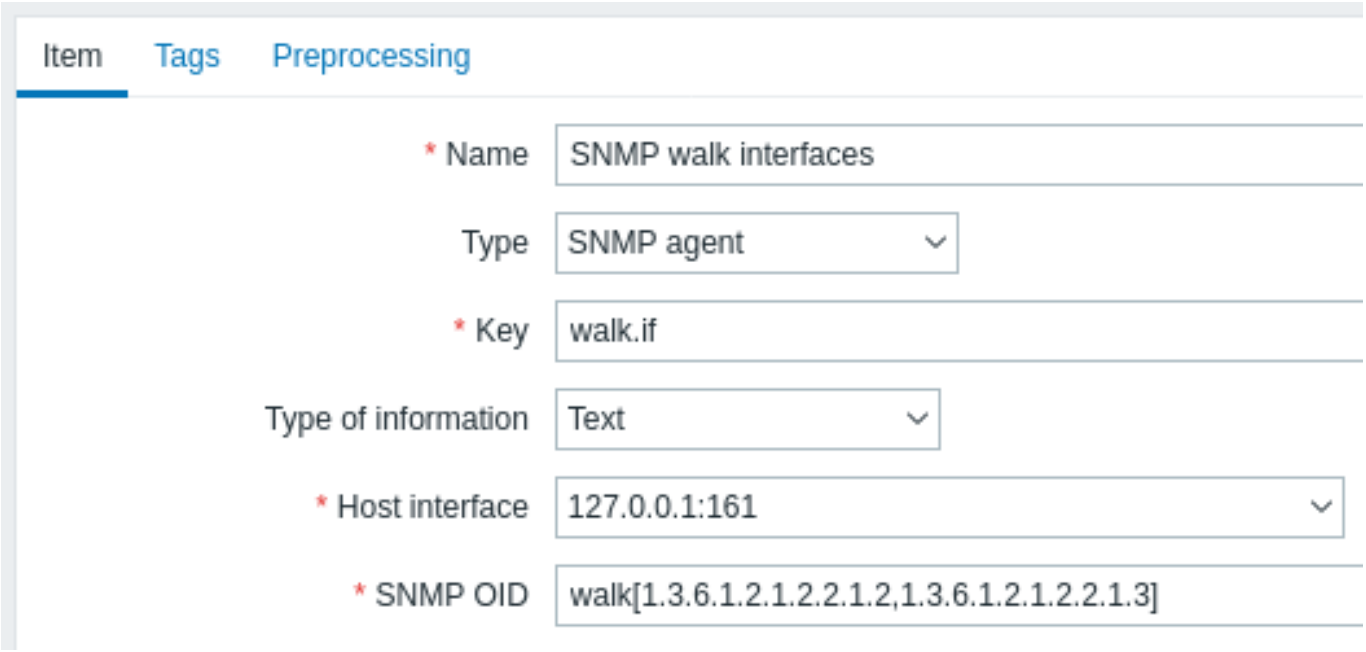
In this section we will perform an SNMP discovery on a switch.

This discovery method of SNMP OIDs has been supported since Zabbix server/proxy 6.4.

Item key

Create an SNMP item, using the following item key in the *SNMP OID* field:

`walk[1.3.6.1.2.1.2.2.1.2,1.3.6.1.2.1.2.2.1.3]`



This item will perform an snmpwalk for the OIDs specified in the parameters (1.3.6.1.2.1.2.2.1.2, 1.3.6.1.2.1.2.2.1.3), returning a concatenated list of values, e.g.:

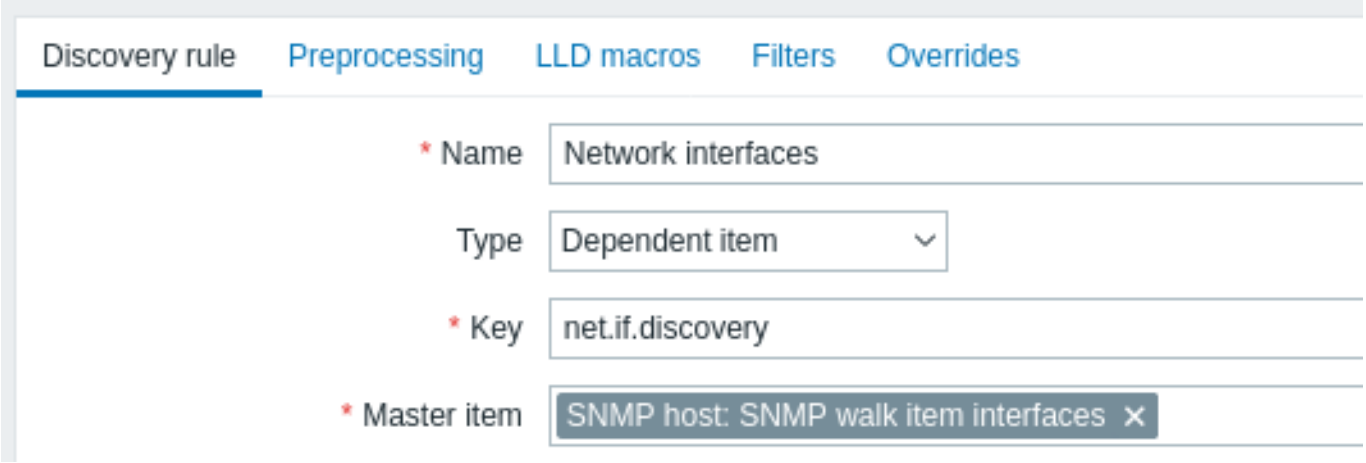
```
.1.3.6.1.2.1.2.2.1.2.1 = STRING: "lo"
.1.3.6.1.2.1.2.2.1.2.2 = STRING: "ens33"
.1.3.6.1.2.1.2.2.1.2.3 = STRING: "ens37"
.1.3.6.1.2.1.2.2.1.3.1 = INTEGER: 24
.1.3.6.1.2.1.2.2.1.3.2 = INTEGER: 6
.1.3.6.1.2.1.2.2.1.3.3 = INTEGER: 6
```

Dependent discovery rule

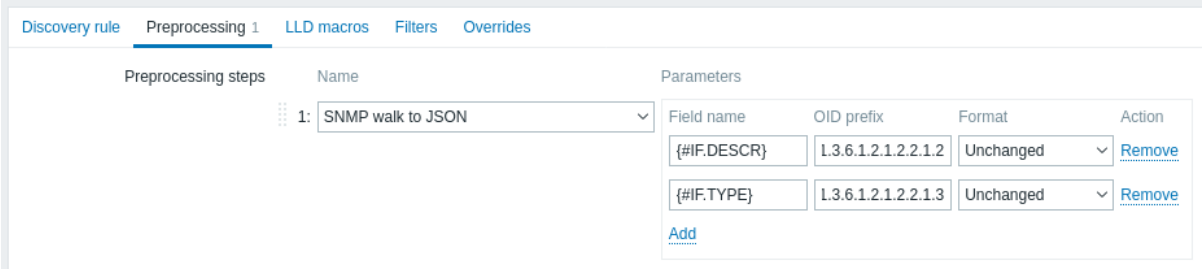
Go to the discovery rules of your template/host. Click on *Create discovery rule* in the upper right corner of the screen.

Fill in the required details in the **Discovery rule** tab:

- Select *Dependent item* as item type
- Select the previously created SNMP walk item as the master item
- Fill the name and key with meaningful values



In the **Preprocessing** tab, select the *SNMP walk to JSON* preproccesing step.



In the field name specify a valid LLD macro name. Select the corresponding OID path to discover values from.

This rule will discover entities and set:

- {#IFDESCR} macros to `lo`, `ens33`, and `ens37`;
- {#IFTYPE} macros set to 24, 6, and 6.

A built-in macro *{#SNMPINDEX}* containing the index of the discovered OIDs is applied to discovered entities. The discovered entities are grouped by *{#SNMPINDEX}* macro value: **1**, **2** and **3**:

```
[
    {
        "{#SNMPINDEX}": "1",
        "{#IFDESCR}": "lo",
        "{#IFTYPE}": "24"
    },
    {
        "{#SNMPINDEX}": "2",
        "{#IFDESCR}": "ens33",
        "{#IFTYPE}": "6"
    },
    {
        "{#SNMPINDEX}": "3",
        "{#IFDESCR}": "ens37",
        "{#IFTYPE}": "6"
```

```
    }
]
```

If an entity does not have the specified OID, then the corresponding macro will be omitted for this entity.

Item, trigger, graph prototypes

Item prototypes must be created as **dependent** item prototypes, using macros from the discovery rule.

Dependent items will obtain their values from the `walk[]` master item. Thus it will not be necessary for each discovered item to query the SNMP device independently.

Trigger and graph prototypes may also be created also by using macros from the discovery rule.

Discovered entities

When server runs, it will create real dependent items, triggers and graphs based on the values the SNMP discovery rule returns.

5 Discovery of SNMP OIDs (legacy)

Overview

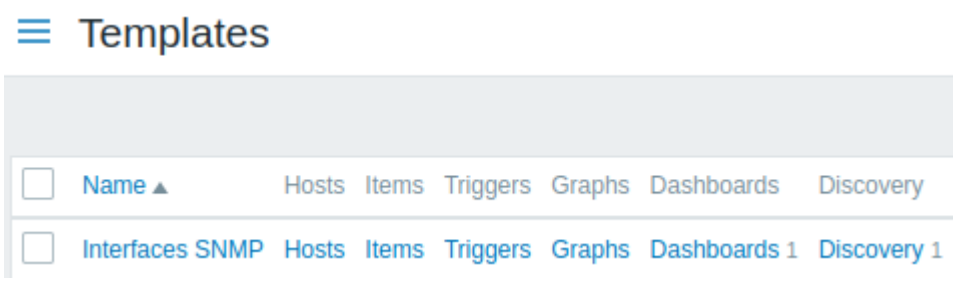
In this section we will perform an SNMP discovery on a switch.

Item key

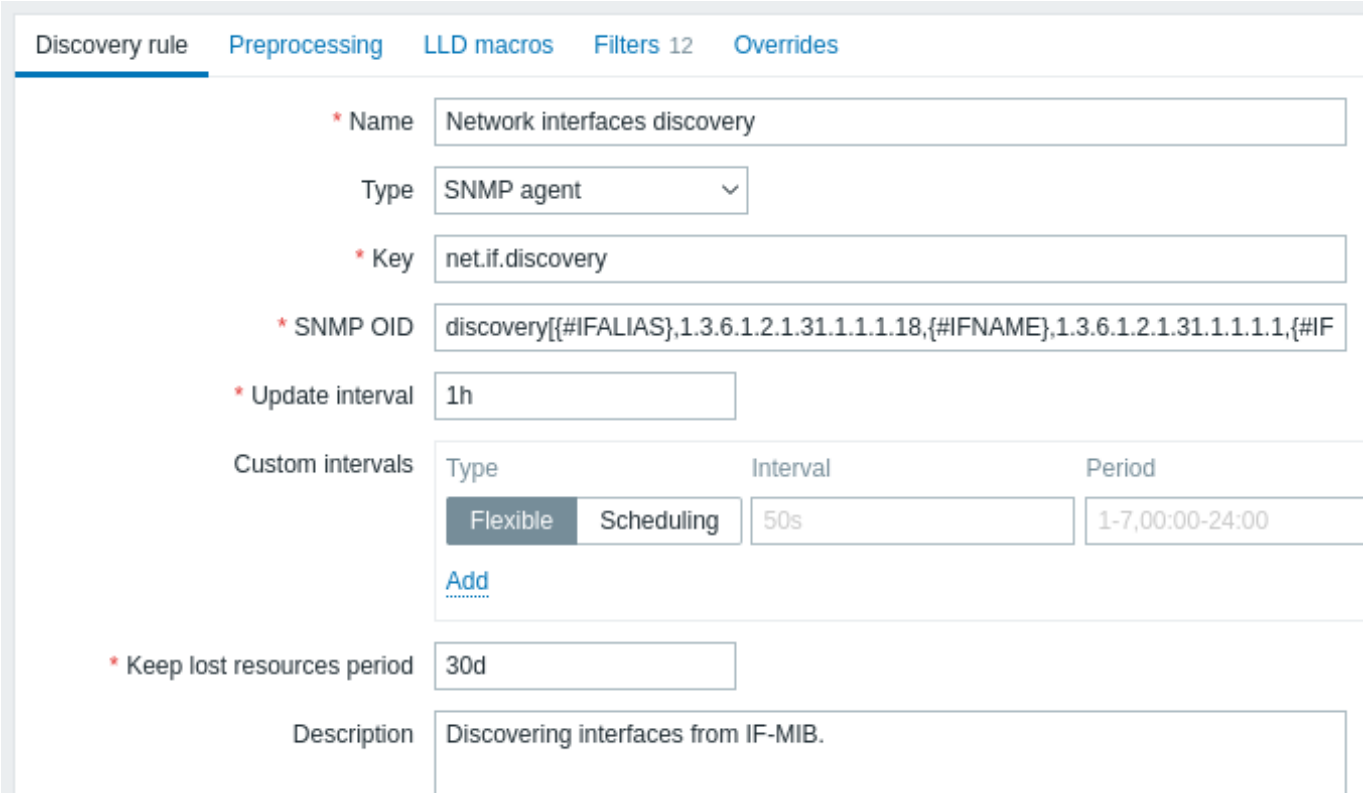
Unlike with file system and network interface discovery, the item does not necessarily has to have an "snmp.discovery" key - item type of SNMP agent is sufficient.

To configure the discovery rule, do the following:

- Go to: *Data collection → Templates*
- Click on *Discovery* in the row of an appropriate template



- Click on *Create discovery rule* in the upper right corner of the screen
- Fill in the discovery rule form with the required details as in the screenshot below



All mandatory input fields are marked with a red asterisk.

The OIDs to discover are defined in SNMP OID field in the following format: `discovery[{#MACRO1}, oid1, {#MACRO2}, oid2, …,]`

where *{#MACRO1}*, *{#MACRO2}* ... are valid lld macro names and *oid1*, *oid2*... are OIDs capable of generating meaningful values for these macros. A built-in macro *{#SNMPINDEX}* containing index of the discovered OID is applied to discovered entities. The discovered entities are grouped by *{#SNMPINDEX}* macro value.

To understand what we mean, let us perform few snmpwalks on our switch:

```
$ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifDescr
IF-MIB::ifDescr.1 = STRING: WAN
IF-MIB::ifDescr.2 = STRING: LAN1
IF-MIB::ifDescr.3 = STRING: LAN2

$ snmpwalk -v 2c -c public 192.168.1.1 IF-MIB::ifPhysAddress
IF-MIB::ifPhysAddress.1 = STRING: 8:0:27:90:7a:75
IF-MIB::ifPhysAddress.2 = STRING: 8:0:27:90:7a:76
IF-MIB::ifPhysAddress.3 = STRING: 8:0:27:2b:af:9e
```

And set SNMP OID to: `discovery[{#IFDESCR}, ifDescr, {#IFPHYSADDRESS}, ifPhysAddress]`

Now this rule will discover entities with {#IFDESCR} macros set to **WAN**, **LAN1** and **LAN2**, {#IFPHYSADDRESS} macros set to **8:0:27:90:7a:75**, **8:0:27:90:7a:76**, and **8:0:27:2b:af:9e**, {#SNMPINDEX} macros set to the discovered OIDs indexes **1**, **2** and **3**:

```
[
    {
        "{#SNMPINDEX}": "1",
        "{#IFDESCR}": "WAN",
        "{#IFPHYSADDRESS}": "8:0:27:90:7a:75"
    },
    {
        "{#SNMPINDEX}": "2",
        "{#IFDESCR}": "LAN1",
        "{#IFPHYSADDRESS}": "8:0:27:90:7a:76"
    },
    {
        "{#SNMPINDEX}": "3",
        "{#IFDESCR}": "LAN2",
```

```
        "{#IFPHYSADDRESS}": "8:0:27:2b:af:9e"
    }
]
```

If an entity does not have the specified OID, then the corresponding macro will be omitted for this entity. For example if we have the following data:

```
ifDescr.1 "Interface #1"
ifDescr.2 "Interface #2"
ifDescr.4 "Interface #4"

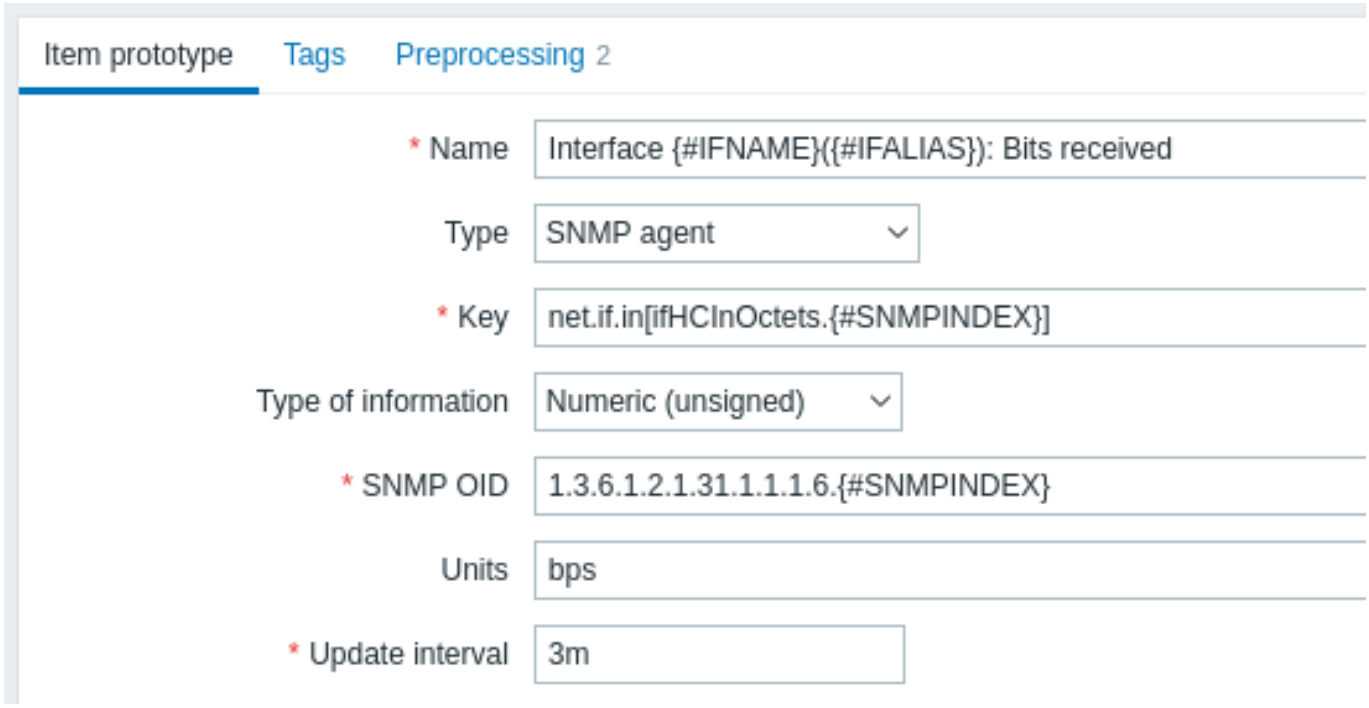
ifAlias.1 "eth0"
ifAlias.2 "eth1"
ifAlias.3 "eth2"
ifAlias.5 "eth4"
```

Then in this case SNMP discovery `discovery[{#IFDESCR}, ifDescr, {#IFALIAS}, ifAlias]` will return the following structure:

```
[
    {
        "{#SNMPINDEX}": 1,
        "{#IFDESCR}": "Interface #1",
        "{#IFALIAS}": "eth0"
    },
    {
        "{#SNMPINDEX}": 2,
        "{#IFDESCR}": "Interface #2",
        "{#IFALIAS}": "eth1"
    },
    {
        "{#SNMPINDEX}": 3,
        "{#IFALIAS}": "eth2"
    },
    {
        "{#SNMPINDEX}": 4,
        "{#IFDESCR}": "Interface #4"
    },
    {
        "{#SNMPINDEX}": 5,
        "{#IFALIAS}": "eth4"
    }
]
```

Item prototypes

The following screenshot illustrates how we can use these macros in item prototypes:



You can create as many item prototypes as needed:

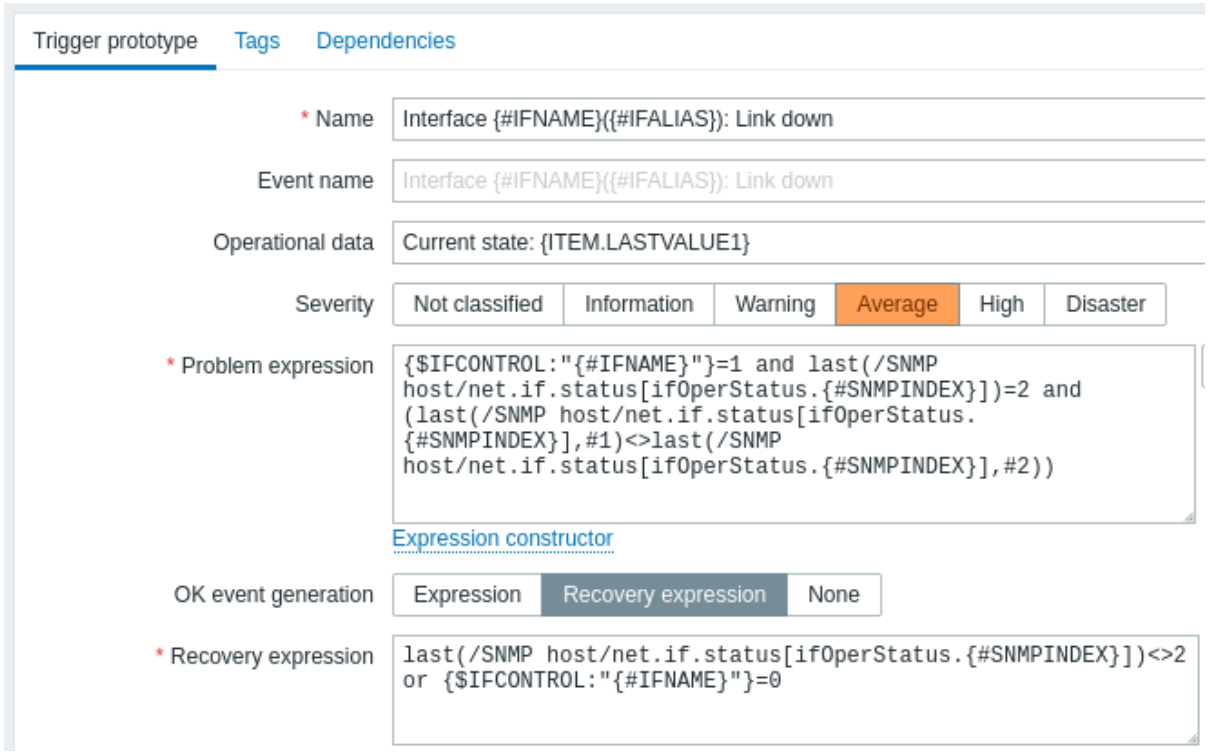
Item prototypes

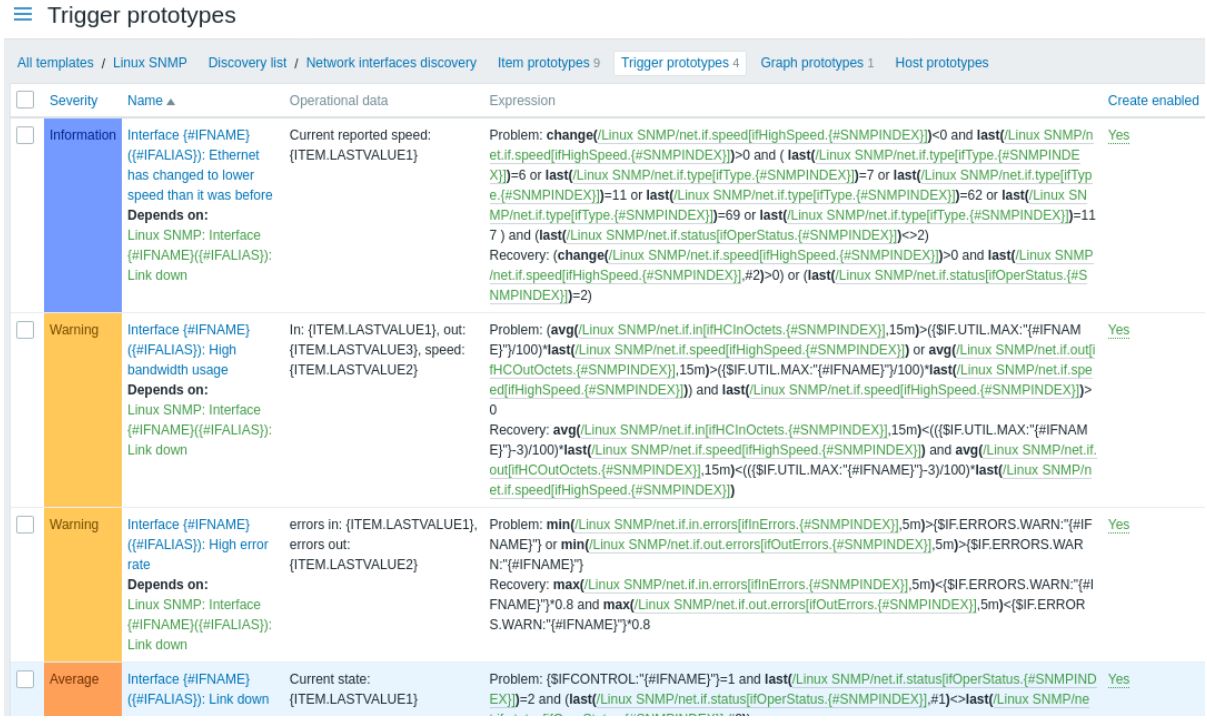
All templates / Linux SNMP Discovery list / Network interfaces discovery Item prototypes 9 Trigger prototypes 4 Graph prototypes 1 Host prototypes

Name ▲	Key	Interval	History	Trends	Type	Create enabled
Interface {#IFNAME}({#IFALIAS}): Bits received	net.if.in[ifHCInOctets.{#SNMPINDEX}]	3m	7d	365d	SNMP agent	Yes
Interface {#IFNAME}({#IFALIAS}): Bits sent	net.if.out[ifHCOutOctets.{#SNMPINDEX}]	3m	7d	365d	SNMP agent	Yes
Interface {#IFNAME}({#IFALIAS}): Inbound packets discarded	net.if.in.discards[ifInDiscards.{#SNMPINDEX}]	3m	7d	365d	SNMP agent	Yes
Interface {#IFNAME}({#IFALIAS}): Inbound packets with errors	net.if.in.errors[ifInErrors.{#SNMPINDEX}]	3m	7d	365d	SNMP agent	Yes
Interface {#IFNAME}({#IFALIAS}): Interface type	net.if.type[ifType.{#SNMPINDEX}]	1h	7d	0d	SNMP agent	Yes
Interface {#IFNAME}({#IFALIAS}): Operational status	net.if.status[ifOperStatus.{#SNMPINDEX}]	1m	7d	0	SNMP agent	Yes
Interface {#IFNAME}({#IFALIAS}): Outbound packets discarded	net.if.out.discards[ifOutDiscards.{#SNMPINDEX}]	3m	7d	365d	SNMP agent	Yes
Interface {#IFNAME}({#IFALIAS}): Outbound packets with errors	net.if.out.errors[ifOutErrors.{#SNMPINDEX}]	3m	7d	365d	SNMP agent	Yes
Interface {#IFNAME}({#IFALIAS}): Speed	net.if.speed[ifHighSpeed.{#SNMPINDEX}]	5m	7d	0d	SNMP agent	Yes

Trigger prototypes

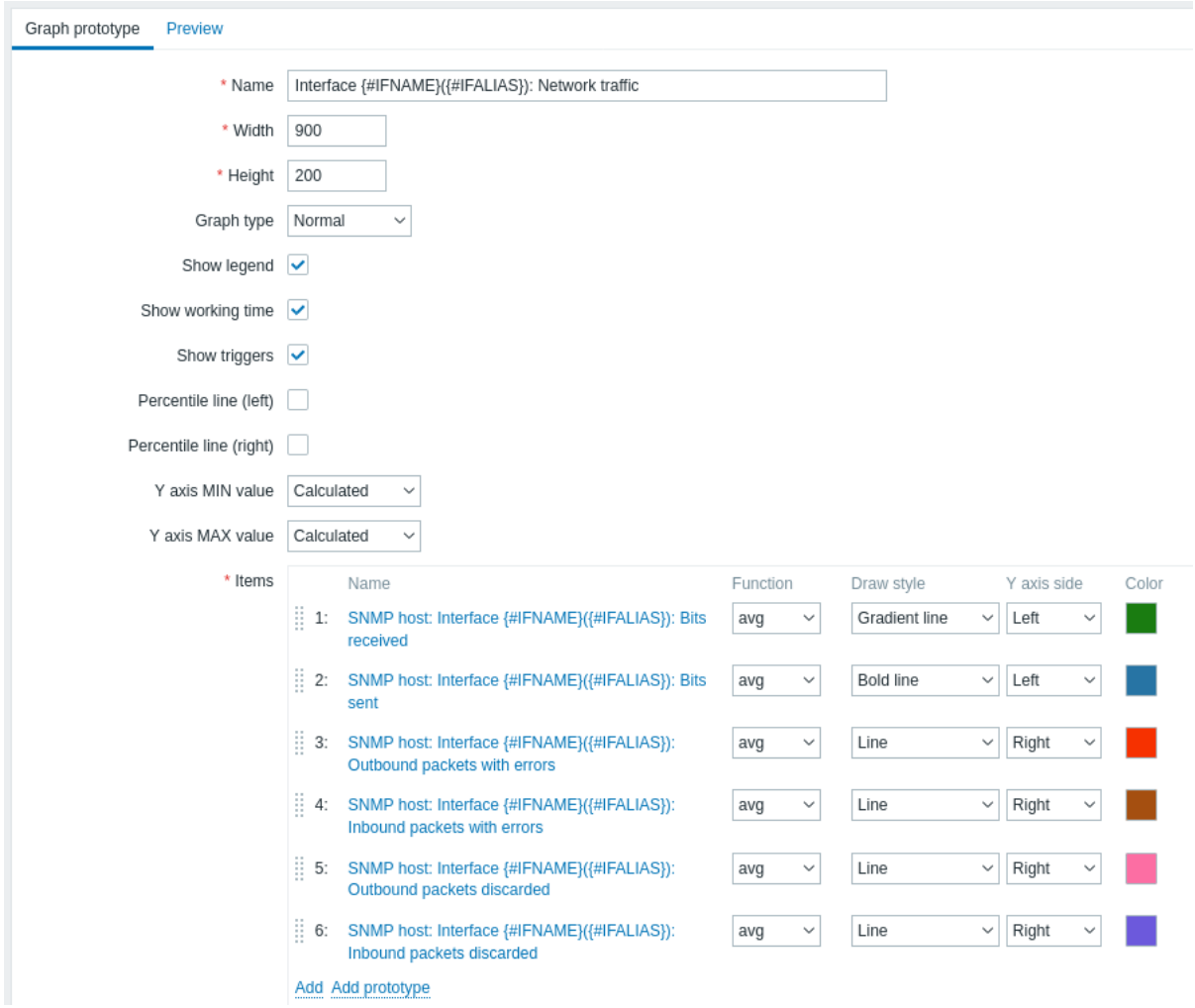
The following screenshot illustrates how we can use these macros in trigger prototypes:

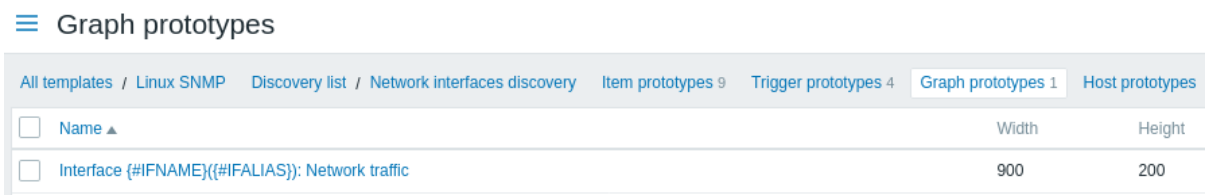




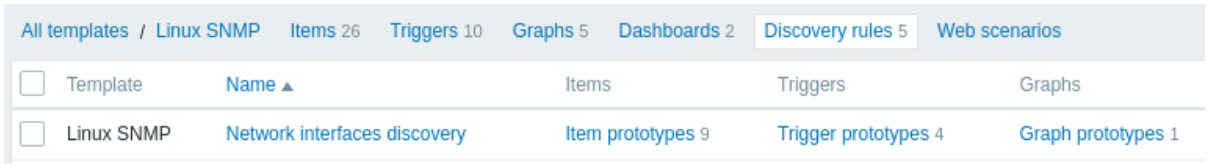
Graph prototypes

The following screenshot illustrates how we can use these macros in graph prototypes:





A summary of our discovery rule:



Discovered entities

When server runs, it will create real items, triggers and graphs based on the values the SNMP discovery rule returns. In the host configuration they are prefixed with an orange link to a discovery rule they come from.

Items

All hosts / SNMP host Enabled SNMP Items 81 Triggers 23 Graphs 14 Discovery rules 6 Web scenarios

Name ▲	Triggers	Key	Interval	History	Trends	Type	Status
Network interfaces discovery: Interface enp4s0(): Bits received	Triggers 1	net.if.in[ifHCInOctets.2]	3m	7d	365d	SNMP agent	Enabled
Network interfaces discovery: Interface enp4s0(): Bits sent	Triggers 1	net.if.out[ifHCOutOctets.2]	3m	7d	365d	SNMP agent	Enabled
Network interfaces discovery: Interface enp4s0(): Inbound packets discarded		net.if.in.discards[ifInDiscards.2]	3m	7d	365d	SNMP agent	Enabled
Network interfaces discovery: Interface enp4s0(): Inbound packets with errors	Triggers 1	net.if.in.errors[ifInErrors.2]	3m	7d	365d	SNMP agent	Enabled
Network interfaces discovery: Interface enp4s0(): Interface type	Triggers 1	net.if.type[ifType.2]	1h	7d	0d	SNMP agent	Enabled
Network interfaces discovery: Interface enp4s0(): Operational status	Triggers 2	net.if.status[ifOperStatus.2]	1m	7d	0	SNMP agent	Enabled
Network interfaces discovery: Interface enp4s0(): Outbound packets discarded		net.if.out.discards[ifOutDiscards.2]	3m	7d	365d	SNMP agent	Enabled
Network interfaces discovery: Interface enp4s0(): Outbound packets with errors	Triggers 1	net.if.out.errors[ifOutErrors.2]	3m	7d	365d	SNMP agent	Enabled
Network interfaces discovery: Interface enp4s0(): Speed	Triggers 2	net.if.speed[ifHighSpeed.2]	5m	7d	0d	SNMP agent	Enabled

Triggers

All hosts / SNMP host Enabled SNMP Items 81 Triggers 23 Graphs 14 Discovery rules 6 Web scenarios

Severity	Value	Name ▲	Operational data	Expression
Information	OK	Network interfaces discovery: Interface enp4s0(): Ethernet has changed to lower speed than it was before **Depends on:** SNMP host: Interface enp4s0(): Link down	Current reported speed: {ITEM.LASTVALUE1}	Problem: **change(**/SNMP host/net.if.speed[ifHighSpeed.2]**)**<0 and **last(**/SNMP host/net.if.speed[ifHighSpeed.2]**)**>0 and (**last(**/SNMP host/net.if.type[ifType.2]**)**=6 or **last(**/SNMP host/net.if.type[ifType.2]**)**=7 or **last(**/SNMP host/net.if.type[ifType.2]**)**=11 or **last(**/SNMP host/net.if.type[ifType.2]**)**=62 or **last(**/SNMP host/net.if.type[ifType.2]**)**=69 or **last(**/SNMP host/net.if.type[ifType.2]**)**=117) and (**last(**/SNMP host/net.if.status[ifOperStatus.2]**)**<>2) Recovery: (**change(**/SNMP host/net.if.speed[ifHighSpeed.2]**)**>0 and **last(**/SNMP host/net.if.speed[ifHighSpeed.2],#2**)**>0) or (**last(**/SNMP host/net.if.status[ifOperStatus.2]**)**=2)
Warning	OK	Network interfaces discovery: Interface enp4s0(): High bandwidth usage **Depends on:** SNMP host: Interface enp4s0(): Link down	In: {ITEM.LASTVALUE1}, out: {ITEM.LASTVALUE3}, speed: {ITEM.LASTVALUE2}	Problem: (**avg(**/SNMP host/net.if.in[ifHCInOctets.2],15m**)**>({$IF.UTIL.MAX:"enp4s0"}/100)***last(**/SNMP host/net.if.speed[ifHighSpeed.2]**)**) or **avg(**/SNMP host/net.if.out[ifHCOutOctets.2],15m**)**>({$IF.UTIL.MAX:"enp4s0"}/100)***last(**/SNMP host/net.if.speed[ifHighSpeed.2]**)**) and **last(**/SNMP host/net.if.speed[ifHighSpeed.2]**)**>0 Recovery: **avg(**/SNMP host/net.if.in[ifHCInOctets.2],15m**)**<(({$IF.UTIL.MAX:"enp4s0"}-3)/100)***last(**/SNMP host/net.if.speed[ifHighSpeed.2]**)** and **avg(**/SNMP host/net.if.out[ifHCOutOctets.2],15m**)**<(({$IF.UTIL.MAX:"enp4s0"}-3)/100)***last(**/SNMP host/net.if.speed[ifHighSpeed.2]**)**
Warning	OK	Network interfaces discovery: Interface enp4s0(): High error rate **Depends on:** SNMP host: Interface enp4s0(): Link down	errors in: {ITEM.LASTVALUE1}, errors out: {ITEM.LASTVALUE2}	Problem: **min(**/SNMP host/net.if.in.errors[ifInErrors.2],5m**)**>{$IF.ERRORS.WARN:"enp4s0"} or **min(**/SNMP host/net.if.out.errors[ifOutErrors.2],5m**)**>{$IF.ERRORS.WARN:"enp4s0"} Recovery: **max(**/SNMP host/net.if.in.errors[ifInErrors.2],5m**)**<{$IF.ERRORS.WARN:"enp4s0"}*0.8 and **max(**/SNMP host/net.if.out.errors[ifOutErrors.2],5m**)**<{$IF.ERRORS.WARN:"enp4s0"}*0.8
Average	OK	Network interfaces discovery: Interface enp4s0(): Link down	Current state: {ITEM.LASTVALUE1}	Problem: {$IFCONTROL:"enp4s0"}=1 and **last(**/SNMP host/net.if.status[ifOperStatus.2]**)**=2 and (**last(**/SNMP host/net.if.status[ifOperStatus.2],#1**)**<>**last(**/SNMP host/net.if.status[ifOperStatus.2],#2**)**) Recovery: **last(**/SNMP host/net.if.status[ifOperStatus.2]**)**<>2 or {$IFCONTROL:"enp4s0"}=0

Graphs

All hosts / SNMP host Enabled SNMP Items 81 Triggers 23 Graphs 14 Discovery rules 6 Web scenarios

Name ▲
Mounted filesystem discovery: /: Disk space usage
Linux SNMP: CPU jumps
CPU discovery: CPU usage
CPU discovery: CPU utilization
Network interfaces discovery: Interface enp4s0(): Network traffic

6 Discovery of JMX objects

Overview

It is possible to discover all JMX MBeans or MBean attributes or to specify a pattern for the discovery of these objects.

It is mandatory to understand the difference between an MBean and MBean attributes for discovery rule configuration. An MBean is an object which can represent a device, an application, or any resource that needs to be managed.

For example, there is an MBean which represents a web server. Its attributes are connection count, thread count, request timeout, http file cache, memory usage, etc. Expressing this thought in human comprehensive language we can define a coffee machine as an MBean which has the following attributes to be monitored: water amount per cup, average consumption of water for a certain period of time, number of coffee beans required per cup, coffee beans and water refill time, etc.

Item key

In discovery rule configuration, select **JMX agent** in the *Type* field.

Two item keys are supported for JMX object discovery - jmx.discovery[] and jmx.get[]:

Item key	Return value	Parameters	Comment
jmx.discovery[<discovery mode>,<object name>,<unique short description>]	This item returns a JSON array with LLD macros describing MBean objects or their attributes.	**discovery mode** - one of the following: *attributes* (retrieve JMX MBean attributes, default) or *beans* (retrieve JMX MBeans) **object name** - object name pattern (see documentation) identifying the MBean names to be retrieved (empty by default, retrieving all registered beans) **unique short description** - a unique description that allows multiple JMX items with the same discovery mode and object name on the host (optional)	Examples: → jmx.discovery - retrieve all JMX MBean attributes → jmx.discovery[beans] - retrieve all JMX MBeans → jmx.discovery[attributes,"*:type=GarbageCollector,name - retrieve all garbage collector attributes → jmx.discovery[beans,"*:type=GarbageCollector,name=* - retrieve all garbage collectors There are some limitations to what MBean properties this item can return based on limited characters that are supported in macro name generation (supported characters can be described by the following regular expression: `A-Z0-9_\.`). So, for example, to discover MBean properties with a hyphenated word or non-ASCII characters, you need to use `jmx.get[]`. Supported since Zabbix Java gateway 3.4.
jmx.get[<discovery mode>,<object name>,<unique short description>]			

Item key			
	This item returns a JSON array with MBean objects or their attributes. Compared to `jmx.discovery[]` it does not define LLD macros.	**discovery mode** - one of the following: *attributes* (retrieve JMX MBean attributes, default) or *beans* (retrieve JMX MBeans) **object name** - object name pattern (see documentation) identifying the MBean names to be retrieved (empty by default, retrieving all registered beans) **unique short description** - a unique description that allows multiple JMX items with the same discovery mode and object name on the host (optional)	When using this item, it is needed to define custom low-level discovery macros, pointing to values extracted from the returned JSON using JSONPath. Supported since Zabbix Java gateway 4.4.

Attention:
If no parameters are passed, all MBean attributes from JMX are requested. Not specifying parameters for JMX discovery or trying to receive all attributes for a wide range like `*:type=*,name=*` may lead to potential performance problems.

Using jmx.discovery

This item returns a JSON object with low-level discovery macros describing MBean objects or attributes. For example, in the discovery of MBean attributes (reformatted for clarity):

```
[
    {
        "{#JMXVALUE}":"0",
        "{#JMXTYPE}":"java.lang.Long",
        "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
        "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS Scavenge,CollectionCount",
        "{#JMXATTR}":"CollectionCount"
    },
    {
        "{#JMXVALUE}":"0",
        "{#JMXTYPE}":"java.lang.Long",
        "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
        "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS Scavenge,CollectionTime",
        "{#JMXATTR}":"CollectionTime"
    },
    {
        "{#JMXVALUE}":"true",
        "{#JMXTYPE}":"java.lang.Boolean",
        "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
        "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS Scavenge,Valid",
        "{#JMXATTR}":"Valid"
    },
    {
        "{#JMXVALUE}":"PS Scavenge",
        "{#JMXTYPE}":"java.lang.String",
        "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
        "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS Scavenge,Name",
        "{#JMXATTR}":"Name"
    },
    {
        "{#JMXVALUE}":"java.lang:type=GarbageCollector,name=PS Scavenge",
        "{#JMXTYPE}":"javax.management.ObjectName",
        "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
        "{#JMXDESC}":"java.lang:type=GarbageCollector,name=PS Scavenge,ObjectName",
        "{#JMXATTR}":"ObjectName"
    }
```

```
]
```

In the discovery of MBeans (reformatted for clarity):

```
[
    {
        "{#JMXDOMAIN}":"java.lang",
        "{#JMXTYPE}":"GarbageCollector",
        "{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenge",
        "{#JMXNAME}":"PS Scavenge"
    }
]
```

Supported macros

The following macros are supported for use in the discovery rule filter and prototypes of items, triggers and graphs:

Macro	Description
Discovery of MBean attributes	
{#JMXVALUE}	Attribute value.
{#JMXTYPE}	Attribute type.
{#JMXOBJ}	Object name.
{#JMXDESC}	Object name including attribute name.
{#JMXATTR}	Attribute name.
Discovery of MBeans	
{#JMXDOMAIN}	MBean domain. (*Zabbix reserved name*)
{#JMXOBJ}	Object name. (*Zabbix reserved name*)
{#JMX<key property>}	MBean properties (like {#JMXTYPE}, {#JMXNAME}) (see Limitations below).

Limitations

There are some limitations associated with the algorithm of creating LLD macro names from MBean property names:

- attribute names are changed to uppercase
- attribute names are ignored (no LLD macros are generated) if they consist of unsupported characters for LLD macro names. Supported characters can be described by the following regular expression: `A-Z0-9_\.`
- if an attribute is called "obj" or "domain" they will be ignored because of the overlap with the values of the reserved Zabbix properties {#JMXOBJ} and {#JMXDOMAIN} (supported since Zabbix 3.4.3.)

Please consider this jmx.discovery (with "beans" mode) example. MBean has the following properties defined (some of which will be ignored; see below):

```
name=test
  =Type
attributes []=1,2,3
Name=NameOfTheTest
domAin=some
```

As a result of JMX discovery, the following LLD macros will be generated:

- {#JMXDOMAIN} - Zabbix internal, describing the domain of MBean
- {#JMXOBJ} - Zabbix internal, describing MBean object
- {#JMXNAME} - created from "name" property

Ignored properties are:

- тип : its name contains unsupported characters (non-ASCII)
- attributes[] : its name contains unsupported characters (square brackets are not supported)
- Name : it's already defined (name=test)
- domAin : it's a Zabbix reserved name

Examples

Let's review two more practical examples of an LLD rule creation with the use of MBean. To understand the difference between an LLD rule collecting MBeans and an LLD rule collecting MBean attributes better please take a look at following table:

MBean1	**MBean2**	**MBean3**
MBean1Attribute1	MBean2Attribute1	MBean3Attribute1

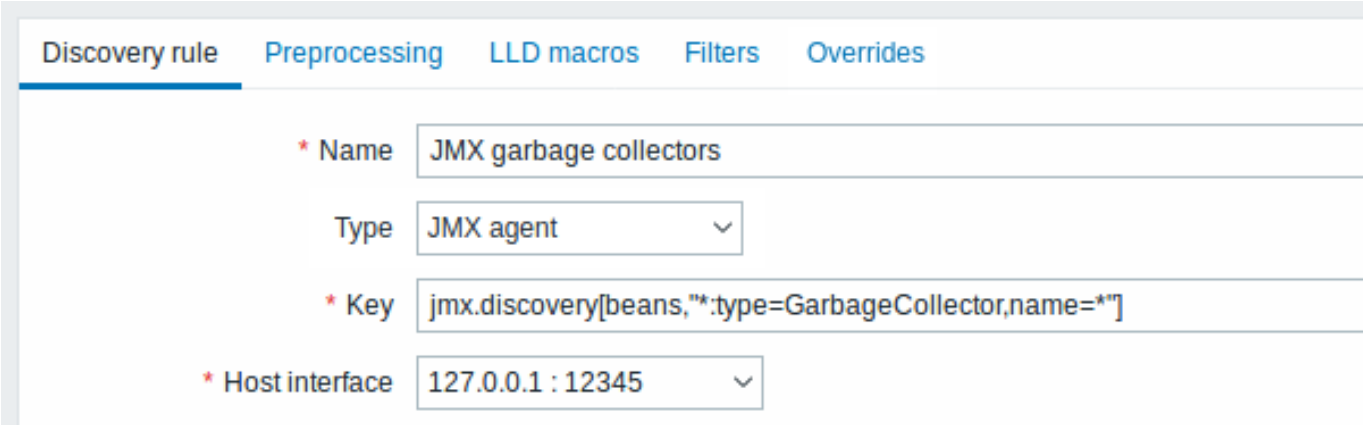
MBean1Attribute2	MBean2Attribute2	MBean3Attribute2
MBean1Attribute3	MBean2Attribute3	MBean3Attribute3

Example 1: Discovering Mbeans

This rule will return 3 objects: the top row of the column: MBean1, MBean2, MBean3.

For more information about objects please refer to supported macros table, *Discovery of MBeans* section.

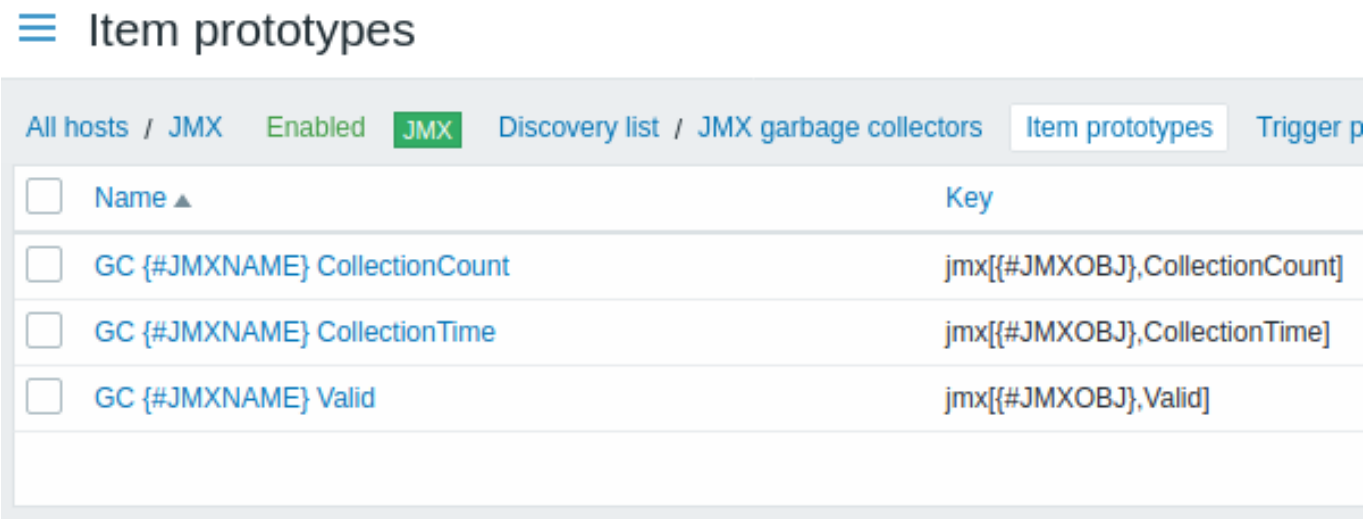
Discovery rule configuration collecting MBeans (without the attributes) looks like the following:



The key used here:

```
jmx.discovery[beans,"*:type=GarbageCollector,name=*"]
```

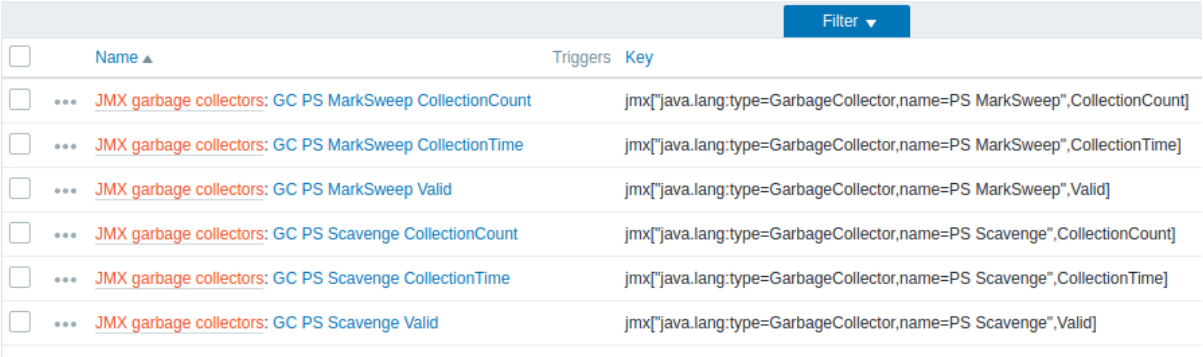
All the garbage collectors without attributes will be discovered. As Garbage collectors have the same attribute set, we can use desired attributes in item prototypes the following way:



The keys used here:

```
jmx[{#JMXOBJ},CollectionCount]
jmx[{#JMXOBJ},CollectionTime]
jmx[{#JMXOBJ},Valid]
```

LLD discovery rule will result in something close to this (items are discovered for two Garbage collectors):

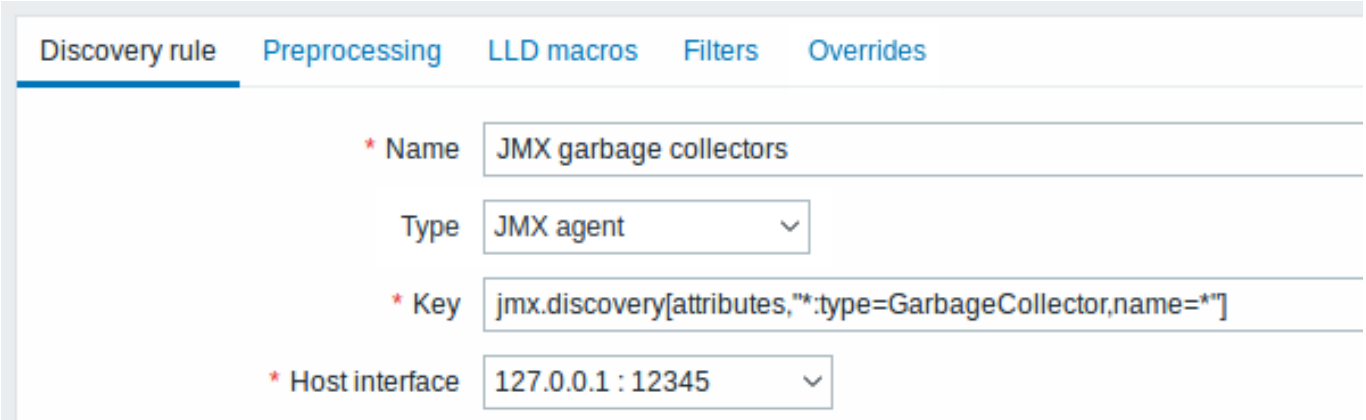


Example 2: Discovering Mbean attributes

This rule will return 9 objects with the following fields: MBean1Attribute1, MBean2Attribute1, MBean3Attribute1, MBean1Attribute2, MBean2Attribute2, MBean3Attribute2, MBean1Attribute3, MBean2Attribute3, MBean3Attribute3.

For more information about objects please refer to supported macros table, *Discovery of MBean attributes* section.

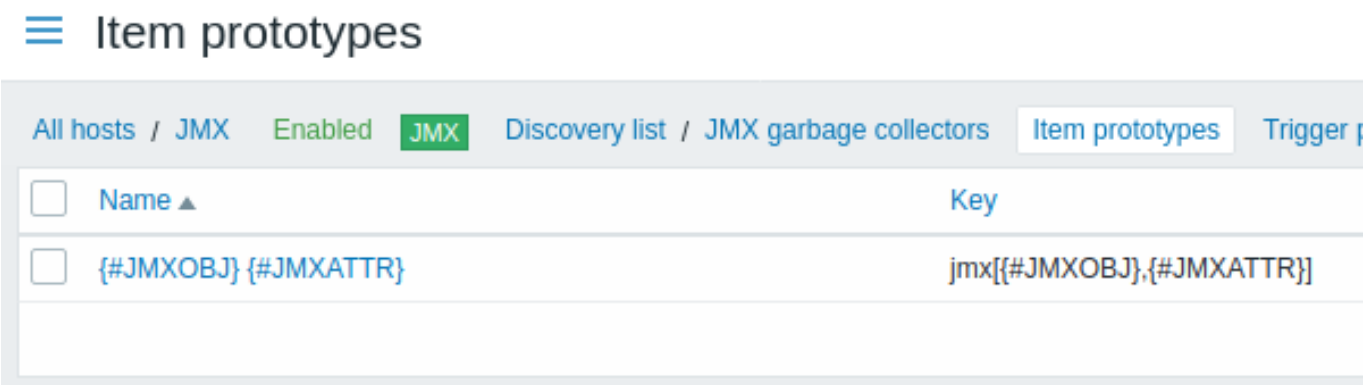
Discovery rule configuration collecting MBean attributes looks like the following:



The key used here:

```
jmx.discovery[attributes,"*:type=GarbageCollector,name=*"]
```

All the garbage collectors with a single item attribute will be discovered.



In this particular case an item will be created from prototype for every MBean attribute. The main drawback of this configuration is that trigger creation from trigger prototypes is impossible as there is only one item prototype for all attributes. So this setup can be used for data collection, but is not recommended for automatic monitoring.

Using jmx.get

`jmx.get[]` is similar to the `jmx.discovery[]` item, but it does not turn Java object properties into low-level discovery macro names and therefore can return values without limitations that are associated with LLD macro name generation such as hyphens or non-ASCII characters.

When using `jmx.get[]` for discovery, low-level discovery macros can be defined separately in the custom LLD macro tab of the discovery rule configuration, using JSONPath to point to the required values.

Discovering MBeans

Discovery item: `jmx.get[beans,"com.example:type=*,*"]`

Response:

```
[
    {
        "object": "com.example:type=Hello,data-src=data-base,  =    ",
        "domain": "com.example",
        "properties": {
            "data-src": "data-base",
            "  ": "    ",
            "type": "Hello"
        }
    },
    {
        "object": "com.example:type=Atomic",
        "domain": "com.example",
        "properties": {
            "type": "Atomic"
        }
    }
]
```

Discovering MBean attributes

Discovery item: `jmx.get[attributes,"com.example:type=*,*"]`

Response:

```
[
    {
        "object": "com.example:type=*",
        "domain": "com.example",
        "properties": {
            "type": "Simple"
        }
    },
    {
        "object": "com.zabbix:type=yes,domain=zabbix.com,data-source=/dev/rand,  =    ,obj=true",
        "domain": "com.zabbix",
        "properties": {
            "type": "Hello",
            "domain": "com.example",
            "data-source": "/dev/rand",
            "  ": "    ",
            "obj": true
        }
    }
]
```

7 Discovery of IPMI sensors

Overview

It is possible to automatically discover IPMI sensors.

To do that, you may use a combination of:

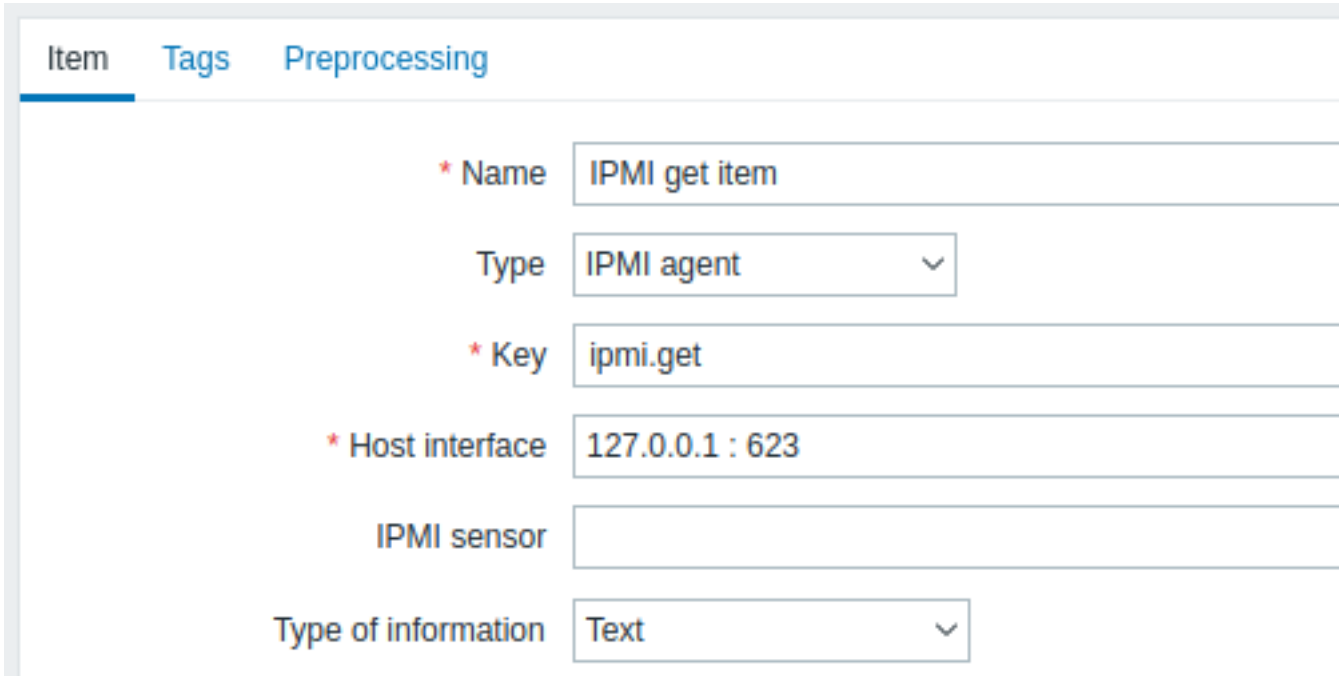
- the `ipmi.get` IPMI item (supported since Zabbix **5.0.0**) as the master item
- dependent low-level discovery rule and item prototypes

Configuration

Master item

Create an IPMI item using the following key:

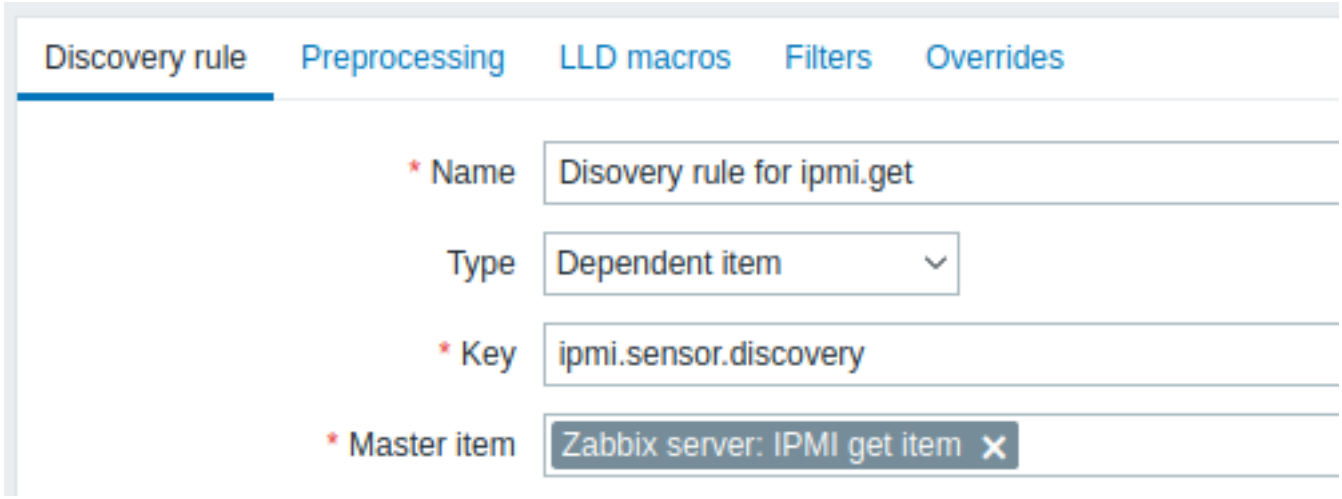
`ipmi.get`



Set the type of information to ”Text” for possibly big JSON data.

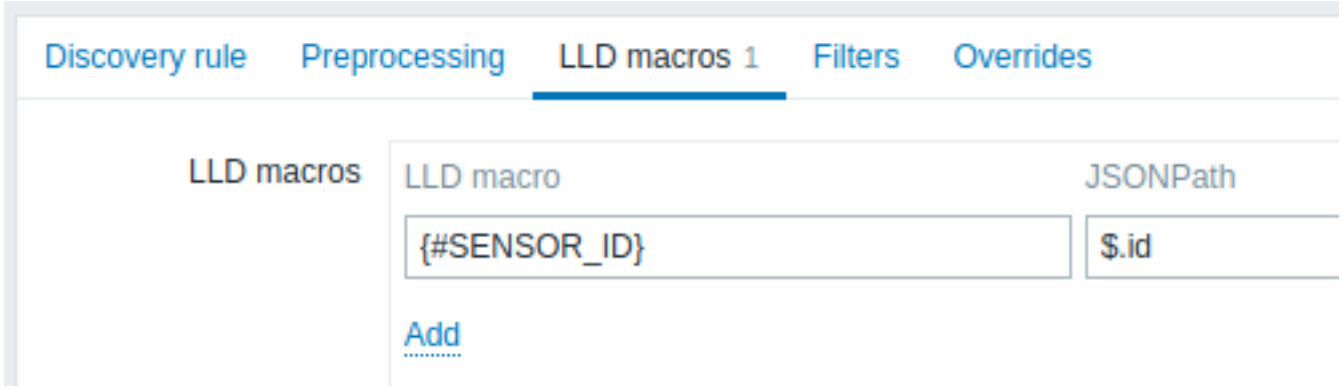
Dependent LLD rule

Create a low-level discovery rule as ”Dependent item” type:



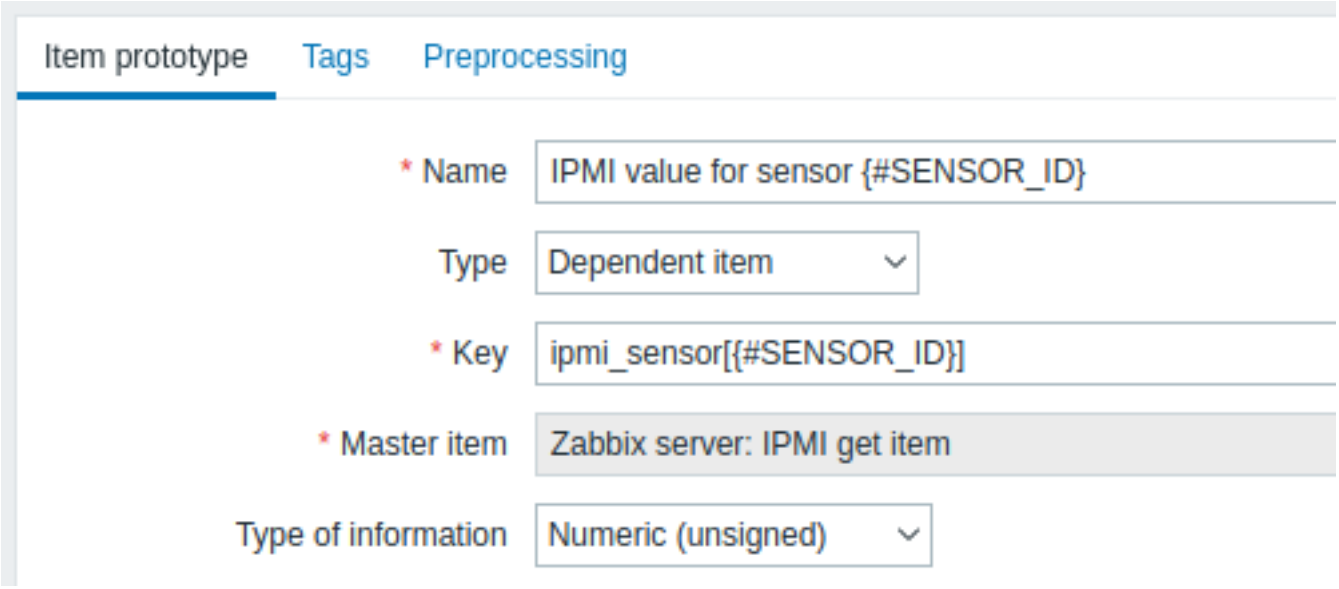
As master item select the `ipmi.get` item we created.

In the ”LLD macros” tab define a custom macro with the corresponding JSONPath:



Dependent item prototype

Create an item prototype with ”Dependent item” type in this LLD rule. As master item for this prototype select the `ipmi.get` item we created.



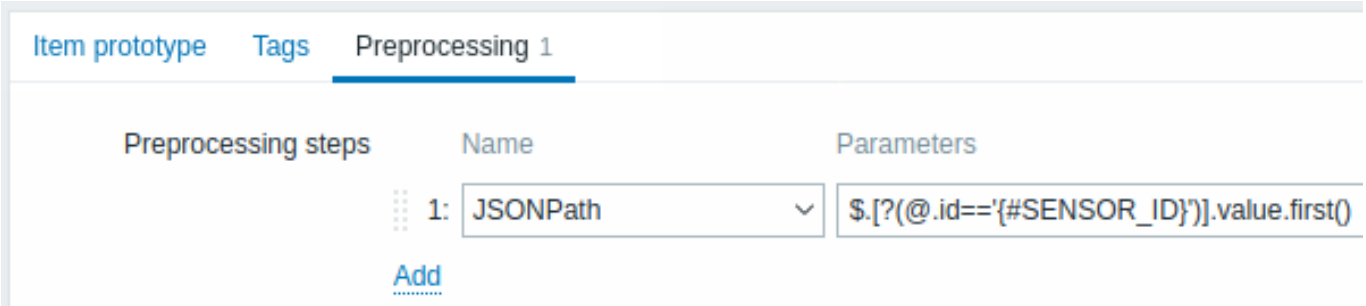
Note the use of the {#SENSOR_ID} macro in the item prototype name and key:

- *Name*: IPMI value for sensor {#SENSOR_ID}
- *Key*: ipmi_sensor[{#SENSOR_ID}]

As type of information, *Numeric (unsigned)*.

In the item prototype "Preprocessing" tab select JSONPath and use the following JSONPath expression as parameter:

```
$.[?(@.id=='{#SENSOR_ID}')].value.first()
```



When discovery starts, one item per each IPMI sensor will be created. This item will return the integer value of the given sensor.

8 Discovery of systemd services

Overview

It is possible to discover systemd units (services, by default) with Zabbix.

Item key

The item to use in the discovery rule is the

`systemd.unit.discovery`

> **Attention:**
> This item key is only supported in Zabbix agent 2.

This item returns a JSON with information about systemd units, for example:

```
[{
    "{#UNIT.NAME}": "mysqld.service",
    "{#UNIT.DESCRIPTION}": "MySQL Server",
    "{#UNIT.LOADSTATE}": "loaded",
    "{#UNIT.ACTIVESTATE}": "active",
    "{#UNIT.SUBSTATE}": "running",
    "{#UNIT.FOLLOWED}": "",
    "{#UNIT.PATH}": "/org/freedesktop/systemd1/unit/mysqld_2eservice",
```

```
    "{#UNIT.JOBID}": 0,
    "{#UNIT.JOBTYPE}": "",
    "{#UNIT.JOBPATH}": "/",
    "{#UNIT.UNITFILESTATE}": "enabled"
}, {
    "{#UNIT.NAME}": "systemd-journald.socket",
    "{#UNIT.DESCRIPTION}": "Journal Socket",
    "{#UNIT.LOADSTATE}": "loaded",
    "{#UNIT.ACTIVESTATE}": "active",
    "{#UNIT.SUBSTATE}": "running",
    "{#UNIT.FOLLOWED}": "",
    "{#UNIT.PATH}": "/org/freedesktop/systemd1/unit/systemd_2djournald_2esocket",
    "{#UNIT.JOBID}": 0,
    "{#UNIT.JOBTYPE}": "",
    "{#UNIT.JOBPATH}": "/",
    "{#UNIT.UNITFILESTATE}": "enabled"
}]
```

Discovery of disabled systemd units

Since Zabbix 6.0.1 it is also possible to discover **disabled** systemd units. In this case three macros are returned in the resulting JSON:

- {#UNIT.PATH}
- {#UNIT.ACTIVESTATE}
- {#UNIT.UNITFILESTATE}.

Attention:
To have items and triggers created from prototypes for disabled systemd units, make sure to adjust (or remove) prohibiting LLD filters for {#UNIT.ACTIVESTATE} and {#UNIT.UNITFILESTATE}.

Supported macros

The following macros are supported for use in the discovery rule filter and prototypes of items, triggers and graphs:

Macro	Description
{#UNIT.NAME}	Primary unit name.
{#UNIT.DESCRIPTION}	Human readable description.
{#UNIT.LOADSTATE}	Load state (i.e. whether the unit file has been loaded successfully)
{#UNIT.ACTIVESTATE}	Active state (i.e. whether the unit is currently started or not)
{#UNIT.SUBSTATE}	Sub state (a more fine-grained version of the active state that is specific to the unit type, which the active state is not)
{#UNIT.FOLLOWED}	Unit that is being followed in its state by this unit, if there is any; otherwise an empty string.
{#UNIT.PATH}	Unit object path.
{#UNIT.JOBID}	Numeric job ID if there is a job queued for the job unit; 0 otherwise.
{#UNIT.JOBTYPE}	Job type.
{#UNIT.JOBPATH}	Job object path.
{#UNIT.UNITFILESTATE}	The install state of the unit file.

Item prototypes

Item prototypes that can be created based on systemd service discovery include, for example:

- Item name: `{#UNIT.DESCRIPTION} active state info`; item key: `systemd.unit.info["{#UNIT.NAME}"]`
- Item name: `{#UNIT.DESCRIPTION} load state info`; item key: `systemd.unit.info["{#UNIT.NAME}",LoadState]`

`systemd.unit.info` agent items are supported since Zabbix 4.4.

9 Discovery of Windows services

Overview

In a similar way as file systems are discovered, it is possible to also discover Windows services.

Item key

The item to use in the discovery rule is

`service.discovery`

This item is supported since Zabbix Windows agent 3.0.

Supported macros

The following macros are supported for use in the discovery rule filter and prototypes of items, triggers and graphs:

Macro	Description
{#SERVICE.NAME}	Service name.
{#SERVICE.DISPLAYNAME}	Displayed service name.
{#SERVICE.DESCRIPTION}	Service description.
{#SERVICE.STATE}	Numerical value of the service state. See the service.info item for details.
{#SERVICE.STATENAME}	Name of the service state. See the service.info item for details.
{#SERVICE.PATH}	Service path.
{#SERVICE.USER}	Service user.
{#SERVICE.STARTUP}	Numerical value of the service startup type. See the service.info item for details.
{#SERVICE.STARTUPNAME}	Name of the service startup type. See the service.info item for details.
{#SERVICE.STARTUPTRIGGER}	Numerical value to indicate if the service startup type has: 0 - no startup triggers 1 - has startup triggers This macro is supported since Zabbix 3.4.4. It is useful to discover such service startup types as *Automatic (trigger start)*, *Automatic delayed (trigger start)* and *Manual (trigger start).*

Based on Windows service discovery you may create an item prototype like

`service.info[{#SERVICE.NAME},<param>]`

where `param` accepts the following values: *state*, *displayname*, *path*, *user*, *startup* or *description*.

For example, to acquire the display name of a service you may use a "service.info[{#SERVICE.NAME},displayname]" item. If `param` value is not specified ("service.info[{#SERVICE.NAME}]"), the default *state* parameter is used.

10 Discovery of Windows performance counter instances

Overview

It is possible to discover object instances of Windows performance counters. This is useful for multi-instance performance counters.

Item key

To configure the discovery rule, use the following item:

- `perf_instance.discovery[object]`

Note that the object name may be localized. For example:

```
perf_instance.discovery[Processor] # The object name is in English.
perf_instance.discovery[Processador] # The object name is in Portuguese.
```

Alternatively, to ensure that the object name is provided in English, independent of OS localization, use the following item:

- `perf_instance_en.discovery[object]`

For example:

```
perf_instance_en.discovery[Processor]
perf_instance_en.discovery[Memory]
```

These items are supported since Zabbix Windows agent 5.0.1.

Supported macros

The discovery process will return all instances of the specified object in the {#INSTANCE} macro:

```
[
    {"{#INSTANCE}":"0"},
    {"{#INSTANCE}":"1"},
    {"{#INSTANCE}":"_Total"}
]
```

This macro may be used in the prototypes of perf_counter[] and perf_counter_en[] items.

For example, if the item key used in the discovery rule is perf_instance.discovery[Processor], you may create the following item prototype:

```
perf_counter["\Processor({#INSTANCE})\% Processor Time"]
```

Note:

- If the specified object is not found or does not support variable instances, the discovery item will become NOTSUPPORTED.
- If the specified object supports variable instances but currently does not have any instances, an empty JSON array will be returned.
- Duplicate instances will be skipped.

11 Discovery using WMI queries

Overview

WMI is a powerful interface in Windows that can be used for retrieving various information about Windows components, services, state and software installed.

It can be used for physical disk discovery and their performance data collection, network interface discovery, Hyper-V guest discovery, monitoring Windows services and many other things in Windows OS.

This type of low-level discovery is done using WQL queries whose results get automatically transformed into a JSON object suitable for low-level discovery.

Item key

The item to use in the discovery rule is

wmi.getall[<namespace>,<query>]

This item transforms the query result into a JSON array. For example:

```
select * from Win32_DiskDrive where Name like '%PHYSICALDRIVE%'
```

may return something like this:

```
[
    {
        "DeviceID" : "\\.\PHYSICALDRIVE0",
        "BytesPerSector" : 512,
        "Capabilities" : [
            3,
            4
        ],
        "CapabilityDescriptions" : [
            "Random Access",
            "Supports Writing"
        ],
        "Caption" : "VBOX HARDDISK ATA Device",
        "ConfigManagerErrorCode" : "0",
        "ConfigManagerUserConfig" : "false",
        "CreationClassName" : "Win32_DiskDrive",
        "Description" : "Disk drive",
        "FirmwareRevision" : "1.0",
        "Index" : 0,
        "InterfaceType" : "IDE"
```

```
    },
    {
        "DeviceID" : "\\.\PHYSICALDRIVE1",
        "BytesPerSector" : 512,
        "Capabilities" : [
            3,
            4
        ],
        "CapabilityDescriptions" : [
            "Random Access",
            "Supports Writing"
        ],
        "Caption" : "VBOX HARDDISK ATA Device",
        "ConfigManagerErrorCode" : "0",
        "ConfigManagerUserConfig" : "false",
        "CreationClassName" : "Win32_DiskDrive",
        "Description" : "Disk drive",
        "FirmwareRevision" : "1.0",
        "Index" : 1,
        "InterfaceType" : "IDE"
    }
]
```

This item is supported since Zabbix Windows agent 4.4.

Low-level discovery macros

Even though no low-level discovery macros are created in the returned JSON, these macros can be defined by the user as an additional step, using the custom LLD macro functionality with JSONPath pointing to the discovered values in the returned JSON.

The macros then can be used to create item, trigger, etc prototypes.

12 Discovery using ODBC SQL queries

Overview

This type of low-level discovery is done using SQL queries, whose results get automatically transformed into a JSON object suitable for low-level discovery.

Item key

SQL queries are performed using a "Database monitor" item type. Therefore, most of the instructions on ODBC monitoring page apply in order to get a working "Database monitor" discovery rule.

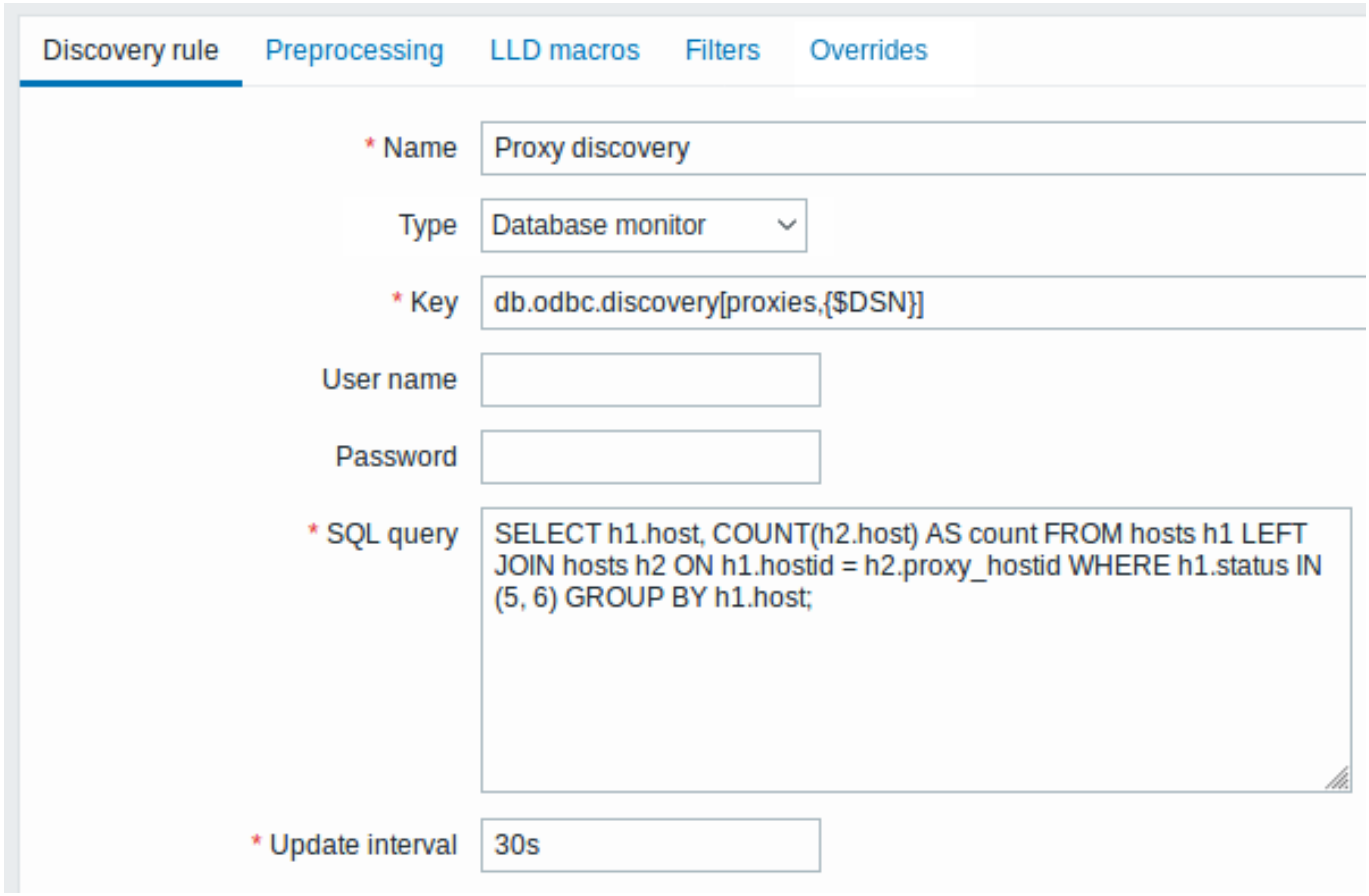
Two item keys may be used in "Database monitor" discovery rules:

- **db.odbc.discovery**[<unique short description>,<dsn>,<connection string>] - this item transforms the SQL query result into a JSON array, turning the column names from the query result into low-level discovery macro names paired with the discovered field values. These macros can be used in creating item, trigger, etc prototypes. See also: Using db.odbc.discovery.
- **db.odbc.get**[<unique short description>,<dsn>,<connection string>] - this item transforms the SQL query result into a JSON array, keeping the original column names from the query result as a field name in JSON paired with the discovered values. Compared to `db.odbc.discovery[]`, this item does not create low-level discovery macros in the returned JSON, therefore there is no need to check if the column names can be valid macro names. The low-level discovery macros can be defined as an additional step as required, using the custom LLD macro functionality with JSONPath pointing to the discovered values in the returned JSON. See also: Using db.odbc.get.

Using db.odbc.discovery

As a practical example to illustrate how the SQL query is transformed into JSON, let us consider low-level discovery of Zabbix proxies by performing an ODBC query on Zabbix database. This is useful for automatic creation of "zabbix[proxy,<name>,lastaccess]" internal items to monitor which proxies are alive.

Let us start with discovery rule configuration:



All mandatory input fields are marked with a red asterisk.

Here, the following direct query on Zabbix database is used to select all Zabbix proxies, together with the number of hosts they are monitoring. The number of hosts can be used, for instance, to filter out empty proxies:

```
mysql> SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_ho
+---------+-------+
| host    | count |
+---------+-------+
| Japan 1 |     5 |
| Japan 2 |    12 |
| Latvia  |     3 |
+---------+-------+
3 rows in set (0.01 sec)
```

By the internal workings of "db.odbc.discovery[,{$DSN}]" item, the result of this query gets automatically transformed into the following JSON:

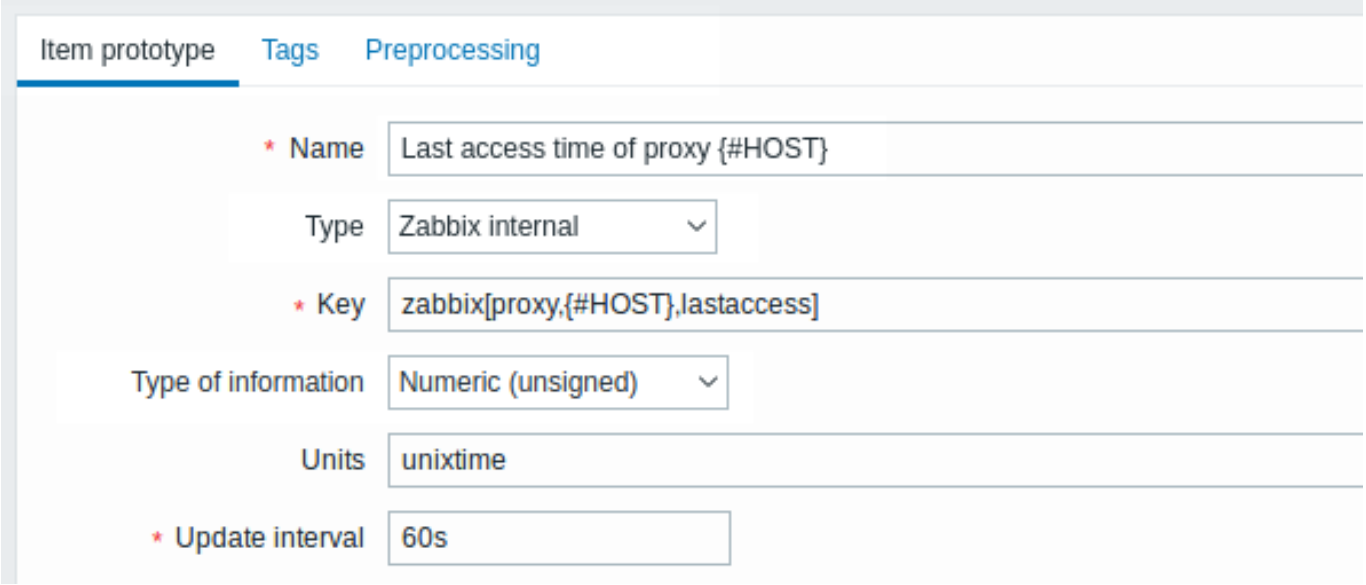
```
[
    {
        "{#HOST}": "Japan 1",
        "{#COUNT}": "5"
    },
    {
        "{#HOST}": "Japan 2",
        "{#COUNT}": "12"
    },
    {
        "{#HOST}": "Latvia",
        "{#COUNT}": "3"
    }
]
```

It can be seen that column names become macro names and selected rows become the values of these macros.

Note:
If it is not obvious how a column name would be transformed into a macro name, it is suggested to use column aliases like "COUNT(h2.host) AS count" in the example above.
In case a column name cannot be converted into a valid macro name, the discovery rule becomes not supported, with the error message detailing the offending column number. If additional help is desired, the obtained column names are provided under DebugLevel=4 in Zabbix server log file:

```
$ grep db.odbc.discovery /tmp/zabbix_server.log
 ...
 23876:20150114:153410.856 In db_odbc_discovery() query:'SELECT h1.host, COUNT(h2.host) FROM hosts h1 L
 23876:20150114:153410.860 db_odbc_discovery() column[1]:'host'
 23876:20150114:153410.860 db_odbc_discovery() column[2]:'COUNT(h2.host)'
 23876:20150114:153410.860 End of db_odbc_discovery():NOTSUPPORTED
 23876:20150114:153410.860 Item [Zabbix server:db.odbc.discovery[proxies,{$DSN}]] error: Cannot convert
```

Now that we understand how a SQL query is transformed into a JSON object, we can use {#HOST} macro in item prototypes:



Once discovery is performed, an item will be created for each proxy:



Using db.odbc.get

Using `db.odbc.get[,{$DSN}]` and the following SQL example:

```
mysql> SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_ho
+---------+-------+
| host    | count |
+---------+-------+
| Japan 1 |     5 |
| Japan 2 |    12 |
| Latvia  |     3 |
+---------+-------+
3 rows in set (0.01 sec)
```

this JSON will be returned:

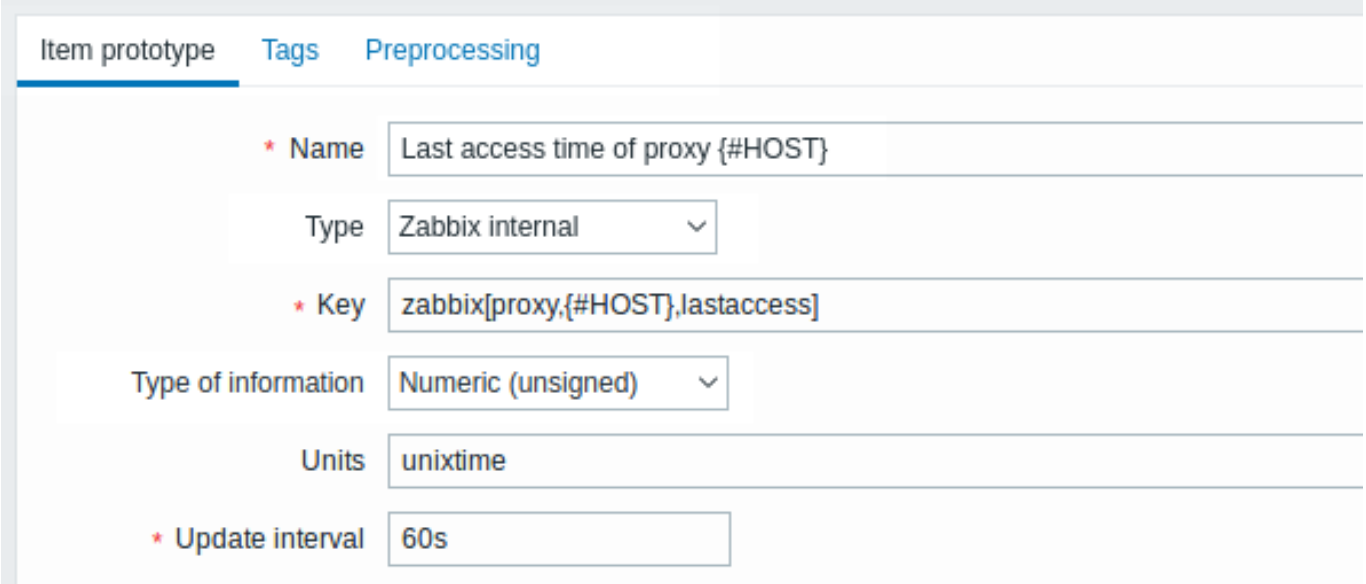
```
[
    {
```

```
        "host": "Japan 1",
        "count": "5"
    },
    {
        "host": "Japan 2",
        "count": "12"
    },
    {
        "host": "Latvia",
        "count": "3"
    }
]
```

As you can see, there are no low-level discovery macros there. However, custom low-level discovery macros can be created in the LLD macros tab of a discovery rule using JSONPath, for example:

`{#HOST}` → `$.host`

Now this `{#HOST}` macro may be used in item prototypes:



13 Discovery using Prometheus data

Overview

Data provided in Prometheus line format can be used for low-level discovery.

See Prometheus checks for details how Prometheus data querying is implemented in Zabbix.

Configuration

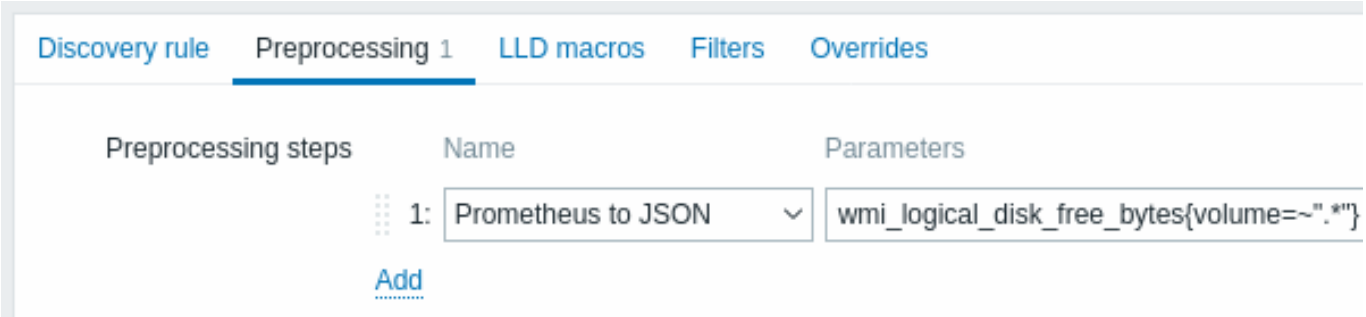
The low-level discovery rule should be created as a dependent item to the HTTP master item that collects Prometheus data.

Prometheus to JSON

In the discovery rule, go to the Preprocessing tab and select the *Prometheus to JSON* preprocessing option. Data in JSON format are needed for discovery and the *Prometheus to JSON* preprocessing option will return exactly that, with the following attributes:

- metric name
- metric value
- help (if present)
- type (if present)
- labels (if present)
- raw line

For example, querying `wmi_logical_disk_free_bytes`:



from these Prometheus lines:

```
# HELP wmi_logical_disk_free_bytes Free space in bytes (LogicalDisk.PercentFreeSpace)
# TYPE wmi_logical_disk_free_bytes gauge
wmi_logical_disk_free_bytes{volume="C:"} 3.5180249088e+11
wmi_logical_disk_free_bytes{volume="D:"} 2.627731456e+09
wmi_logical_disk_free_bytes{volume="HarddiskVolume4"} 4.59276288e+08
```

will return:

```
[
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "C:"
         },
        "value": "3.5180249088e+11",
        "line_raw": "wmi_logical_disk_free_bytes{volume=\"C:\"} 3.5180249088e+11"
    },
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "D:"
         },
        "value": "2.627731456e+09",
        "line_raw": "wmi_logical_disk_free_bytes{volume=\"D:\"} 2.627731456e+09"
    },
    {
        "name": "wmi_logical_disk_free_bytes",
        "help": "Free space in bytes (LogicalDisk.PercentFreeSpace)",
        "type": "gauge",
        "labels": {
            "volume": "HarddiskVolume4"
         },
        "value": "4.59276288e+08",
        "line_raw": "wmi_logical_disk_free_bytes{volume=\"HarddiskVolume4\"} 4.59276288e+08"
    }
]
```

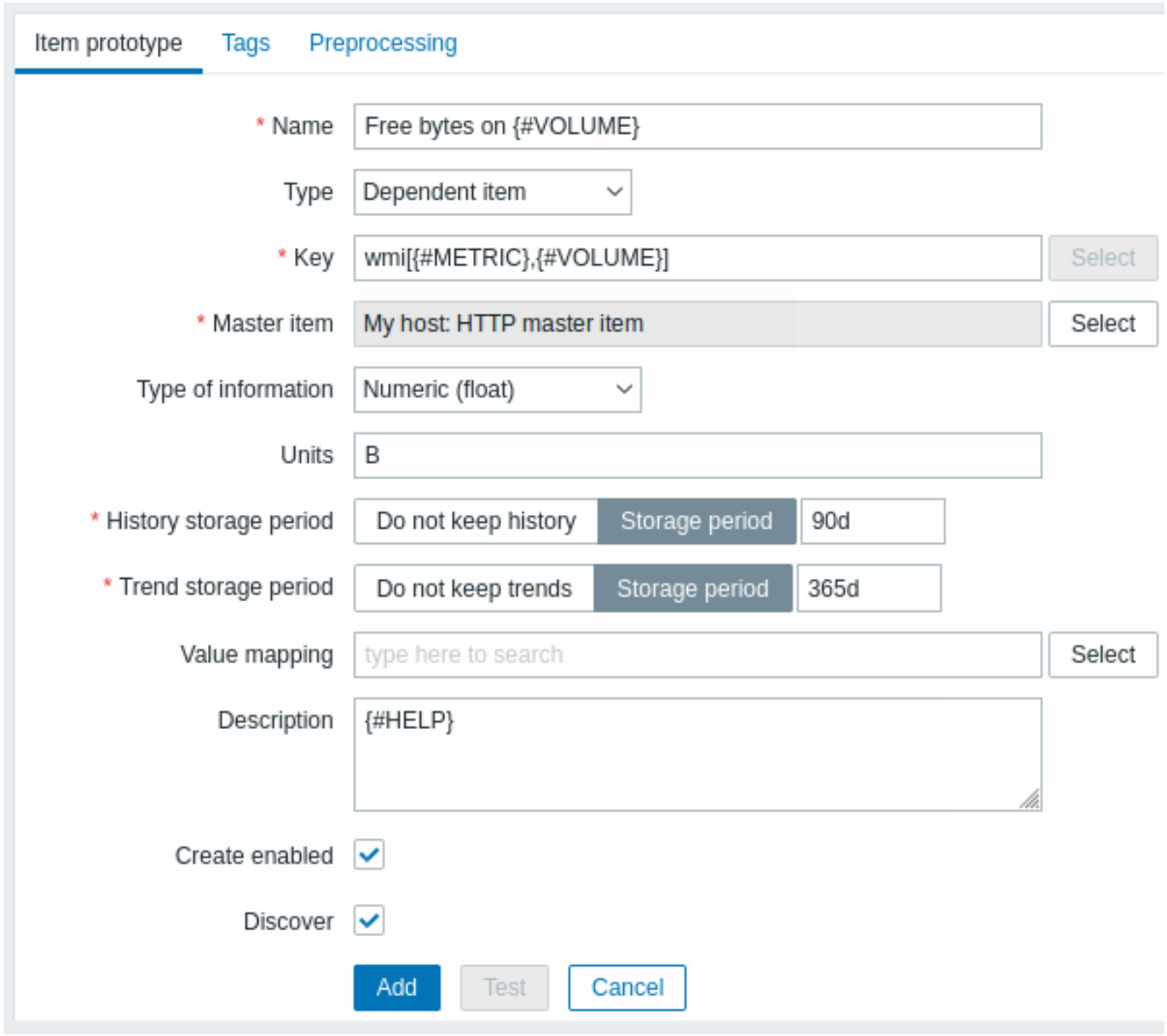
Mapping LLD macros

Next you have to go to the LLD macros tab and make the following mappings:

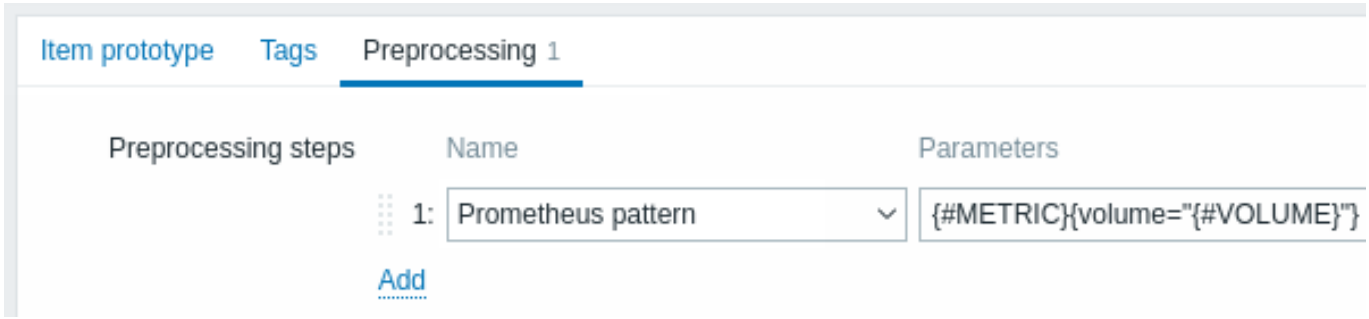
```
{#VOLUME}=$.labels['volume']
{#METRIC}=$['name']
{#HELP}=$['help']
```

Item prototype

You may want to create an item prototype like this:



with preprocessing options:



14 Discovery of block devices

In a similar way as file systems are discovered, it is possible to also discover block devices and their type.

Item key

The item key to use in the discovery rule is

`vfs.dev.discovery`

This item is supported on Linux platforms only, since Zabbix agent 4.4.

You may create discovery rules using this discovery item and:

- filter: **{#DEVNAME} matches** `sd[\D]$` - to discover devices named "sd0", "sd1", "sd2", ...
- filter: **{#DEVTYPE} matches** `disk` **AND {#DEVNAME} does not match** `^loop.*` - to discover disk type devices whose name does not start with "loop"

Supported macros

This discovery key returns two macros - {#DEVNAME} and {#DEVTYPE} identifying the block device name and type respectively, e.g.:

```
[
   {
      "{#DEVNAME}":"loop1",
      "{#DEVTYPE}":"disk"
   },
   {
      "{#DEVNAME}":"dm-0",
      "{#DEVTYPE}":"disk"
   },
   {
      "{#DEVNAME}":"sda",
      "{#DEVTYPE}":"disk"
   },
   {
      "{#DEVNAME}":"sda1",
      "{#DEVTYPE}":"partition"
   }
]
```

Block device discovery allows to use `vfs.dev.read[]` and `vfs.dev.write[]` items to create item prototypes using the {#DEVNAME} macro, for example:

- "vfs.dev.read[{#DEVNAME},sps]"
- "vfs.dev.write[{#DEVNAME},sps]"

{#DEVTYPE} is intended for device filtering.

15 Discovery of host interfaces in Zabbix

Overview

It is possible to discover all interfaces configured in Zabbix frontend for a host.

Item key

The item to use in the discovery rule is the

`zabbix[host,discovery,interfaces]`

internal item. This item is supported since Zabbix server 3.4.

This item returns a JSON with the description of interfaces, including:

- IP address/DNS hostname (depending on the "Connect to" host setting)
- Port number
- Interface type (Zabbix agent, SNMP, JMX, IPMI)
- If it is the default interface or not
- If the bulk request feature is enabled - for SNMP interfaces only.

For example:

```
[{"{#IF.CONN}":"192.168.3.1","{#IF.IP}":"192.168.3.1","{#IF.DNS}":"","{#IF.PORT}":"10050","{#IF.TYPE}":"AG
```

With multiple interfaces their records in JSON are ordered by:

- Interface type,
- Default - the default interface is put before non-default interfaces,
- Interface ID (in ascending order).

Supported macros

The following macros are supported for use in the discovery rule filter and prototypes of items, triggers and graphs:

Macro	Description
{#IF.CONN}	Interface IP address or DNS host name.
{#IF.IP}	Interface IP address.
{#IF.DNS}	Interface DNS host name.
{#IF.PORT}	Interface port number.
{#IF.TYPE}	Interface type (”AGENT”, ”SNMP”, ”JMX”, or ”IPMI”).
{#IF.DEFAULT}	Default status for the interface: 0 - not default interface 1 - default interface
{#IF.SNMP.BULK}	SNMP bulk processing status for the interface: 0 - disabled 1 - enabled This macro is returned only if interface type is “SNMP”.

16 Distributed monitoring

Overview Zabbix provides an effective and reliable way of monitoring a distributed IT infrastructure using Zabbix proxies.

Proxies can be used to collect data locally on behalf of a centralized Zabbix server and then report the data to the server.

Proxy features

When making a choice of using/not using a proxy, several considerations must be taken into account.

	Proxy
Lightweight	**Yes**
GUI	No
Works independently	**Yes**
Easy maintenance	**Yes**
Automatic DB creation	**Yes**[1]
Local administration	No
Ready for embedded hardware	**Yes**
One way TCP connections	**Yes**
Centralized configuration	**Yes**
Generates notifications	No

[1] Automatic DB creation feature works only with SQLite. Other databases require manual setup.

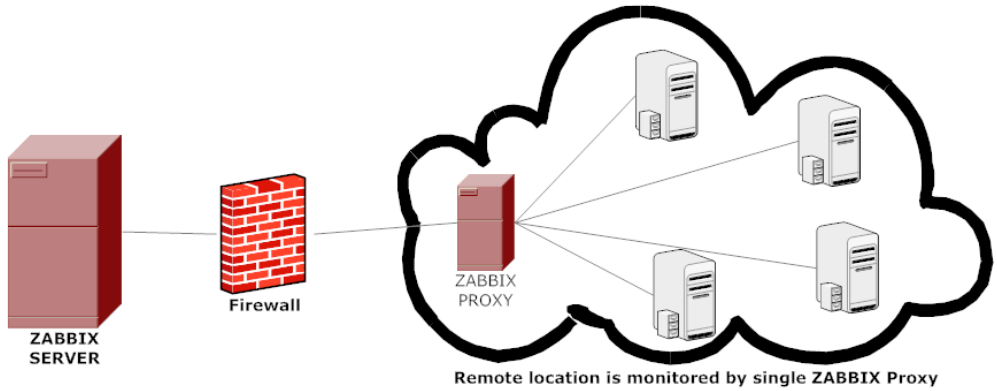
1 Proxies

Overview A Zabbix proxy can collect performance and availability data on behalf of the Zabbix server. This way, a proxy can take on itself some of the load of collecting data and offload the Zabbix server.

Also, using a proxy is the easiest way of implementing centralized and distributed monitoring, when all agents and proxies report to one Zabbix server and all data is collected centrally.

A Zabbix proxy can be used to:

- Monitor remote locations
- Monitor locations having unreliable communications
- Offload the Zabbix server when monitoring thousands of devices
- Simplify the maintenance of distributed monitoring



The proxy requires only one TCP connection to the Zabbix server. This way it is easier to get around a firewall as you only need to configure one firewall rule.

Attention:
Zabbix proxy must use a separate database. Pointing it to the Zabbix server database will break the configuration.

All data collected by the proxy is stored locally before transmitting it over to the server. This way no data is lost due to any temporary communication problems with the server. The *ProxyLocalBuffer* and *ProxyOfflineBuffer* parameters in the proxy configuration file control for how long the data are kept locally.

Attention:
It may happen that a proxy, which receives the latest configuration changes directly from Zabbix server database, has a more up-to-date configuration than Zabbix server whose configuration may not be updated as fast due to the value of CacheUpdateFrequency. As a result, proxy may start gathering data and send them to Zabbix server that ignores these data.

Zabbix proxy is a data collector. It does not calculate triggers, process events or send alerts. For an overview of what proxy functionality is, review the following table:

Function		Supported by proxy
Items		
	Zabbix agent checks	**Yes**
	Zabbix agent checks (active)	**Yes** [1]
	Simple checks	**Yes**
	Trapper items	**Yes**
	SNMP checks	**Yes**
	SNMP traps	**Yes**
	IPMI checks	**Yes**
	JMX checks	**Yes**
	Log file monitoring	**Yes**
	Internal checks	**Yes**
	SSH checks	**Yes**
	Telnet checks	**Yes**
	External checks	**Yes**
	Dependent items	**Yes**
	Script items	**Yes**
Built-in web monitoring		**Yes**
Item value preprocessing		**Yes**
Network discovery		**Yes**
Active agent autoregistration		**Yes**
Low-level discovery		**Yes**
Remote commands		**Yes**
Calculating triggers		*No*
Processing events		*No*
Event correlation		*No*
Sending alerts		*No*

Note:
[1] To make sure that an agent asks the proxy (and not the server) for active checks, the proxy must be listed in the **ServerActive** parameter in the agent configuration file.

Protection from overloading

If Zabbix server was down for some time, and proxies have collected a lot of data, and then the server starts, it may get overloaded (history cache usage stays at 95-100% for some time). This overload could result in a performance hit, where checks are processed slower than they should. Protection from this scenario was implemented to avoid problems that arise due to overloading history cache.

When Zabbix server history cache is full the history cache write access is being throttled, stalling server data gathering processes. The most common history cache overload case is after server downtime when proxies are uploading gathered data. To avoid this proxy throttling was added (currently it cannot be disabled).

Zabbix server will stop accepting data from proxies when history cache usage reaches 80%. Instead those proxies will be put on a throttling list. This will continue until the cache usage falls down to 60%. Now server will start accepting data from proxies one by one, defined by the throttling list. This means the first proxy that attempted to upload data during the throttling period will be served first and until it's done the server will not accept data from other proxies.

This throttling mode will continue until either cache usage hits 80% again or falls down to 20% or the throttling list is empty. In the first case the server will stop accepting proxy data again. In the other two cases the server will start working normally, accepting data from all proxies.

The above information can be illustrated in the following table:

History write cache usage	Zabbix server mode	Zabbix server action
Reaches 80%	Wait	Stops accepting proxy data, but maintains a *throttling list* (prioritized list of proxies to be contacted later).
Drops to 60%	Throttled	Starts processing the throttling list, but still not accepting proxy data.
Drops to 20%	Normal	Drops the throttling list and starts accepting proxy data normally.

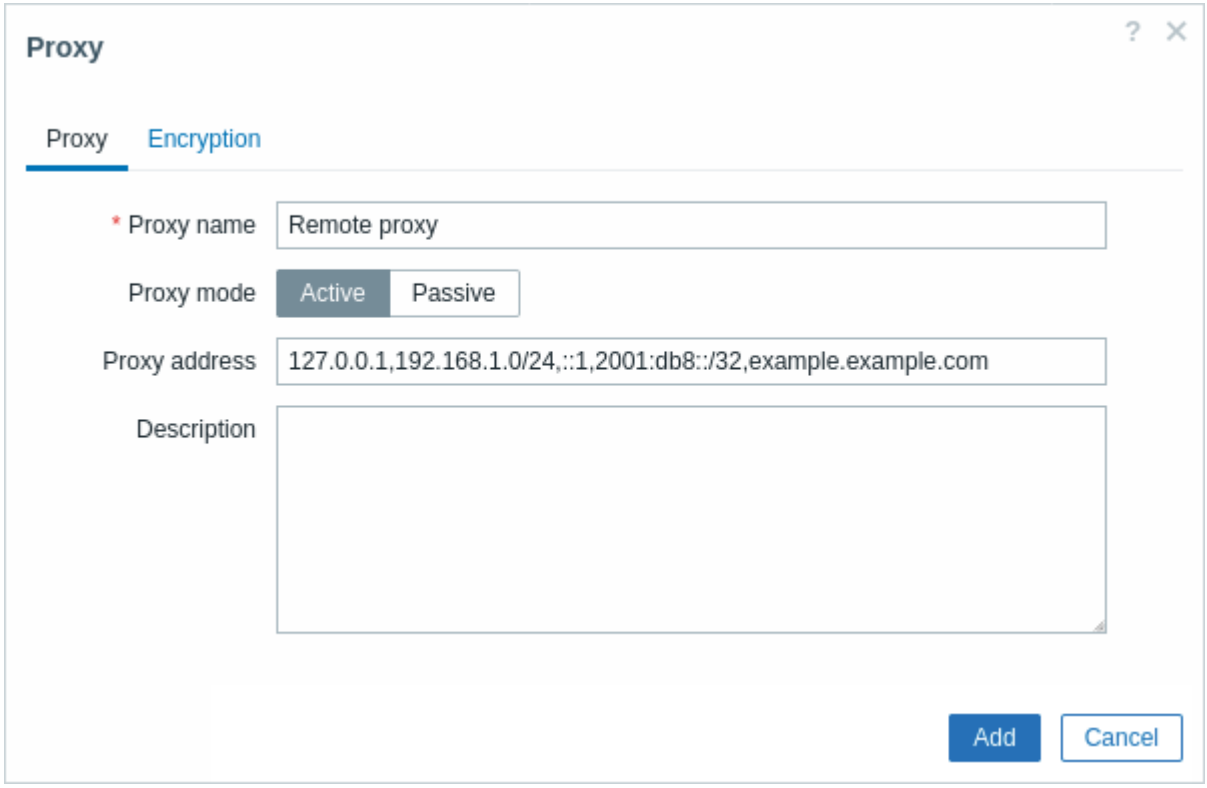
You may use the `zabbix[wcache,history,pused]` internal item to correlate this behavior of Zabbix server with a metric.

Configuration Once you have installed and configured a proxy, it is time to configure it in the Zabbix frontend.

Adding proxies

To configure a proxy in Zabbix frontend:

- Go to: *Administration* → *Proxies*
- Click on *Create proxy*



Parameter	Description
Proxy name	Enter the proxy name. It must be the same name as in the *Hostname* parameter in the proxy configuration file.
Proxy mode	Select the proxy mode. **Active** - the proxy will connect to the Zabbix server and request configuration data **Passive** - Zabbix server connects to the proxy *Note* that without encrypted communications (sensitive) proxy configuration data may become available to parties having access to the Zabbix server trapper port when using an active proxy. This is possible because anyone may pretend to be an active proxy and request configuration data if authentication does not take place or proxy addresses are not limited in the *Proxy address* field.
Proxy address	If specified then active proxy requests are only accepted from this list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of active Zabbix proxy. This field is only available if an active proxy is selected in the *Proxy mode* field. Macros are not supported. This option is supported since Zabbix 4.0.0.
Interface	Enter interface details for the passive proxy. This field is only available if a passive proxy is selected in the *Proxy mode* field.
IP address	IP address of the passive proxy (optional).
DNS name	DNS name of the passive proxy (optional).
Connect to	Clicking the respective button will tell Zabbix server what to use to retrieve data from proxy: **IP** - Connect to the proxy IP address (recommended) **DNS** - Connect to the proxy DNS name
Port	TCP port number of the passive proxy (10051 by default).
Description	Enter the proxy description.

The **Encryption** tab allows you to require encrypted connections with the proxy.

Parameter	Description
Connections to proxy	How the server connects to the passive proxy: no encryption (default), using PSK (pre-shared key) or certificate.
Connections from proxy	Select what type of connections are allowed from the active proxy. Several connection types can be selected at the same time (useful for testing and switching to other connection type). Default is ”No encryption”.

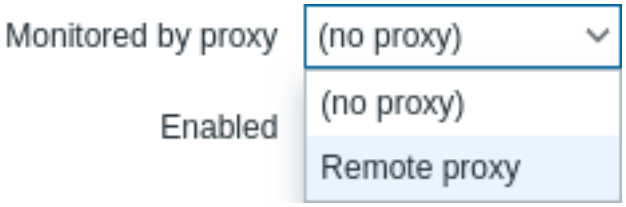
Parameter	Description
Issuer	Allowed issuer of certificate. Certificate is first validated with CA (certificate authority). If it is valid, signed by the CA, then the *Issuer* field can be used to further restrict allowed CA. This field is optional, intended to use if your Zabbix installation uses certificates from multiple CAs.
Subject	Allowed subject of certificate. Certificate is first validated with CA. If it is valid, signed by the CA, then the *Subject* field can be used to allow only one value of *Subject* string. If this field is empty then any valid certificate signed by the configured CA is accepted.
PSK identity	Pre-shared key identity string. Do not put sensitive information in the PSK identity, it is transmitted unencrypted over the network to inform a receiver which PSK to use.
PSK	Pre-shared key (hex-string). Maximum length: 512 hex-digits (256-byte PSK) if Zabbix uses GnuTLS or OpenSSL library, 64 hex-digits (32-byte PSK) if Zabbix uses mbed TLS (PolarSSL) library. Example: 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952

The editing form of an existing proxy has the following additional buttons:

- *Refresh configuration* - refresh configuration of the proxy
- *Clone* - create a new proxy based on the properties of the existing proxy
- *Delete* - delete the proxy

Host configuration

You can specify that an individual host should be monitored by a proxy in the host configuration form, using the *Monitored by proxy* field.



Host mass update is another way of specifying that hosts should be monitored by a proxy.

1 Synchronization of monitoring configuration

Overview

This page provides details on the monitoring configuration update for the proxy, i.e. how changes made to the monitoring configuration on the server are synchronized to the proxy.

Incremental update

The proxy configuration update is incremental. During a configuration sync only the modified entities are updated (thus, if no entities have been modified, nothing will be sent). This approach allows to save resources and set a smaller interval (almost instant) for the proxy configuration update.

Proxy configuration changes are tracked using revision numbers. Only entities with revisions larger than the proxy configuration revision are included in configuration data sent to the proxy.

The entities for a configuration sync are as follows:

Entity	Details
autoregistration tls data	All autoregistration TLS data.
expressions	All expressions (regular expressions, expression tables).
global configuration	Global configuration defined in the 'config' table
host	All properties, interfaces, inventory, items, item preprocessing, item parameters, web scenarios of a host.
host macros	All macros defined on a host and all template IDs linked to it.
proxy discovery rule	Discovery rules and checks assigned to a proxy.

That means:

- If an item is changed on a **host**, all configuration of that host will be synced.
- If a **regular expression** is changed, all regular expressions will be synced.

An exception are the host macros which are sent also if anything on the host has been changed.

The `-R config_cache_reload` command on the proxy will also initiate an incremental update.

Note that a full configuration sync will take place on a proxy start/restart, HA failover, if the session token has changed, or if the configuration update failed on the proxy, for example, if the connection was broken while receiving configuration data.

Configuration parameters

The **ProxyConfigFrequency** parameter determines how often the proxy configuration is synced with the server (10 seconds by default).

Note that ProxyConfigFrequency is:

- server parameter for passive proxies
- proxy parameter for active proxies

On active proxies ProxyConfigFrequency is a new parameter since Zabbix 6.4 and must be used instead of the now-deprecated ConfigFrequency.

> **Attention:**
> If both ProxyConfigFrequency and ConfigFrequency are used, the proxy will log an error and terminate.

17 Encryption

Overview Zabbix supports encrypted communications between Zabbix components using Transport Layer Security (TLS) protocol v.1.2 and 1.3 (depending on the crypto library). Certificate-based and pre-shared key-based encryption is supported.

Encryption can be configured for connections:

- Between Zabbix server, Zabbix proxy, Zabbix agent, Zabbix web service, zabbix_sender and zabbix_get utilities
- To Zabbix database from Zabbix frontend and server/proxy

Encryption is optional and configurable for individual components:

- Some proxies and agents can be configured to use certificate-based encryption with the server, while others can use pre-shared key-based encryption, and yet others continue with unencrypted communications (as before)
- Server (proxy) can use different encryption configurations for different hosts

Zabbix daemon programs use one listening port for encrypted and unencrypted incoming connections. Adding an encryption does not require opening new ports on firewalls.

Limitations

- Private keys are stored in plain text in files readable by Zabbix components during startup
- Pre-shared keys are entered in Zabbix frontend and stored in Zabbix database in plain text
- Built-in encryption does not protect communications:
  - Between the web server running Zabbix frontend and user web browser
  - Between Zabbix frontend and Zabbix server
- Currently each encrypted connection opens with a full TLS handshake, no session caching and tickets are implemented
- Adding encryption increases the time for item checks and actions, depending on network latency:
  - For example, if packet delay is 100ms then opening a TCP connection and sending unencrypted request takes around 200ms. With encryption about 1000 ms are added for establishing the TLS connection;
  - Timeouts may need to be increased, otherwise some items and actions running remote scripts on agents may work with unencrypted connections, but fail with timeout with encrypted.
- Encryption is not supported by network discovery. Zabbix agent checks performed by network discovery will be unencrypted and if Zabbix agent is configured to reject unencrypted connections such checks will not succeed.

Compiling Zabbix with encryption support To support encryption Zabbix must be compiled and linked with one of the supported crypto libraries:

- GnuTLS - from version 3.1.18
- OpenSSL - versions 1.0.1, 1.0.2, 1.1.0, 1.1.1, 3.0.x
- LibreSSL - tested with versions 2.7.4, 2.8.2:
  - LibreSSL 2.6.x is not supported

- LibreSSL is supported as a compatible replacement of OpenSSL; the new `tls_*()` LibreSSL-specific API functions are not used. Zabbix components compiled with LibreSSL will not be able to use PSK, only certificates can be used.

Note:
You can find out more about setting up SSL for Zabbix frontend by referring to these best practices.

The library is selected by specifying the respective option to "configure" script:

- `--with-gnutls[=DIR]`
- `--with-openssl[=DIR]` (also used for LibreSSL)

For example, to configure the sources for server and agent with *OpenSSL* you may use something like:

```
./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --wit
```

Different Zabbix components may be compiled with different crypto libraries (e.g. a server with *OpenSSL*, an agent with *GnuTLS*).

Attention:
If you plan to use pre-shared keys (PSK), consider using *GnuTLS* or *OpenSSL 1.1.0* (or newer) libraries in Zabbix components using PSKs. *GnuTLS* and *OpenSSL 1.1.0* libraries support PSK ciphersuites with Perfect Forward Secrecy. Older versions of the *OpenSSL* library (1.0.1, 1.0.2c) also support PSKs, but available PSK ciphersuites do not provide Perfect Forward Secrecy.

Connection encryption management Connections in Zabbix can use:

- no encryption (default)
- RSA certificate-based encryption
- PSK-based encryption

There are two important parameters used to specify encryption between Zabbix components:

- TLSConnect - specifies what encryption to use for outgoing connections (unencrypted, PSK or certificate)
- TLSAccept - specifies what types of connections are allowed for incoming connections (unencrypted, PSK or certificate). One or more values can be specified.

`TLSConnect` is used in the configuration files for Zabbix proxy (in active mode, specifies only connections to server) and Zabbix agent (for active checks). In Zabbix frontend the TLSConnect equivalent is the *Connections to host* field in *Data collection → Hosts → <some host> → Encryption* tab and the *Connections to proxy* field in *Administration → Proxies → <some proxy> → Encryption* tab. If the configured encryption type for connection fails, no other encryption types will be tried.

`TLSAccept` is used in the configuration files for Zabbix proxy (in passive mode, specifies only connections from server) and Zabbix agent (for passive checks). In Zabbix frontend the TLSAccept equivalent is the *Connections from host* field in *Data collection → Hosts → <some host> → Encryption* tab and the *Connections from proxy* field in *Administration → Proxies → <some proxy> → Encryption* tab.

Normally you configure only one type of encryption for incoming encryptions. But you may want to switch the encryption type, e.g. from unencrypted to certificate-based with minimum downtime and rollback possibility. To achieve this:

- Set `TLSAccept=unencrypted,cert` in the agent configuration file and restart Zabbix agent
- Test connection with zabbix_get to the agent using certificate. If it works, you can reconfigure encryption for that agent in Zabbix frontend in the *Data collection → Hosts → <some host> → Encryption* tab by setting *Connections to host* to "Certificate".
- When server configuration cache gets updated (and proxy configuration is updated if the host is monitored by proxy) then connections to that agent will be encrypted
- If everything works as expected you can set `TLSAccept=cert` in the agent configuration file and restart Zabbix agent. Now the agent will be accepting only encrypted certificate-based connections. Unencrypted and PSK-based connections will be rejected.

In a similar way it works on server and proxy. If in Zabbix frontend in host configuration *Connections from host* is set to "Certificate" then only certificate-based encrypted connections will be accepted from the agent (active checks) and zabbix_sender (trapper items).

Most likely you will configure incoming and outgoing connections to use the same encryption type or no encryption at all. But technically it is possible to configure it asymmetrically, e.g. certificate-based encryption for incoming and PSK-based for outgoing connections.

Encryption configuration for each host is displayed in the Zabbix frontend, in *Data collection → Hosts* in the *Agent encryption* column. For example:

Example	Connections to host	Allowed connections from host	Rejected connections from host
NONE	Unencrypted	Unencrypted	Encrypted, certificate and PSK-based encrypted
CERT NONE PSK CERT	Encrypted, certificate-based	Encrypted, certificate-based	Unencrypted and PSK-based encrypted
PSK NONE PSK CERT	Encrypted, PSK-based	Encrypted, PSK-based	Unencrypted and certificate-based encrypted
PSK NONE PSK CERT	Encrypted, PSK-based	Unencrypted and PSK-based encrypted	Certificate-based encrypted
CERT NONE PSK CERT	Encrypted, certificate-based	Unencrypted, PSK or certificate-based encrypted	-

Attention:
Connections are unencrypted by default. Encryption must be configured for each host and proxy individually.

zabbix_get and zabbix_sender with encryption See zabbix_get and zabbix_sender manpages for using them with encryption.

Ciphersuites Ciphersuites by default are configured internally during Zabbix startup and, before Zabbix 4.0.19, 4.4.7, are not user-configurable.

Since Zabbix 4.0.19, 4.4.7 also user-configured ciphersuites are supported for GnuTLS and OpenSSL. Users may configure ciphersuites according to their security policies. Using this feature is optional (built-in default ciphersuites still work).

For crypto libraries compiled with default settings Zabbix built-in rules typically result in the following ciphersuites (in order from higher to lower priority):

Library	Certificate ciphersuites	PSK ciphersuites
GnuTLS 3.1.18	TLS_ECDHE_RSA_AES_128_GCM_SHA256 TLS_ECDHE_RSA_AES_128_CBC_SHA256 TLS_ECDHE_RSA_AES_128_CBC_SHA1 TLS_RSA_AES_128_GCM_SHA256 TLS_RSA_AES_128_CBC_SHA256 TLS_RSA_AES_128_CBC_SHA1	TLS_ECDHE_PSK_AES_128_CBC_SHA256 TLS_ECDHE_PSK_AES_128_CBC_SHA1 TLS_PSK_AES_128_GCM_SHA256 TLS_PSK_AES_128_CBC_SHA256 TLS_PSK_AES_128_CBC_SHA1
OpenSSL 1.0.2c	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-SHA256 AES128-SHA	PSK-AES128-CBC-SHA
OpenSSL 1.1.0	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-CCM8 AES128-CCM AES128-SHA256 AES128-SHA	ECDHE-PSK-AES128-CBC-SHA256 ECDHE-PSK-AES128-CBC-SHA PSK-AES128-GCM-SHA256 PSK-AES128-CCM8 PSK-AES128-CCM PSK-AES128-CBC-SHA256 PSK-AES128-CBC-SHA
OpenSSL 1.1.1d	TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-CCM8 AES128-CCM AES128-SHA256 AES128-SHA	TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-PSK-AES128-CBC-SHA256 ECDHE-PSK-AES128-CBC-SHA PSK-AES128-GCM-SHA256 PSK-AES128-CCM8 PSK-AES128-CCM PSK-AES128-CBC-SHA256 PSK-AES128-CBC-SHA

User-configured ciphersuites The built-in ciphersuite selection criteria can be overridden with user-configured ciphersuites.

> **Attention:**
> User-configured ciphersuites is a feature intended for advanced users who understand TLS ciphersuites, their security and consequences of mistakes, and who are comfortable with TLS troubleshooting.

The built-in ciphersuite selection criteria can be overridden using the following parameters:

Override scope	Parameter	Value	Description
Ciphersuite selection for certificates	TLSCipherCert13	Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).	Certificate-based ciphersuite selection criteria for TLS 1.3 Only OpenSSL 1.1.1 or newer.
	TLSCipherCert	Valid OpenSSL cipher strings for TLS 1.2 or valid GnuTLS priority strings. Their values are passed to the SSL_CTX_set_cipher_list() or gnutls_priority_init() functions, respectively.	Certificate-based ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)
Ciphersuite selection for PSK	TLSCipherPSK13	Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).	PSK-based ciphersuite selection criteria for TLS 1.3 Only OpenSSL 1.1.1 or newer.
	TLSCipherPSK	Valid OpenSSL cipher strings for TLS 1.2 or valid GnuTLS priority strings. Their values are passed to the SSL_CTX_set_cipher_list() or gnutls_priority_init() functions, respectively.	PSK-based ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)
Combined ciphersuite list for certificate and PSK	TLSCipherAll13	Valid OpenSSL 1.1.1 cipher strings for TLS 1.3 protocol (their values are passed to the OpenSSL function SSL_CTX_set_ciphersuites()).	Ciphersuite selection criteria for TLS 1.3 Only OpenSSL 1.1.1 or newer.
	TLSCipherAll	Valid OpenSSL cipher strings for TLS 1.2 or valid GnuTLS priority strings. Their values are passed to the SSL_CTX_set_cipher_list() or gnutls_priority_init() functions, respectively.	Ciphersuite selection criteria for TLS 1.2/1.3 (GnuTLS), TLS 1.2 (OpenSSL)

To override the ciphersuite selection in zabbix_get and zabbix_sender utilities - use the command-line parameters:

- `--tls-cipher13`
- `--tls-cipher`

The new parameters are optional. If a parameter is not specified, the internal default value is used. If a parameter is defined it cannot be empty.

If the setting of a TLSCipher* value in the crypto library fails then the server, proxy or agent will not start and an error is logged.

It is important to understand when each parameter is applicable.

Outgoing connections

The simplest case is outgoing connections:

- For outgoing connections with certificate - use TLSCipherCert13 or TLSCipherCert
- For outgoing connections with PSK - use TLSCipherPSK13 or TLSCipherPSK
- In case of zabbix_get and zabbix_sender utilities the command-line parameters `--tls-cipher13` or `--tls-cipher` can be used (encryption is unambiguously specified with a `--tls-connect` parameter)

Incoming connections

It is a bit more complicated with incoming connections because rules are specific for components and configuration.

For Zabbix **agent**:

Agent connection setup	Cipher configuration
TLSConnect=cert	TLSCipherCert, TLSCipherCert13
TLSConnect=psk	TLSCipherPSK, TLSCipherPSK13
TLSAccept=cert	TLSCipherCert, TLSCipherCert13
TLSAccept=psk	TLSCipherPSK, TLSCipherPSK13
TLSAccept=cert,psk	TLSCipherAll, TLSCipherAll13

For Zabbix **server** and **proxy**:

Connection setup	Cipher configuration
Outgoing connections using PSK	TLSCipherPSK, TLSCipherPSK13
Incoming connections using certificates	TLSCipherAll, TLSCipherAll13
Incoming connections using PSK if server has no certificate	TLSCipherPSK, TLSCipherPSK13
Incoming connections using PSK if server has certificate	TLSCipherAll, TLSCipherAll13

Some pattern can be seen in the two tables above:

- TLSCipherAll and TLSCipherAll13 can be specified only if a combined list of certificate- **and** PSK-based ciphersuites is used. There are two cases when it takes place: server (proxy) with a configured certificate (PSK ciphersuites are always configured on server, proxy if crypto library supports PSK), agent configured to accept both certificate- and PSK-based incoming connections
- in other cases TLSCipherCert* and/or TLSCipherPSK* are sufficient

The following tables show the `TLSCipher*` built-in default values. They could be a good starting point for your own custom values.

Parameter	GnuTLS 3.6.12
TLSCipherCert	NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509
TLSCipherPSK	NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL
TLSCipherAll	NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIGN-ALL:+CTYPE-X.509

Parameter	OpenSSL 1.1.1d [1]
TLSCipherCert13	
TLSCipherCert	EECDH+aRSA+AES128:RSA+aRSA+AES128
TLSCipherPSK13	TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
TLSCipherPSK	kECDHEPSK+AES128:kPSK+AES128
TLSCipherAll13	
TLSCipherAll	EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128

[1] Default values are different for older OpenSSL versions (1.0.1, 1.0.2, 1.1.0), for LibreSSL and if OpenSSL is compiled without PSK support.

Examples of user-configured ciphersuites

See below the following examples of user-configured ciphersuites:

- Testing cipher strings and allowing only PFS ciphersuites
- Switching from AES128 to AES256

Testing cipher strings and allowing only PFS ciphersuites

To see which ciphersuites have been selected you need to set 'DebugLevel=4' in the configuration file, or use the `-vv` option for zabbix_sender.

Some experimenting with TLSCipher* parameters might be necessary before you get the desired ciphersuites. It is inconvenient to restart Zabbix server, proxy or agent multiple times just to tweak TLSCipher* parameters. More convenient options are using zabbix_sender or the openssl command. Let's show both.

1. Using zabbix_sender.

Let's make a test configuration file, for example /home/zabbix/test.conf, with the syntax of a zabbix_agentd.conf file:

```
Hostname=nonexisting
ServerActive=nonexisting

TLSConnect=cert
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/agent.crt
TLSKeyFile=/home/zabbix/agent.key
TLSPSKIdentity=nonexisting
TLSPSKFile=/home/zabbix/agent.psk
```

You need valid CA and agent certificates and PSK for this example. Adjust certificate and PSK file paths and names for your environment.

If you are not using certificates, but only PSK, you can make a simpler test file:

```
Hostname=nonexisting
ServerActive=nonexisting

TLSConnect=psk
TLSPSKIdentity=nonexisting
TLSPSKFile=/home/zabbix/agentd.psk
```

The selected ciphersuites can be seen by running zabbix_sender (example compiled with OpenSSL 1.1.d):

```
$ zabbix_sender -vv -c /home/zabbix/test.conf -k nonexisting_item -o 1 2>&1 | grep ciphersuites
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() certificate ciphersuites: TLS_AES_256_GCM_SHA384 TLS_
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() PSK ciphersuites: TLS_CHACHA20_POLY1305_SHA256 TLS_AE
zabbix_sender [41271]: DEBUG: zbx_tls_init_child() certificate and PSK ciphersuites: TLS_AES_256_GCM_SHA
```

Here you see the ciphersuites selected by default. These default values are chosen to ensure interoperability with Zabbix agents running on systems with older OpenSSL versions (from 1.0.1).

With newer systems you can choose to tighten security by allowing only a few ciphersuites, e.g. only ciphersuites with PFS (Perfect Forward Secrecy). Let's try to allow only ciphersuites with PFS using TLSCipher* parameters.

Attention:
The result will not be interoperable with systems using OpenSSL 1.0.1 and 1.0.2, if PSK is used. Certificate-based encryption should work.

Add two lines to the test.conf configuration file:

```
TLSCipherCert=EECDH+aRSA+AES128
TLSCipherPSK=kECDHEPSK+AES128
```

and test again:

```
$ zabbix_sender -vv -c /home/zabbix/test.conf -k nonexisting_item -o 1 2>&1 | grep ciphersuites
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() certificate ciphersuites: TLS_AES_256_GCM_SHA384 TLS_
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() PSK ciphersuites: TLS_CHACHA20_POLY1305_SHA256 TLS_AE
zabbix_sender [42892]: DEBUG: zbx_tls_init_child() certificate and PSK ciphersuites: TLS_AES_256_GCM_SHA
```

The "certificate ciphersuites" and "PSK ciphersuites" lists have changed - they are shorter than before, only containing TLS 1.3 ciphersuites and TLS 1.2 ECDHE-* ciphersuites as expected.

2. TLSCipherAll and TLSCipherAll13 cannot be tested with zabbix_sender; they do not affect "certificate and PSK ciphersuites" value shown in the example above. To tweak TLSCipherAll and TLSCipherAll13 you need to experiment with the agent, proxy or server.

So, to allow only PFS ciphersuites you may need to add up to three parameters

```
TLSCipherCert=EECDH+aRSA+AES128
TLSCipherPSK=kECDHEPSK+AES128
TLSCipherAll=EECDH+aRSA+AES128:kECDHEPSK+AES128
```

to zabbix_agentd.conf, zabbix_proxy.conf and zabbix_server_conf if each of them has a configured certificate and agent has also PSK.

If your Zabbix environment uses only PSK-based encryption and no certificates, then only one:

```
TLSCipherPSK=kECDHEPSK+AES128
```

Now that you understand how it works you can test the ciphersuite selection even outside of Zabbix, with the `openssl` command. Let's test all three `TLSCipher*` parameter values:

```
$ openssl ciphers EECDH+aRSA+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 E
$ openssl ciphers kECDHEPSK+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-PSK-AES128-CBC-SHA256 E
$ openssl ciphers EECDH+aRSA+AES128:kECDHEPSK+AES128 | sed 's/:/ /g'
TLS_AES_256_GCM_SHA384 TLS_CHACHA20_POLY1305_SHA256 TLS_AES_128_GCM_SHA256 ECDHE-RSA-AES128-GCM-SHA256 E
```

You may prefer `openssl ciphers` with option `-V` for a more verbose output:

```
$ openssl ciphers -V EECDH+aRSA+AES128:kECDHEPSK+AES128
           0x13,0x02 - TLS_AES_256_GCM_SHA384  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(256) Mac=AEAD
           0x13,0x03 - TLS_CHACHA20_POLY1305_SHA256 TLSv1.3 Kx=any      Au=any  Enc=CHACHA20/POLY1305(256
           0x13,0x01 - TLS_AES_128_GCM_SHA256  TLSv1.3 Kx=any      Au=any  Enc=AESGCM(128) Mac=AEAD
           0xC0,0x2F - ECDHE-RSA-AES128-GCM-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AESGCM(128) Mac=AEAD
           0xC0,0x27 - ECDHE-RSA-AES128-SHA256 TLSv1.2 Kx=ECDH     Au=RSA  Enc=AES(128)  Mac=SHA256
           0xC0,0x13 - ECDHE-RSA-AES128-SHA    TLSv1 Kx=ECDH     Au=RSA  Enc=AES(128)  Mac=SHA1
           0xC0,0x37 - ECDHE-PSK-AES128-CBC-SHA256 TLSv1 Kx=ECDHEPSK Au=PSK  Enc=AES(128)  Mac=SHA256
           0xC0,0x35 - ECDHE-PSK-AES128-CBC-SHA TLSv1 Kx=ECDHEPSK Au=PSK  Enc=AES(128)  Mac=SHA1
```

Similarly, you can test the priority strings for GnuTLS:

```
$ gnutls-cli -l --priority=NONE:+VERS-TLS1.2:+ECDHE-RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+CURVE-A
Cipher suites for NONE:+VERS-TLS1.2:+ECDHE-RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+CURVE-ALL:+COMP-
TLS_ECDHE_RSA_AES_128_GCM_SHA256                        0xc0, 0x2f      TLS1.2
TLS_ECDHE_RSA_AES_128_CBC_SHA256                        0xc0, 0x27      TLS1.2

Protocols: VERS-TLS1.2
Ciphers: AES-128-GCM, AES-128-CBC
MACs: AEAD, SHA256
Key Exchange Algorithms: ECDHE-RSA
Groups: GROUP-SECP256R1, GROUP-SECP384R1, GROUP-SECP521R1, GROUP-X25519, GROUP-X448, GROUP-FFDHE2048, GR
PK-signatures: SIGN-RSA-SHA256, SIGN-RSA-PSS-SHA256, SIGN-RSA-PSS-RSAE-SHA256, SIGN-ECDSA-SHA256, SIGN-E
```

Switching from AES128 to AES256

Zabbix uses AES128 as the built-in default for data. Let's assume you are using certificates and want to switch to AES256, on OpenSSL 1.1.1.

This can be achieved by adding the respective parameters in `zabbix_server.conf`:

```
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/server.crt
TLSKeyFile=/home/zabbix/server.key
TLSCipherCert13=TLS_AES_256_GCM_SHA384
TLSCipherCert=EECDH+aRSA+AES256:-SHA1:-SHA384
TLSCipherPSK13=TLS_CHACHA20_POLY1305_SHA256
TLSCipherPSK=kECDHEPSK+AES256:-SHA1
TLSCipherAll13=TLS_AES_256_GCM_SHA384
TLSCipherAll=EECDH+aRSA+AES256:-SHA1:-SHA384
```

Attention:
Although only certificate-related ciphersuites will be used, `TLSCipherPSK*` parameters are defined as well to avoid their default values which include less secure ciphers for wider interoperability. PSK ciphersuites cannot be completely disabled on server/proxy.

And in `zabbix_agentd.conf`:

```
TLSConnect=cert
```

```
TLSAccept=cert
TLSCAFile=/home/zabbix/ca.crt
TLSCertFile=/home/zabbix/agent.crt
TLSKeyFile=/home/zabbix/agent.key
TLSCipherCert13=TLS_AES_256_GCM_SHA384
TLSCipherCert=EECDH+aRSA+AES256:-SHA1:-SHA384
```

1 Using certificates

Overview

Zabbix can use RSA certificates in PEM format, signed by a public or an in-house certificate authority (CA).

Certificate verification is performed against a pre-configured CA certificate. Optionally, Certificate Revocation Lists (CRL) can be used.

Each Zabbix component can have only one certificate configured.

For more information on setting up and operating an internal CA, generating and signing certificate requests, and revoking certificates, refer to tutorials such as the OpenSSL PKI Tutorial v2.0.

Carefully consider and test your certificate extensions. For more details, see Limitations on using X.509 v3 certificate extensions.

Certificate configuration parameters

The following configuration parameters are supported for setting up certificates on Zabbix components.

Parameter	Mandatory	Description
TLSCAFile	yes	Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification. If using a certificate chain with multiple members, order the certificates with lower level CA(s) certificates first, followed by higher level CA(s) certificates. Certificates from multiple CAs can be included in a single file.
TLSCRLFile	no	Full pathname of a file containing Certificate Revocation Lists (CRL).
TLSCertFile	yes	Full pathname of a file containing the certificate. If using a certificate chain with multiple members, order the certificates with the server, proxy, or agent certificate first, followed by lower level CA(s) certificates, and concluded by higher level CA(s) certificates.
TLSKeyFile	yes	Full pathname of a file containing the private key. Ensure that this file is readable only by the Zabbix user by setting appropriate access rights.
TLSServerCertIssuer	no	Allowed server certificate issuer.
TLSServerCertSubject	no	Allowed server certificate subject.

Configuration examples

After setting up the necessary certificates, configure Zabbix components to use certificate-based encryption.

Below are detailed steps for configuring:

- Zabbix server
- Zabbix proxy
- Zabbix agent

Zabbix server

1. Prepare the CA certificate file.

In order to verify peer certificates, Zabbix server must have access to the file containing the top-level, self-signed root CA certificates. For example, if certificates from two independent root CAs are needed, place them into a file at `/home/zabbix/zabbix_ca_file.crt`

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
```

```
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
            ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
            ...
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            ...
-----BEGIN CERTIFICATE-----
MIID2jCCAsKgAwIBAgIBATANBgkqhkiG9w0BAQUFADB+MRMwEQYKCZImiZPyLGQB
....
9wEzdN8uTrqoyU78gi12npLj08LegRKjb5hFTVmO
-----END CERTIFICATE-----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root2 CA
            ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root2 CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
            ....
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            ....
-----BEGIN CERTIFICATE-----
MIID3DCCAsSgAwIBAgIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQB
...
vdGNYoSfvu41GQAR5Vj5FnRJRzv5XQOZ3B6894GY1zY=
-----END CERTIFICATE-----
```

2. Place the Zabbix server certificate/certificate chain into a file, for example, at `/home/zabbix/zabbix_server.crt`. The first certificate is the Zabbix server certificate, followed by the intermediate CA certificate:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
        ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Zabbix server
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                ...
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Basic Constraints:
                CA:FALSE
            ...
```

```
-----BEGIN CERTIFICATE-----
MIIECDCCAvCgAwIBAgIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixk
...
h02u1GHiy46GI+xfR3LsPwFKlkTaaLaL/6aaoQ==
-----END CERTIFICATE-----
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2 (0x2)
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Root1 CA
        ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
            ...
        X509v3 extensions:
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
        ...
-----BEGIN CERTIFICATE-----
MIID4TCCAsmgAwIBAgIBAjANBgkqhkiG9w0BAQUFADB+MRMwEQYKCZImiZPyLGQB
...
dyCeWnvL7u5sd6ffo8iRny0QzbHKmQt/wUtcVIvWXdMIFJM0Hw==
-----END CERTIFICATE-----
```

Note:
Use only the attributes mentioned above for both client and server certificates to avoid affecting the certificate verification process. For example, OpenSSL might fail to establish an encrypted connection if *X509v3 Subject Alternative Name* or *Netscape Cert Type* extensions are used. For more information, see Limitations on using X.509 v3 certificate extensions.

3. Place the Zabbix server private key into a file, for example, at `/home/zabbix/zabbix_server.key`:

```
-----BEGIN PRIVATE KEY-----
MIIEwAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQC9tIXIJoVnNXDl
...
IJLkhbybBYEf47MLhffWa7XvZTY=
-----END PRIVATE KEY-----
```

4. Edit the TLS configuration parameters in the Zabbix server configuration file:

```
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSCertFile=/home/zabbix/zabbix_server.crt
TLSKeyFile=/home/zabbix/zabbix_server.key
```

Zabbix proxy

1. Prepare files with the top-level CA certificates, the Zabbix proxy certificate/certificate chain, and the private key as described in the Zabbix server section. Then, edit the `TLSCAFile`, `TLSCertFile`, and `TLSKeyFile` parameters in the Zabbix proxy configuration file accordingly.

2. Edit additional TLS parameters in the Zabbix proxy configuration file:

- For active proxy: `TLSConnect=cert`
- For passive proxy: `TLSAccept=cert`

Note:
To improve proxy security, you can also set the `TLSServerCertIssuer` and `TLSServerCertSubject` parameters. For more information, see Restricting allowed certificate issuer and subject.

TLS parameters in the final proxy configuration file may look as follows:

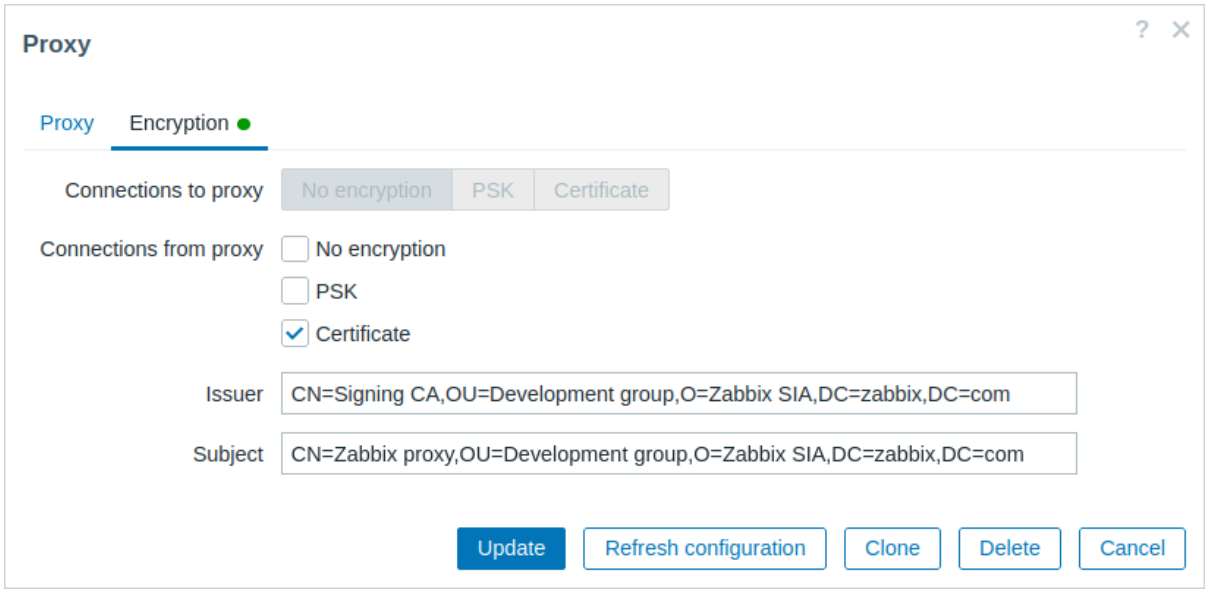
```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_proxy.crt
TLSKeyFile=/home/zabbix/zabbix_proxy.key
```

3. Configure encryption for this proxy in Zabbix frontend:

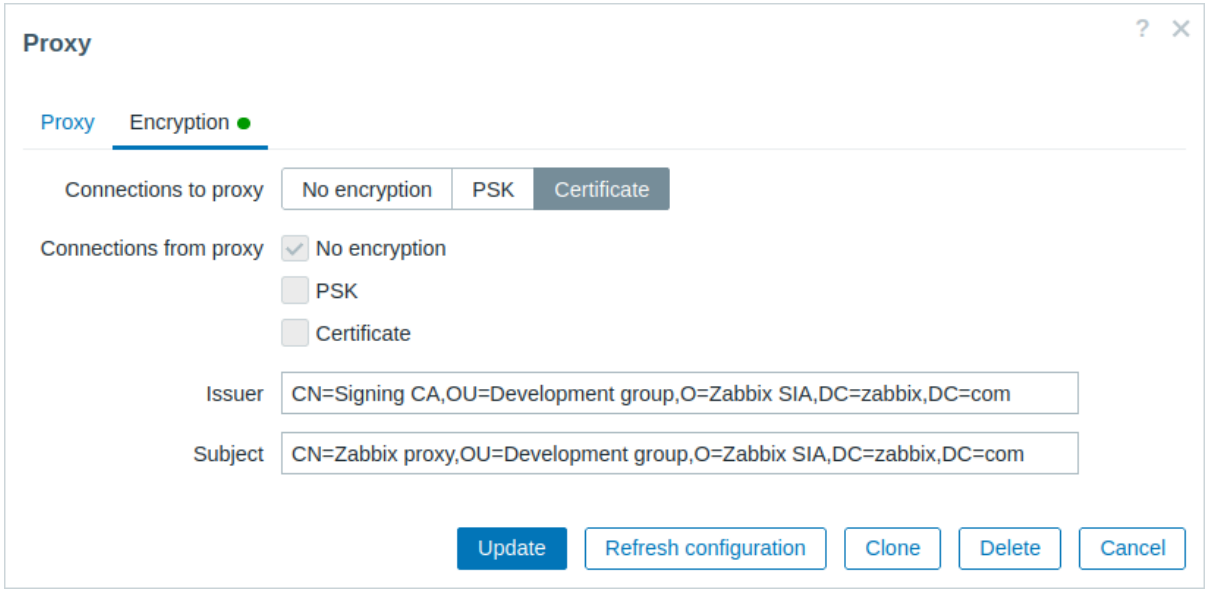
- Go to: *Administration* → *Proxies*.
- Select the proxy and click the *Encryption* tab.

In the examples below, the *Issuer* and *Subject* fields are filled in. For more information on why and how to use these fields, see Restricting allowed certificate issuer and subject.

For active proxy:



For passive proxy:



Zabbix agent

1. Prepare files with the top-level CA certificates, the Zabbix agent certificate/certificate chain, and the private key as described in the Zabbix server section. Then, edit the `TLSCAFile`, `TLSCertFile`, and `TLSKeyFile` parameters in the Zabbix agent configuration file accordingly.

2. Edit additional TLS parameters in the Zabbix agent configuration file:

- For active agent: `TLSConnect=cert`

- For passive agent: `TLSAccept=cert`

> **Note:**
> To improve agent security, you can set the `TLSServerCertIssuer` and `TLSServerCertSubject` parameters. For more information, see Restricting allowed certificate issuer and subject.

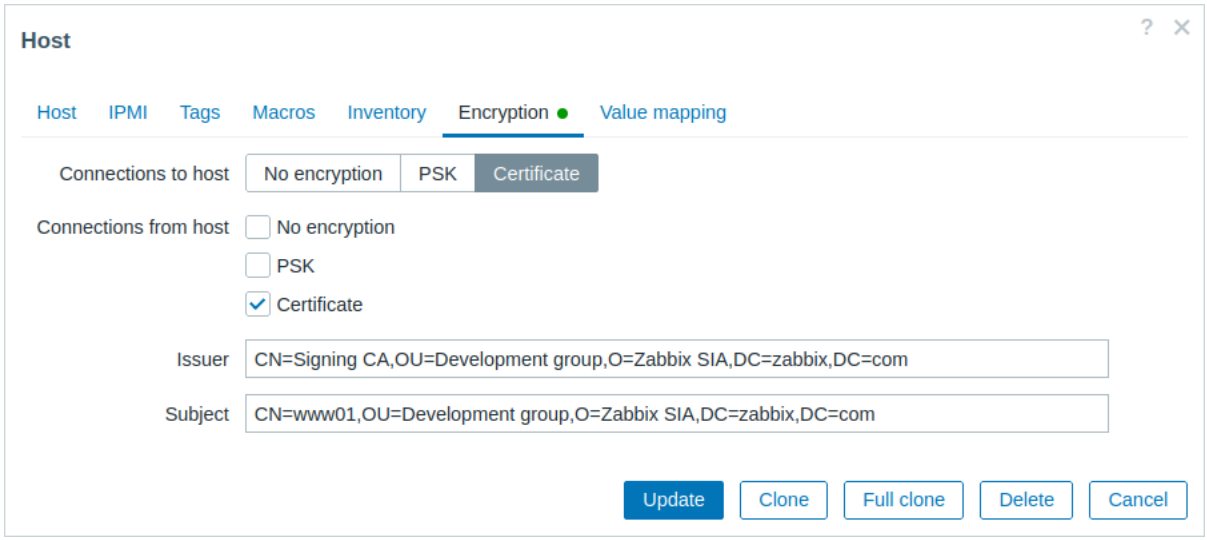
The TLS parameters in the final agent configuration file may look as follows. Note that the example assumes that the host is monitored by a proxy, hence it is specified as the certificate Subject:

```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_agentd.crt
TLSKeyFile=/home/zabbix/zabbix_agentd.key
```

3. Configure encryption in Zabbix frontend for the host monitored by this agent.

- Go to: *Data collection* → *Hosts*.
- Select the host and click the *Encryption* tab.

In the example below, the *Issuer* and *Subject* fields are filled in. For more information on why and how to use these fields, see Restricting allowed certificate issuer and subject.



Zabbix web service

1. Prepare files with the top-level CA certificates, the Zabbix web service certificate/certificate chain, and the private key as described in the Zabbix server section. Then, edit the `TLSCAFile`, `TLSCertFile`, and `TLSKeyFile` parameters in the Zabbix web service configuration file accordingly.

2. Edit an additional TLS parameter in the Zabbix web service configuration file: `TLSAccept=cert`

TLS parameters in the final web service configuration file may look as follows:

```
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSCertFile=/home/zabbix/zabbix_web_service.crt
TLSKeyFile=/home/zabbix/zabbix_web_service.key
```

3. Configure Zabbix server to connect to the TLS-configured Zabbix web service by editing the `WebServiceURL` parameter in the Zabbix server configuration file:

```
WebServiceURL=https://example.com
```

Restricting allowed certificate issuer and subject

When two Zabbix components (for example, server and agent) establish a TLS connection, they validate each other's certificates. If a peer certificate is signed by a trusted CA (with a pre-configured top-level certificate in `TLSCAFile`), is valid, has not expired, and passes other checks, then the communication between components can proceed. In this simplest case, the certificate issuer and subject are not verified.

However, this presents a risk: anyone with a valid certificate can impersonate anyone else (for example, a host certificate could be used to impersonate a server). While this may be acceptable in small environments where certificates are signed by a dedicated in-house CA and the risk of impersonation is low, it may not be sufficient in larger or more security-sensitive environments.

If your top-level CA issues certificates that should not be accepted by Zabbix or if you want to reduce the risk of impersonation, you can restrict allowed certificates by specifying their issuer and subject.

For example, in the Zabbix proxy configuration file, you could specify:

```
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

With these settings, an active proxy will not communicate with a Zabbix server whose certificate has a different issuer or subject. Similarly, a passive proxy will not accept requests from such a server.

Rules for matching `Issuer` and `Subject` strings

The rules for matching `Issuer` and `Subject` strings are as follows:

- `Issuer` and `Subject` strings are checked independently. Both are optional.
- An unspecified string means that any string is accepted.
- Strings are compared *as is* and must match exactly.
- UTF-8 characters are supported. However, wildcards (*) or regular expressions are not supported.
- The following RFC 4514 requirements are implemented - characters that require escaping (with a '\' backslash, U+005C):
  - anywhere in the string: '"' (U+0022), '+' (U+002B), ',' (U+002C), ';' (U+003B), '<' (U+003C), '>' (U+003E), '\\' (U+005C);
  - at the beginning of the string: space (' ', U+0020) or number sign ('#', U+0023);
  - at the end of the string: space (' ', U+0020).
- Null characters (U+0000) are not supported. If a null character is encountered, the matching will fail.
- RFC 4517 and RFC 4518 standards are not supported.

For example, if `Issuer` and `Subject` organization (O) strings contain trailing spaces and the `Subject` organizational unit (OU) string contains double quotes, these characters must be escaped:

```
TLSServerCertIssuer=CN=Signing CA,OU=Development head,O=\ Example SIA\ ,DC=example,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group \"5\",O=\ Example SIA\ ,DC=example,DC=com
```

Field order and formatting

Zabbix follows the recommendations of RFC 4514, which specifies a "reverse" order for these fields, starting with the lowest-level fields (CN), proceeding to the mid-level fields (OU, O), and concluding with the highest-level fields (DC).

```
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

In contrast, OpenSSL by default displays the `Issuer` and `Subject` strings in top-level to low-level order. In the following example, `Issuer` and `Subject` fields start with the top-level (DC) and end with the low-level (CN) field. The formatting with spaces and field separators also varies based on the options used, and thus will not match the format required by Zabbix.

```
$ openssl x509 -noout -in /home/zabbix/zabbix_proxy.crt -issuer -subject
issuer= /DC=com/DC=zabbix/O=Zabbix SIA/OU=Development group/CN=Signing CA
subject= /DC=com/DC=zabbix/O=Zabbix SIA/OU=Development group/CN=Zabbix proxy

$ openssl x509 -noout -text -in /home/zabbix/zabbix_proxy.crt
Certificate:
    ...
        Issuer: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Signing CA
        ...
        Subject: DC=com, DC=zabbix, O=Zabbix SIA, OU=Development group, CN=Zabbix proxy
```

To format *Issuer* and *Subject* strings correctly for Zabbix, invoke OpenSSL with the following options:

```
$ openssl x509 -noout -issuer -subject \
    -nameopt esc_2253,esc_ctrl,utf8,dump_nostr,dump_unknown,dump_der,sep_comma_plus,dn_rev,sname\
    -in /home/zabbix/zabbix_proxy.crt
```

The output will then be in reverse order, comma-separated, and usable in Zabbix configuration files and frontend:

```
issuer= CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
subject= CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
```

Limitations on using X.509 v3 certificate extensions

When implementing X.509 v3 certificates within Zabbix, certain extensions may not be fully supported or could result in inconsistent behavior.

Subject Alternative Name extension

Zabbix does not support the *Subject Alternative Name* extension, which is used to specify alternative DNS names such as IP addresses or email addresses. Zabbix can only validate the value in the *Subject* field of the certificate (see Restricting Allowed Certificate Issuer and Subject). If certificates include the `subjectAltName` field, the outcome of certificate validation may vary depending on the specific crypto toolkits used to compile Zabbix components. As a result, Zabbix may either accept or reject certificates based on these combinations.

Extended Key Usage extension

Zabbix supports the *Extended Key Usage* extension. However, if used, it is generally required that both *clientAuth* (for TLS WWW client authentication) and *serverAuth* (for TLS WWW server authentication) attributes are specified. For example:

- In passive checks, where Zabbix agent operates as a TLS server, the *serverAuth* attribute must be included in the agent's certificate.
- For active checks, where the agent operates as a TLS client, the *clientAuth* attribute must be included in the agent's certificate.

While GnuTLS may issue a warning for key usage violations, it typically allows communication to proceed despite these warnings.

Name Constraints extension

Support for the *Name Constraints* extension varies among crypto toolkits. Ensure that your chosen toolkit supports this extension. This extension may restrict Zabbix from loading CA certificates if this section is marked as critical, depending on the specific toolkit in use.

Certificate Revocation Lists (CRL)

If a certificate is compromised, the Certificate Authority (CA) can revoke it by including the certificate in a Certificate Revocation List (CRL). CRLs are managed through configuration files and can be specified using the `TLSCRLFile` parameter in server, proxy, and agent configuration files. For example:

```
TLSCRLFile=/home/zabbix/zabbix_crl_file.crt
```

In this case, `zabbix_crl_file.crt` may contain CRLs from multiple CAs, and could look like this:

```
-----BEGIN X509 CRL-----
MIIB/DCB5QIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixkARkWA2Nv
...
treZeUPjb7LSmZ3K2hpbZN7SoOZcAoHQ3GWd9npuctg=
-----END X509 CRL-----
-----BEGIN X509 CRL-----
MIIB+TCB4gIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQBGRYDY29t
...
CAEebS2CND3ShBedZ8YSil59O6JvaDP61lR5lNs=
-----END X509 CRL-----
```

The CRL file is loaded only when Zabbix starts. To update the CRL, restart Zabbix.

> **Attention:**
> If Zabbix components are compiled with OpenSSL and CRLs are used, ensure that each top-level and intermediate CA in the certificate chains has a corresponding CRL (even if it is empty) included in the `TLSCRLFile`.

2 Using pre-shared keys

Overview

Each pre-shared key (PSK) in Zabbix actually is a pair of:

- non-secret PSK identity string,
- secret PSK string value.

PSK identity string is a non-empty UTF-8 string. For example, "PSK ID 001 Zabbix agentd". It is a unique name by which this specific PSK is referred to by Zabbix components. Do not put sensitive information in PSK identity string - it is transmitted over the network unencrypted.

PSK value is a hard to guess string of hexadecimal digits, for example, "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327b

Size limits

There are size limits for PSK identity and value in Zabbix, in some cases a crypto library can have lower limit:

Component	PSK identity max size	PSK value min size	PSK value max size
Zabbix	128 UTF-8 characters	128-bit (16-byte PSK, entered as 32 hexadecimal digits)	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
GnuTLS	128 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
OpenSSL 1.0.x, 1.1.0	127 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
OpenSSL 1.1.1	127 bytes (may include UTF-8 characters)	-	512-bit (64-byte PSK, entered as 128 hexadecimal digits)
OpenSSL 1.1.1a and later	127 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)

Attention:
Zabbix frontend allows configuring up to 128-character long PSK identity string and 2048-bit long PSK regardless of crypto libraries used.
If some Zabbix components support lower limits, it is the user's responsibility to configure PSK identity and value with allowed length for these components.
Exceeding length limits results in communication failures between Zabbix components.

Before Zabbix server connects to agent using PSK, the server looks up the PSK identity and PSK value configured for that agent in database (actually in configuration cache). Upon receiving a connection the agent uses PSK identity and PSK value from its configuration file. If both parties have the same PSK identity string and PSK value the connection may succeed.

Attention:
Each PSK identity must be paired with only one value. It is the user's responsibility to ensure that there are no two PSKs with the same identity string but different values. Failing to do so may lead to unpredictable errors or disruptions of communication between Zabbix components using PSKs with this PSK identity string.

Generating PSK

For example, a 256-bit (32 bytes) PSK can be generated using the following commands:

- with *OpenSSL*:

```
$ openssl rand -hex 32
af8ced32dfe8714e548694e2d29e1a14ba6fa13f216cb35c19d0feb1084b0429
```

- with *GnuTLS*:

```
$ psktool -u psk_identity -p database.psk -s 32
Generating a random key for user 'psk_identity'
Key stored to database.psk

$ cat database.psk
psk_identity:9b8eafedfaae00cece62e85d5f4792c7d9c9bcc851b23216a1d300311cc4f7cb
```

Note that "psktool" above generates a database file with a PSK identity and its associated PSK. Zabbix expects just a PSK in the PSK file, so the identity string and colon (':') should be removed from the file.

Configuring PSK for server-agent communication (example)

On the agent host, write the PSK value into a file, for example, `/home/zabbix/zabbix_agentd.psk`. The file must contain PSK in the first text string, for example:

```
1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

Set access rights to PSK file - it must be readable only by Zabbix user.

Edit TLS parameters in agent configuration file `zabbix_agentd.conf`, for example, set:

```
TLSConnect=psk
TLSAccept=psk
TLSPSKFile=/home/zabbix/zabbix_agentd.psk
TLSPSKIdentity=PSK 001
```

The agent will connect to server (active checks) and accept from server and `zabbix_get` only connections using PSK. PSK identity will be ”PSK 001”.

Restart the agent. Now you can test the connection using `zabbix_get`, for example:

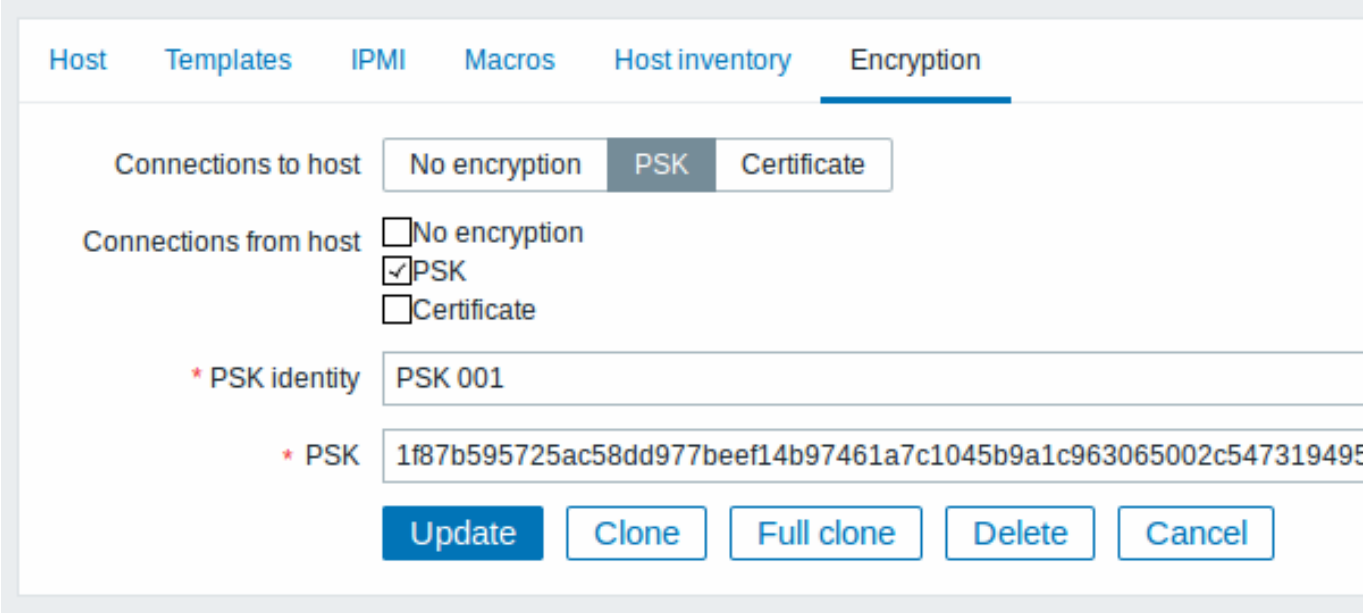
```
$ zabbix_get -s 127.0.0.1 -k "system.cpu.load[all,avg1]" --tls-connect=psk \
            --tls-psk-identity="PSK 001" --tls-psk-file=/home/zabbix/zabbix_agentd.psk
```

(To minimize downtime see how to change connection type in Connection encryption management).

Configure PSK encryption for this agent in Zabbix frontend:

- Go to: *Data collection → Hosts*
- Select host and click on **Encryption** tab

Example:



All mandatory input fields are marked with a red asterisk.

When configuration cache is synchronized with database the new connections will use PSK. Check server and agent logfiles for error messages.

Configuring PSK for server - active proxy communication (example)

On the proxy, write the PSK value into a file, for example, `/home/zabbix/zabbix_proxy.psk`. The file must contain PSK in the first text string, for example:

```
e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9
```

Set access rights to PSK file - it must be readable only by Zabbix user.

Edit TLS parameters in proxy configuration file `zabbix_proxy.conf`, for example, set:

```
TLSConnect=psk
TLSPSKFile=/home/zabbix/zabbix_proxy.psk
TLSPSKIdentity=PSK 002
```

The proxy will connect to server using PSK. PSK identity will be ”PSK 002”.

(To minimize downtime see how to change connection type in Connection encryption management).

Configure PSK for this proxy in Zabbix frontend. Go to *Administration→Proxies*, select the proxy, go to "Encryption" tab. In "Connections from proxy" mark PSK. Paste into "PSK identity" field "PSK 002" and "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d083 into "PSK" field. Click "Update".

Restart proxy. It will start using PSK-based encrypted connections to server. Check server and proxy logfiles for error messages.

For a passive proxy the procedure is very similar. The only difference - set `TLSAccept=psk` in proxy configuration file and set "Connections to proxy" in Zabbix frontend to PSK.

3 Troubleshooting

General recommendations

- Start with understanding which component acts as a TLS client and which one acts as a TLS server in problem case.
  Zabbix server, proxies and agents, depending on interaction between them, all can work as TLS servers and clients.
  For example, Zabbix server connecting to agent for a passive check, acts as a TLS client. The agent is in role of TLS server.
  Zabbix agent, requesting a list of active checks from proxy, acts as a TLS client. The proxy is in role of TLS server.
  `zabbix_get` and `zabbix_sender` utilities always act as TLS clients.
- Zabbix uses mutual authentication.
  Each side verifies its peer and may refuse connection.
  For example, Zabbix server connecting to agent can close connection immediately if agent's certificate is invalid. And vice versa - Zabbix agent accepting a connection from server can close connection if server is not trusted by agent.
- Examine logfiles in both sides - in TLS client and TLS server.
  The side which refuses connection may log a precise reason why it was refused. Other side often reports rather general error (e.g. "Connection closed by peer", "connection was non-properly terminated").
- Sometimes misconfigured encryption results in confusing error messages in no way pointing to real cause.
  In subsections below we try to provide a (far from exhaustive) collection of messages and possible causes which could help in troubleshooting.
  Please note that different crypto toolkits (OpenSSL, GnuTLS) often produce different error messages in same problem situations.
  Sometimes error messages depend even on particular combination of crypto toolkits on both sides.

1 Connection type or permission problems

Server is configured to connect with PSK to agent but agent accepts only unencrypted connections

In server or proxy log (with *GnuTLS* 3.3.16)

```
Get value from agent failed: zbx_tls_connect(): gnutls_handshake() failed: \
    -110 The TLS connection was non-properly terminated.
```

In server or proxy log (with *OpenSSL* 1.0.2c)

```
Get value from agent failed: TCP connection successful, cannot establish TLS to [[127.0.0.1]:10050]: \
    Connection closed by peer. Check allowed connection types and access rights
```

One side connects with certificate but other side accepts only PSK or vice versa

In any log (with *GnuTLS*):

```
failed to accept an incoming connection: from 127.0.0.1: zbx_tls_accept(): gnutls_handshake() failed:\
    -21 Could not negotiate a supported cipher suite.
```

In any log (with *OpenSSL* 1.0.2c):

```
failed to accept an incoming connection: from 127.0.0.1: TLS handshake returned error code 1:\
    file .\ssl\s3_srvr.c line 1411: error:1408A0C1:SSL routines:ssl3_get_client_hello:no shared cipher:\
    TLS write fatal alert "handshake failure"
```

Attempting to use Zabbix sender compiled with TLS support to send data to Zabbix server/proxy compiled without TLS

In connecting-side log:

Linux:

```
...In zbx_tls_init_child()
...OpenSSL library (version OpenSSL 1.1.1  11 Sep 2018) initialized
...
```

```
...In zbx_tls_connect(): psk_identity:"PSK test sender"
...End of zbx_tls_connect():FAIL error:'connection closed by peer'
...send value error: TCP successful, cannot establish TLS to [[localhost]:10051]: connection closed by pee
```

Windows:

```
...OpenSSL library (version OpenSSL 1.1.1a  20 Nov 2018) initialized
...
...In zbx_tls_connect(): psk_identity:"PSK test sender"
...zbx_psk_client_cb() requested PSK identity "PSK test sender"
...End of zbx_tls_connect():FAIL error:'SSL_connect() I/O error: [0x00000000] The operation completed succ
...send value error: TCP successful, cannot establish TLS to [[192.168.1.2]:10051]: SSL_connect() I/O erro
```

In accepting-side log:

```
...failed to accept an incoming connection: from 127.0.0.1: support for TLS was not compiled in
```

One side connects with PSK but other side uses LibreSSL or has been compiled without encryption support

LibreSSL does not support PSK.

In connecting-side log:

```
...TCP successful, cannot establish TLS to [[192.168.1.2]:10050]: SSL_connect() I/O error: [0] Success
```

In accepting-side log:

```
...failed to accept an incoming connection: from 192.168.1.2: support for PSK was not compiled in
```

In Zabbix frontend:

```
Get value from agent failed: TCP successful, cannot establish TLS to [[192.168.1.2]:10050]: SSL_connect()
```

One side connects with PSK but other side uses OpenSSL with PSK support disabled

In connecting-side log:

```
...TCP successful, cannot establish TLS to [[192.168.1.2]:10050]: SSL_connect() set result code to SSL_ERR
```

In accepting-side log:

```
...failed to accept an incoming connection: from 192.168.1.2: TLS handshake set result code to 1: file ssl
```

2 Certificate problems

OpenSSL used with CRLs and for some CA in the certificate chain its CRL is not included in TLSCRLFile

In TLS server log in case of *OpenSSL* peer:

```
failed to accept an incoming connection: from 127.0.0.1: TLS handshake with 127.0.0.1 returned error code
    file s3_srvr.c line 3251: error:14089086: SSL routines:ssl3_get_client_certificate:certificate verify
    TLS write fatal alert "unknown CA"
```

In TLS server log in case of *GnuTLS* peer:

```
failed to accept an incoming connection: from 127.0.0.1: TLS handshake with 127.0.0.1 returned error code
    file rsa_pk1.c line 103: error:0407006A: rsa routines:RSA_padding_check_PKCS1_type_1:\
    block type is not 01 file rsa_eay.c line 705: error:04067072: rsa routines:RSA_EAY_PUBLIC_DECRYPT:padd
```

CRL expired or expires during server operation

OpenSSL, in server log:

- before expiration:

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
    SSL_connect() returned SSL_ERROR_SSL: file s3_clnt.c line 1253: error:14090086:\
    SSL routines:ssl3_get_server_certificate:certificate verify failed:\
    TLS write fatal alert "certificate revoked"
```

- after expiration:

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
    SSL_connect() returned SSL_ERROR_SSL: file s3_clnt.c line 1253: error:14090086:\
    SSL routines:ssl3_get_server_certificate:certificate verify failed:\
    TLS write fatal alert "certificate expired"
```

The point here is that with valid CRL a revoked certificate is reported as "certificate revoked". When CRL expires the error message changes to "certificate expired" which is quite misleading.

GnuTLS, in server log:

- before and after expiration the same:

```
cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
      invalid peer certificate: The certificate is NOT trusted. The certificate chain is revoked.
```

Self-signed certificate, unknown CA

OpenSSL, in log:

```
error:'self signed certificate: SSL_connect() set result code to SSL_ERROR_SSL: file ../ssl/statem/statem_
      line 1924: error:1416F086:SSL routines:tls_process_server_certificate:certificate verify failed:\
      TLS write fatal alert "unknown CA"'
```

This was observed when server certificate by mistake had the same Issuer and Subject string, although it was signed by CA. Issuer and Subject are equal in top-level CA certificate, but they cannot be equal in server certificate. (The same applies to proxy and agent certificates.)

To check whether a certificate contains the same Issuer and Subject entries, run:

```
openssl x509 -in <yourcertificate.crt> -noout -text
```

It is acceptable for the root (top-level) certificate to have identical values for Issuer and Subject.

3 PSK problems

PSK contains an odd number of hex-digits

Proxy or agent does not start, message in the proxy or agent log:

```
invalid PSK in file "/home/zabbix/zabbix_proxy.psk"
```

PSK identity string longer than 128 bytes is passed to GnuTLS

In TLS client side log:

```
gnutls_handshake() failed: -110 The TLS connection was non-properly terminated.
```

In TLS server side log.

```
gnutls_handshake() failed: -90 The SRP username supplied is illegal.
```

Too long PSK value used with OpenSSL 1.1.1

In connecting-side log:

```
...OpenSSL library (version OpenSSL 1.1.1  11 Sep 2018) initialized
...
...In zbx_tls_connect(): psk_identity:"PSK 1"
...zbx_psk_client_cb() requested PSK identity "PSK 1"
...End of zbx_tls_connect():FAIL error:'SSL_connect() set result code to SSL_ERROR_SSL: file ssl\statem\ex
```

In accepting-side log:

```
...Message from 123.123.123.123 is missing header. Message ignored.
```

This problem typically arises when upgrading OpenSSL from 1.0.x or 1.1.0 to 1.1.1 and if the PSK value is longer than 512-bit (64-byte PSK, entered as 128 hexadecimal digits).

See also: Value size limits

18 Web interface

Overview For an easy access to Zabbix from anywhere and from any platform, the web-based interface is provided.

> **Note:**
> If using more than one frontend instance make sure that the locales and libraries (LDAP, SAML etc.) are installed and configured identically for all frontends.

Frontend help A help link is provided in Zabbix frontend forms with direct links to the corresponding parts of the documentation.

1 Menu

Overview

A vertical menu in a sidebar provides access to various Zabbix frontend sections.

The menu is dark blue in the default theme.

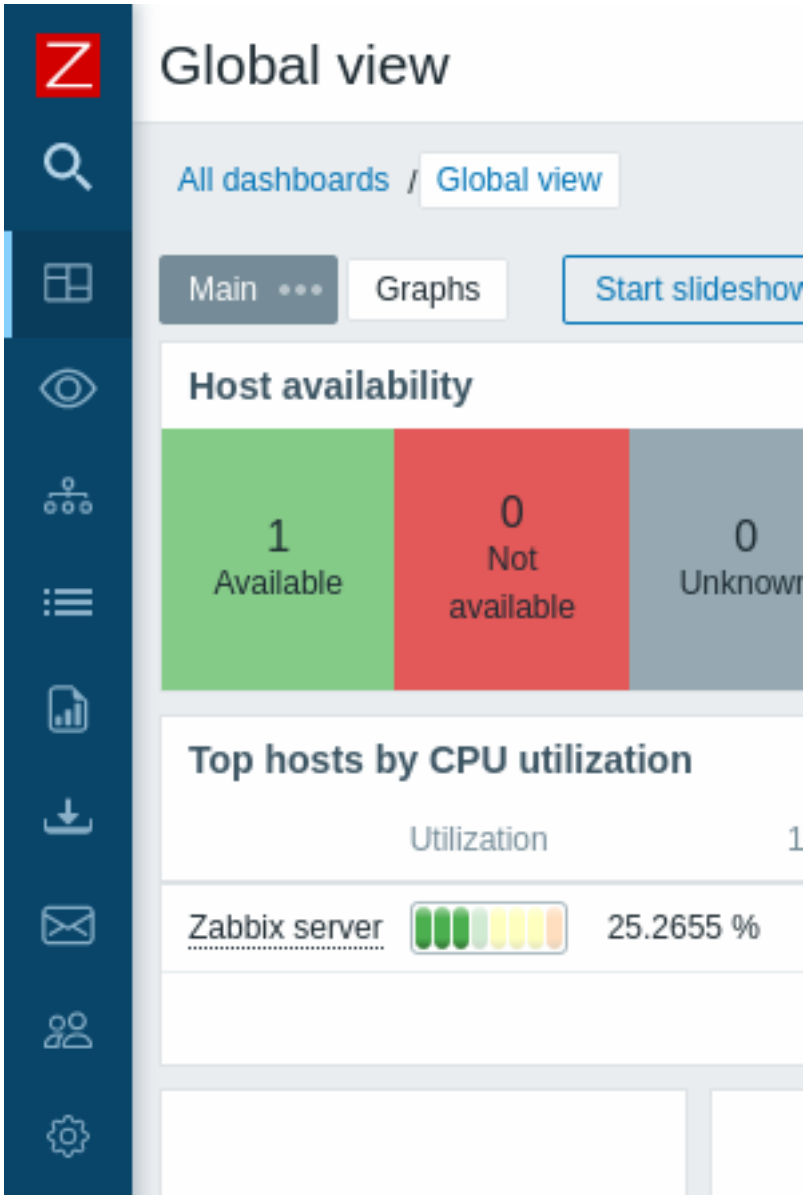


Working with the menu

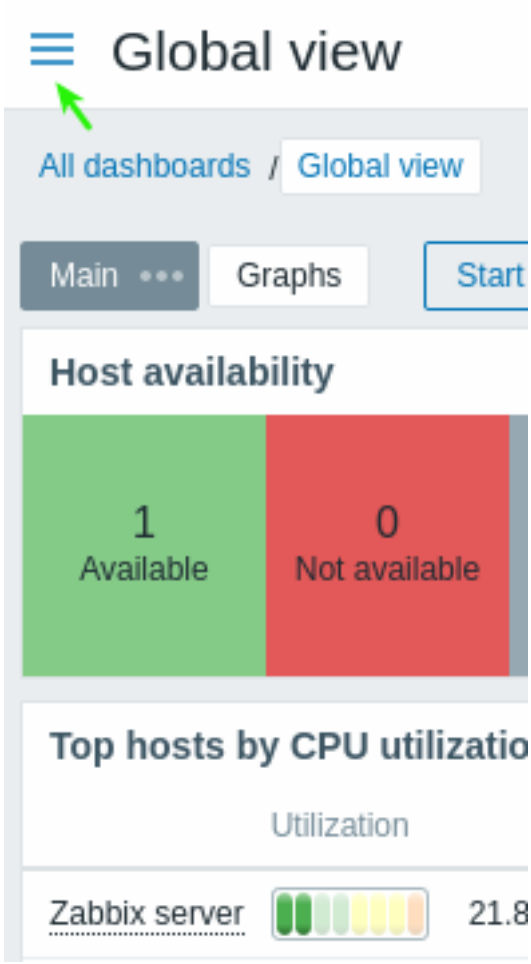
A global search box is located below the Zabbix logo.

The menu can be collapsed or hidden completely:

- To collapse, click on next to Zabbix logo. In the collapsed menu only the icons are visible.



- To hide, click on  next to Zabbix logo. In the hidden menu everything is hidden.



Collapsed menu

When the menu is collapsed to icons only, a full menu reappears as soon as the mouse cursor is placed upon it. Note that it reappears over page content; to move page content to the right you have to click on the expand button. If the mouse cursor again is placed outside the full menu, the menu will collapse again after two seconds.

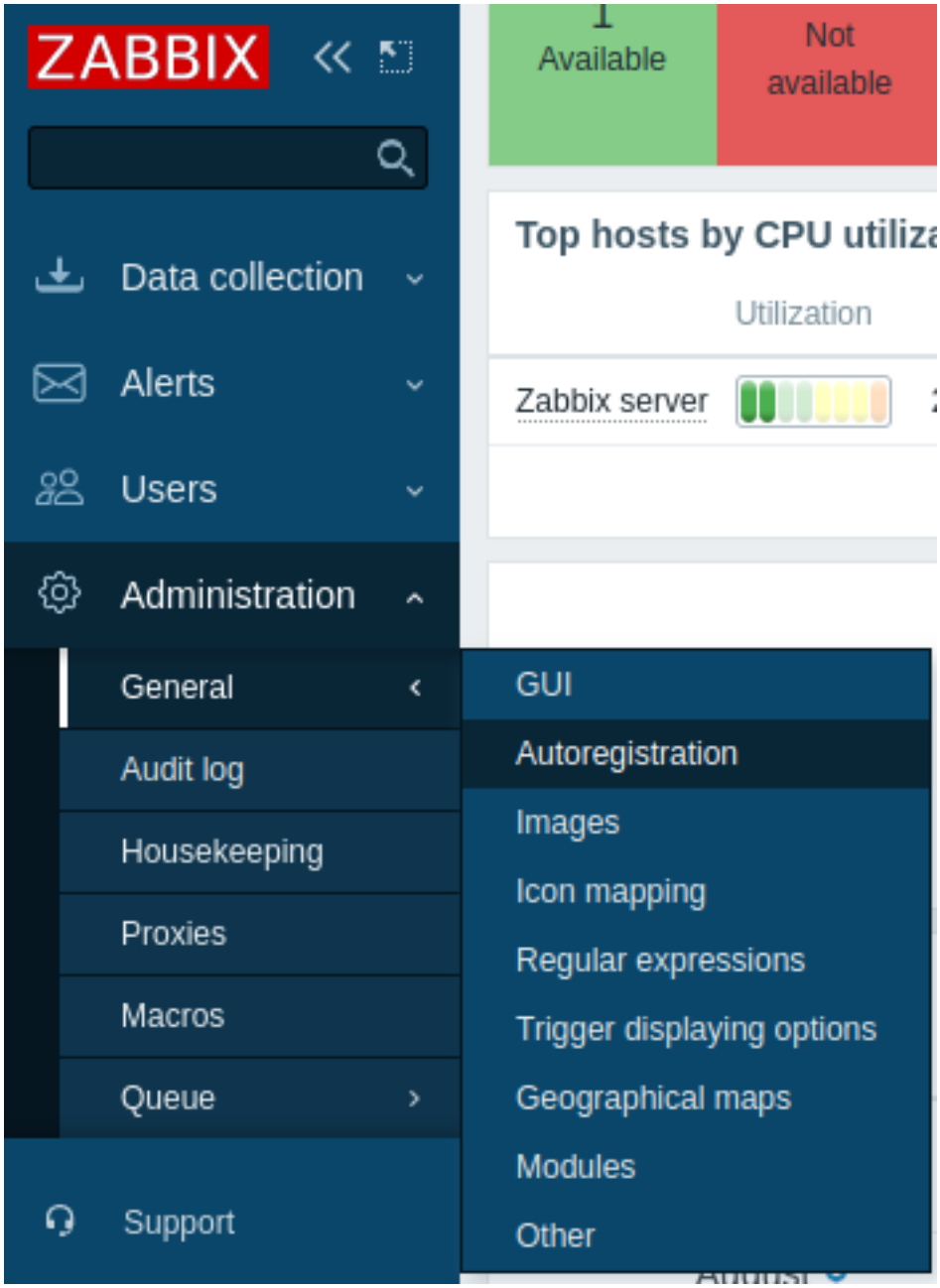
You can also make a collapsed menu reappear fully by hitting the Tab key. Hitting the Tab key repeatedly will allow to focus on the next menu element.

Hidden menu

Even when the menu is hidden completely, a full menu is just one mouse click away, by clicking on the burger icon. Note that it reappears over page content; to move page content to the right you have to unhide the menu by clicking on the show sidebar button.

Menu levels

There are up to three levels in the menu.



Context menus

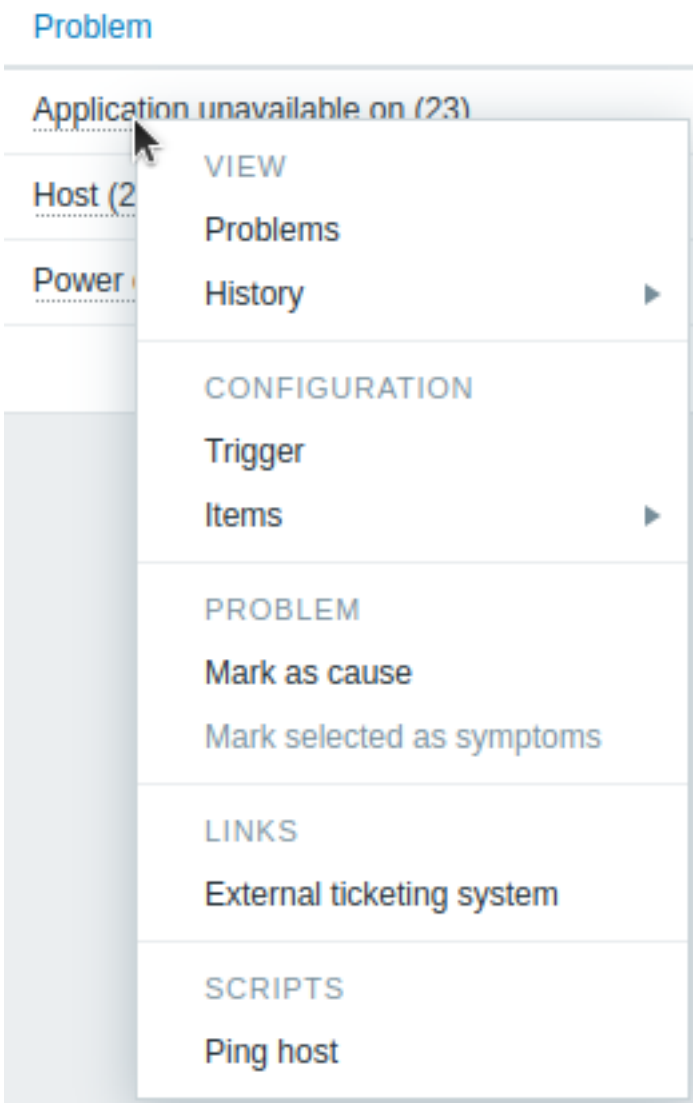
In addition to the main menu, Zabbix provides context menus for host, item, and event for quick access to frequently used entities, such as the latest values, simple graph, configuration form, related scripts or external links.

The context menus are accessible by clicking on the host, item or problem/trigger name in supported locations.

1 Event menu

Overview

The event context menu provides shortcuts for executing frequently required actions or navigating to UI sections related to an event.



Content

The event context menu has six sections: *View*, *Actions*, *Configuration*, *Problem*, *Links*, and *Scripts*.

For the entities that are not configured, links are disabled and displayed in gray. The sections *Scripts* and *Links* are displayed if their entities are configured.

The *View* section contains links to:

- **Problems** - opens the list of unresolved problems of the underlying trigger;
- **History** - leads to the *Latest data* graph/item history for the underlying item(s). If a trigger uses several items, links will be available for each of them.

The *Actions* section is available in *Trigger overview* widgets only. It contains a link to:

- **Update problem** - opens the problem update screen.

The *Configuration* section contains links to the configuration of:

- **Trigger** that fired the problem;
- **Items** used in the trigger expression.

Note:
Note that configuration section is available only for Admin and Super admin users.

The *Problem* section contains the options to:

- **Mark as cause** - mark the problem as cause;
- **Mark selected as symptoms** - mark the selected problems as symptoms of this problem.

The *Links* section contains links to:

- access a configured trigger URL;

- access custom links configured in Global scripts (with scope 'Manual event action' and type 'URL');
- access a configured external ticket for the problem (see the *Include event menu entry* option when configuring webhooks).

The *Scripts* section contains links to execute a global script (with scope *Manual event action*). This feature may be handy for running scripts used for managing problem tickets in external systems.

Supported locations

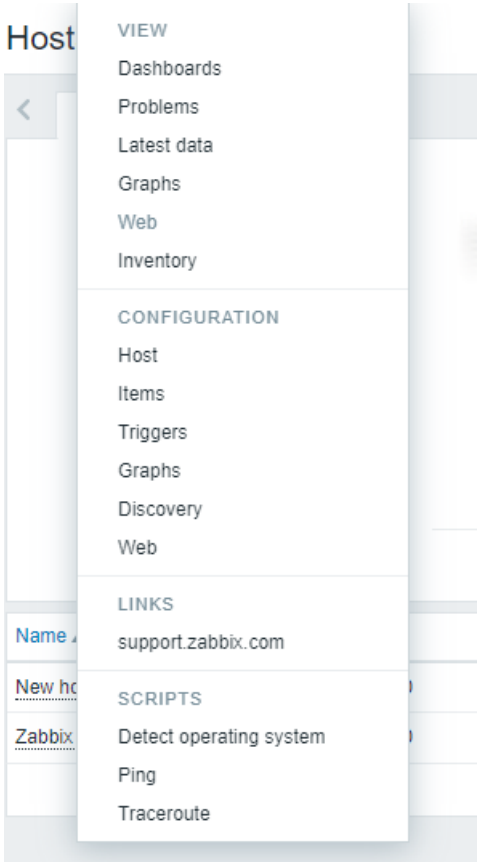
The event context menu is accessible by clicking on a problem or event name in various frontend sections, for example:

- Dashboards widgets, such as *Problems* widget, *Trigger overview* widget, etc.
- Monitoring → Problems
- Monitoring → Problems → Event details
- Reports → Triggers top 100 (global scripts and access to external ticket are not supported in this location)

2 Host menu

Overview

The host context menu provides shortcuts for executing frequently required actions or navigating to UI sections related to a host.



Content

The host context menu has four sections: *View*, *Configuration*, *Links*, and *Scripts*. For the entities that are not configured, links are disabled and displayed in grey color. The sections *Scripts* and *Links* are displayed if their entities are configured.

View section contains links to:

- **Dashboards** - opens widgets and graphs.
- **Problems** - opens the *Problems* section with the list of unresolved problems of the underlying trigger.
- **Latest data** - opens the *Latest data* section with the list of all the latest data of the current host.
- **Graphs** - opens simple graphs of the current host.
- **Web**- opens the link to the configured web scenarios.
- **Inventory** - opens the link to the inventory of the current host.

Configuration section contains links to:

- **Host** - configuration form of the current host.
- **Items** - the list of the current host items.
- **Triggers** - the list of the current host triggers.

- **Graphs** - simple graphs of the current host.
- **Discovery** - the list of low-level discovery rules of the current host.
- **Web** - the list of web scenarios of the current host.

> **Note:**
> Note that configuration section is available only for Admin and Super admin users.

Links section contains links to:

- access a configured trigger URL.
- access custom links configured in Global scripts (with scope *Manual host action* and type 'URL').

Scripts section allows to execute global scripts configured for the current host. These scripts need to have their scope defined as *Manual host action* to be available in the host menu.

Supported locations

The host menu is accessible by clicking on a host in various frontend sections, for example:

- Dashboards widgets, such as Data overview, Trigger overview, Problems, etc.
- Monitoring → Problems
- Monitoring → Problems → Event details
- Monitoring → Hosts
- Monitoring → Hosts → Web Monitoring
- Monitoring → Latest data
- Monitoring → Maps
- Inventory → Hosts
- Reports → Triggers top 100

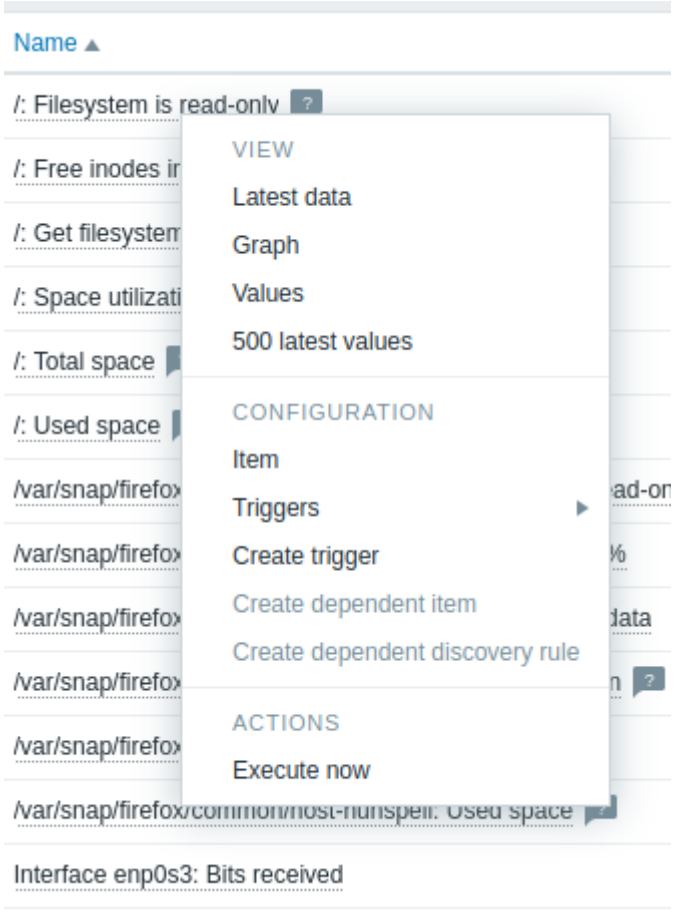
3 Item menu

Overview

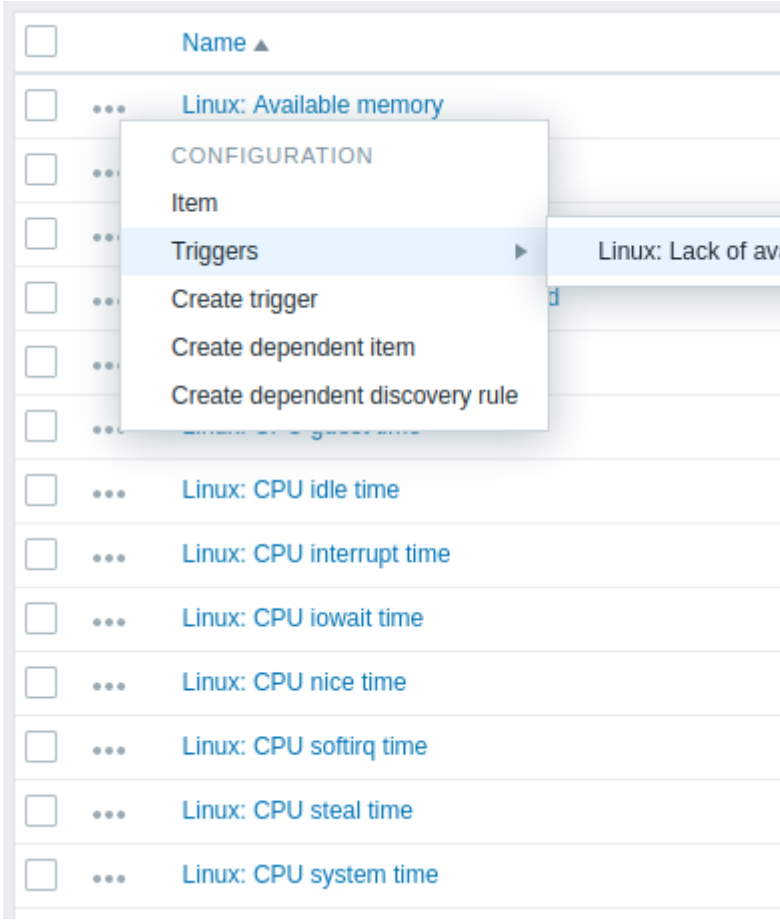
The item context menu contains shortcuts to actions or frontend sections that are frequently required for an item.

The item context menu can be opened by clicking on the item name or the three-dot icon, depending on the frontend section, for example:

Monitoring → *Latest data*



Data Collection → *Hosts* → *Items*



Content

The item context menu has three sections: *View*, *Configuration*, and *Actions*.

The *View* section contains the following options:

- **Latest data** - opens the *Latest data* section filtered by the current host and item;
- **Graph** - opens a simple graph of the current item;
- **Values** - opens the list of all values received for the current item within the past 60 minutes;
- **500 latest values** - opens the list of 500 latest values for the current item.

The *Configuration* section (available only for *Admin* and *Super admin* type users) contains the following options:

- **Item** - opens the item configuration form of the current item;
- **Triggers** - on mouseover, opens a list of item's triggers, if any; clicking on a trigger opens its trigger configuration form;
- **Create trigger** - opens the trigger configuration form to create a trigger for this item;
- **Create dependent item** - opens the item configuration form to create a dependent item, with the current item as the master item (available only when the item context menu is accessed from the *Data collection* section);
- **Create dependent discovery rule** - opens the discovery rule configuration form to create a dependent discovery rule, with the current item as the master item (available only when the item context menu is accessed from the *Data collection* section).

The *Actions* section contains the following option:

- **Execute now** - executes a check for a new item value immediately.

Supported locations

The item context menu is accessible by clicking on an item name in various frontend sections, for example:

- Monitoring → Latest data
- Data Collection → Hosts → Items
- Data collection → Hosts → Discovery rules → Item prototypes

2 Frontend sections

Menu structure The Zabbix frontend menu has the following structure:

- Dashboards
- Monitoring
  - Problems
  - Hosts
  - Latest data
  - Maps
  - Discovery
- Services
  - Services
  - SLA
  - SLA report
- Inventory
  - Overview
  - Hosts
- Reports
  - System information
  - Scheduled reports
  - Availability report
  - Triggers top 100
  - Audit log
  - Action log
  - Notifications
- Data collection
  - Template groups
  - Host groups
  - Templates
  - Hosts
  - Maintenance
  - Event correlation
  - Discovery
- Alerts
  - Actions
    - Trigger actions
    - Service actions
    - Discovery actions
    - Autoregistration actions
    - Internal actions
  - Media types
  - Scripts
- Users
  - User groups
  - User roles
  - Users
  - API tokens
  - Authentication
- Administration
  - General
    - GUI
    - Autoregistration
    - Images
    - Icon mapping
    - Regular expressions
    - Trigger displaying options
    - Geographical maps
    - Modules
    - Other
  - Audit log

- Housekeeping
- Proxies
- Macros
- Queue
    - Queue overview
    - Queue overview by proxy
    - Queue details

1 Dashboards

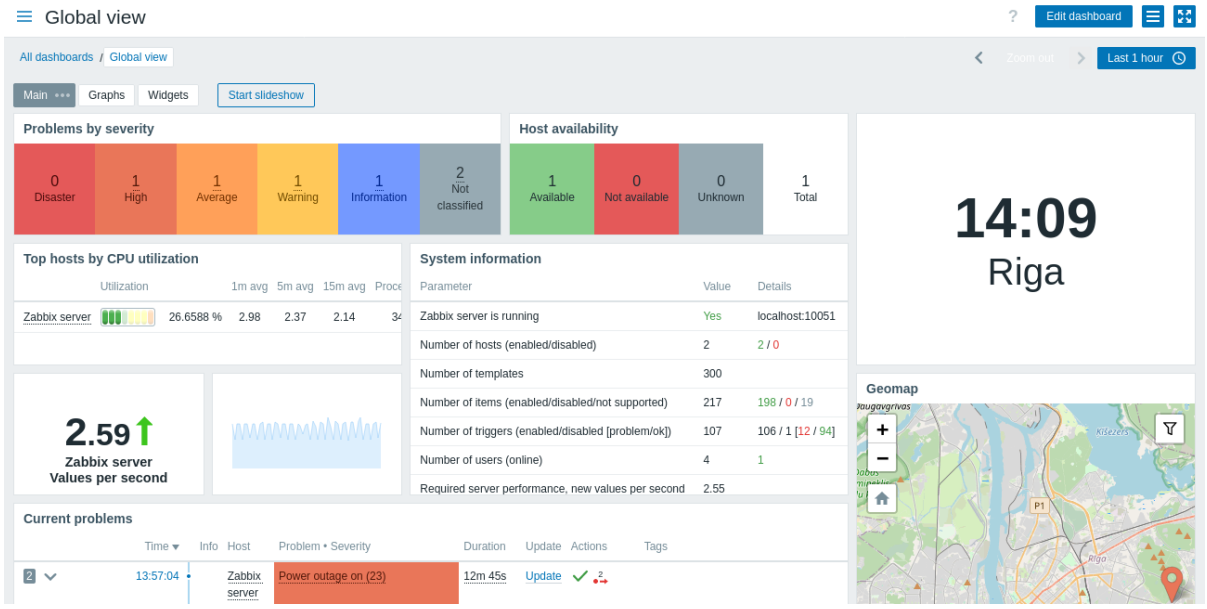
Overview

The *Dashboards* section is designed to display summaries of all the important information in a **dashboard**.

While only one dashboard can displayed at one time, it is possible to configure several dashboards. Each dashboard may contain one or several pages that can be rotated in a slideshow.

A dashboard page consists of widgets and each widget is designed to display information of a certain kind and source, which can be a summary, a map, a graph, the clock, etc.

Access to hosts in the widgets depends on host permissions.



Pages and widgets are added to the dashboard and edited in the dashboard editing mode. Pages can be viewed and rotated in the dashboard viewing mode.

The time period that is displayed in graph widgets is controlled by the time period selector located above the widgets. The time period selector label, located to the right, displays the currently selected time period. Clicking the tab label allows expanding and collapsing the time period selector.

Note that when the dashboard is displayed in kiosk mode and widgets only are displayed, it is possible to zoom out the graph period by double-clicking in the graph.

Dashboard size

The minimum width of a dashboard is 1200 pixels. The dashboard will not shrink below this width; instead a horizontal scrollbar is displayed if the browser window is smaller than that.

The maximum width of a dashboard is the browser window width. Dashboard widgets stretch horizontally to fit the window. At the same time, a dashboard widget cannot be stretched horizontally beyond the window limits.

Technically the dashboard consists of 12 horizontal columns of always equal width that stretch/shrink dynamically (but not to less than 1200 pixels total).

Vertically the dashboard may contain a maximum of 64 rows. Each row has a fixed height of 70 pixels. A widget may be up to 32 rows high.

Viewing dashboards

To view all configured dashboards, click on *All dashboards* just below the section title.



Dashboards are displayed with a sharing tag:

- *My* - indicates a private dashboard
- *Shared* - indicates a public dashboard or a private dashboard shared with any user or user group

The filter located to the right above the list allows to filter dashboards by name and by those created by the current user.

To delete one or several dashboards, mark the checkboxes of the respective dashboards and click on *Delete* below the list.

Viewing a dashboard

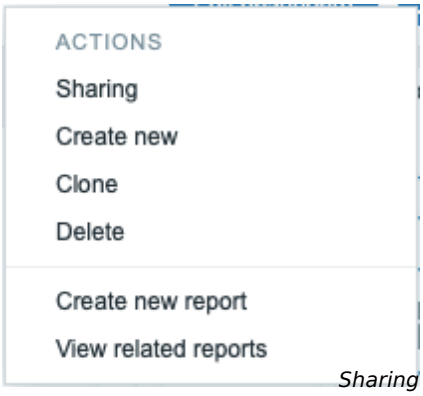
To view a single dashboard, click on its name in the list of dashboards.

When **viewing** a dashboard, the following options are available:

Edit dashboard

Switch to the dashboard **editing** mode.
The editing mode is also opened when a new dashboard is being created and when you click on the edit button of a widget.

Open the action menu (see action descriptions below).



Sharing - edit sharing preferences for the dashboard.
Create new - create a new dashboard.
Clone - create a new dashboard by copying properties of the existing one. First you are prompted to enter dashboard parameters. Then, the new dashboard opens in editing mode with all the widgets of the original dashboard.
Delete - delete the dashboard.
Create new report - open a pop-up window with report configuration form. Disabled if the user does not have permission to manage scheduled reports.
View related reports - open a pop-up window with a list of existing reports based on the current dashboard. Disabled if there are no related reports or the user does not have permission to view scheduled reports.

	Display only page content (kiosk mode). Kiosk mode can also be accessed with the following URL parameters: `/zabbix.php?action=dashboard.view&kiosk=1`. To exit to normal mode: `/zabbix.php?action=dashboard.view&kiosk=0`

Sharing

Dashboards can be made public or private.

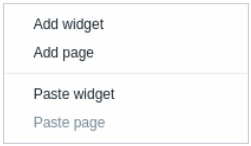
Public dashboards are visible to all users. Private dashboards are visible only to their owner. Private dashboards can be shared by the owner with other users and user groups.

The sharing status of a dashboard is displayed in the list of all dashboards. To edit the sharing status of a dashboard, click on the *Sharing* option in the action menu when viewing a single dashboard:

Parameter	Description
Type	Select dashboard type: **Private** - dashboard is visible only to selected user groups and users **Public** - dashboard is visible to all
List of user group shares	Select user groups that the dashboard is accessible to. You may allow read-only or read-write access.
List of user shares	Select users that the dashboard is accessible to. You may allow read-only or read-write access.

Editing a dashboard

When **editing** a dashboard, the following options are available:

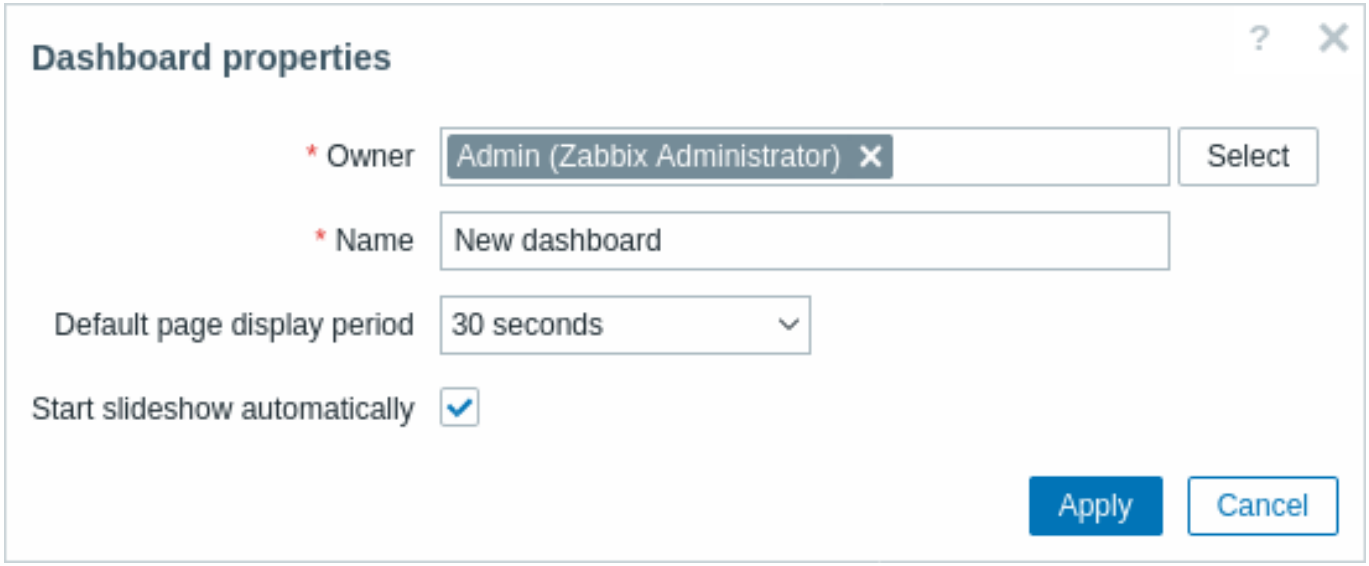
	Edit general dashboard parameters.
	Add a new widget. Clicking on the arrow button will open the action menu (see action descriptions below).
	*Add widget* - add a new widget *Add page* - add a new page *Paste widget* - paste a copied widget. This option is grayed out if no widget has been copied. Only one entity (widget or page) can be copied at one time. *Paste page* - paste a copied page. This option is grayed out if no page has been copied.
	Save dashboard changes.
	Cancel dashboard changes.

Creating a dashboard

It is possible to create a new dashboard in two ways:

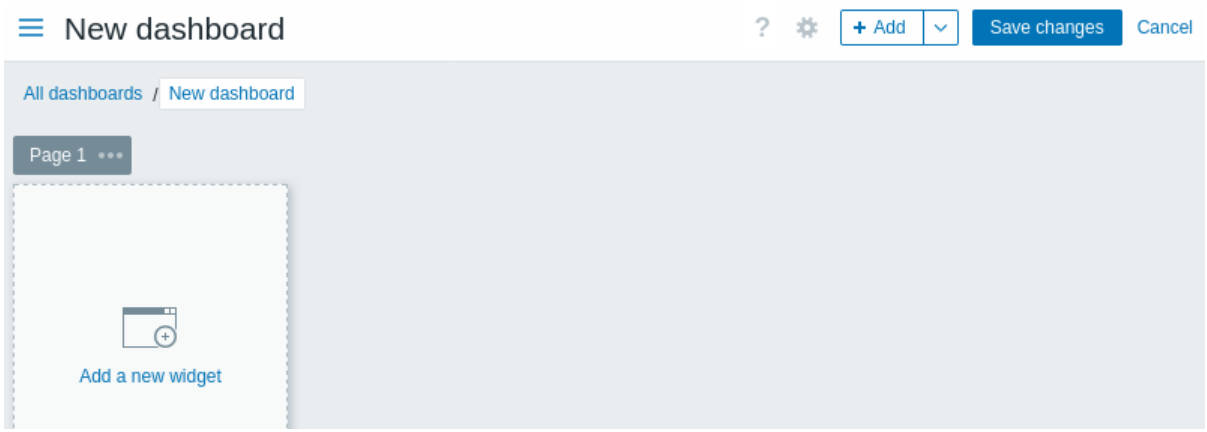
- Click on *Create dashboard*, when viewing all dashboards
- Select *Create new* from the action menu, when viewing a single dashboard

You will be first asked to enter general dashboard parameters:



Parameter	Description
Owner	Select system user that will be the dashboard owner.
Name	Enter dashboard name.
Default page display period	Select period for how long a dashboard page is displayed before rotating to the next page in a slideshow.
Start slideshow automatically	Mark this checkbox to run a slideshow automatically one more than one dashboard page exists.

When you click on *Apply*, an empty dashboard is opened:



To populate the dashboard, you can add widgets and pages.

Click on the *Save changes* button to save the dashboard. If you click on *Cancel*, the dashboard will not be created.

Adding widgets

To add a widget to a dashboard:

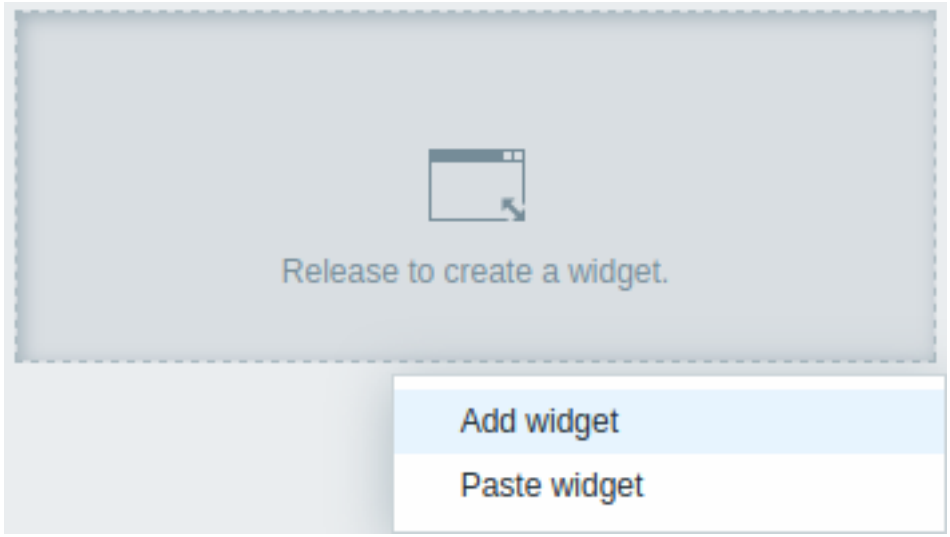
- Click on the 

button or the *Add widget* option in the action menu that can be opened by clicking on the arrow. Fill the widget configuration form. The widget will be created in its default size and placed after the existing widgets (if any);

Or

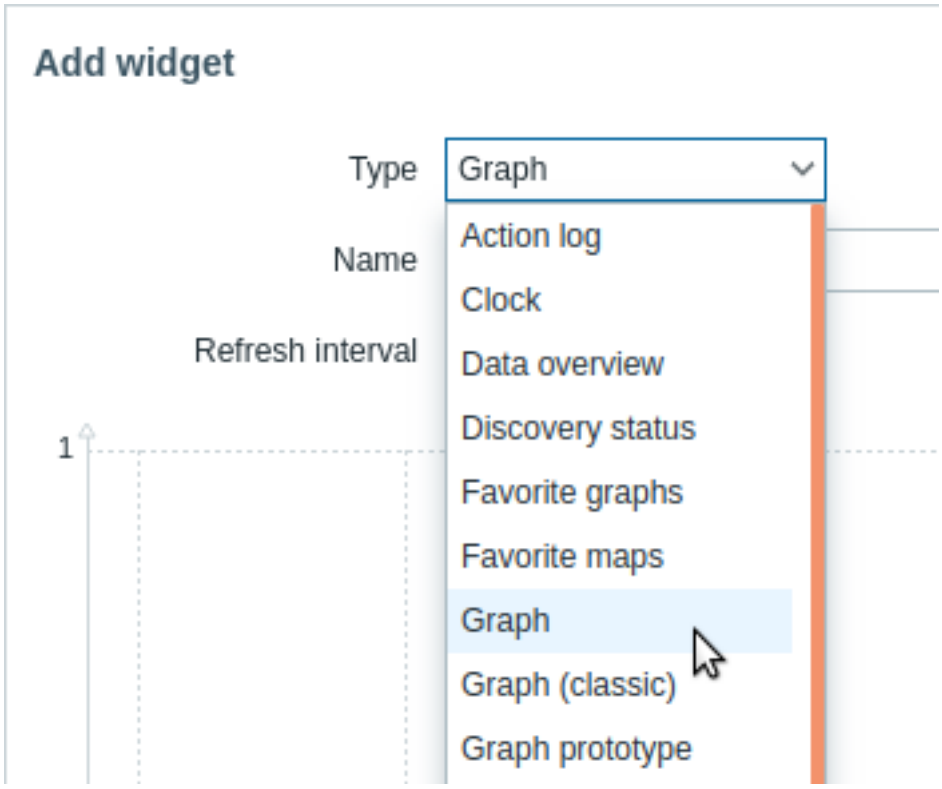
- Move your mouse to the desired empty spot for the new widget. Notice how a placeholder appears, on mouseover, on any empty slot on the dashboard. Then click to open the widget configuration form. After filling the form the widget will be created in its default size or, if its default size is bigger than is available, take up the available space. Alternatively, you may click and drag the placeholder to the desired widget size, then release, and then fill the widget configuration form. (Note that when there is a widget copied onto the clipboard, you will be first prompted to select between *Add widget* and *Paste widget* options to create a widget.)





In the widget configuration form:

- Select the *Type* of widget
- Enter widget parameters
- Click on *Add*



Widgets

A wide variety of widgets (e.g. Clock, Host availability or Trigger overview) can be added to a dashboard: these can be resized and moved around the dashboard in dashboard editing mode by clicking on the widget title bar and dragging it to a new location.

Also, you can click on the following buttons in the top-right corner of the widget to:

-  - edit a widget;
- - access the widget menu

Click on *Save changes* for the dashboard to make any changes to the widgets permanent.

Copying/pasting widgets

Dashboard widgets can be copied and pasted, allowing to create a new widget with the properties of an existing one. They can be copy-pasted within the same dashboard, or between dashboards opened in different tabs.

A widget can be copied using the widget menu. To paste the widget:

- click on the arrow next to the *Add* button and selecting the *Paste widget* option, when editing the dashboard
- use the *Paste widget* option when adding a new widget by selecting some area in the dashboard (a widget must be copied first for the paste option to become available)

A copied widget can be used to paste over an existing widget using the *Paste* option in the widget menu.

Creating a slideshow

A slideshow will run automatically if the dashboard contains two or more pages (see Adding pages) and if one of the following is true:

- The *Start slideshow automatically* option is marked in dashboard properties
- The dashboard URL contains a `slideshow=1` parameter

The pages rotate according to the intervals given in the properties of the dashboard and individual pages. Click on:

- *Stop slideshow* - to stop the slideshow
- *Start slideshow* - to start the slideshow



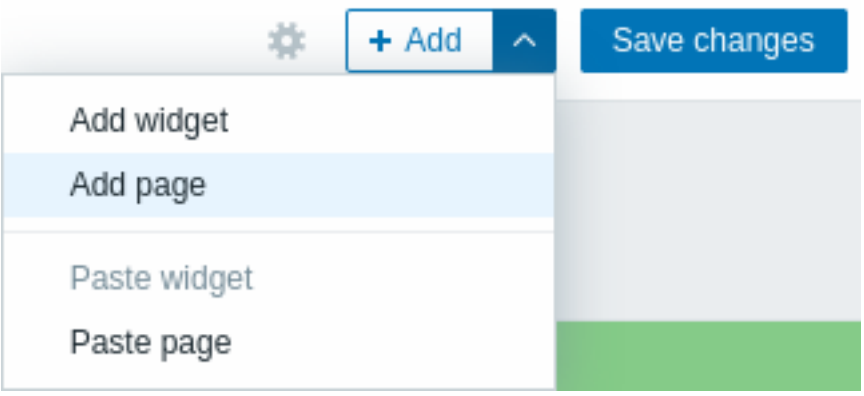
Slideshow-related controls are also available in kiosk mode (where only the page content is shown):

-  - stop slideshow
-  - start slideshow
-  - go back one page
-  - go to the next page

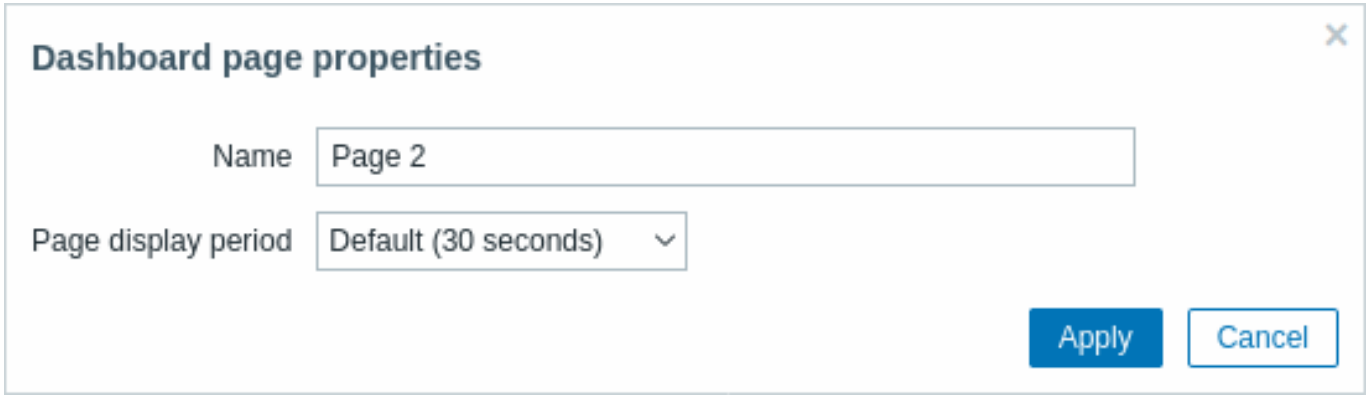
Adding pages

To add a new page to a dashboard:

- Make sure the dashboard is in the editing mode
- Click on the arrow next to the *Add* button and select the *Add page* option



- Fill the general page parameters and click on *Apply*. If you leave the name empty, the page will be added with a `Page N` name where 'N' is the incremental number of the page. The page display period allows to customize how long a page is displayed in a slideshow.



A new page will be added, indicated by a new tab (*Page 2*).



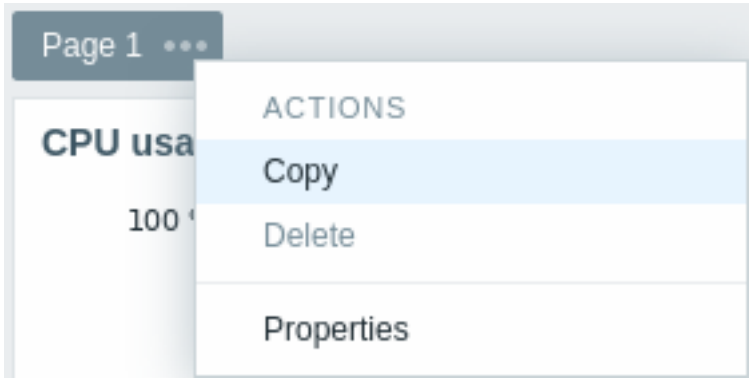
The pages can be reordered by dragging-and-dropping the page tabs. Reordering maintains the original page naming. It is always possible to go to each page by clicking on its tab.

When a new page is added, it is empty. You can add widgets to it as described above.

Copying/pasting pages

Dashboard pages can be copied and pasted, allowing to create a new page with the properties of an existing one. They can be pasted from the same dashboard or a different dashboard.

To paste an existing page to the dashboard, first copy it, using the page menu:

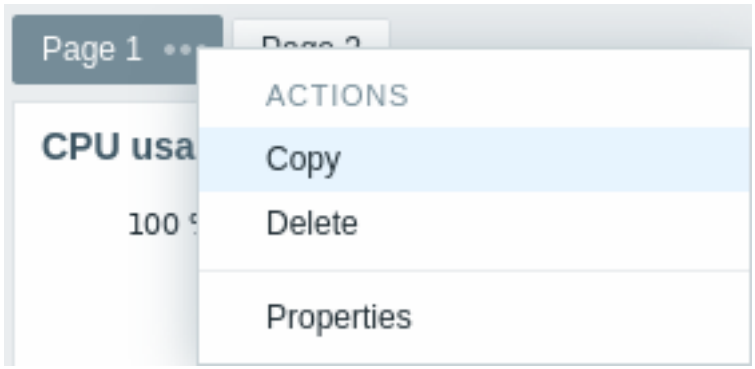


To paste the copied page:

- Make sure the dashboard is in the editing mode
- Click on the arrow next to the *Add* button and select the *Paste page* option

Page menu

The page menu can be opened by clicking on the three dots  next to the page name:



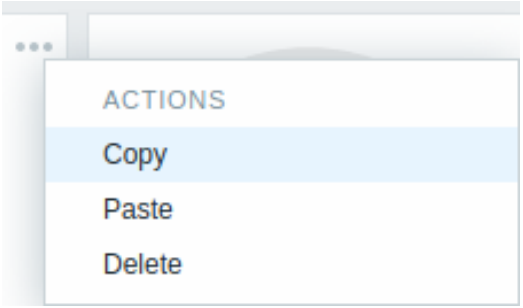
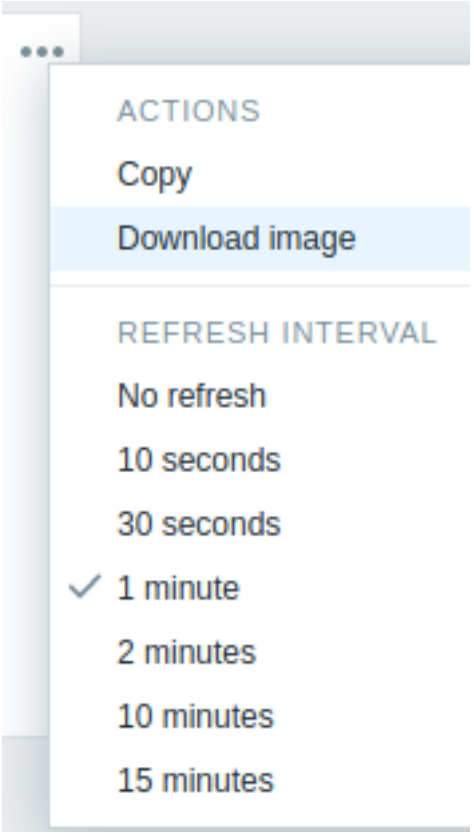
It contains the following options:

- *Copy* - copy the page
- *Delete* - delete the page (pages can only be deleted in the dashboard editing mode)

- *Properties* - customize the page parameters (the name and the page display period in a slideshow)

Widget menu

The widget menu contains different options based on whether the dashboard is in the edit or view mode:

Widget menu	Options
In dashboard edit mode: 	*Copy* - copy the widget *Paste* - paste a copied widget over this widget This option is grayed out if no widget has been copied. *Delete* - delete the widget
In dashboard view mode: 	*Copy* - copy the widget *Download image* - download the widget as a PNG image (only available for graph/classic graph widgets) *Refresh interval* - select the frequency of refreshing the widget contents

Dynamic widgets

When configuring some of the widgets:

- Classic graph
- Graph prototype
- Item value
- Plain text
- URL

there is an extra option called *Enable host selection*. You can check this box to make the widget dynamic - i.e. capable of displaying different content based on the selected host.

Now, when saving the dashboard, you will notice that a new host selection field has appeared atop the dashboard for selecting the host (while the *Select* button allows selecting the host group in a popup):



Thus you have a widget, which can display content that is based on the data from the host that is selected. The benefit of this is that you do not need to create extra widgets just because, for example, you want to see the same graph containing data from various hosts.

Permissions to dashboards

Permissions to dashboards for regular users and users of 'Admin' type are limited in the following way:

- They can see and clone a dashboard if they have at least READ rights to it;
- They can edit and delete dashboard only if they have READ/WRITE rights to it;
- They cannot change the dashboard owner.

1 Dashboard widgets

Overview

This section provides the details of parameters that are common for dashboard widgets.

Common parameters

The following parameters are common for every single widget:

Name	Enter a widget name.
Refresh interval	Configure default refresh interval. Default refresh intervals for widgets range from *No refresh* to *15 minutes* depending on the type of widget. For example: *No refresh* for URL widget, *1 minute* for action log widget, *15 minutes* for clock widget.
Show header	Mark the checkbox to show the header permanently. When unchecked the header is hidden to save space and only slides up and becomes visible again when the mouse is positioned over the widget, both in view and edit modes. It is also semi-visible when dragging a widget to a new place.

Refresh intervals for a widget can be set to a default value for all the corresponding users and also each user can set his own refresh interval value:

- To set a default value for all the corresponding users switch to editing mode (click the *Edit dashboard* button, find the right widget, click the *Edit* button opening the editing form of a widget), and choose the required refresh interval from the dropdown list.
- Setting a unique refresh interval for each user separately is possible in view mode by clicking the button for a certain widget.

Unique refresh interval set by a user has priority over the widget setting and once it's set it's always preserved when the widget's setting is modified.

To see **specific parameters** for each widget, go to individual widget pages for:

- Action log
- Clock
- Data overview
- Discovery status
- Favorite graphs
- Favorite maps
- Geomap
- Graph
- Graph (classic)
- Graph prototype
- Host availability
- Item value
- Map
- Map navigation tree
- Plain text
- Problem hosts
- Problems
- SLA report
- System information
- Problems by severity
- Top hosts

- Trigger overview
- URL
- Web monitoring

Deprecated widgets:

- Data overview

Attention:
Deprecated widgets will be removed in the upcoming major release.

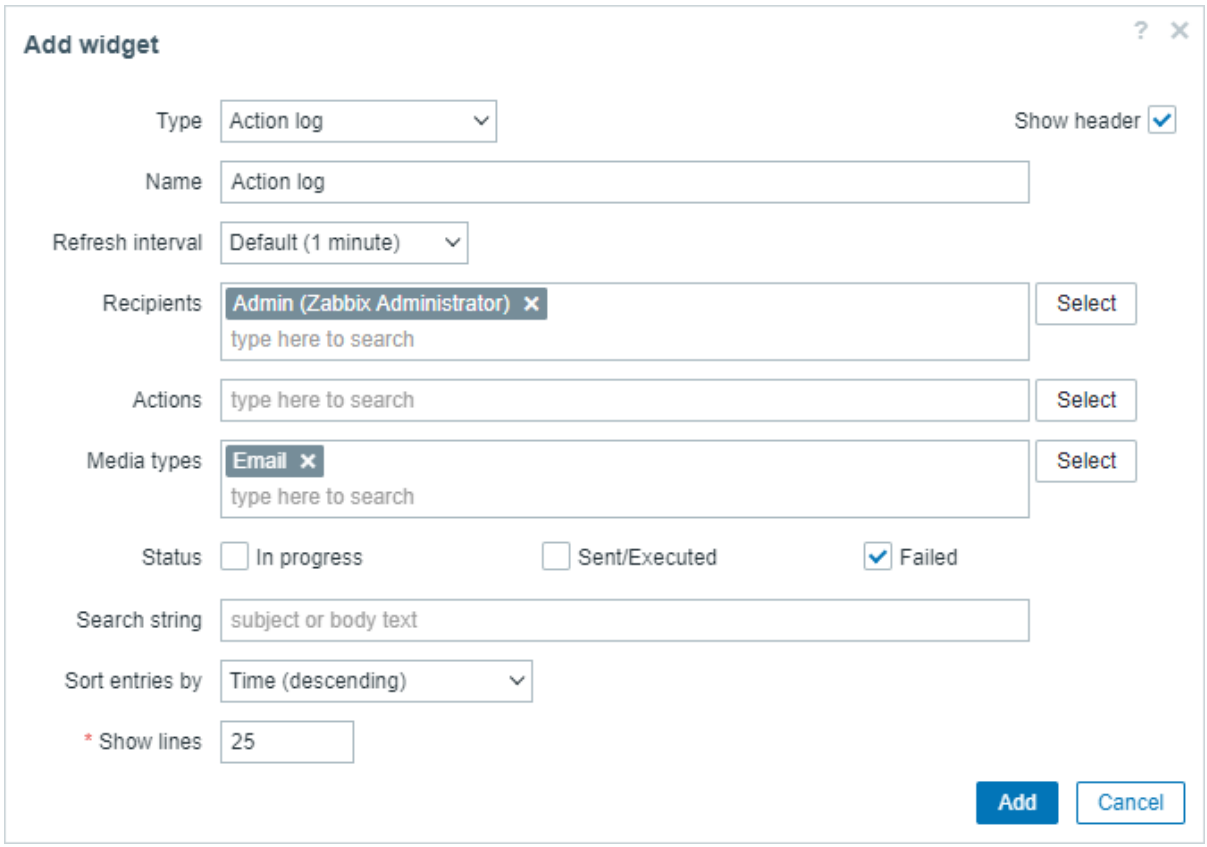
1 Action log

Overview

In the action log widget, you can display details of action operations (notifications, remote commands). It replicates information from *Reports* → *Action log*.

Configuration

To configure, select *Action log* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

Recipients	Filter entries by recipients. This field is auto-complete, so starting to type the name of a recipient will offer a dropdown of matching recipients. If no recipients are selected, details of action operations for all recipients will be displayed.
Actions	Filter entries by actions. This field is auto-complete, so starting to type the name of an action will offer a dropdown of matching actions. If no actions are selected, details of action operations for all actions will be displayed.
Media types	Filter entries by media types. This field is auto-complete, so starting to type the name of a media type will offer a dropdown of matching media types. If no media types are selected, details of action operations for all media types will be displayed.

Status	Mark the checkbox to filter entries by the respective status: **In progress** - action operations that are in progress are displayed **Sent/Executed** - action operations that have sent a notification or have been executed are displayed **Failed** - action operations that have failed are displayed
Search string	Filter entries by the content of the message/remote command. If you enter a string here, only those action operations whose message/remote command contains the entered string will be displayed. Macros are not resolved.
Sort entries by	Sort entries by: **Time** (descending or ascending) **Type** (descending or ascending) **Status** (descending or ascending) **Recipient** (descending or ascending)
Show lines	Set how many action log lines will be displayed in the widget.

2 Clock

Overview

In the clock widget, you may display local, server, or specified host time.

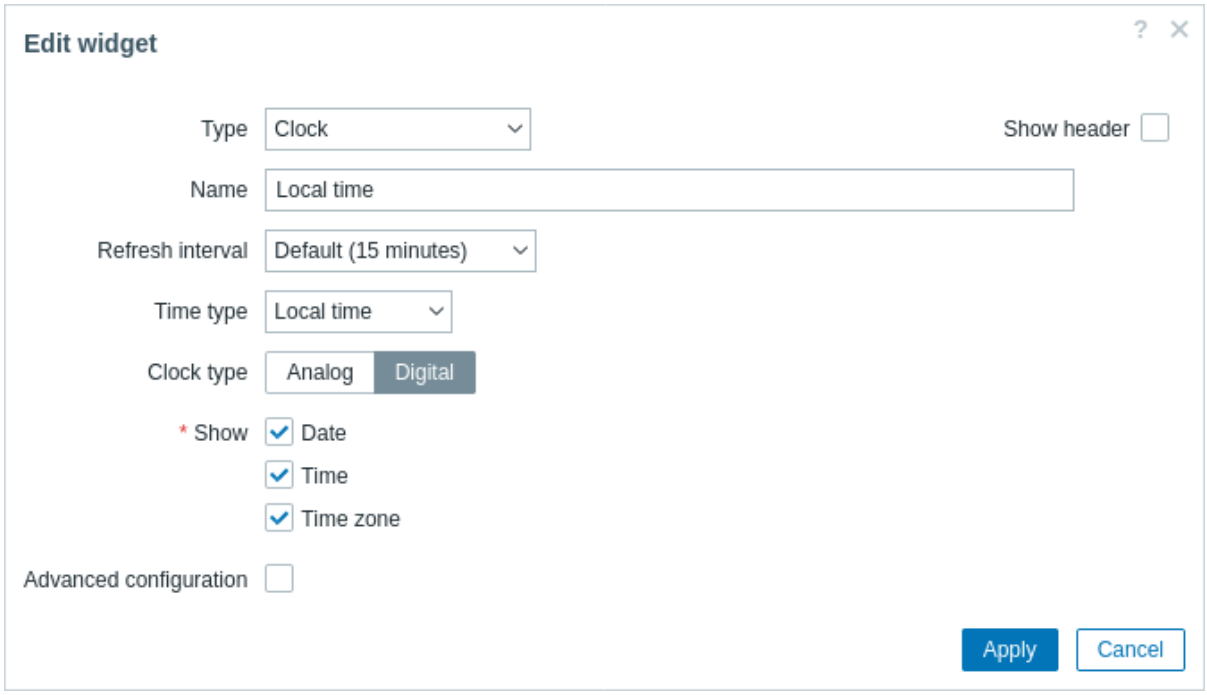
Both analog and digital clocks can be displayed:





Configuration

To configure, select *Clock* as type:



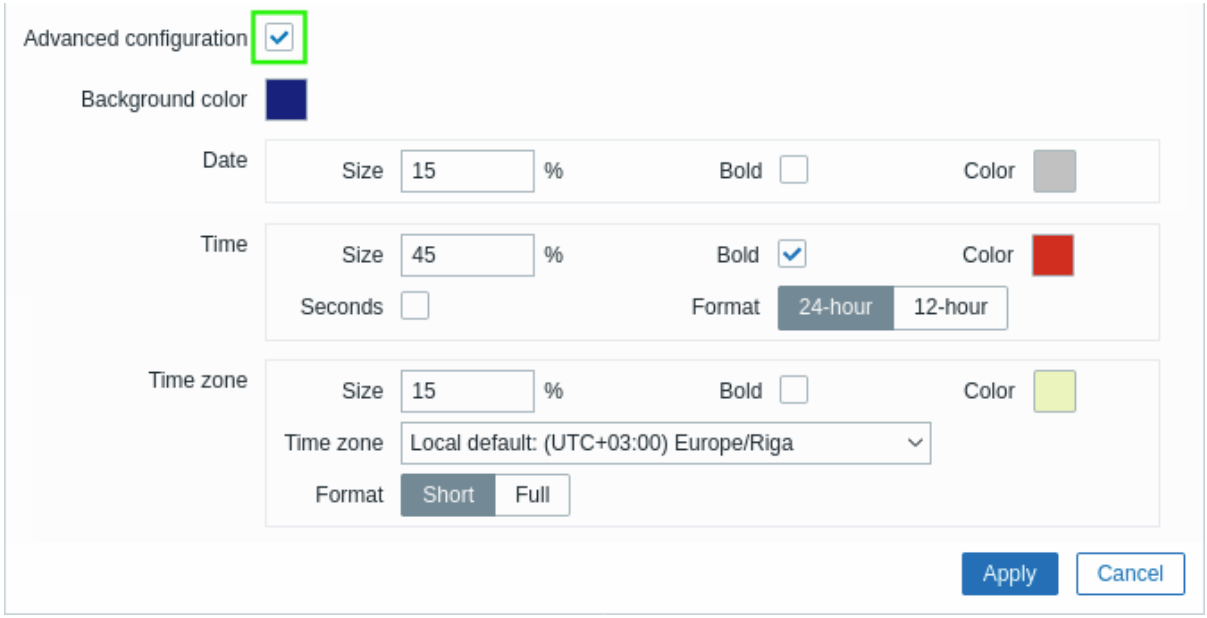
In addition to the parameters that are common for all widgets, you may set the following specific options:

Time type	Select local, server, or specified host time. Server time will be identical to the time zone set globally or for the Zabbix user.
Item	Select the item for displaying time. To display host time, use the `system.localtime[local]` item. This item must exist on the host. This field is available only when *Host time* is selected.
Clock type	Select clock type: **Analog** - analog clock **Digital** - digital clock
Show	Select information units to display in the digital clock (date, time, time zone). This field is available only if "Digital" is selected in the *Clock type* field.
Advanced configuration	Mark the checkbox to display advanced configuration options for the digital clock. This field is available only if "Digital" is selected in the *Clock type* field.

Advanced configuration

Advanced configuration options become available if the *Advanced configuration* checkbox is marked (see screenshot) and only for those elements that are selected in the *Show* field (see above).

Additionally, advanced configuration allows to change the background color for the whole widget.



Background color	Select the background color from the color picker. D stands for default color (depends on the frontend theme). To return to the default value, click the *Use default* button in the color picker.
Date	
Size	Enter font size height for the date (in percent relative to total widget height).
Bold	Mark the checkbox to display date in bold type.
Color	Select the date color from the color picker. D stands for default color (depends on the frontend theme). To return to the default value, click the *Use default* button in the color picker.
Time	
Size	Enter font size height for the time (in percent relative to total widget height).
Bold	Mark the checkbox to display time in bold type.
Color	Select the time color from the color picker. D stands for default color (depends on the frontend theme). To return to the default value, click the *Use default* button in the color picker.
Seconds	Mark the checkbox to display seconds. Otherwise only hours and minutes will be displayed.
Format	Select to display a 24-hour or 12-hour time.
Time zone	
Size	Enter font size height for the time zone (in percent relative to total widget height).
Bold	Mark the checkbox to display time zone in bold type.
Color	Select the time zone color from the color picker. D stands for default color (depends on the frontend theme). To return to the default value, click the *Use default* button in the color picker.
Time zone	Select the time zone.
Format	Select to display time zone in short format (e.g. `New York`) or full format (e.g. `(UTC-04:00) America/New York`).

3 Data overview

Attention:
This widget is deprecated and will be removed in the upcoming major release. Consider using the *Top hosts* widget instead.

Overview

In the data overview widget, you can display the latest data for a group of hosts.

The color of problem items is based on the problem severity color, which can be adjusted in the problem update screen.

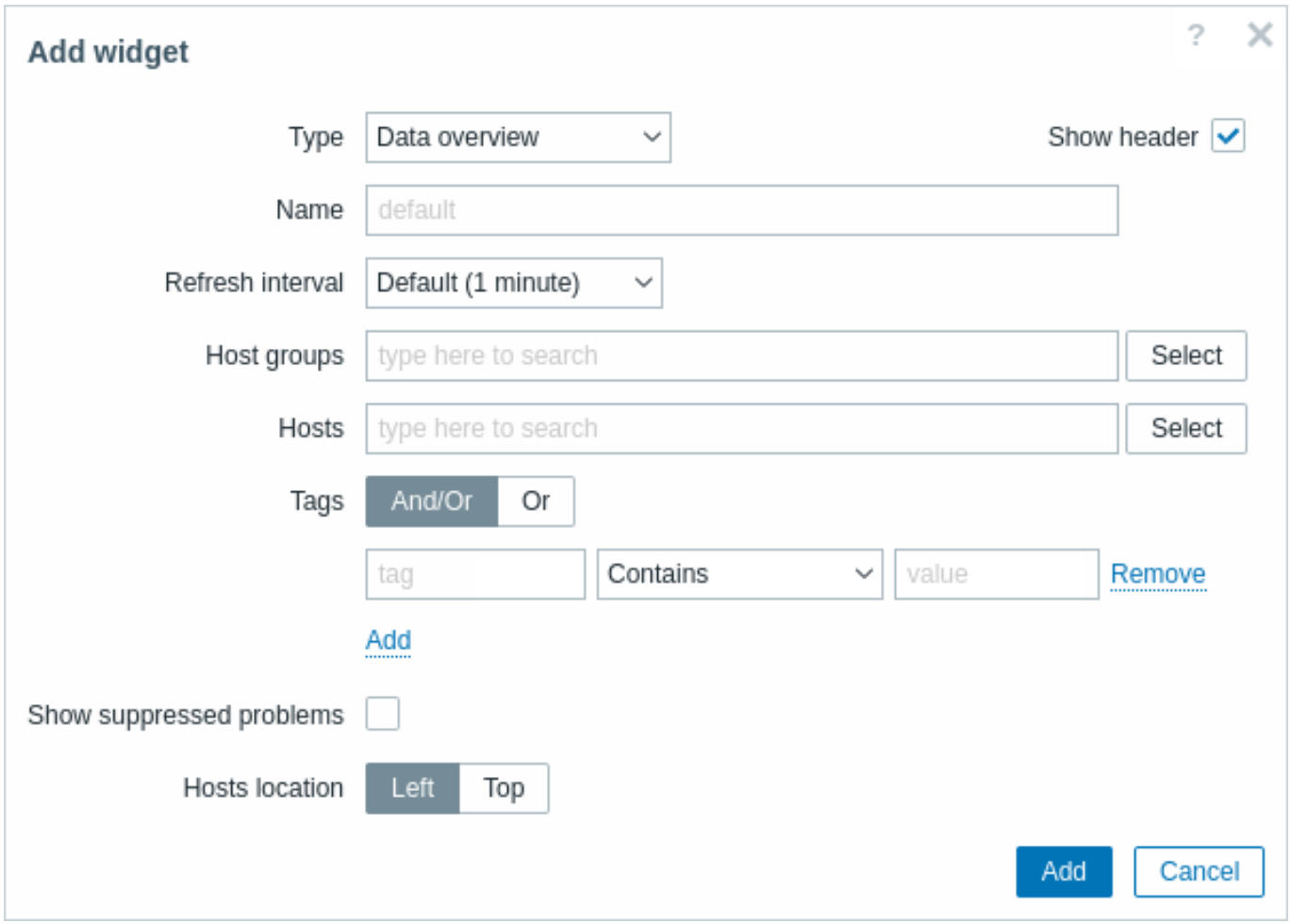
By default, only values that fall within the last 24 hours are displayed. This limit has been introduced with the aim of improving initial loading times for large pages of latest data. This limit is configurable in *Administration* → *General* → *GUI*, using the *Max history display period* option.

Clicking on a piece of data offers links to some predefined graphs or latest values.

Note that 50 records are displayed by default (configurable in *Administration* → *General* → *GUI*, using the *Max number of columns and rows in overview tables* option). If more records exist than are configured to display, a message is displayed at the bottom of the table, asking to provide more specific filtering criteria. There is no pagination. Note that this limit is applied first, before any further filtering of data, for example, by tags.

Configuration

To configure, select *Data overview* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

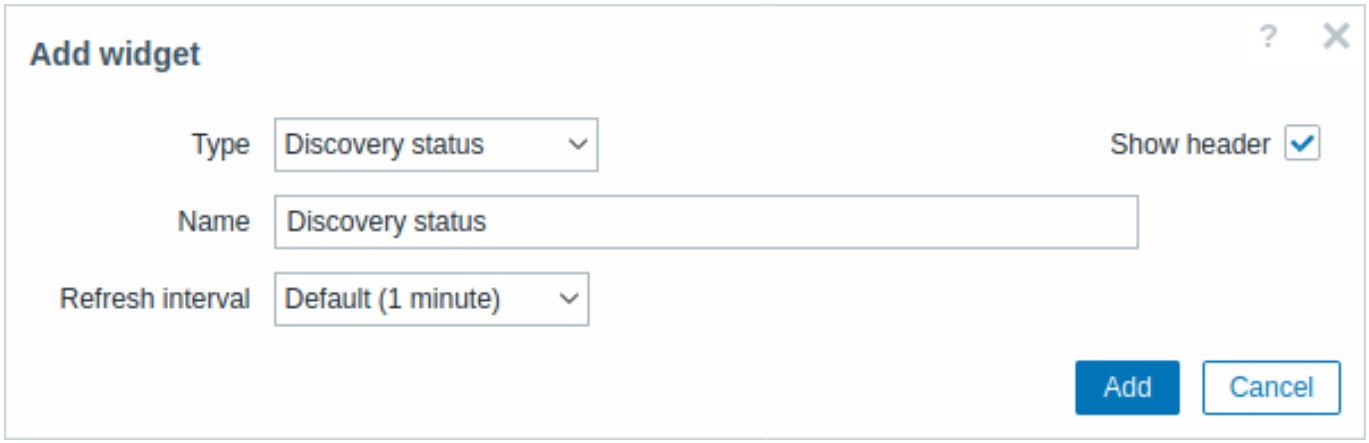
Host groups	Select host groups. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Scroll down to select. Click on 'x' to remove the selected.
Hosts	Select hosts. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. Scroll down to select. Click on 'x' to remove the selected.
Tags	Specify tags to limit the number of item data displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met

Show suppressed problems	Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance.
Hosts location	Select host location - left or top.

4 Discovery status

Overview

This widget displays a status summary of the active network discovery rules.



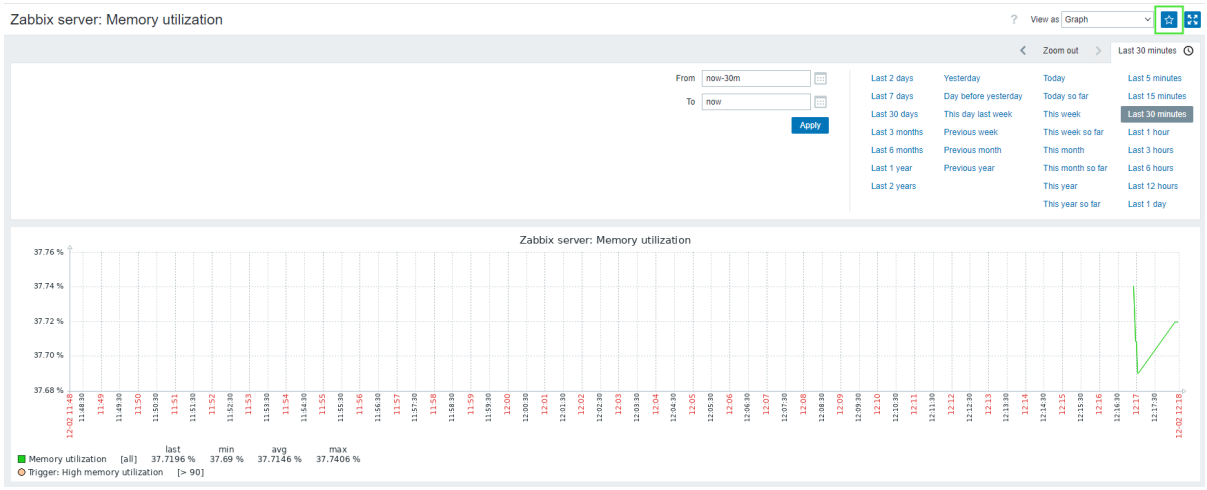
All configuration parameters are common for all widgets.

5 Favorite graphs

Overview

This widget contains shortcuts to the most needed graphs, sorted alphabetically.

The list of shortcuts is populated when you view a graph in Monitoring -> Latest data -> Graphs, and then click on its *Add to favorites* button.



All configuration parameters are common for all widgets.

6 Favorite maps

Overview

This widget contains shortcuts to the most needed maps, sorted alphabetically.

The list of shortcuts is populated when you view a map and then click on its *Add to favorites* button.

All configuration parameters are common for all widgets.

7 Geomap

Overview

Geomap widget displays hosts as markers on a geographical map using open-source JavaScript interactive maps library Leaflet.

Note:
Zabbix offers multiple predefined map tile service providers and an option to add a custom tile service provider or even host tiles themselves (configurable in the *Administration* → *General* → *Geographical maps* menu section).

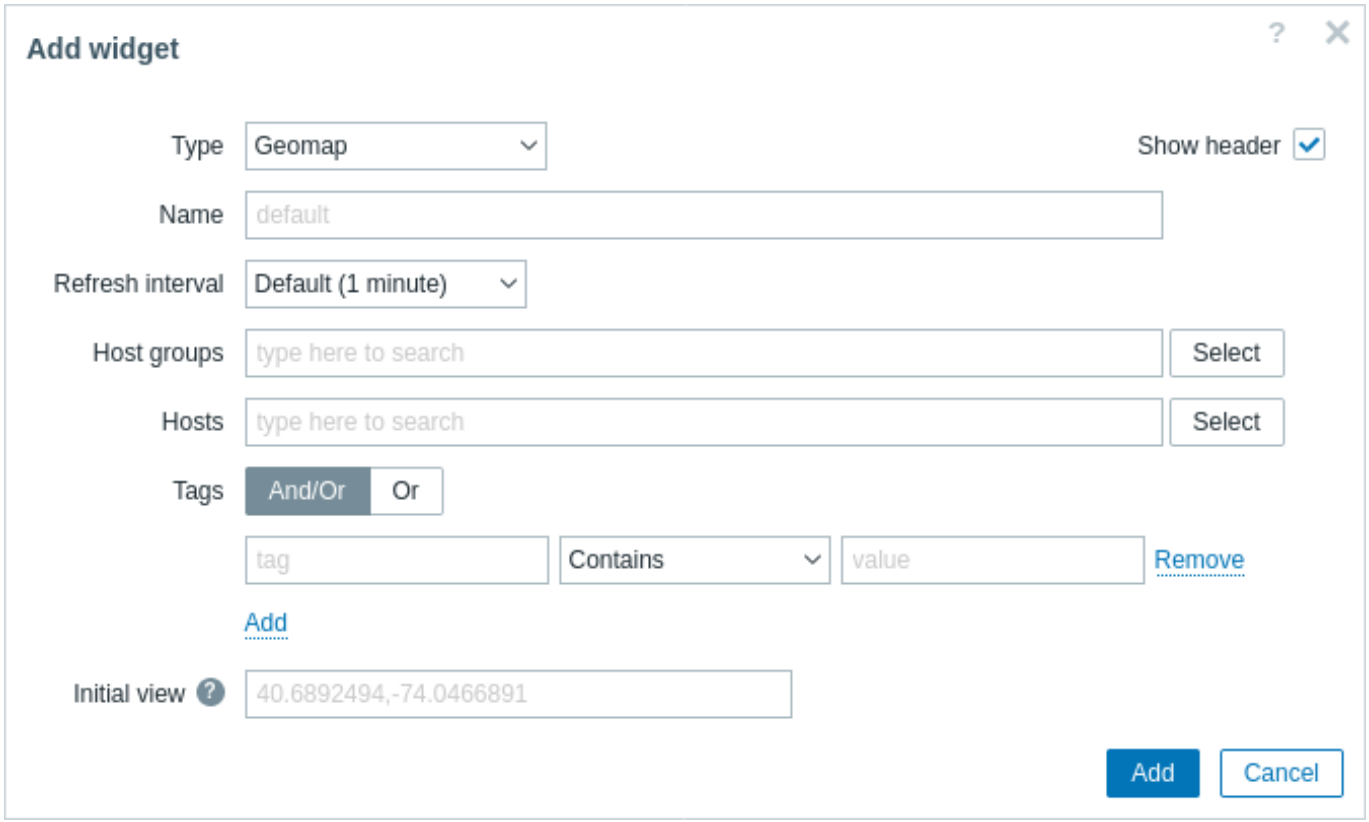
By default, the widget displays all enabled hosts with valid geographical coordinates defined in the host configuration. It is possible to configure host filtering in the widget parameters.

The valid host coordinates are:

- Latitude: from -90 to 90 (can be integer or float number)
- Longitude: from -180 to 180 (can be integer or float number)

Configuration

To add the widget, select *Geomap* as type.



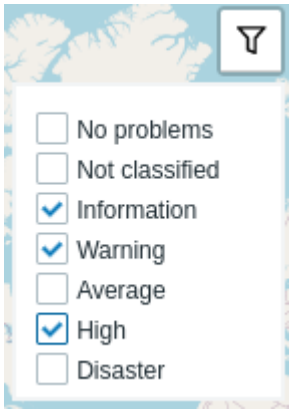
In addition to the parameters that are common for all widgets, you may set the following specific options:

Host groups	Select host groups to be displayed on the map. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Scroll down to select. Click on 'x' to remove selected groups. If nothing is selected in both *Host groups* and *Hosts* fields, all hosts with valid coordinates will be displayed.

Hosts	Select hosts to be displayed all the map. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. Scroll down to select. Click on 'x' to remove selected hosts. If nothing is selected in both *Host groups* and *Hosts* fields, all hosts with valid coordinates will be displayed.
Tags	Specify tags to limit the number of hosts displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met
Initial view	Comma-separated center coordinates and an optional zoom level to display when the widget is initially loaded in the format `<latitude>,<longitude>,<zoom>` If initial zoom is specified, the Geomap widget is loaded at the given zoom level. Otherwise, initial zoom is calculated as half of the max zoom for the particular tile provider. The initial view is ignored if the default view is set (see below). Examples: => 40.6892494,-74.0466891,14 => 40.6892494,-122.0466891

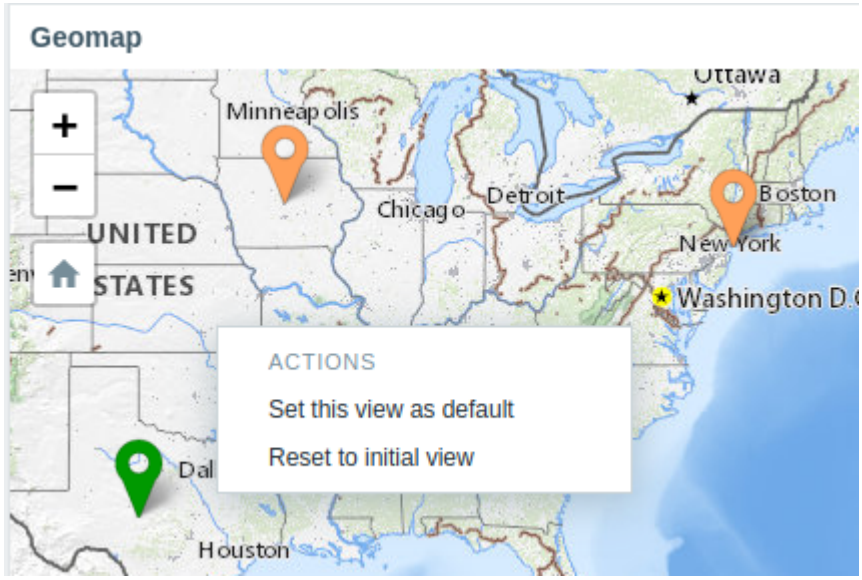
Host markers displayed on the map have the color of the host's most serious problem and green color if a host has no problems. Clicking on a host marker allows viewing the host's visible name and the number of unresolved problems grouped by severity. Clicking on the visible name will open host menu.

Hosts displayed on the map can be filtered by problem severity. Press on the filter icon in the widget's upper right corner and mark the required severities.



It is possible to zoom in and out the map by using the plus and minus buttons in the widget's upper left corner or by using the mouse scroll wheel or touchpad. To set the current view as default, right-click anywhere on the map and select *Set this view as default*. This setting will override *Initial view* widget parameter for the current user. To undo this action, right-click anywhere on the map again and select *Reset the initial view*.

When *Initial view* or *Default view* is set, you can return to this view at any time by pressing on the home icon on the left.



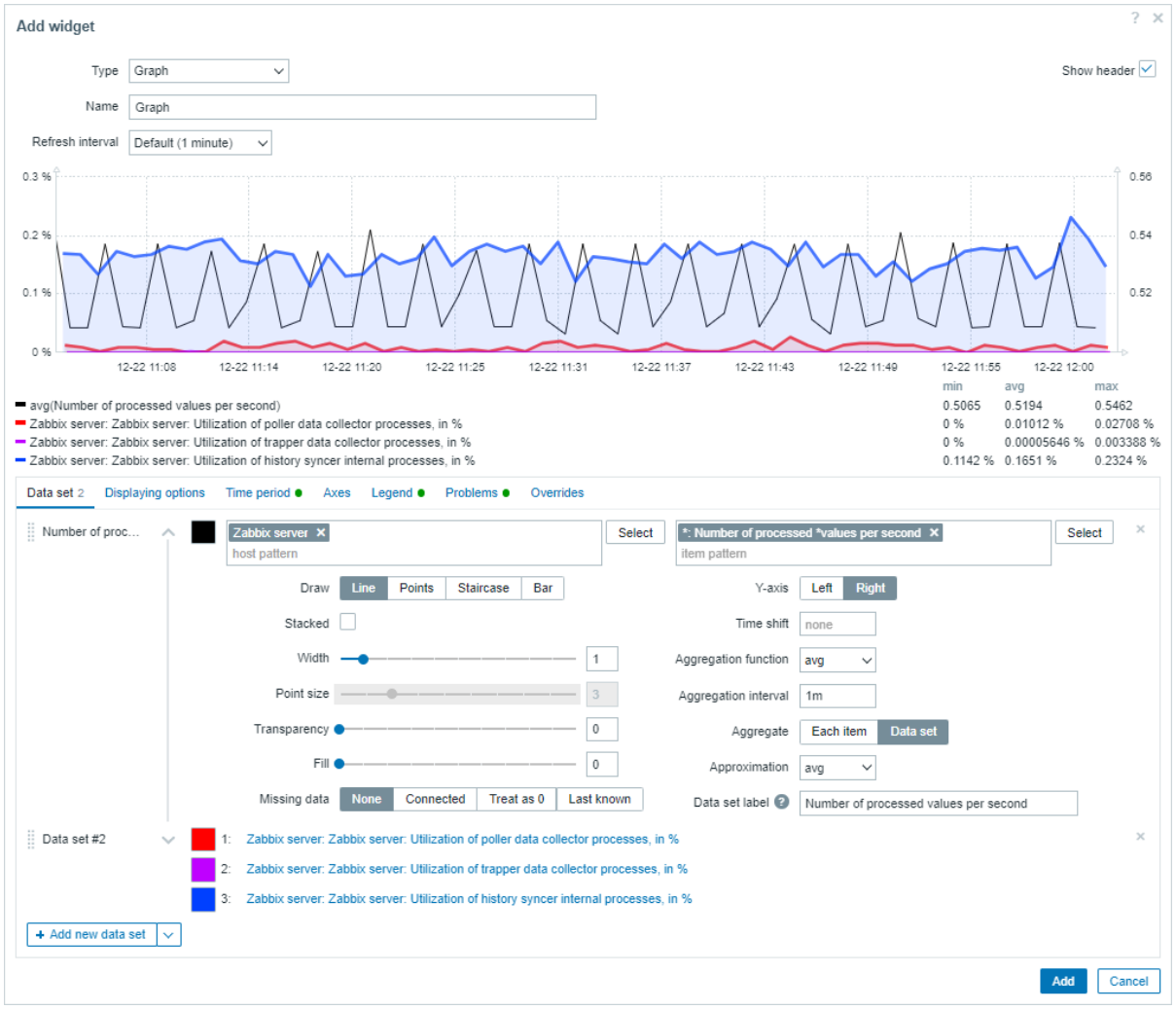
8 Graph

Overview

The graph widget provides a modern and versatile way of visualizing data collected by Zabbix using a vector image drawing technique. This graph widget is supported since Zabbix 4.0. Note that the graph widget supported before Zabbix 4.0 can still be used as Graph (classic). See also Adding widgets section on Dashboards page for more details.

Configuration

To configure, select *Graph* as type:



Data set

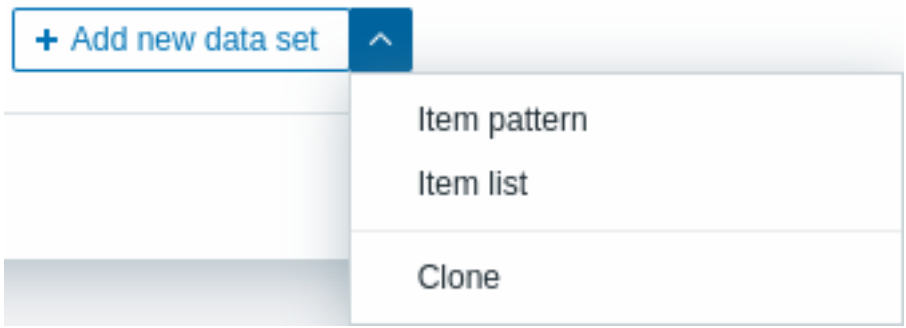
The **Data set** tab allows to add data sets and define their visual representation:

Data set	Enter the host and item patterns; data of items that match the entered patterns is displayed on the graph; up to 50 items may be displayed. Host pattern and item pattern fields are mandatory. Wildcard patterns may be used for selection. For example, ∗ will return results that match zero or more characters. To specify a wildcard pattern, just enter the string manually and press *Enter*. The wildcard symbol is always interpreted. Therefore, it is not possible to add, for example, an item named *item** individually, if there are other matching items (e.g. item2, item3). See data set configuration details. **Alternatively** to specifying item patterns, you may select a list of items, if the data set has been added with the *Item list* option (see the description of the *Add new data set* button).
Draw	Choose the draw type of the metric. Possible draw types are *Line* (set by default), *Points*, *Staircase* and *Bar*. Note that if there's only one data point in the line/staircase graph it is drawn as a point regardless of the draw type. The point size is calculated from the line width, but it cannot be smaller than 3 pixels, even if the line width is less.
Stacked	Mark the checkbox to display data as stacked (filled areas displayed). This option is disabled when *Points* draw type is selected.
Width	Set the line width. This option is available when *Line* or *Staircase* draw type is selected.
Point size	Set the point size. This option is available when *Points* draw type is selected.
Transparency	Set the transparency level.
Fill	Set the fill level. This option is available when *Line* or *Staircase* draw type is selected.
Missing data	Select the option for displaying missing data: **None** - the gap is left empty **Connected** - two border values are connected **Treat as 0** - the missing data is displayed as 0 values **Last known** - the missing data is displayed with the same value as the last known value Not applicable for the *Points* and *Bar* draw type.
Y-axis	Select the side of the graph where the Y-axis will be displayed.
Time shift	Specify time shift if required. You may use time suffixes in this field. Negative values are allowed.
Aggregation function	Specify which aggregation function to use: **min** - display the smallest value **max** - display the largest value **avg** - display the average value **sum** - display the sum of values **count** - display the count of values **first** - display the first value **last** - display the last value **none** - display all values (no aggregation) Aggregation allows to display an aggregated value for the chosen interval (5 minutes, an hour, a day), instead of all values. See also: Aggregation in graphs.
Aggregation interval	Specify the interval for aggregating values. You may use time suffixes in this field. A numeric value without a suffix will be regarded as seconds.
Aggregate	Specify whether to aggregate: **Each item** - each item in the dataset will be aggregated and displayed separately. **Data set** - all dataset items will be aggregated and displayed as one value.
Approximation	Specify what value to display when more than one value exists per vertical graph pixel: **all** - display the smallest, the largest and the average values **min** - display the smallest value **max** - display the largest value **avg** - display the average value This setting is useful when displaying a graph for a large time period with frequent update interval (such as one year of values collected every 10 minutes).

Data set label	Specify the data set label that is displayed in graph *Data set* configuration and in graph *Legend* (for aggregated data sets). All data sets are numbered including those with a specified *Data set label*. If no label is specified, the data set will be labeled automatically according to its number (e.g. "Data set #2", "Data set #3", etc.). Data set numbering is recalculated after reordering/dragging data sets. Data set labels that are too long will be shortened to fit where displayed (e.g. "Number of proc...").

Existing data sets are displayed in a list. You may:

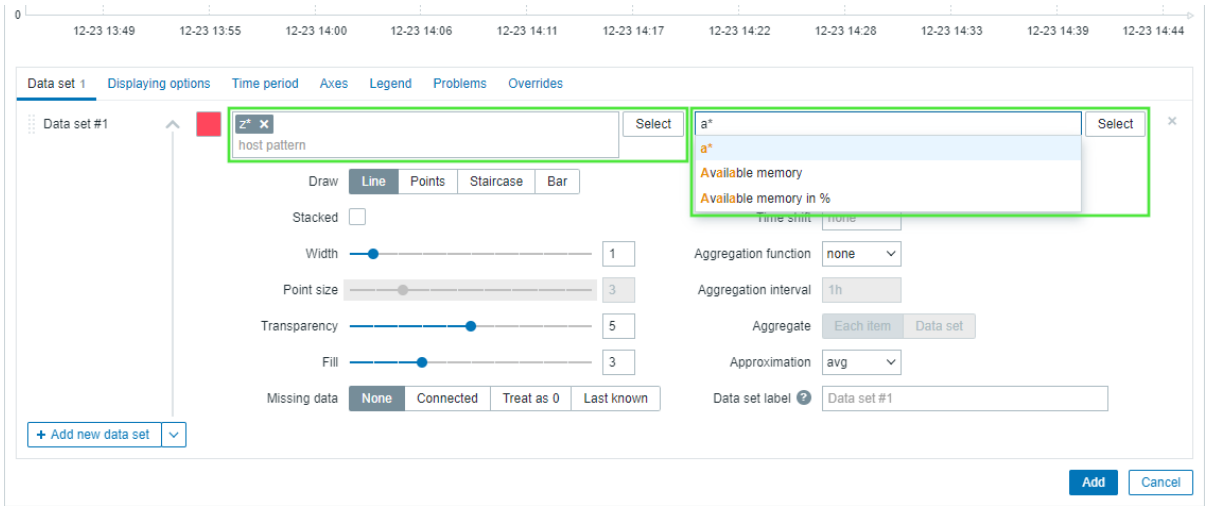
- - click on the move icon and drag a data set to a new place in the list
- - click on the expand icon to expand data set details. When expanded, this icon turns into a collapse icon.
- - click on the color icon to change the base color, either from the color picker or manually. The base color is used to calculate different colors for each item of the data set.
- + Add new data set - click on this button to add an empty data set allowing to select the host/item pattern.
  - If you click on the downward pointing icon next to the *Add new data set* button, a drop-down menu appears allowing to add a new data set with item pattern/item list or by cloning the currently open data set. If all data sets are collapsed, the *Clone* option is not available.



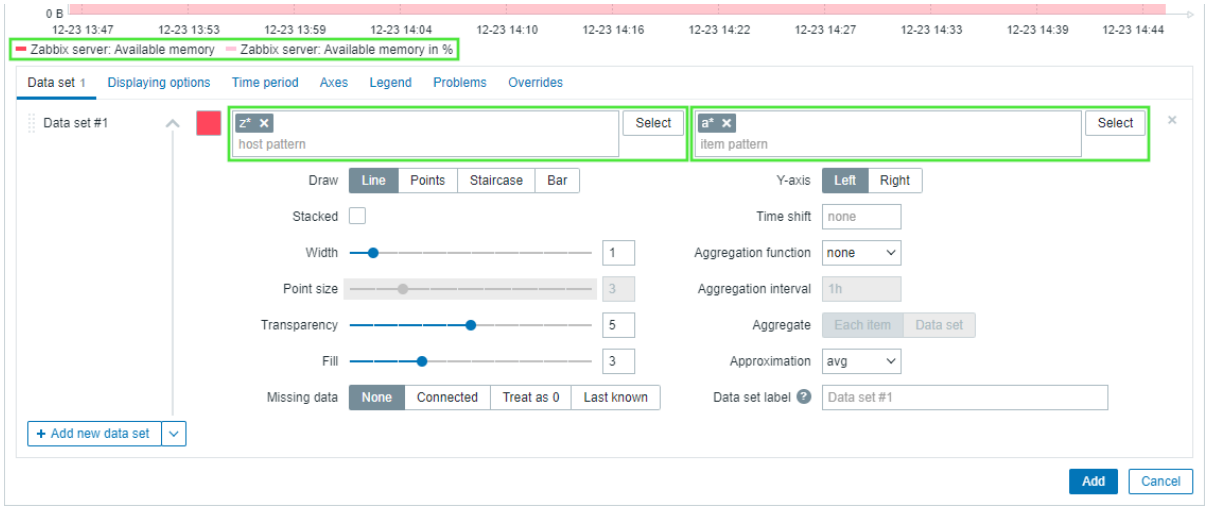
Data set configuration details

The *host pattern* and *item pattern* fields in the *Data set* tab both recognize full names or patterns containing a wildcard symbol (*). This functionality enables to select all the host names and item names containing the selected pattern. Most importantly, while typing the item name or item pattern in the *item pattern* field, only the items that belong to the selected host name(s) are displayed on a drop-down list. For example, having typed a pattern **z*** in the *host pattern* field, the drop-down list displays all the host names containing this pattern: z*, Zabbix server, and Zabbix proxy. After pressing *Enter*, this pattern is accepted and is displayed as **z***. Similarly, the pattern can be created in the *item pattern* field. For example, having typed the pattern **a*** in the *item pattern* field, the drop-down list displays all the item names containing this pattern: a*, Available memory, Available memory in %.

See the image of the *Data set* tab below.

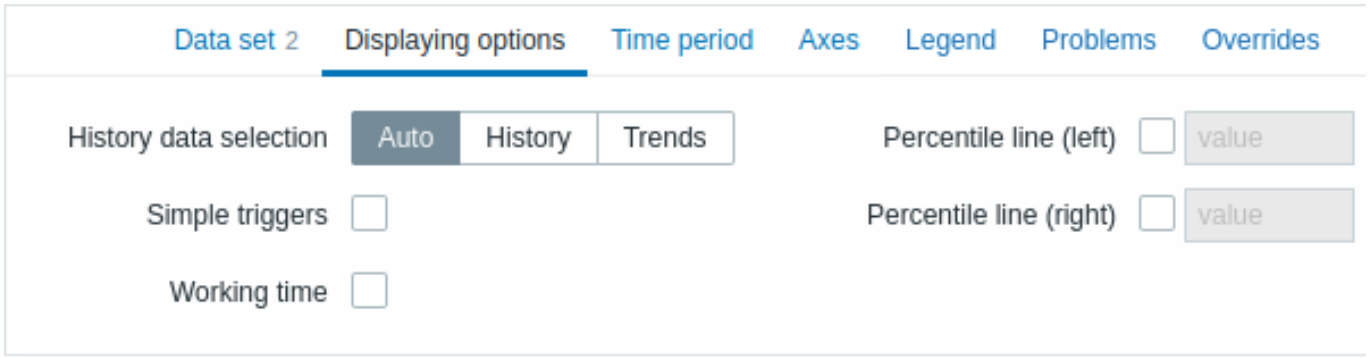


After pressing *Enter*, this pattern is accepted and is displayed as **a***, and all the selected items that belong to the selected host name(s) are displayed above the *Data set* tab. See the image of the *Data set* tab below.



Displaying options

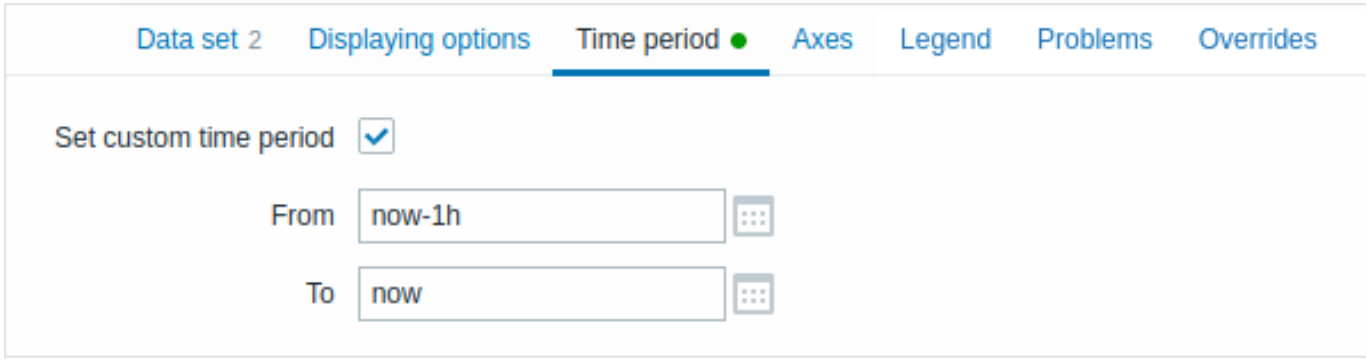
The **Displaying options** tab allows to define history data selection:



History data selection	Set the source of graph data: **Auto** - data are sourced according to the classic graph algorithm (default) **History** - data from history **Trends** - data from trends
Simple triggers	Mark the checkbox to show the trigger thresholds for simple triggers. The thresholds will be drawn as dashed lines using the trigger severity color. A simple trigger is a trigger with one function (only `last`, `max`, `min`, `avg`) for one item in the expression. A maximum of three triggers can be drawn. Note that the trigger has to be within the drawn range to be visible.
Working time	Mark the checkbox to show working time on the graph. Working time (working days) is displayed in graphs as a white background, while non-working time is displayed in gray (with the *Original blue* default frontend theme).
Percentile line (left)	Mark the checkbox and enter the percentile value to show the specified percentile as a line on the left Y-axis of the graph. If, for example, a 95% percentile is set, then the percentile line will be at the level where 95 percent of the values fall under.
Percentile line (right)	Mark the checkbox and enter the percentile value to show the specified percentile as a line on the right Y-axis of the graph. If, for example, a 95% percentile is set, then the percentile line will be at the level where 95 percent of the values fall under.

Time period

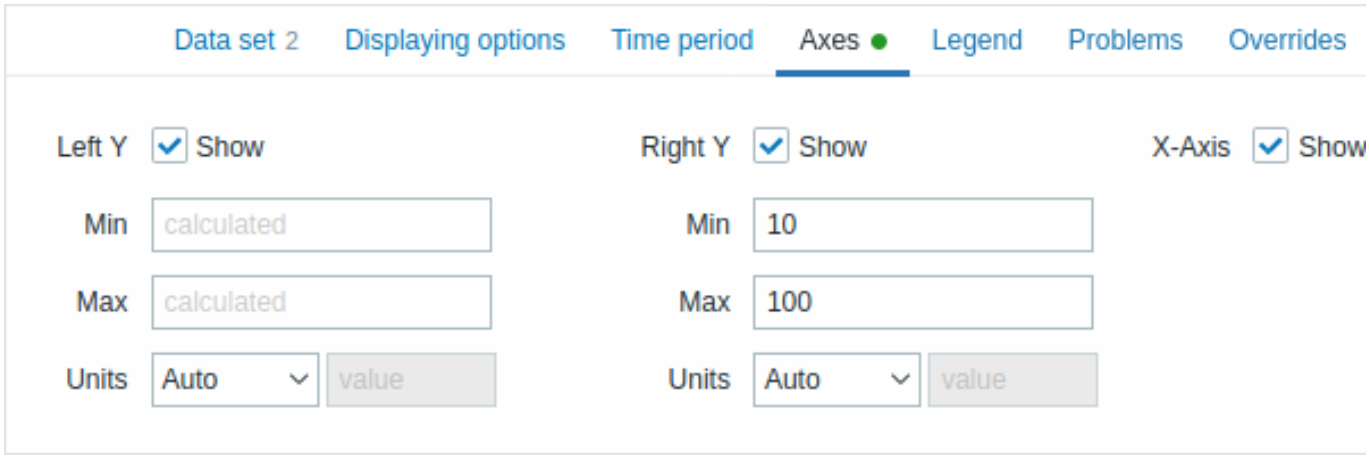
The **Time period** tab allows to set a custom time period:



Set custom time period	Mark this checkbox to set the custom time period for the graph (unmarked by default).
From	Set the start time of the custom time period for the graph.
To	Set the end time of the custom time period for the graph.

Axes

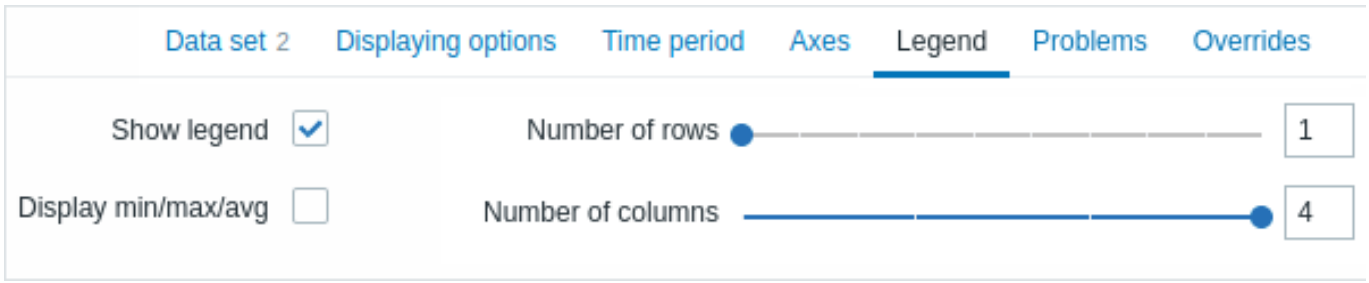
The **Axes** tab allows to customize how axes are displayed:



Left Y	Mark this checkbox to make left Y-axis visible. The checkbox may be disabled if unselected either in *Data set* or in *Overrides* tab.
Right Y	Mark this checkbox to make right Y-axis visible. The checkbox may be disabled if unselected either in *Data set* or in *Overrides* tab.
X-Axis	Unmark this checkbox to hide X-axis (marked by default).
Min	Set the minimum value of the corresponding axis. Visible range minimum value of Y-axis is specified.
Max	Set the maximum value of the corresponding axis. Visible range maximum value of Y-axis is specified.
Units	Choose the unit for the graph axis values from the dropdown. If the *Auto* option is chosen axis values are displayed using units of the first item of the corresponding axis. *Static* option allows you to assign the corresponding axis' custom name. If the *Static* option is chosen and the *value* input field left blank the corresponding axis' name will only consist of a numeric value.

Legend

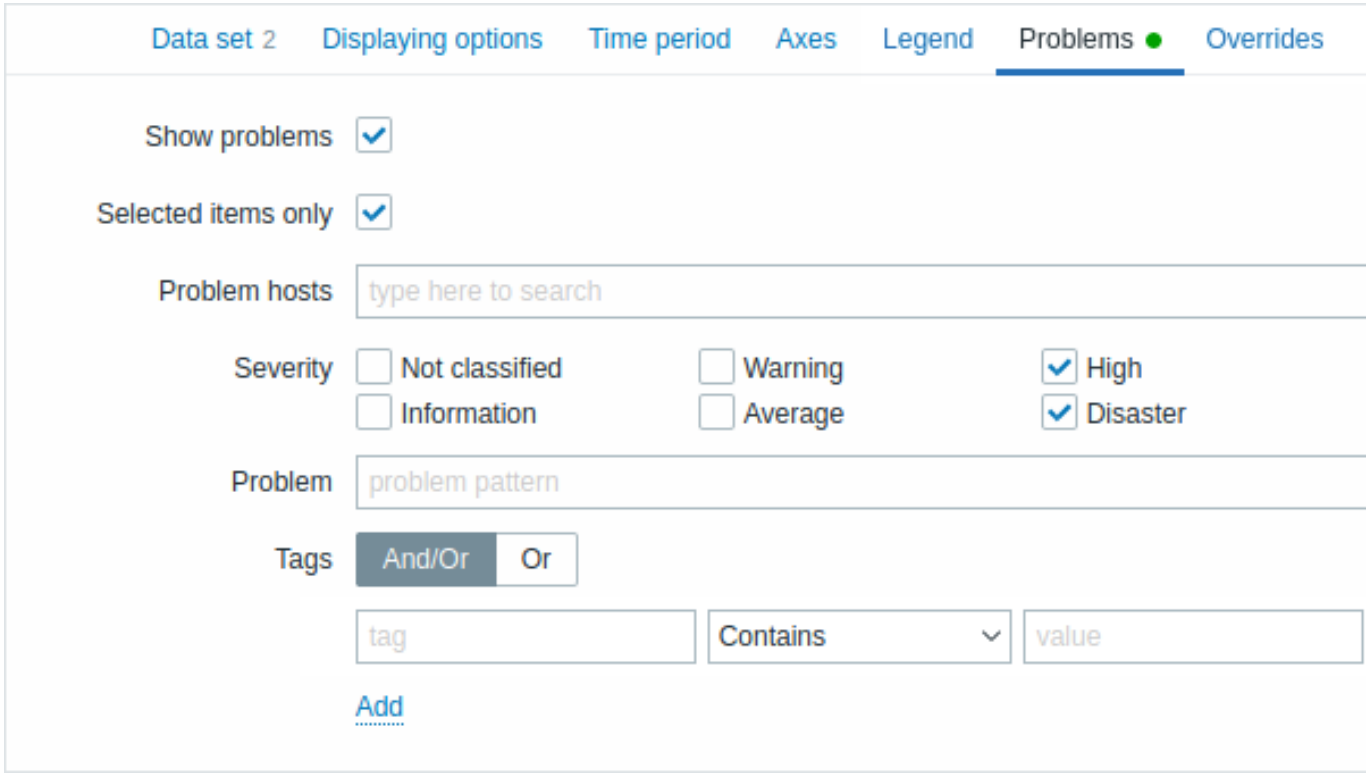
The **Legend** tab allows to customize the graph legend:



Show legend	Unmark this checkbox to hide the legend on the graph (marked by default).
Display min/max/avg	Mark this checkbox to display the minimum, maximum and average values of the item in the legend.
Number of rows	Set the number of legend rows to be displayed.
Number of columns	Set the number of legend columns to be displayed.

Problems

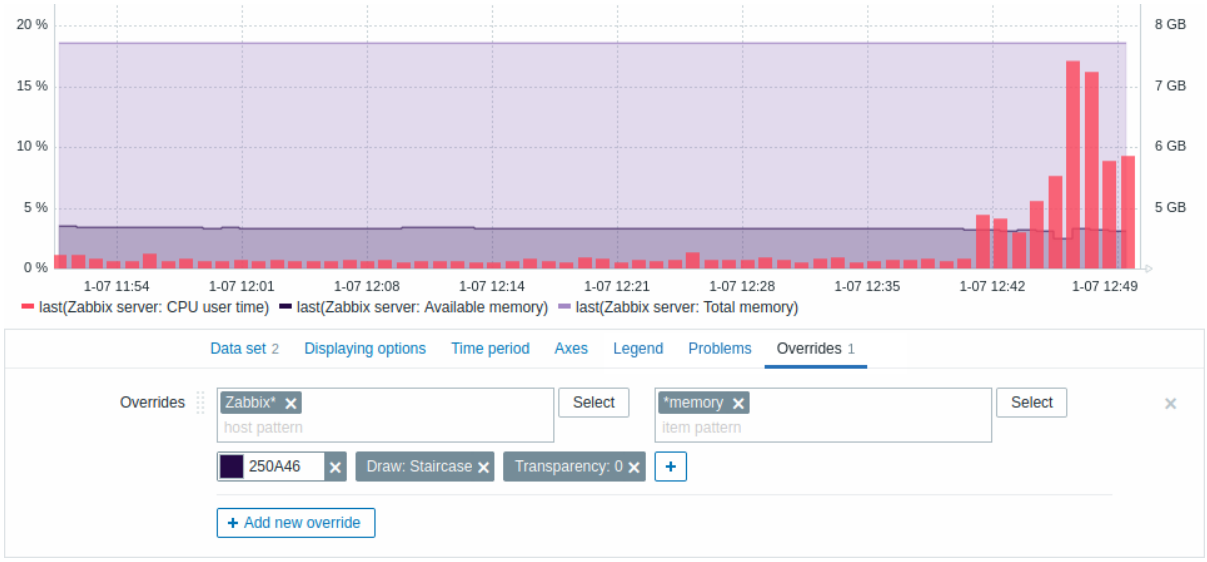
The **Problems** tab allows to customize the problem display:



Show problems	Mark this checkbox to enable problem displaying on the graph (unmarked, i.e. disabled by default).
Selected items only	Mark this checkbox to include problems for the selected items only to be displayed on the graph.
Problem hosts	Select the problem hosts to be displayed on the graph. Wildcard patterns may be used (for example, * will return results that match zero or more characters). To specify a wildcard pattern, just enter the string manually and press *Enter*. While you are typing, note how all matching hosts are displayed in the dropdown.
Severity	Mark problem severities to filter problems to be displayed on the graph. If no severities are marked, all problems will be displayed.
Problem	Specify the problem's name to be displayed on the graph.
Tags	Specify problem tags to limit the number of problems displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met

Overrides

The **Overrides** tab allows to add custom overrides for data sets:

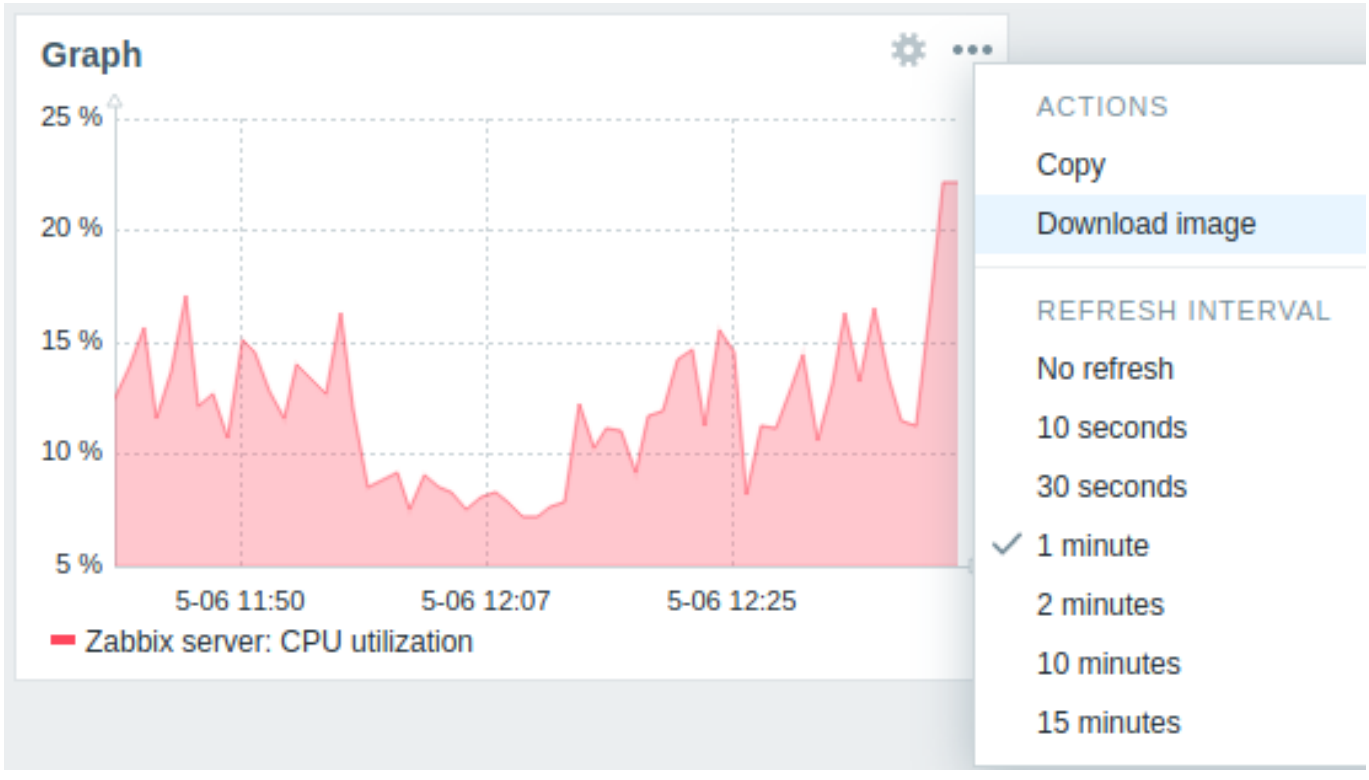


Overrides are useful when several items are selected for a data set using the * wildcard and you want to change how the items are displayed by default (e.g. default base color or any other property).

Existing overrides (if any) are displayed in a list. To add a new override:

- Click on the + Add new override button
- Select hosts and items for the override. Alternatively, you may enter host and item patterns. Wildcard patterns may be used (for example, * will return results that match zero or more characters). To specify a wildcard pattern, just enter the string manually and press *Enter*. While you are typing, note how all matching hosts are displayed in the dropdown. The wildcard symbol is always interpreted, therefore it is not possible to add, for example, an item named "item*" individually if there are other matching items (e.g. item2, item3). Host pattern and item pattern fields are mandatory.
- Click on +, to select override parameters. At least one override parameter should be selected. For parameter descriptions, see the *Data set* tab above.

Information displayed by the graph widget can be downloaded as a .png image using the widget menu:



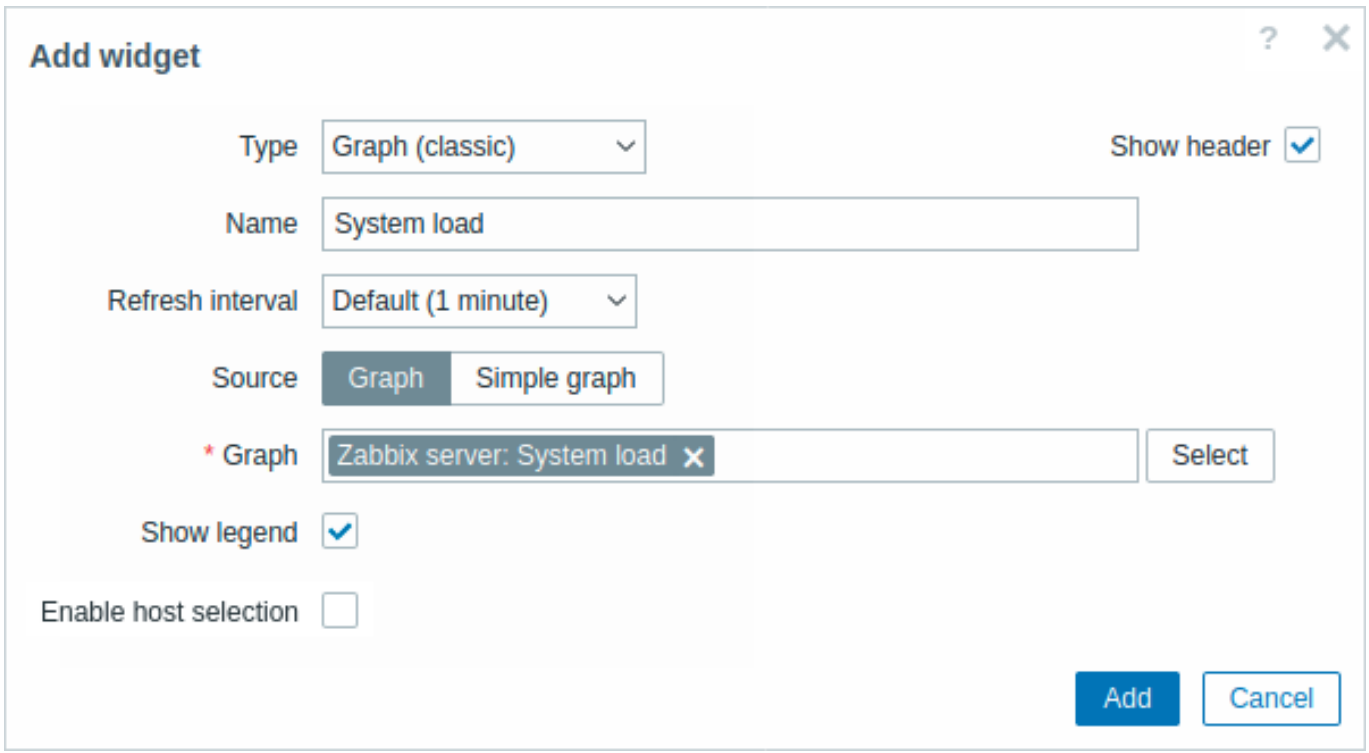
A screenshot of the widget will be saved to the Downloads folder.

Overview

In the classic graph widget, you can display a single custom graph or simple graph.

Configuration

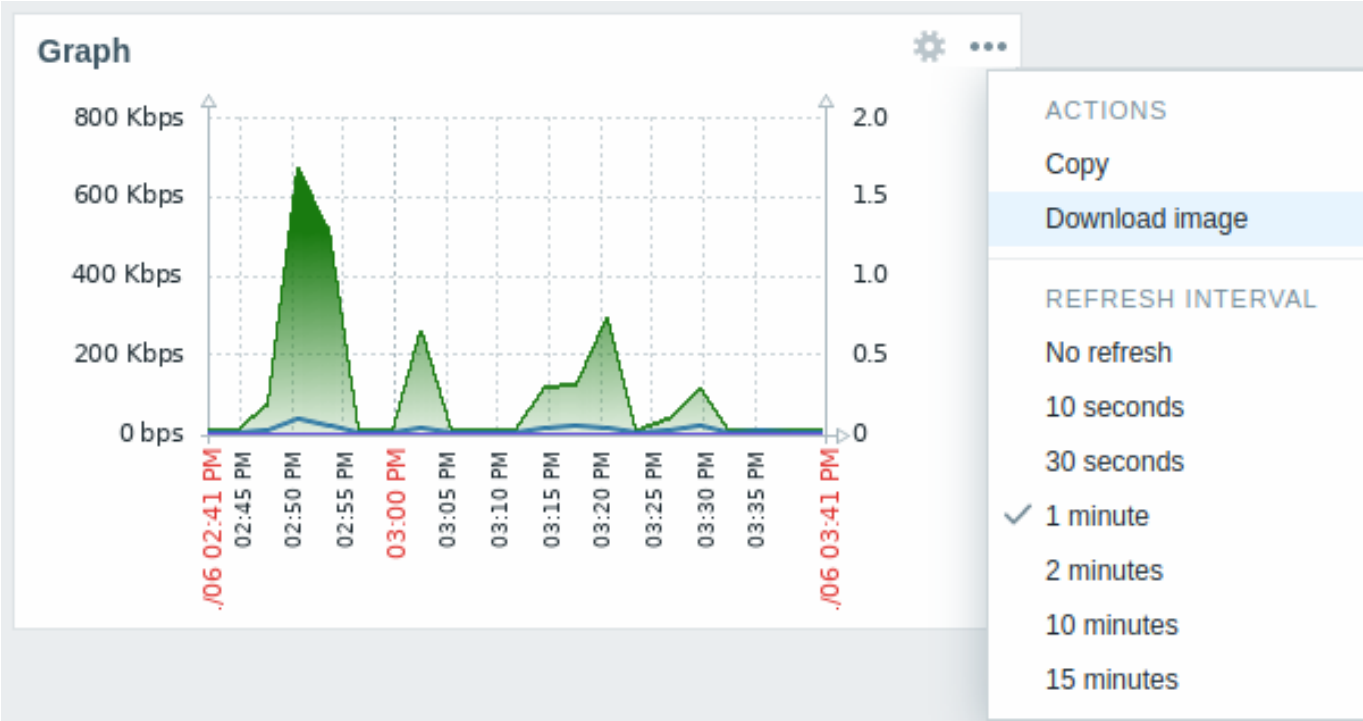
To configure, select *Graph (classic)* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

Source	Select graph type: **Graph** - custom graph **Simple graph** - simple graph
Graph	Select the custom graph to display. This option is available if 'Graph' is selected as *Source*.
Item	Select the item to display in a simple graph. This option is available if 'Simple graph' is selected as *Source*.
Show legend	Unmark this checkbox to hide the legend on the graph (marked by default).
Enable host selection	Set graph to display different data depending on the selected host.

Information displayed by the classic graph widget can be downloaded as .png image using the widget menu:



A screenshot of the widget will be saved to the Downloads folder.

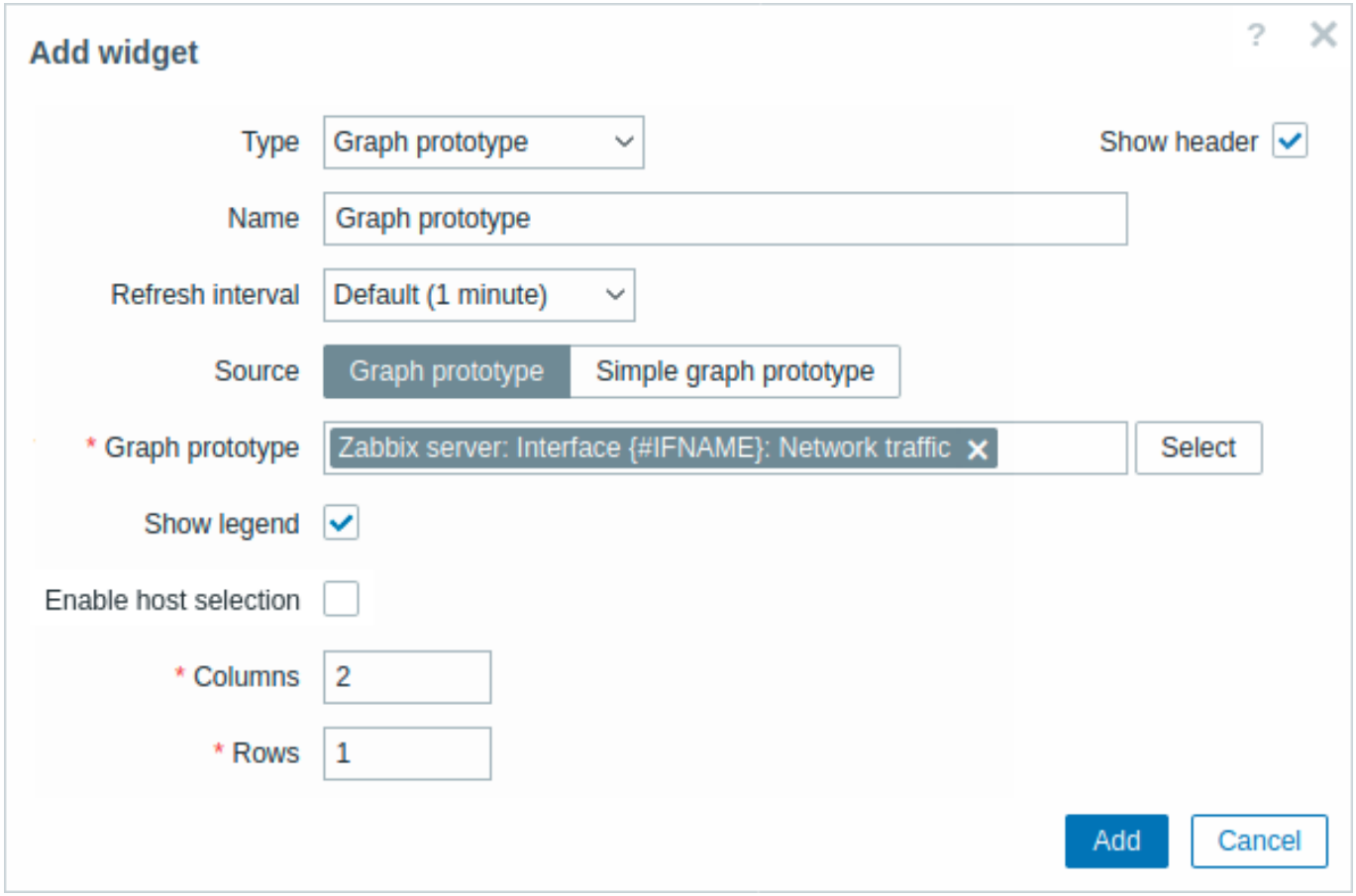
10 Graph prototype

Overview

In the graph prototype widget, you can display a grid of graphs created from either a graph prototype or an item prototype by low-level discovery.

Configuration

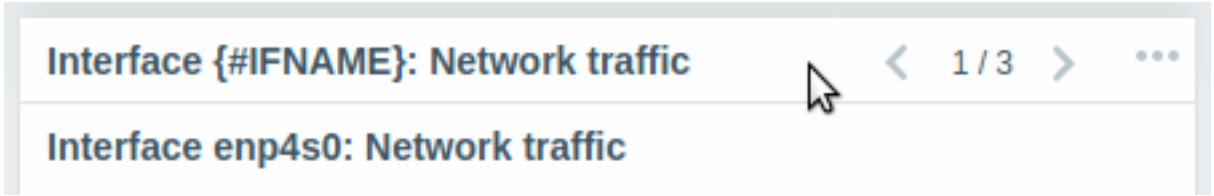
To configure, select *Graph prototype* as widget type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

Source	Select source: either a **Graph prototype** or a **Simple graph prototype**.
Graph prototype	Select a graph prototype to display discovered graphs of the graph prototype. This option is available if 'Graph prototype' is selected as Source.
Item prototype	Select an item prototype to display simple graphs based on discovered items of an item prototype. This option is available if 'Simple graph prototype' is selected as Source.
Show legend	Mark this checkbox to show the legend on the graphs (marked by default).
Enable host selection	Set graphs to display different data depending on the selected host.
Columns	Enter the number of columns of graphs to display within a graph prototype widget.
Rows	Enter the number of rows of graphs to display within a graph prototype widget.

While the *Columns* and *Rows* settings allow fitting more than one graph in the widget, there still may be more discovered graphs than there are columns/rows in the widget. In this case paging becomes available in the widget and a slide-up header allows to switch between pages using the left and right arrows.



11 Host availability

Overview

In the host availability widget, high-level statistics about host availability are displayed in four colored columns/lines.



Horizontal display (columns).



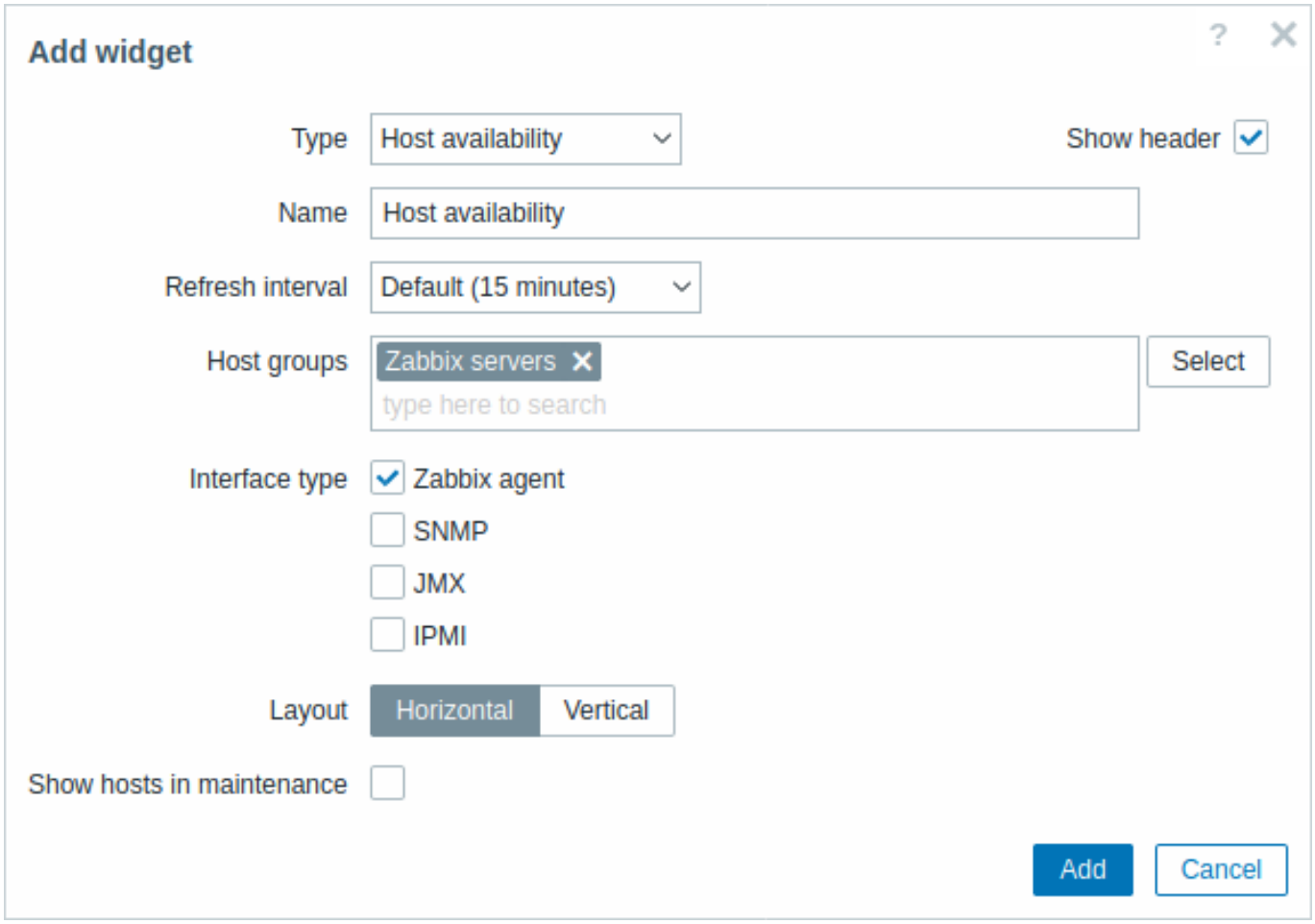
Vertical display (lines).

Host availability in each column/line is counted as follows:

- *Available* - hosts with all interfaces available
- *Not available* - hosts with at least one interface unavailable
- *Unknown* - hosts with at least one interface unknown (none unavailable)
- *Total* - total of all hosts

Configuration

To configure, select *Host availability* as type:



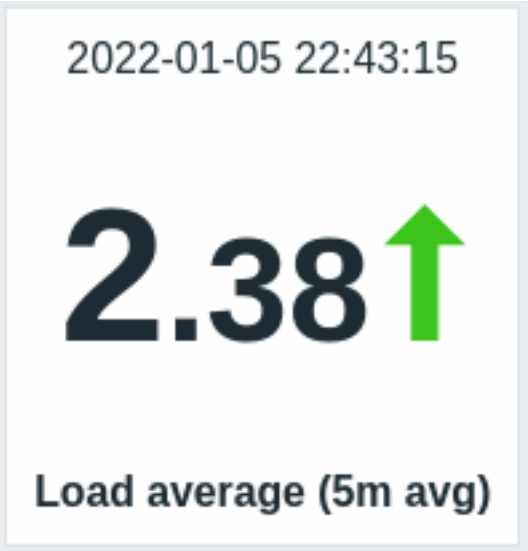
In addition to the parameters that are common for all widgets, you may set the following specific options:

Host groups	Select host group(s). This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Scroll down to select. Click on 'x' to remove the selected.
Interface type	Select which host interfaces you want to see availability data for. Availability of all interfaces is displayed by default if nothing is selected.
Layout	Select horizontal display (columns) or vertical display (lines).
Show hosts in maintenance	Include hosts that are in maintenance in the statistics.

12 Item value

Overview

This widget is useful for displaying the value of a single item prominently.



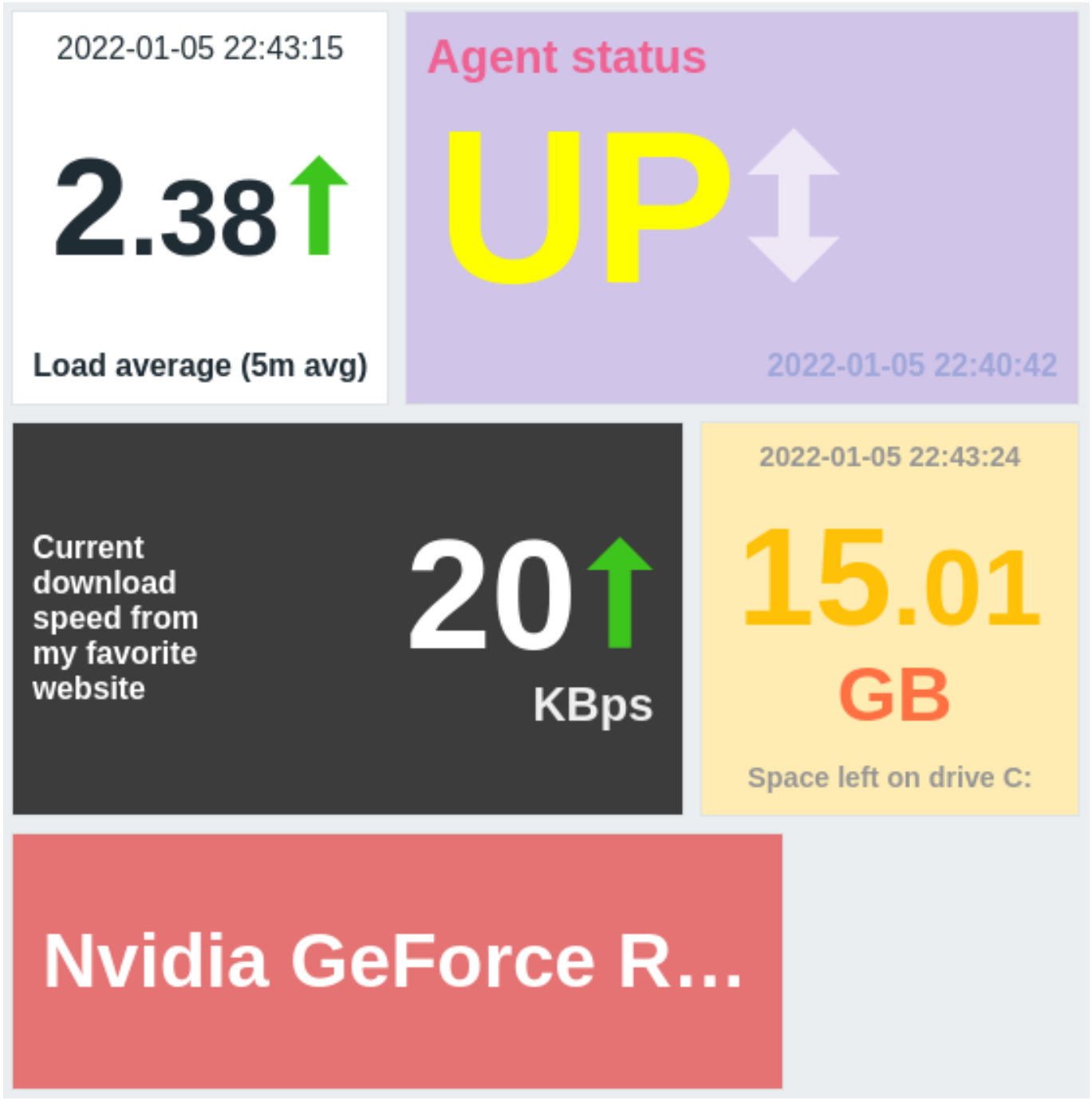
Besides the value itself, additional elements can be displayed, if desired:

- time of the metric
- item description
- change indicator for the value
- background color for the value
- item unit

The widget can display numeric and string values. String values are displayed on a single line and truncated, if needed. "No data" is displayed, if there is no value for the item.

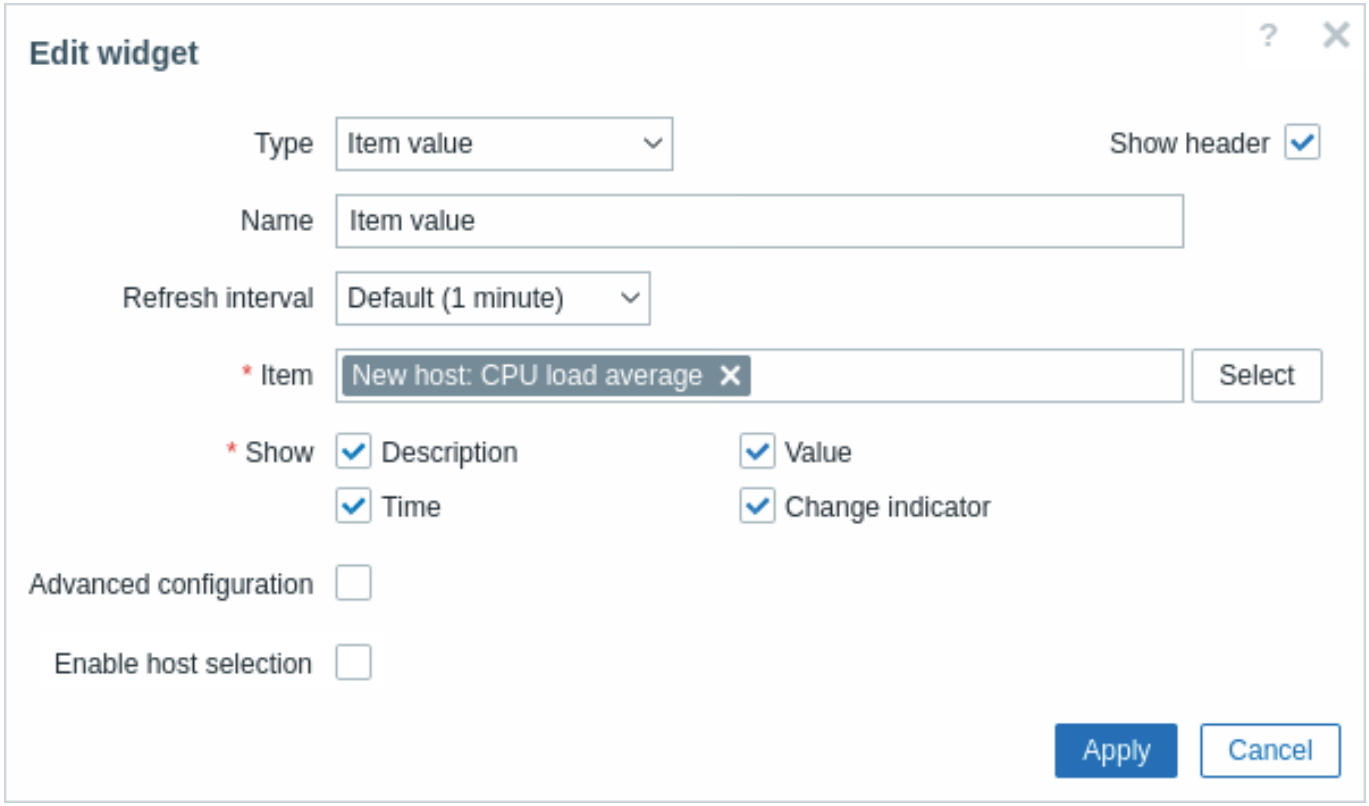
Clicking on the value leads to an ad-hoc graph for numeric items or latest data for string items.

The widget and all elements in it can be visually fine-tuned using advanced configuration options, allowing to create a wide variety of visual styles:



Configuration

To configure, select *Item value* as the widget type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

Item	Select the item.
Show	Mark the checkbox to display the respective element (description, value, time, change indicator). Unmark to hide. At least one element must be selected.
Advanced configuration	Mark the checkbox to display advanced configuration options.
Enable host selection	Mark the checkbox to display a different value depending on the selected host.

Advanced configuration

Advanced configuration options become available if the *Advanced configuration* checkbox is marked (see screenshot) and only for those elements that are selected in the *Show* field (see above).

Additionally, advanced configuration allows to change the background color - static or dynamic - for the whole widget.



Description

Description	Enter the item description. This description may override the default item name. Multiline descriptions are supported. A combination of text and supported macros is possible. {HOST.*}, {ITEM.*}, {INVENTORY.*} and user macros are supported.
Horizontal position	Select horizontal position of the item description - left, right or center.
Vertical position	Select vertical position of the item description - top, bottom or middle.
Size	Enter font size height for the item description (in percent relative to total widget height).
Bold	Mark the checkbox to display item description in bold type.
Color	Select the item description color from the color picker. D stands for default color (depends on the frontend theme). To return to the default value, click the *Use default* button in the color picker.

Value

Decimal places	Select how many decimal places will be displayed with the value. This value will affect only float items.
Size	Enter font size height for the decimal places (in percent relative to total widget height).
Horizontal position	Select horizontal position of the item value - left, right or center.
Vertical position	Select vertical position of the item value - top, bottom or middle.
Size	Enter font size height for the item value (in percent relative to total widget height). Note that the size of item value is prioritized; other elements have to concede space for the value. With the change indicator though, if the value is too large, it will be truncated to show the change indicator.
Bold	Mark the checkbox to display item value in bold type.
Color	Select the item value color from the color picker. D stands for default color (depends on the frontend theme). To return to the default value, click the *Use default* button in the color picker.
Units	
Units	Mark the checkbox to display units with the item value. If you enter a unit name, it will override the unit from item configuration.
Position	Select the item unit position - above, below, before or after the value.
Size	Enter font size height for the item unit (in percent relative to total widget height).
Bold	Mark the checkbox to display item unit in bold type.
Color	Select the item unit color from the color picker. D stands for default color (depends on the frontend theme). To return to the default value, click the *Use default* button in the color picker.
Time *(clock value from item history)*	
Horizontal position	Select horizontal position of the time - left, right or center.
Vertical position	Select vertical position of the time - top, bottom or middle.
Size	Enter font size height for the time (in percent relative to total widget height).
Bold	Mark the checkbox to display time in bold type.
Color	Select the time color from the color picker. D stands for default color (depends on the frontend theme). To return to the default value, click the *Use default* button in the color picker.
Change indicator	Select the color of change indicators from the color picker. The change indicators are as follows: ↑ - item value is up (for numeric items) ↓ - item value is down (for numeric items) ↕ - item value has changed (for string items and items with value mapping) D stands for default color (depends on the frontend theme). To return to the default value, click the *Use default* button in the color picker. Vertical size of the change indicator is equal to the size of the value (integer part of the value for numeric items). Note that up and down indicators are not shown with just one value.
Background color	Select the background color for the whole widget from the color picker. D stands for default color (depends on the frontend theme). To return to the default value, click the *Use default* button in the color picker.
Thresholds	Configure the dynamic background color for the whole widget. Click *Add* to add a threshold, select the background color from the color picker, and specify a numeric value. Once the item value equals or is greater than the threshold value, the background color will change. The list will be sorted in ascending order when saved. Note that the dynamic background color will be displayed correctly only for numeric items.

Note that multiple elements cannot occupy the same space; if they are placed in the same space, an error message will be displayed.

13 Map

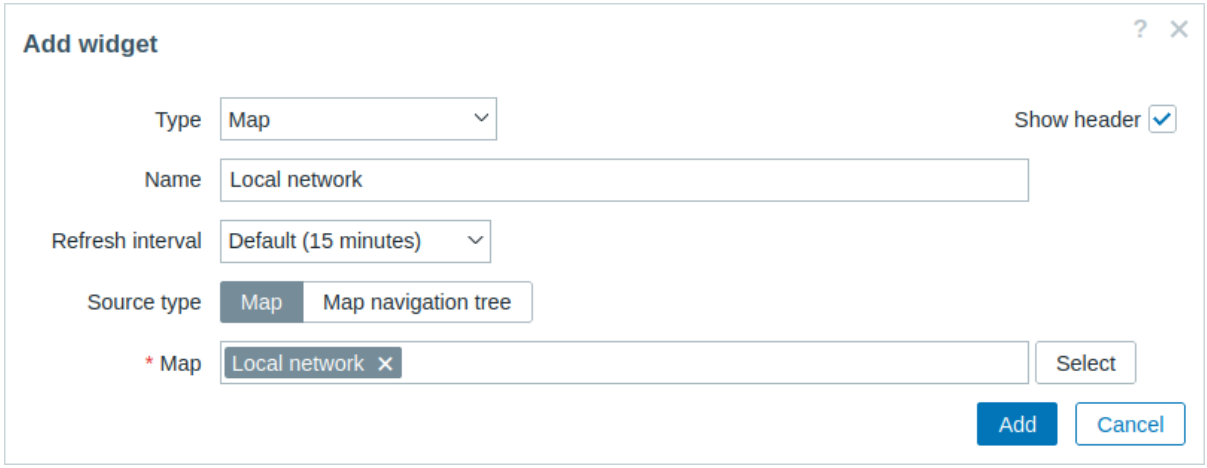
Overview

In the map widget you can display either:

- a single configured network map;
- one of the configured network maps in the map navigation tree (when clicking on the map name in the tree).

Configuration

To configure, select *Map* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

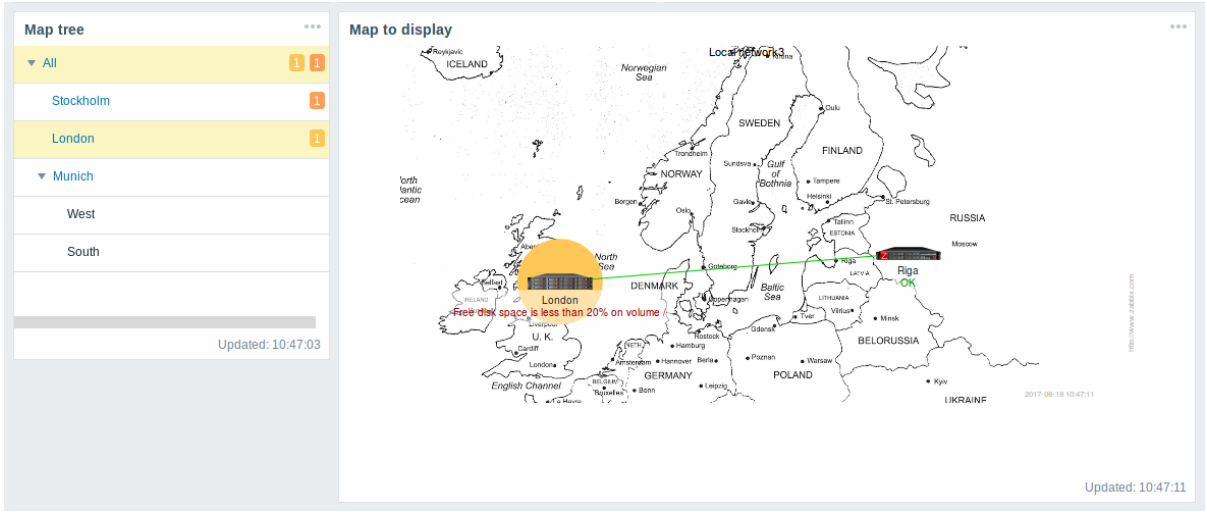
Source type	Select to display: **Map** - network map; **Map navigation tree** - one of the maps in the selected map navigation tree.
Map	Select the map to display. This field is auto-complete so starting to type the name of a map will offer a dropdown of matching maps. This option is available if 'Map' is selected as *Source type*.
Filter	Select the map navigation tree to display the maps of. This option is available if 'Map navigation tree' is selected as *Source type*.

14 Map navigation tree

Overview

This widget allows building a hierarchy of existing maps while also displaying problem statistics with each included map and map group.

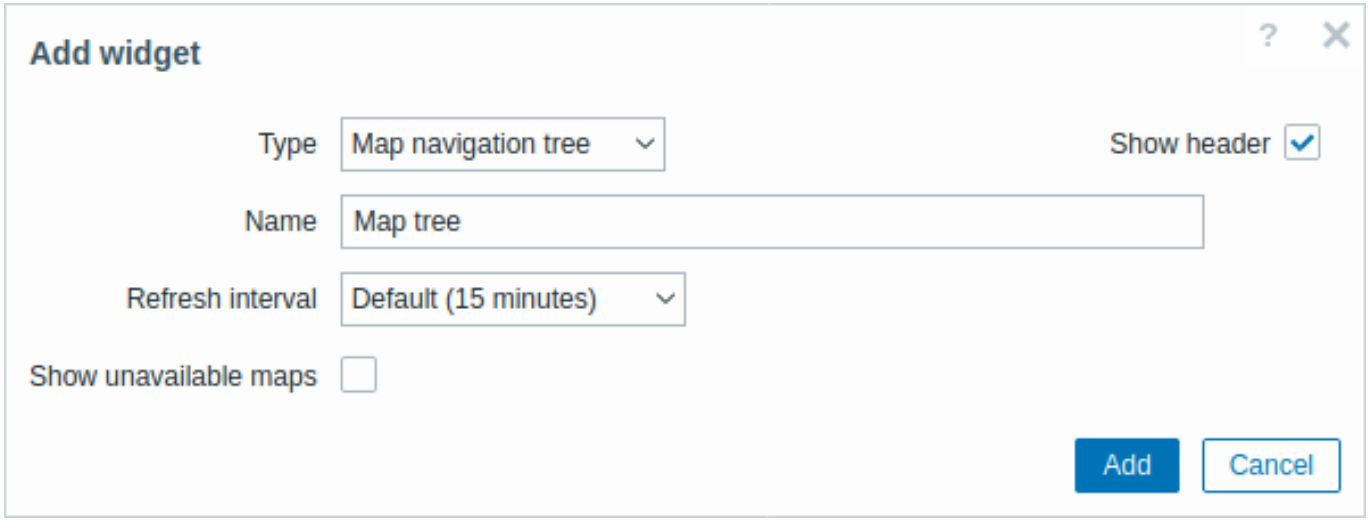
It becomes even more powerful if you link the *Map* widget to the navigation tree. In this case, clicking on a map name in the navigation tree displays the map in full in the *Map* widget.



Statistics with the top-level map in the hierarchy display a sum of problems of all submaps and their own problems.

Configuration

To configure, select *Map navigation tree* as type:

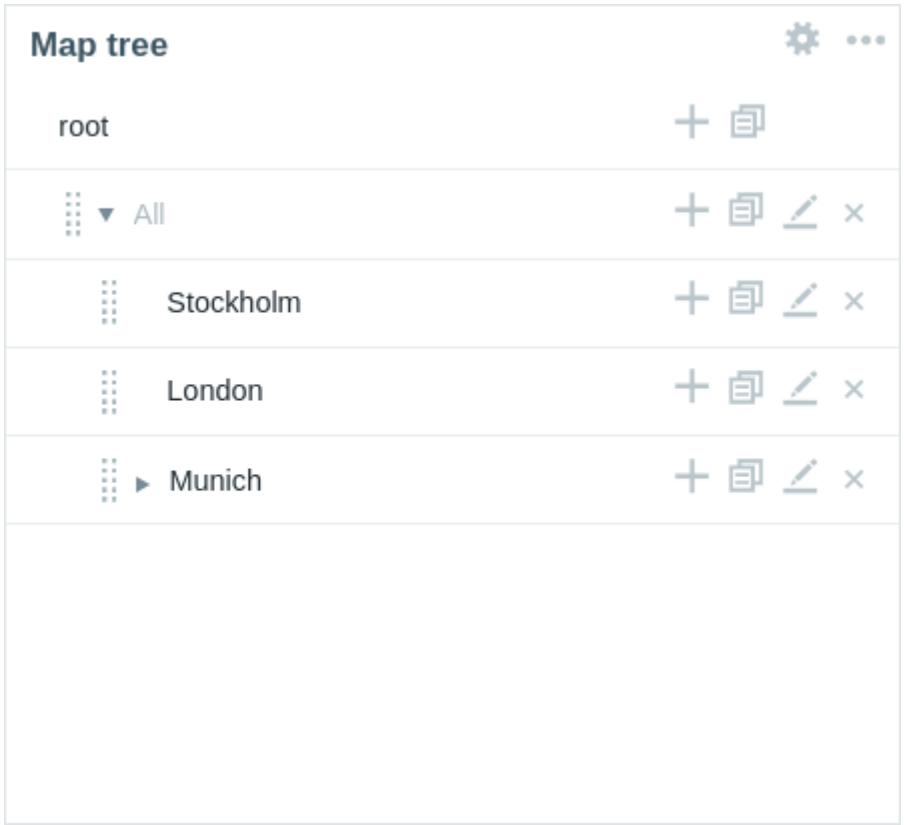


In addition to the parameters that are common for all widgets, you may set the following specific options:

Show unavailable maps	Mark this checkbox to display maps that the user does not have read permission to. Unavailable maps in the navigation tree will be displayed with a grayed-out icon. Note that if this checkbox is marked, available submaps are displayed even if the parent level map is unavailable. If unmarked, available submaps to an unavailable parent map will not be displayed at all. Problem count is calculated based on available maps and available map elements.

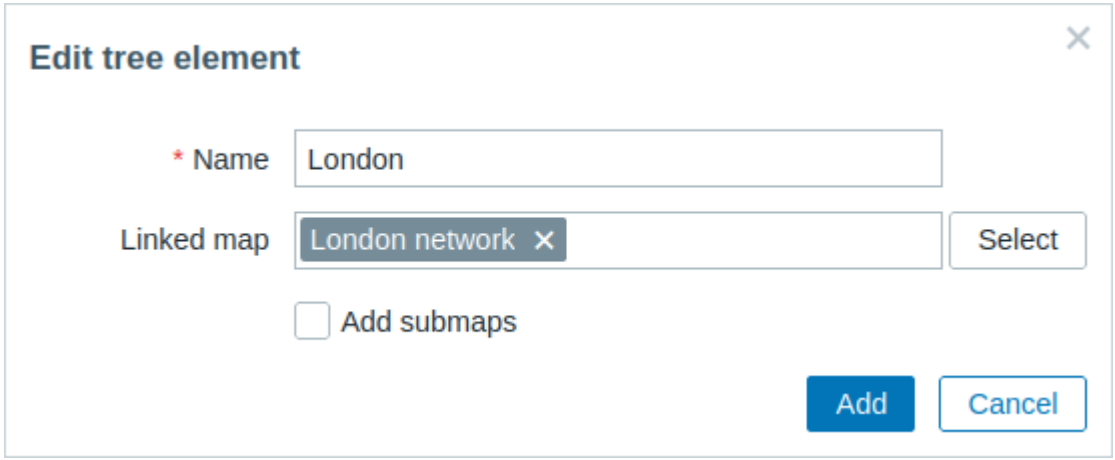
Navigation tree elements are displayed in a list. You can:

- drag an element (including its child elements) to a new place in the list;
- expand or collapse an element to display or hide its child elements;
- add a child element (with or without a linked map) to an element;
- add multiple child elements (with linked maps) to an element;
- edit an element;
- remove an element (including its child elements).



Element configuration

To configure a navigation tree element, either add a new element or edit an existing element.



The following navigation tree element configuration parameters are available:

Name	Enter the navigation tree element name.
Linked map	Select the map to link to the navigation tree element. This field is auto-complete so starting to type the name of a map will offer a dropdown of matching maps.
Add submaps	Mark this checkbox to add the submaps of the linked map as child elements to the navigation tree element.

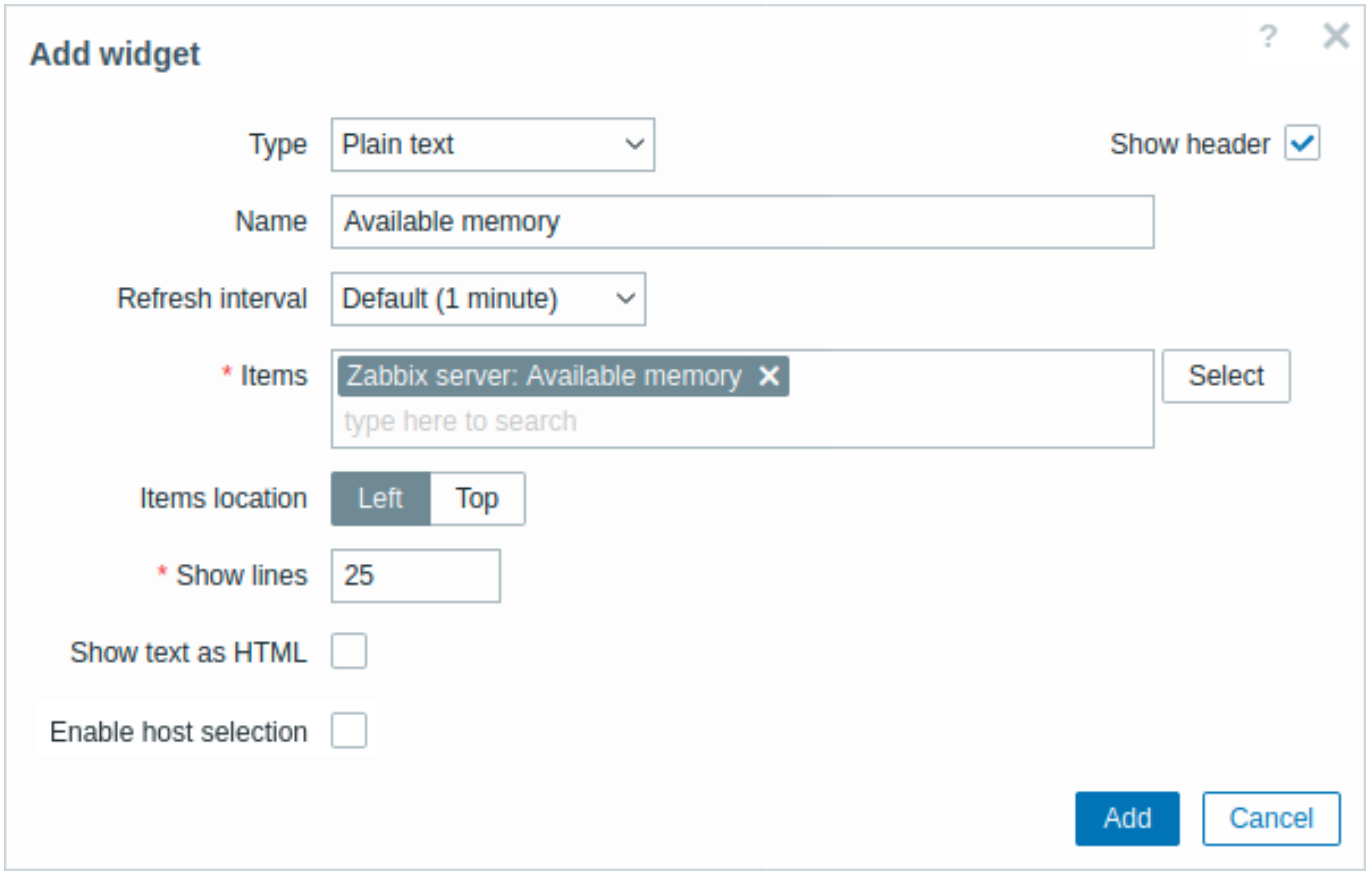
15 Plain text

Overview

In the plain text widget, you can display the latest item data in plain text.

Configuration

To configure, select *Plain text* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

Items	Select the items.
Items location	Choose the location of selected items to be displayed in the widget.
Show lines	Set how many latest data lines will be displayed in the widget.
Show text as HTML	Set to display text as HTML.
Enable host selection	Set to display different data depending on the selected host.

16 Problem hosts

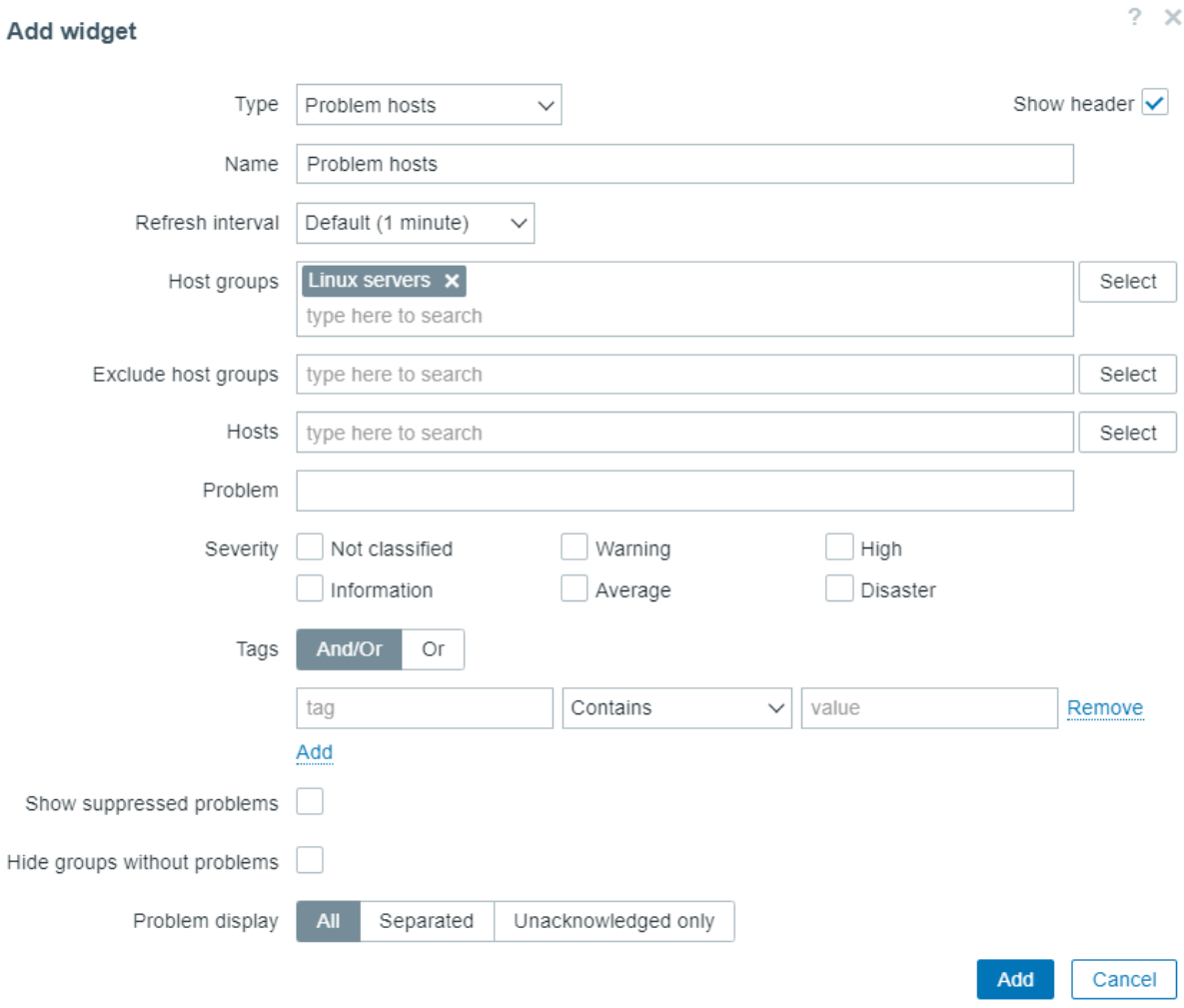
Overview

In the problem host widget, you can display problem count by host group and the highest problem severity within a group.

The problem count is displayed only for cause problems.

Configuration

To configure, select *Problem hosts* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

Host groups	Enter host groups to display in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will be displayed in the widget. If no host groups are entered, all host groups will be displayed.

Exclude host groups	Enter host groups to hide from the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will not be displayed in the widget. For example, hosts 001, 002, 003 may be in Group A and hosts 002, 003 in Group B as well. If we select to *show* Group A and *exclude* Group B at the same time, only data from host 001 will be displayed in the Dashboard.
Hosts	Enter hosts to display in the widget. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. If no hosts are entered, all hosts will be displayed.
Problem	You can limit the number of problem hosts displayed by the problem name. If you enter a string here, only those hosts with problems whose name contains the entered string will be displayed. Macros are not expanded.
Severity	Mark problem severities to filter problems to be displayed in the widget. If no severities are marked, all problems will be displayed.
Tags	Specify problem tags to limit the number of problems displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met
Show suppressed problems	Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance.
Hide groups without problems	Mark the *Hide groups without problems* option to hide data from host groups without problems in the widget.
Problem display	Display problem count as: **All** - full problem count will be displayed **Separated** - unacknowledged problem count will be displayed separated as a number of the total problem count **Unacknowledged only** - only the unacknowledged problem count will be displayed.

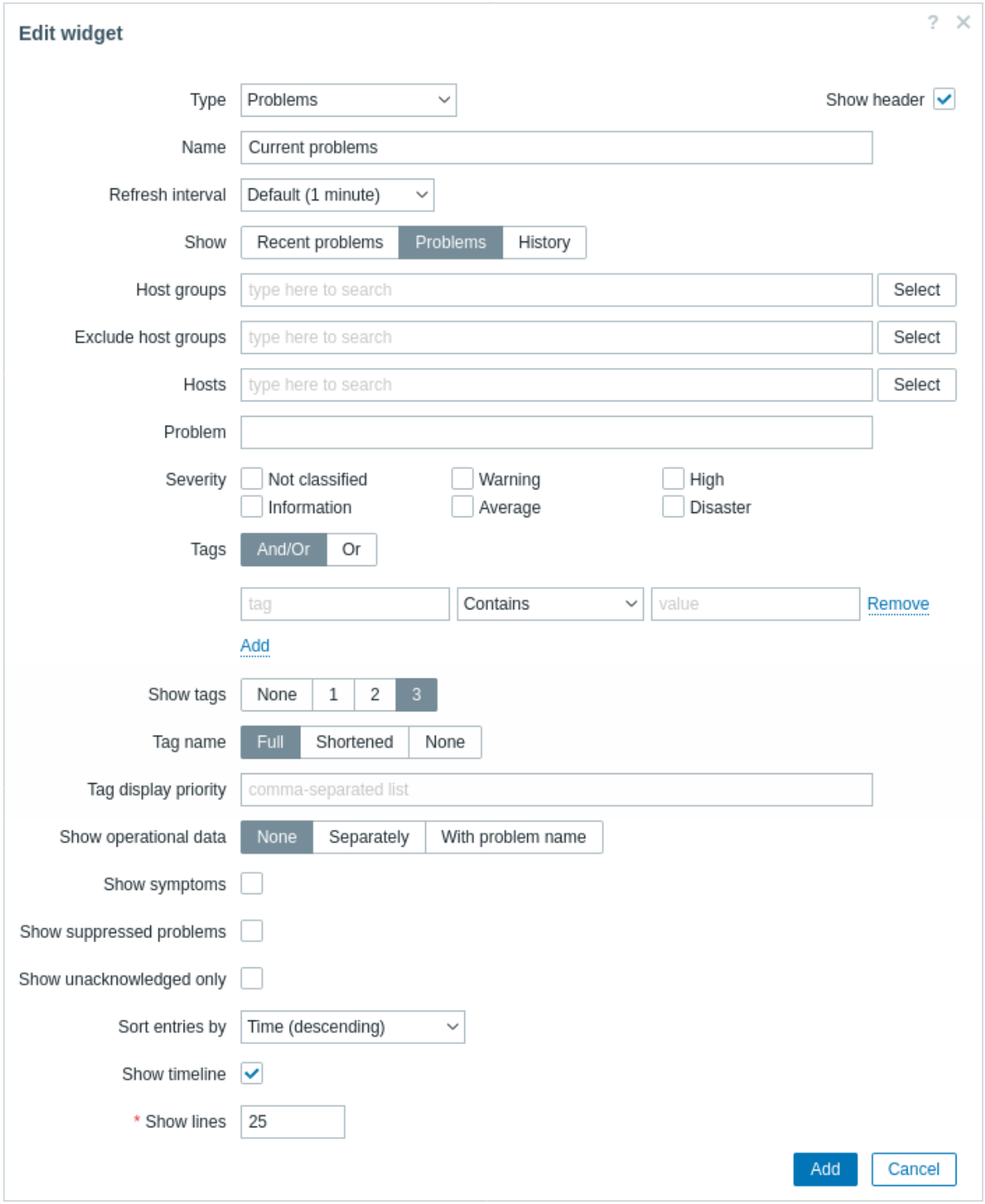
17 Problems

Overview

In this widget you can display current problems. The information in this widget is similar to *Monitoring* → *Problems*.

Configuration

To configure, select *Problems* as type:



You can limit how many problems are displayed in the widget in various ways - by problem status, problem name, severity, host group, host, event tag, acknowledgment status, etc.

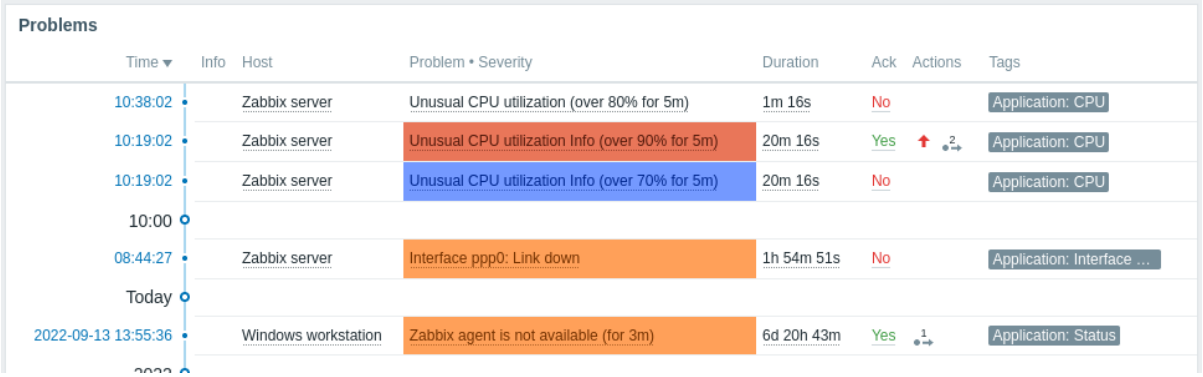
In addition to the parameters that are common for all widgets, you may set the following specific options:

Show	Filter by problem status: **Recent problems** - unresolved and recently resolved problems are displayed (default) **Problems** - unresolved problems are displayed **History** - history of all events is displayed
Host groups	Enter host groups to display problems of in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Problems from these host groups will be displayed in the widget. If no host groups are entered, problems from all host groups will be displayed.

Exclude host groups	Enter host groups to hide problems of from the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Problems from these host groups will not be displayed in the widget. For example, hosts 001, 002, 003 may be in Group A and hosts 002, 003 in Group B as well. If we select to *show* Group A and *exclude* Group B at the same time, only problems from host 001 will be displayed in the widget.
Hosts	Enter hosts to display problems of in the widget. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. If no hosts are entered, problems of all hosts will be displayed.
Problem	You can limit the number of problems displayed by their name. If you enter a string here, only those problems whose name contains the entered string will be displayed. Macros are not expanded.
Severity	Mark problem severities to filter problems to be displayed in the widget. If no severities are marked, all problems will be displayed.
Tags	Specify problem tags to limit the number of problems displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met When filtered, the tags specified here will be displayed first with the problem, unless overridden by the *Tag display priority* (see below) list.
Show tags	Select the number of displayed tags: **None** - no *Tags* column in *Monitoring* → *Problems* **1** - *Tags* column contains one tag **2** - *Tags* column contains two tags **3** - *Tags* column contains three tags To see all tags for the problem roll your mouse over the three dots icon.
Tag name	Select tag name display mode: **Full** - tag names and values are displayed in full **Shortened** - tag names are shortened to 3 symbols; tag values are displayed in full **None** - only tag values are displayed; no names
Tag display priority	Enter tag display priority for a problem, as a comma-separated list of tags (for example: `Services,Applications,Application`). Tag names only should be used, no values. The tags of this list will always be displayed first, overriding the natural ordering by alphabet.
Show operational data	Select the mode for displaying operational data: **None** - no operational data is displayed **Separately** - operational data is displayed in a separate column **With problem name** - append operational data to the problem name, using parentheses for the operational data
Show symptoms	Mark the checkbox to display in its own line problems classified as symptoms.
Show suppressed problems	Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance or single problem suppression.
Show unacknowledged only	Mark the checkbox to display unacknowledged problems only.
Sort entries by	Sort entries by: **Time** (descending or ascending) **Severity** (descending or ascending) **Problem name** (descending or ascending) **Host** (descending or ascending).
Show timeline	Mark the checkbox to display a visual timeline.

Show lines	Specify the number of problem lines to display.

Using the widget

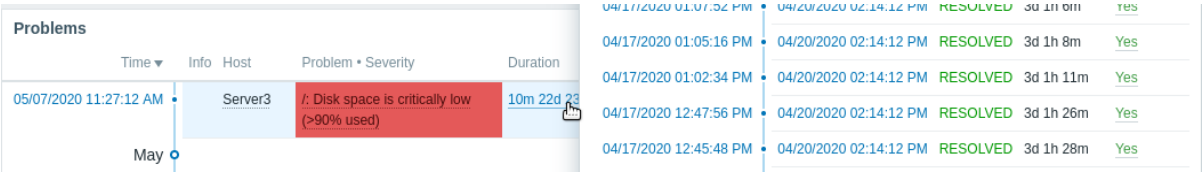


Problems widget offers quick access to additional information:

- Click on the problem date and time to view event details
- If Info column is not empty, you can hover over displayed icon to view additional details
- Click on the host name to open the host menu
- Click on the problem name to open the event menu
- Hover over or click on the problem duration to view problem event popup
- Press on the Yes or No in the Acknowledge (Ack) column to update a problem
- Hover over or press on the gray arrow icon in Actions column to view list of executed actions

Problem event popup

The problem event popup includes the list of problem events for this trigger and, if defined, the trigger description and a clickable URL.



To bring up the problem event popup:

- roll a mouse over the problem duration in the *Duration* column of the *Problems* widget. The popup disappears once you remove the mouse from the duration.
- click on the duration in the *Duration* column of the *Problems* widget. The popup disappears only if you click on the duration again.

18 Problems by severity

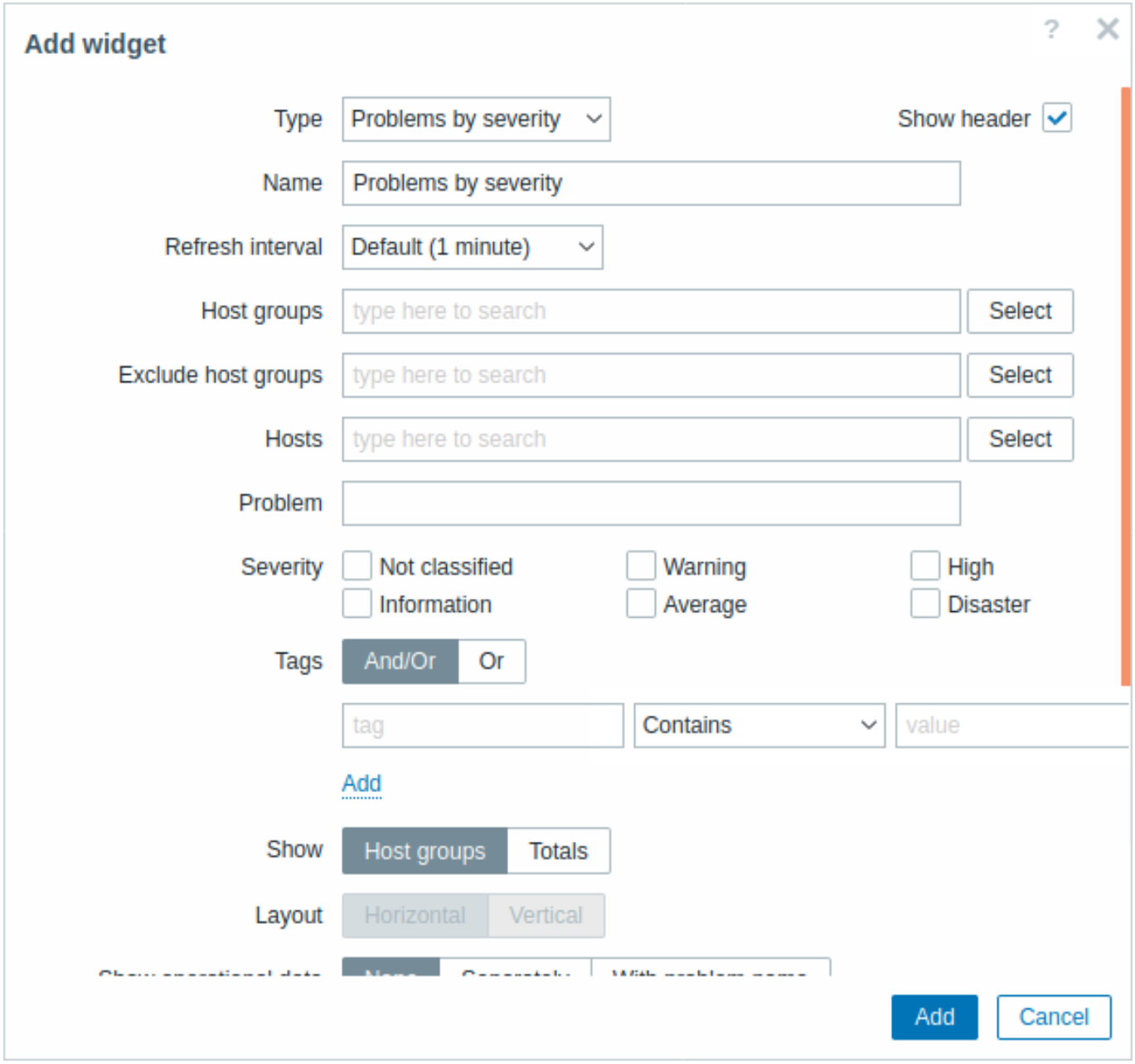
Overview

In this widget, you can display the problem count by severity. You can limit what hosts and triggers are displayed in the widget and define how the problem count is displayed.

The problem count is displayed only for cause problems.

Configuration

To configure, select *Problems by severity* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

Host groups	Enter host groups to display in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will be displayed in the widget. If no host groups are entered, all host groups will be displayed.
Exclude host groups	Enter host groups to hide from the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will not be displayed in the widget. For example, hosts 001, 002, 003 may be in Group A and hosts 002, 003 in Group B as well. If we select to *show* Group A and *exclude* Group B at the same time, only data from host 001 will be displayed in the Dashboard.
Hosts	Enter hosts to display in the widget. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. If no hosts are entered, all hosts will be displayed.
Problem	You can limit the number of problem hosts displayed by the problem name. If you enter a string here, only those hosts with problems whose name contains the entered string will be displayed. Macros are not expanded.
Severity	Mark problem severities to filter problems to be displayed in the widget. If no severities are marked, all problems will be displayed.

Tags	Specify problem tags to limit the number of problems displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met
Show	Select the show option: **Host groups** - display problems per host group **Totals** - display a problem total for all selected host groups in colored blocks corresponding to the problem severity.
Layout	Select the layout option: **Horizontal** - colored blocks of totals will be displayed horizontally **Vertical** - colored blocks of totals will be displayed vertically This field is available for editing if 'Totals' is selected as the *Show* option.
Show suppressed problems	Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance.
Hide groups without problems	Mark the *Hide groups without problems* option to hide data from host groups without problems in the widget.
Show operational data	Mark the checkbox to display operational data (see description of *Operational data* in *Monitoring → Problems*).
Problem display	Display problem count as: **All** - full problem count will be displayed **Separated** - unacknowledged problem count will be displayed separated as a number of the total problem count **Unacknowledged only** - only the unacknowledged problem count will be displayed.
Show timeline	Mark the checkbox to display a visual timeline.

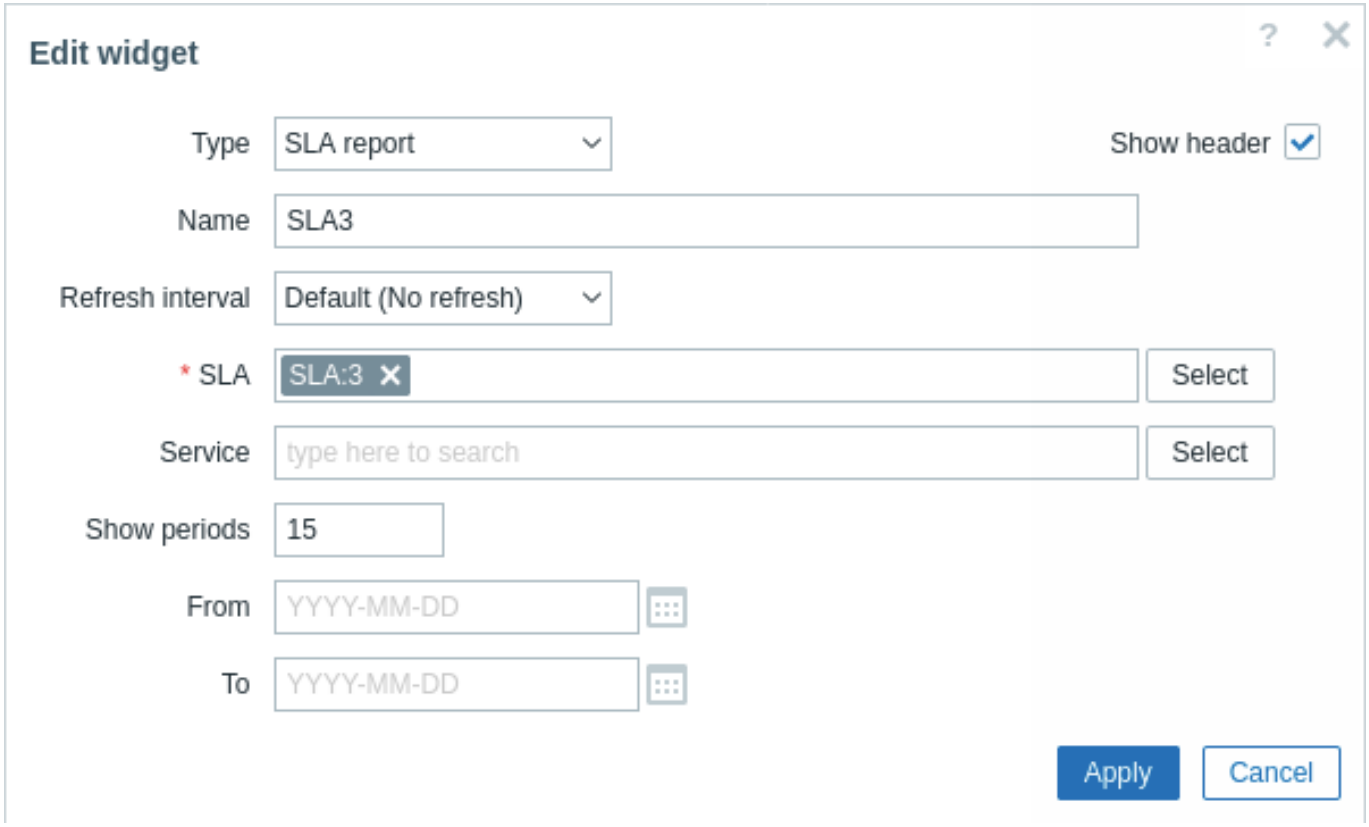
19 SLA report

Overview

This widget is useful for displaying SLA reports. Functionally it is similar to the *Services -> SLA report* section.

Configuration

To configure, select *SLA report* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

SLA	Select the SLA for the report.
Service	Select the service for the report.
Show periods	Set how many periods will be displayed in the widget (20 by default, 100 maximum).
From	Select the beginning date for the report. Relative dates are supported: `now`, `now/d`, `now/w-1w` etc; supported date modifiers: d, w, M, y.
To	Select the end date for the report. Relative dates are supported: `now`, `now/d`, `now/w-1w` etc; supported date modifiers: d, w, M, y.

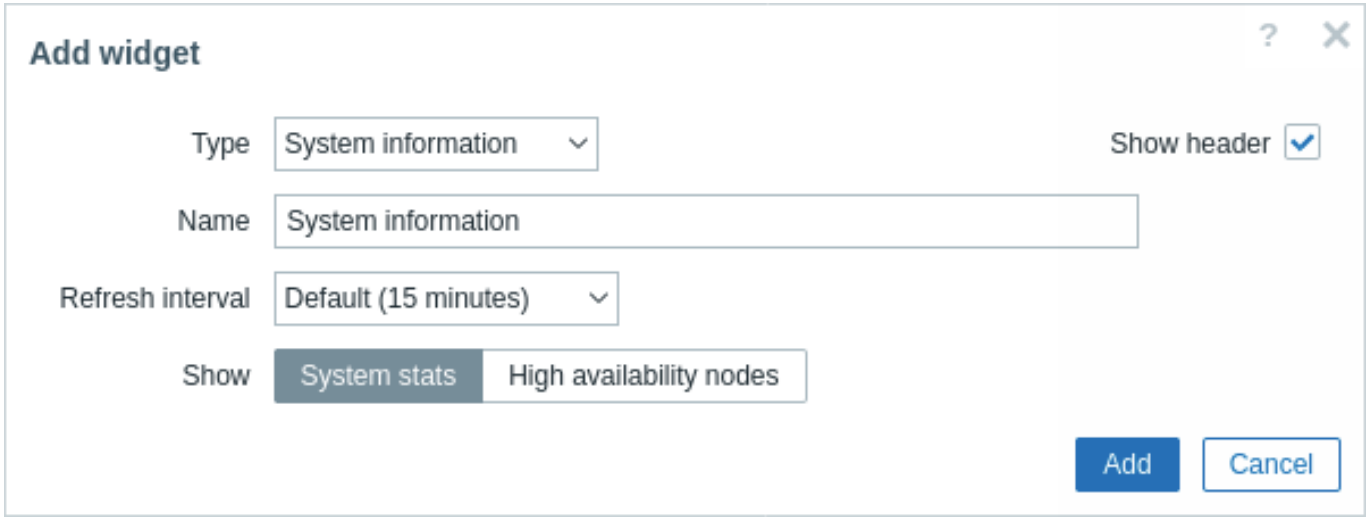
20 System information

Overview

This widget displays the same information as in *Reports* → *System information*, however, a single dashboard widget can only display either the system stats or the high availability nodes at a time (not both).

Configuration

To configure, select *System information* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

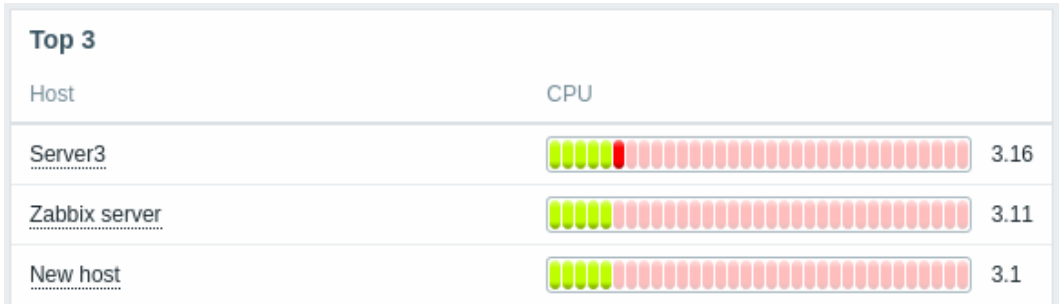
Show	Select what to display: **System stats** - display a summary of key Zabbix server and system data; **High availability nodes** - display the status of high availability nodes (if high availability cluster is enabled).

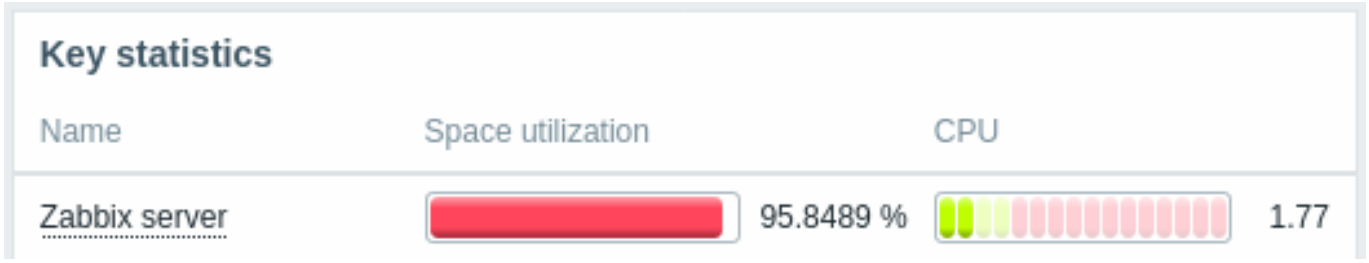
21 Top hosts

Overview

This widget provides a way to create custom tables for displaying the data situation, allowing to display *Top N*-like reports and progress-bar reports useful for capacity planning.

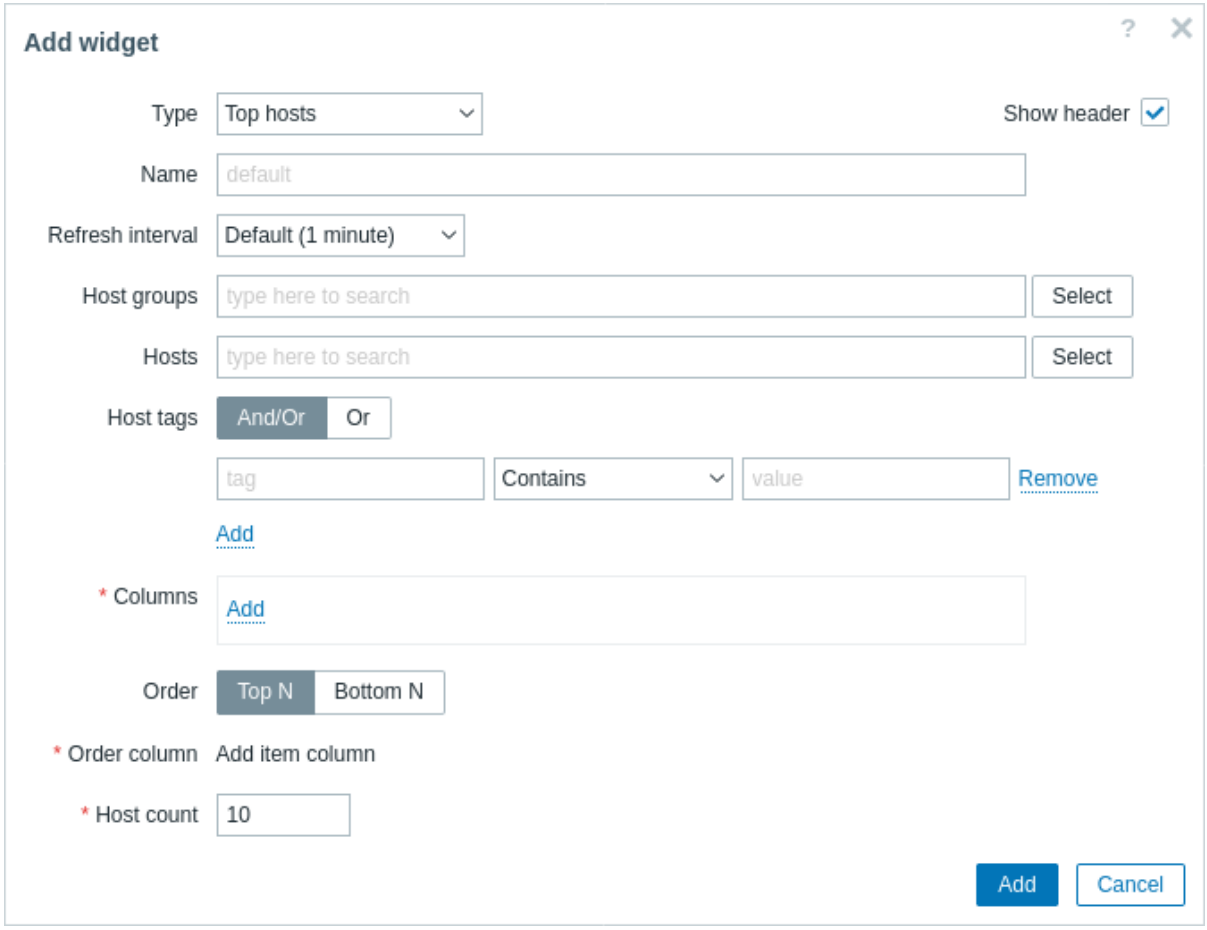
The maximum number of hosts that can be displayed is 100.





Configuration

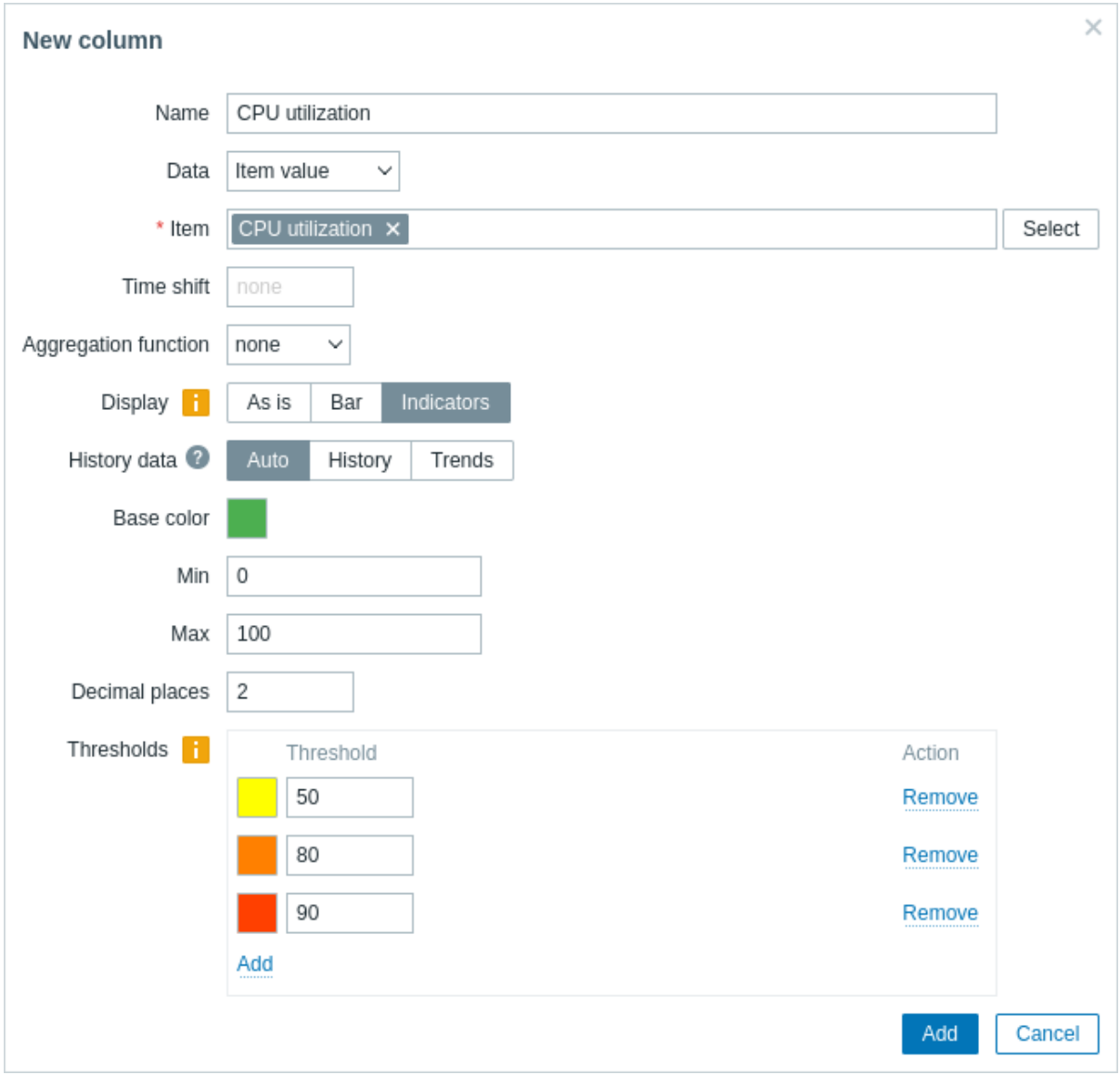
To configure, select *Top hosts* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

Host groups	Host groups to display data for.
Hosts	Hosts to display data for.
Host tags	Specify tags to limit the number of hosts displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names; **Equals** - include the specified tag names and values (case-sensitive); **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive); **Does not exist** - exclude the specified tag names; **Does not equal** - exclude the specified tag names and values (case-sensitive); **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive). There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition; **Or** - enough if one condition is met.
Columns	Add data columns to display. The column order determines their display from left to right. Columns can be reordered by dragging up and down by the handle before the column name.
Order	Specify the ordering of rows: **Top N** - in descending order by the *Order column* aggregated value; **Bottom N** - in ascending order by the *Order column* aggregated value.
Order column	Specify the column from the defined *Columns* list to use for *Top N* or *Bottom N* ordering.
Host count	Count of host rows to be shown (1-100).

Column configuration



Common column parameters:

Name	Name of the column.
Data	Data type to display in the column: **Item value** - value of the specified item; **Host name** - host name of the item specified in the *Item value* column; **Text** - static text string.
Base color	Background color of the column; fill color if *Item value* data is displayed as bar/indicators. For *Item value* data the default color can be overridden by custom color, if the item value is over one of the specified "Thresholds".

Specific parameters for item value columns:

Item	Select the item.
Time shift	Specify time shift if required. You may use time suffixes in this field. Negative values are allowed.

Aggregation function	Specify which aggregation function to use: **min** - display the smallest value; **max** - display the largest value; **avg** - display the average value; **sum** - display the sum of values; **count** - display the count of values; **first** - display the first value; **last** - display the last value; **none** - display all values (no aggregation). Aggregation allows to display an aggregated value for the chosen interval (5 minutes, an hour, a day), instead of all values. Note that only numeric items can be displayed in this column if this setting is not ”none”.
Aggregation interval	Specify the interval for aggregating values. You may use time suffixes in this field. A numeric value without a suffix will be regarded as seconds. This field will not be displayed if *Aggregation function* is ”none”.
Display	Define how the value should be displayed: **As is** - as regular text; **Bar** - as solid, color-filled bar; **Indicators** - as segmented, color-filled bar. Note that only numeric items can be displayed in this column if this setting is not ”as is”.
History	Take data from history or trends: **Auto** - automatic selection; **History** - take history data; **Trends** - take trend data. This setting applies only to numeric data. Non-numeric data will always be taken from history.
Min	Minimum value for bar/indicators.
Max	Maximum value for bar/indicators.
Decimal places	Specify how many decimal places will be displayed with the value. This setting applies only to numeric data.
Thresholds	Specify threshold values when the background/fill color should change. The list will be sorted in ascending order when saved. Note that only numeric items can be displayed in this column if thresholds are used.

Specific parameters for text columns:

Text	Enter the string to display. May contain host and inventory macros.

22 Trigger overview

Overview

In the trigger overview widget, you can display the trigger states for a group of hosts.

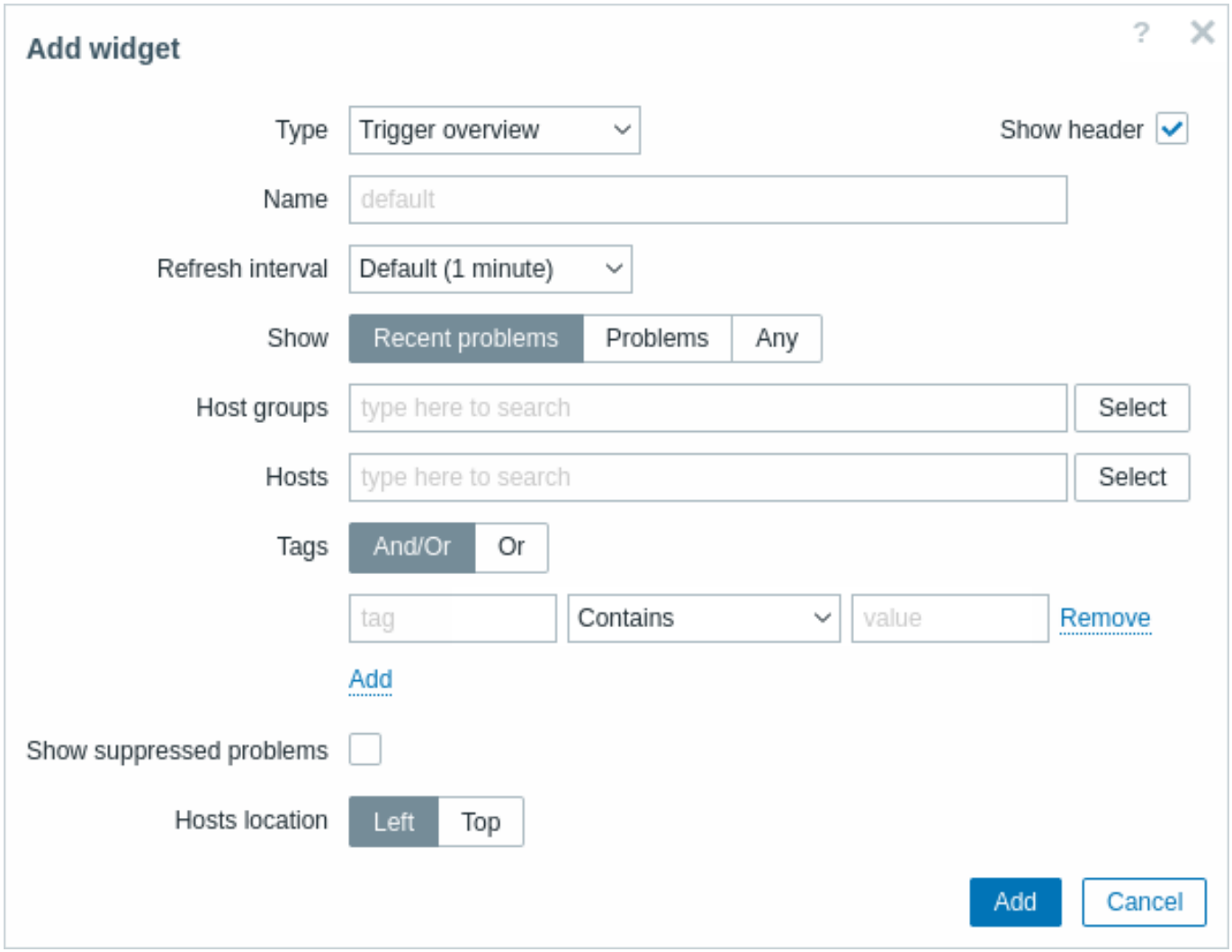
- The trigger states are displayed as colored blocks (the color of the blocks for PROBLEM triggers depends on the problem severity color, which can be adjusted in the problem update screen). Note that recent trigger state changes (within the last 2 minutes) will be displayed as blinking blocks.
- Gray up and down arrows indicate triggers that have dependencies. On mouseover, dependency details are revealed.
- A checkbox icon indicates acknowledged problems. All problems or resolved problems of the trigger must be acknowledged for this icon to be displayed.

Clicking on a trigger block provides context-dependent links to problem events of the trigger, the problem acknowledgment screen, trigger configuration, trigger URL or a simple graph/latest values list.

Note that 50 records are displayed by default (configurable in *Administration* → *General* → *GUI*, using the *Max number of columns and rows in overview tables* option). If more records exist than are configured to display, a message is displayed at the bottom of the table, asking to provide more specific filtering criteria. There is no pagination. Note that this limit is applied first, before any further filtering of data, for example, by tags.

Configuration

To configure, select *Trigger overview* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

Show	Filter triggers by trigger state: **Recent problems** - *(default)* show triggers that recently have been or still are in a PROBLEM state (resolved and unresolved); **Problems** - show triggers that are in a PROBLEM state (unresolved); **Any** - show all triggers.
Host groups	Select the host group(s). This field is auto-complete, so starting to type the name of a group will offer a dropdown of matching groups.
Hosts	Select hosts. This field is auto-complete, so starting to type the name of a host will offer a dropdown of matching hosts. Scroll down to select. Click on 'x' to remove the selected.

Tags	Specify tags to filter the triggers displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. **Note:** If the parameter *Show* is set to 'Any', all triggers will be displayed even if tags are specified. However, while recent trigger state changes (displayed as blinking blocks) will update for all triggers, the trigger state details (problem severity color and whether the problem is acknowledged) will only update for triggers that match the specified tags. There are several operators available for each condition: **Exists** - include the specified tag names; **Equals** - include the specified tag names and values (case-sensitive); **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive); **Does not exist** - exclude the specified tag names; **Does not equal** - exclude the specified tag names and values (case-sensitive); **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive). There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the *Or* condition; **Or** - enough if one condition is met.
Show suppressed problems	Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance.
Hosts location	Select host location - left or top.

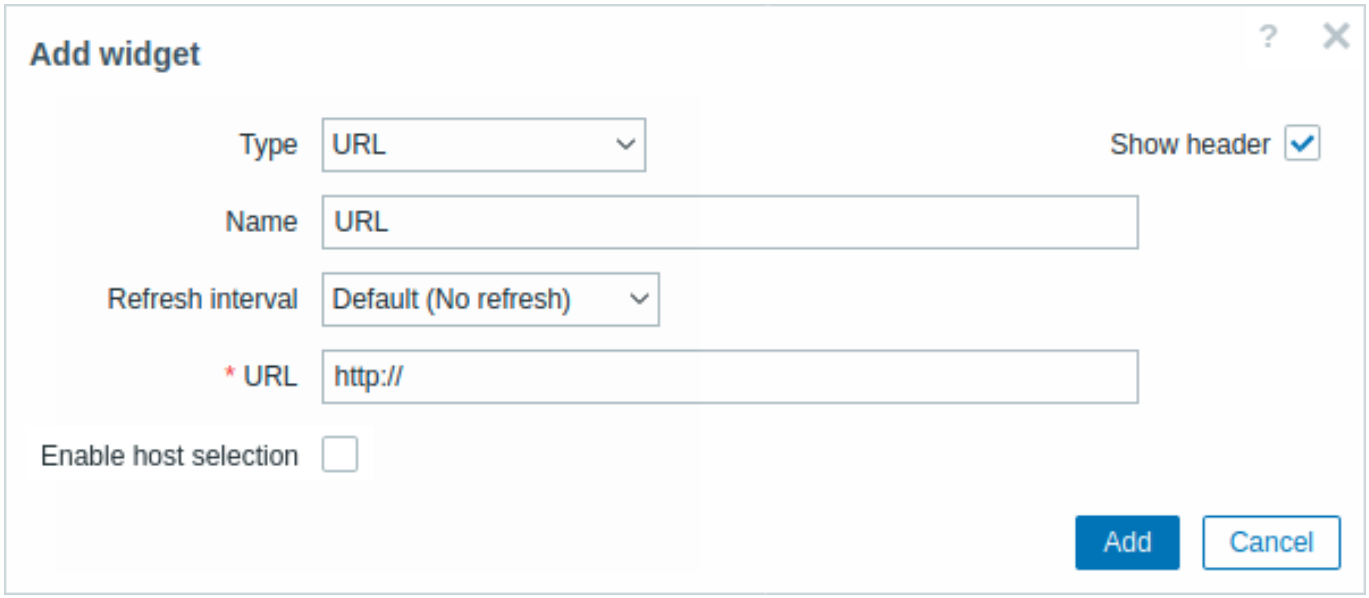
23 URL

Overview

This widget displays the content retrieved from the specified URL.

Configuration

To configure, select *URL* as type:



In addition to the parameters that are common for all widgets, you may set the following specific options:

URL	Enter the URL to display. External URLs must start with `http://` or `https://`. Since Zabbix 4.4.8, internal URLs support relative paths (for example, `zabbix.php?action=report.status`). {HOST.*} macros are supported.
Enable host selection	Set to display different URL content depending on the selected host. This can work if {HOST.*} macros are used in the URL.

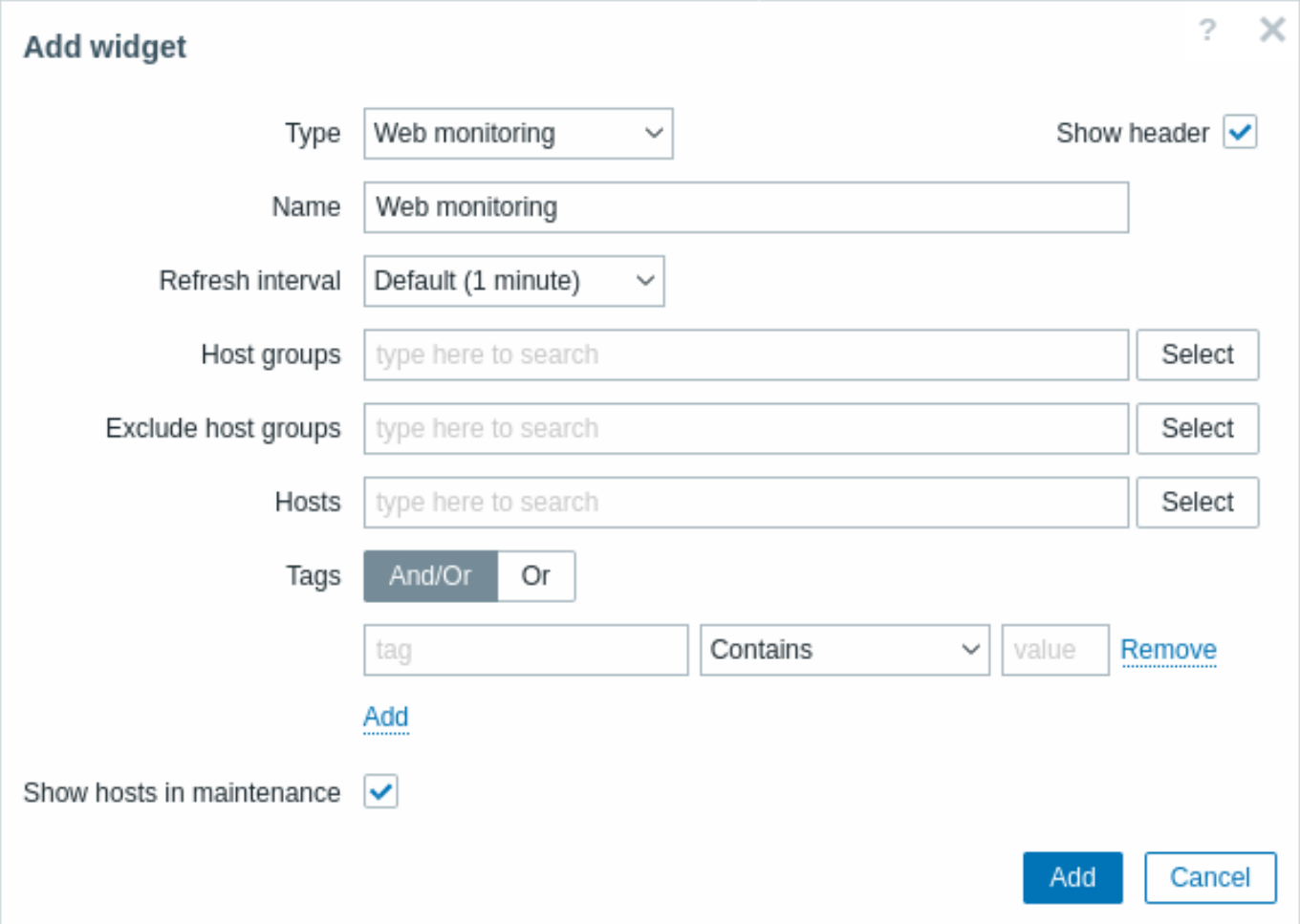
Attention:
Browsers might not load an HTTP page included in the widget if Zabbix frontend is accessed over HTTPS.

24 Web monitoring

Overview

This widget displays a status summary of the active web monitoring scenarios. See the Web monitoring widget section for detailed information.

Configuration



Note:
In cases when a user does not have permission to access certain widget elements, that element's name will appear as *Inaccessible* during the widget's configuration. This results in *Inaccessible Item, Inaccessible Host, Inaccessible Group, Inaccessible Map, and Inaccessible Graph* appearing instead of the "real" name of the element.

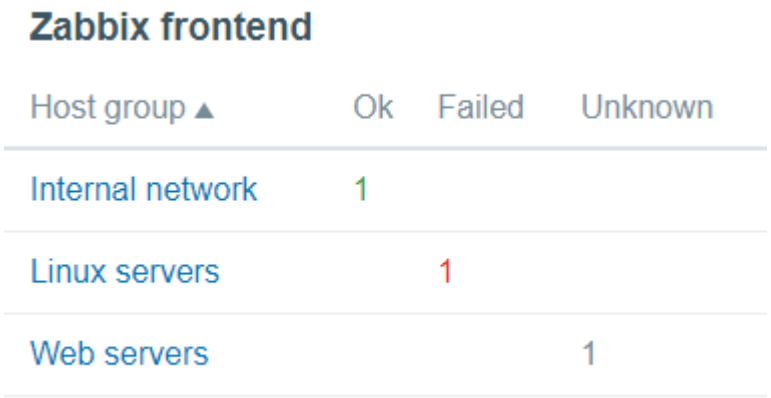
In addition to the parameters that are common for all widgets, you may set the following specific options:

Host groups	Enter host groups to display in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will be displayed in the widget. If no host groups are entered, all host groups will be displayed.
Exclude host groups	Enter host groups to hide from the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will not be displayed in the widget. For example, hosts 001, 002, 003 may be in Group A and hosts 002, 003 in Group B as well. If we select to *show* Group A and *exclude* Group B at the same time, only data from host 001 will be displayed in the Dashboard.
Hosts	Enter hosts to display in the widget. This field is auto-complete so starting to type the name of a host will offer a dropdown of matching hosts. If no hosts are entered, all hosts will be displayed.
Tags	Specify tags to limit the number of web scenarios displayed in the widget. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met
Show hosts in maintenance	Include hosts that are in maintenance in the statistics.

Web monitoring widget

Once you have completed the configuration, you might like to see the widget with the data it displays. To do it, go to *Dashboards*, click on the name of a dashboard where you created the widget.

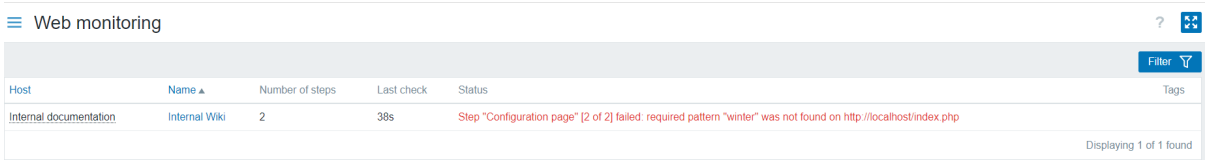
In this example, you can see the widget named ”Zabbix frontend” displaying the status of the web monitoring for three host groups: ”Internal network,” ”Linux servers,” and ”Web servers.”



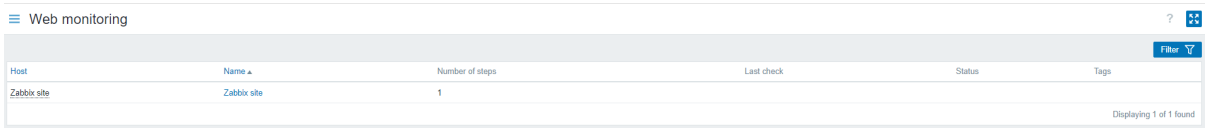
A web monitoring widget displays the following information:

- a name of a widget; below it, there are four columns:
  - *Host group* - displays a list of host groups that contain hosts having web scenarios configured;
  - *Ok* - displays a number of web scenarios (in green color) when two conditions are observed:
    - Zabbix has collected the latest data for a web scenario(s);
    - all steps that were configured in a web scenario are in ”Ok” *Status*.

- *Failed* - displays a number of web scenarios (in red color), which have some failed steps:
  * click on the host name, and it will open a new window; the *Status* column provides detailed information (in red color) on the step where Zabbix failed to collect the data; and also,
  * gives a hint for the parameter that has to be corrected in the configuration form.



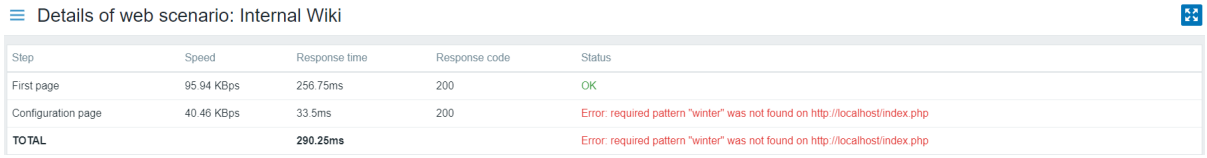
- *Unknown* - displays a number of web scenarios (in grey color) for which Zabbix has neither collected data, nor has an information about the failed steps.



Viewing the status and data

Clickable links in the widget allow to easily navigate and quickly acquire a full information on each web scenario. Thus, to view:

- the Status of a web scenario, click on the name of a host group.
- more detailed statistics, click on the scenario name. In this example it is "Zabbix frontend".
- the details in the case of *Failed* status, click on a host group name; in the window that opens, click on a web scenario name in the *Name* column; it will open more detailed information on the configured steps for which Zabbix failed to collect the data.



Now, you can return to the web scenario configuration form and correct your settings.

To view the details in the case of *Unknown* status, you can repeat the same steps as explained for *Failed*.

Attention:
At the first monitoring instance, a web scenario is always displayed in *Unknown* state, which is switched to *Failed* or *Ok* state right after the first check. In the case when a host is monitored by the proxy, the status change occurs in accordance with the data collection frequency configured on the proxy.

2 Monitoring

Overview

The Monitoring menu is all about displaying data. Whatever information Zabbix is configured to gather, visualize and act upon, it will be displayed in the various sections of the Monitoring menu.

View mode buttons

The following buttons located in the top right corner are common for every section:

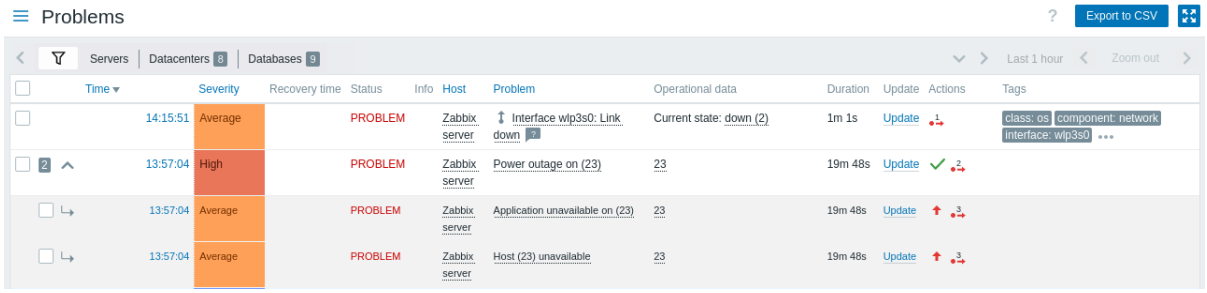
	Display page in kiosk mode. In this mode only page content is displayed. To exit kiosk mode, move the mouse cursor until the  exit button appears and click on it. You will be taken back to normal mode.

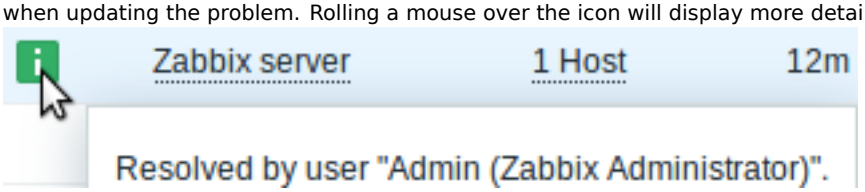
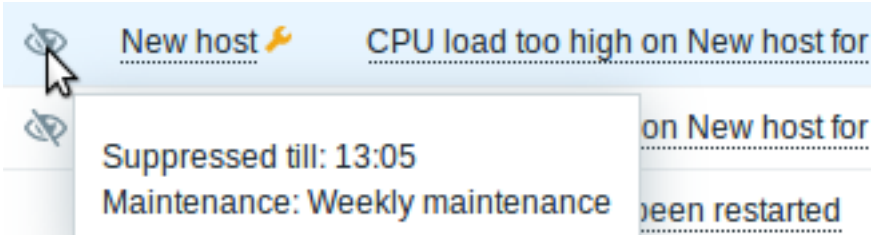
1 Problems

Overview

In *Monitoring* → *Problems* you can see what problems you currently have. Problems are those triggers that are in the "Problem" state.

By default all new problems are classified as cause problems. It is possible to manually reclassify certain problems as symptom problem of the cause problem. For more details, see cause and symptom events.



Column	Description
Checkbox	Checkboxes for problem selection are displayed. Icons, next to the checkboxes, have the following meaning:  - the number of symptom events for the cause problem;  - expand to show symptom events;  - collapse to hide symptom events;  - this is a symptom event.
Time	Problem start time is displayed.
Severity	Problem severity is displayed. Problem severity is originally based on the severity of the underlying problem trigger, however, after the event has happened it can be updated using the *Update problem* screen. Color of the problem severity is used as cell background during problem time.
Recovery time	Problem resolution time is displayed.
Status	Problem status is displayed: **Problem** - unresolved problem **Resolved** - recently resolved problem. You can hide recently resolved problems using the filter. New and recently resolved problems blink for 2 minutes. Resolved problems are displayed for 5 minutes in total. Both of these values are configurable in *Administration* → *General* → *Trigger displaying options*.
Info	A green information icon is displayed if a problem is closed by global correlation or manually when updating the problem. Rolling a mouse over the icon will display more details:  The following icon is displayed if a suppressed problem is being shown (see *Show suppressed problems* option in the filter). Rolling a mouse over the icon will display more details: 
Host	Problem host is displayed. Clicking on the host name brings up the host menu.

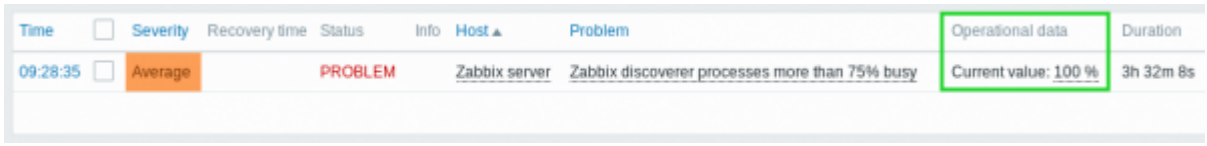
Column	Description
Problem	Problem name is displayed. Problem name is based on the name of the underlying problem trigger. Macros in the trigger name are resolved at the time of the problem happening and the resolved values do not update any more. *Note* that it is possible to append the problem name with operational data showing some latest item values. Clicking on the problem name brings up the event menu. Hovering on the  icon after the problem name will bring up the trigger description (for those problems that have it).
Operational data	Operational data are displayed containing latest item values. Operational data can be a combination of text and item value macros if configured on a trigger level. If no operational data is configured on a trigger level, the latest values of all items from the expression are displayed. This column is only displayed if *Separately* is selected for *Show operational data* in the filter.
Duration	Problem duration is displayed. See also: Negative problem duration
Update	Click on the *Update* link to go to the problem update screen where various actions can be taken on the problem, including commenting and acknowledging the problem.
Actions	History of activities about the problem is displayed using symbolic icons:  - problem has been acknowledged. This icon is always displayed first.  - comments have been made. The number of comments is also displayed.  - problem severity has been increased (e.g. Information → Warning)  - problem severity has been decreased (e.g. Warning → Information)  - problem severity has been changed, but returned to the original level (e.g. Warning → Information → Warning)  - actions have been taken. The number of actions is also displayed.  - actions have been taken, at least one is in progress. The number of actions is also displayed.  - actions have been taken, at least one has failed. The number of actions is also displayed. When rolling the mouse over the icons, popups with details about the activity are displayed. See viewing details to learn more about icons used in the popup for actions taken.
Tags	Tags are displayed (if any). In addition, tags from an external ticketing system may also be displayed (see the *Process tags* option when configuring webhooks).

Operational data of problems

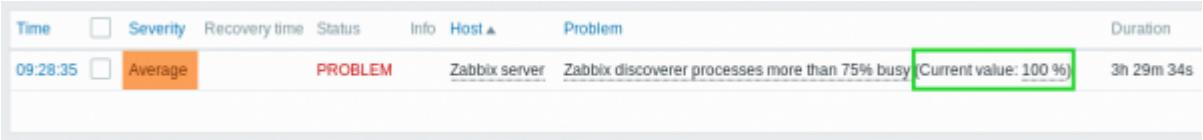
It is possible to display operational data for current problems, i.e. the latest item values as opposed to the item values at the time of the problem.

Operational data display can be configured in the filter of *Monitoring* → *Problems* or in the configuration of the respective dashboard widget, by selecting one of the three options:

- *None* - no operational data is displayed
- *Separately* - operational data is displayed in a separate column



- *With problem name* - operational data is appended to the problem name and in parentheses. Operational data are appended to the problem name only if the *Operational data* field is non-empty in the trigger configuration.



The content of operational data can be configured with each trigger, in the *Operational data* field. This field accepts an arbitrary string with macros, most importantly, the {ITEM.LASTVALUE<1-9>} macro.

{ITEM.LASTVALUE<1-9>} in this field will always resolve to the latest values of items in the trigger expression. {ITEM.VALUE<1-9>} in this field will resolve to the item values at the moment of trigger status change (i.e. change into problem, change into OK, being closed manually by a user or being closed by correlation).

Note that closing the problem manually does not produce a new value so the resolved value of {ITEM.LASTVALUE<1-9>} or {ITEM.VALUE<1-9>} will still show the value from the problem time.

{ITEM.LASTVALUE<1-9>} or {ITEM.VALUE<1-9>} will resolve to *UNKNOWN* if the latest history value has been collected more than the *Max history display period* time ago (see *Administration → General*).

Negative problem duration

It is actually possible in some common situations to have negative problem duration i.e. when the problem resolution time is earlier than problem creation time, e. g.:

- If some host is monitored by proxy and a network error happens, leading to no data received from the proxy for a while, the nodata(/host/key) trigger will be fired by the server. When the connection is restored, the server will receive item data from the proxy having a time from the past. Then, the nodata(/host/key) problem will be resolved and it will have a negative problem duration;
- When item data that resolve the problem event are sent by Zabbix sender and contain a timestamp earlier than the problem creation time, a negative problem duration will also be displayed.

Note:
Negative problem duration is not affecting SLA calculation or Availability report of a particular trigger in any way; it neither reduces nor expands problem time.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Mass update* - update the selected problems by navigating to the problem update screen

To use this option, mark the checkboxes before the respective problems, then click on the *Mass update* button.

Buttons

The button to the right offers the following option:

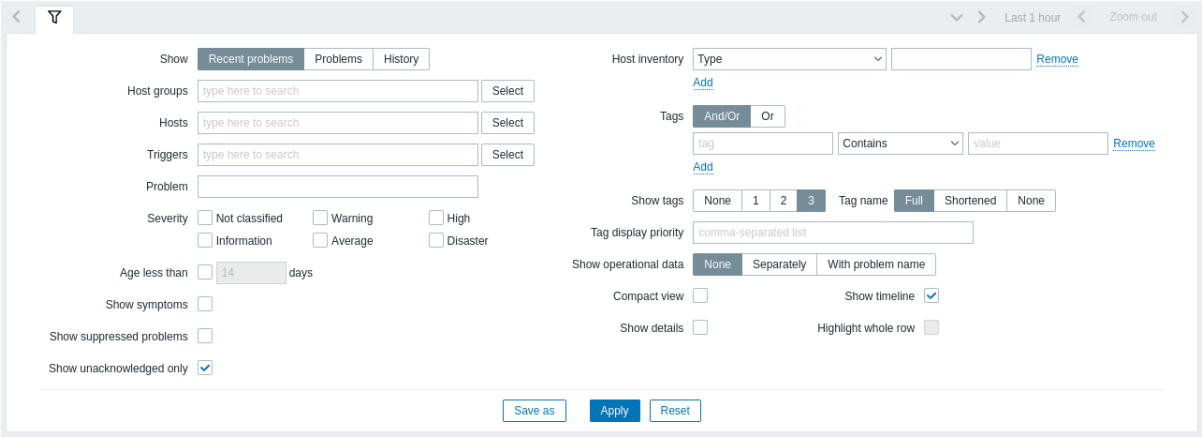
Export to CSV	Export content from all pages to a CSV file.

View mode buttons, being common for all sections, are described on the Monitoring page.

Using filter

You can use the filter to display only the problems you are interested in. For better search performance, data is searched with macros unresolved.

The filter is located above the table. Favorite filter settings can be saved as tabs and then quickly accessed by clicking on the tabs above the filter.



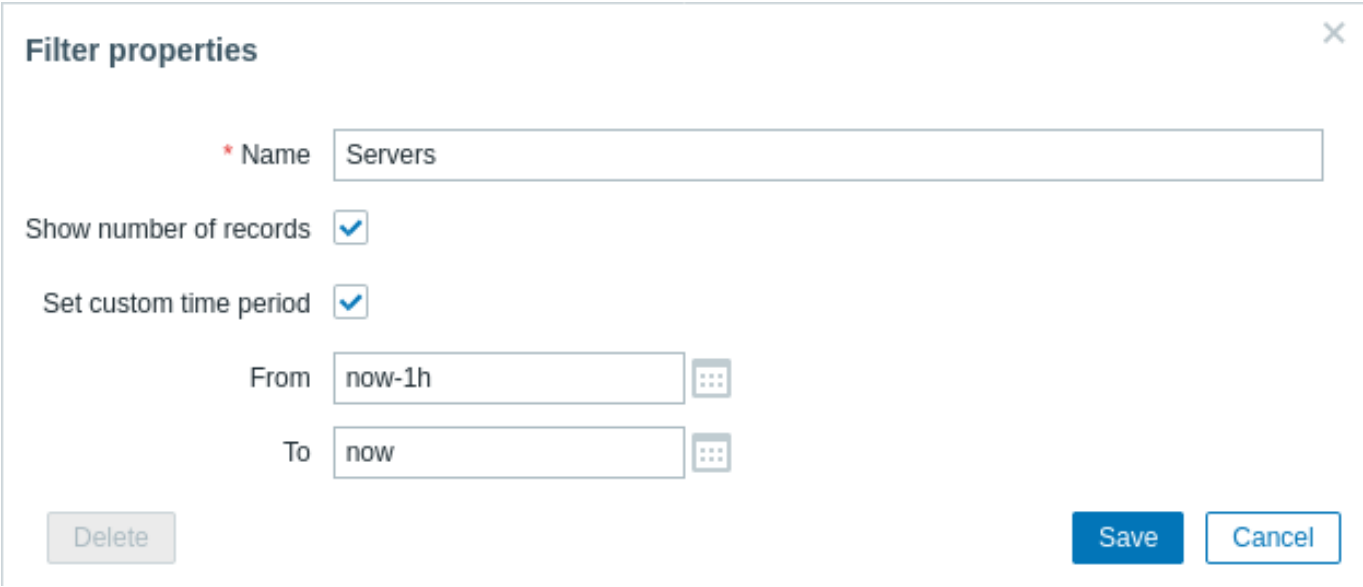
Parameter	Description
Show	Filter by problem status: **Recent problems** - unresolved and recently resolved problems are displayed (default) **Problems** - unresolved problems are displayed **History** - history of all events is displayed
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
Hosts	Filter by one or more hosts.
Triggers	Filter by one or more triggers.
Problem	Filter by problem name.
Severity	Filter by trigger (problem) severity.
Age less than	Filter by how old the problem is.
Show symptoms	Mark the checkbox to display in its own line problems classified as symptoms.
Show suppressed problems	Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance or single problem suppression.
Show unacknowledged only	Mark the checkbox to display unacknowledged problems only.
Host inventory	Filter by inventory type and value.
Tags	Filter by event tag name and value. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met When filtered, the tags specified here will be displayed first with the problem, unless overridden by the *Tag display priority* (see below) list.
Show tags	Select the number of displayed tags: **None** - no *Tags* column in *Monitoring → Problems* **1** - *Tags* column contains one tag **2** - *Tags* column contains two tags **3** - *Tags* column contains three tags To see all tags for the problem roll your mouse over the three dots icon.
Tag name	Select tag name display mode: **Full** - tag names and values are displayed in full **Shortened** - tag names are shortened to 3 symbols; tag values are displayed in full **None** - only tag values are displayed; no names
Tag display priority	Enter tag display priority for a problem, as a comma-separated list of tags (for example: `Services,Applications,Application`). Tag names only should be used, no values. The tags of this list will always be displayed first, overriding the natural ordering by alphabet.

Parameter	Description
Show operational data	Select the mode for displaying operational data: **None** - no operational data is displayed **Separately** - operational data is displayed in a separate column **With problem name** - append operational data to the problem name, using parentheses for the operational data
Compact view	Mark the checkbox to enable compact view.
Show details	Mark the checkbox to display underlying trigger expressions of the problems. Disabled if *Compact view* checkbox is marked.
Show timeline	Mark the checkbox to display the visual timeline and grouping. Disabled if *Compact view* checkbox is marked.
Highlight whole row	Mark the checkbox to highlight the full line for unresolved problems. The problem severity color is used for highlighting. Enabled only if the *Compact view* checkbox is marked in the standard blue and dark themes. *Highlight whole row* is not available in the high-contrast themes.

Tabs for favorite filters

Frequently used sets of filter parameters can be saved in tabs.

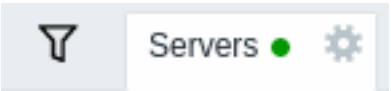
To save a new set of filter parameters, open the main tab, and configure the filter settings, then press the *Save as* button. In a new popup window, define *Filter properties*.



Parameter	Description
Name	The name of the filter to display in the tab list.
Show number of records	Check, if you want the number of problems to be displayed next to the tab name.
Set custom time period	Check to set specific default time period for this filter set. If set, you will only be able to change the time period for this tab by updating filter settings. For tabs without a custom time period, the time range can be changed by pressing the time selector button in the top right corner (button name depends on selected time interval: This week, Last 30 minutes, Yesterday, etc.). This option is available only for filters in *Monitoring→Problems*.
From/To	Time period start and end in absolute (`Y-m-d H:i:s`) or relative time syntax (`now-1d`). Available, if *Set custom time period* is checked.

When saved, the filter is created as a named filter tab and immediately activated.

To edit the filter properties of an existing filter, press the gear symbol next to the active tab name.



Notes:

- To hide the filter area, click on the name of the current tab. Click on the active tab name again to open the filter area again.
- Keyboard navigation is supported: use arrows to switch between tabs, press *Enter* to open.
- The left/right buttons above the filter may be used to switch between saved filters. Alternatively, the downward pointing button opens a drop-down menu with all saved filters and you can click on the one you need.
- Filter tabs can be re-arranged by dragging and dropping.
- If the settings of a saved filter have been changed (but not saved), a green dot is displayed after the filter name. To update the filter according to the new settings, click on the *Update* button, which is displayed instead of the *Save as* button.
- Current filter settings are remembered in the user profile. When the user opens the page again, the filter settings will have stayed the same.

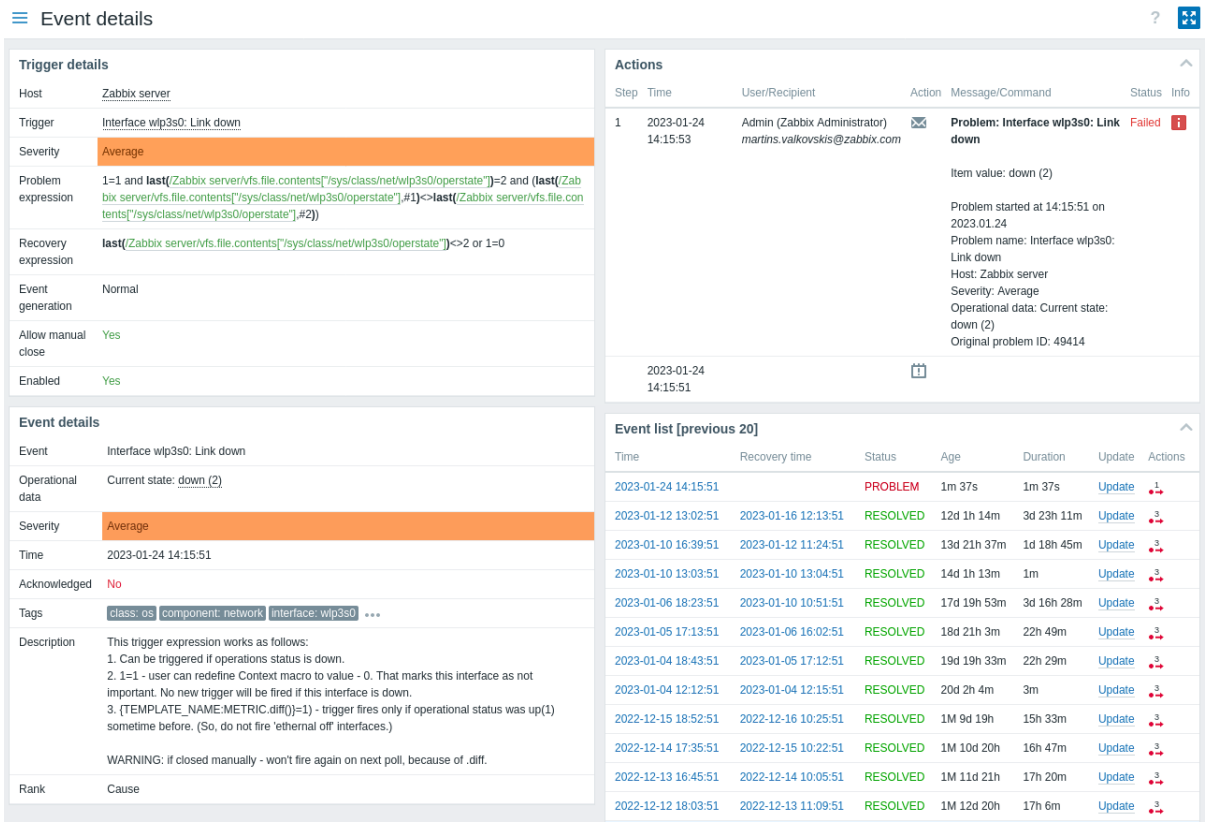
Note:
To share filters, copy and send to others a URL of an active filter. After opening this URL, other users will be able to save this set of parameters as a permanent filter in their Zabbix account.
See also: Page parameters.

Filter buttons

Apply	Apply specified filtering criteria (without saving).
Reset	Reset current filter and return to saved parameters of the current tab. On the main tab, this will clear the filter.
Save as	Save current filter parameters in a new tab. Only available on the main tab.
Update	Replace tab parameters with currently specified parameters. Not available on the main tab.

Viewing details

The times for problem start and recovery in *Monitoring* → *Problems* are links. Clicking on them opens more details of the event.



Note that the problem severity may differ for the trigger and the problem event - if it has been updated for the problem event using the *Update problem* screen.

In the action list, the following icons are used to denote the activity type:

-  - problem event generated
-  - message has been sent
-  - problem event acknowledged
-  - problem event unacknowledged
-  - a comment has been added
-  - problem severity has been increased (e.g. Information → Warning)
-  - problem severity has been decreased (e.g. Warning → Information)
-  - problem severity has been changed, but returned to the original level (e.g. Warning → Information → Warning)
-  - a remote command has been executed
-  - problem event has recovered
-  - the problem has been closed manually
-  - the problem has been suppressed
-  - the problem has been unsuppressed
-  - the problem has been converted to a symptom problem
-  - the problem has been converted to a cause problem

1 Cause and symptom problems

Overview

By default all new problems are classified as cause problems. It is possible to manually reclassify certain problems as symptom problems of the cause problem.

For example, power outage may be the actual root cause why some host is unreachable or some service is down. In this case, ”host is unreachable” and ”service is down” problems must be classified as symptom problems of ”power outage” - the cause problem.

The cause-symptom hierarchy supports only two levels. A problem that is already a symptom cannot be assigned ”subordinate” symptom problems; any problems assigned as symptoms to a symptom problem will become symptoms of the same cause problem.

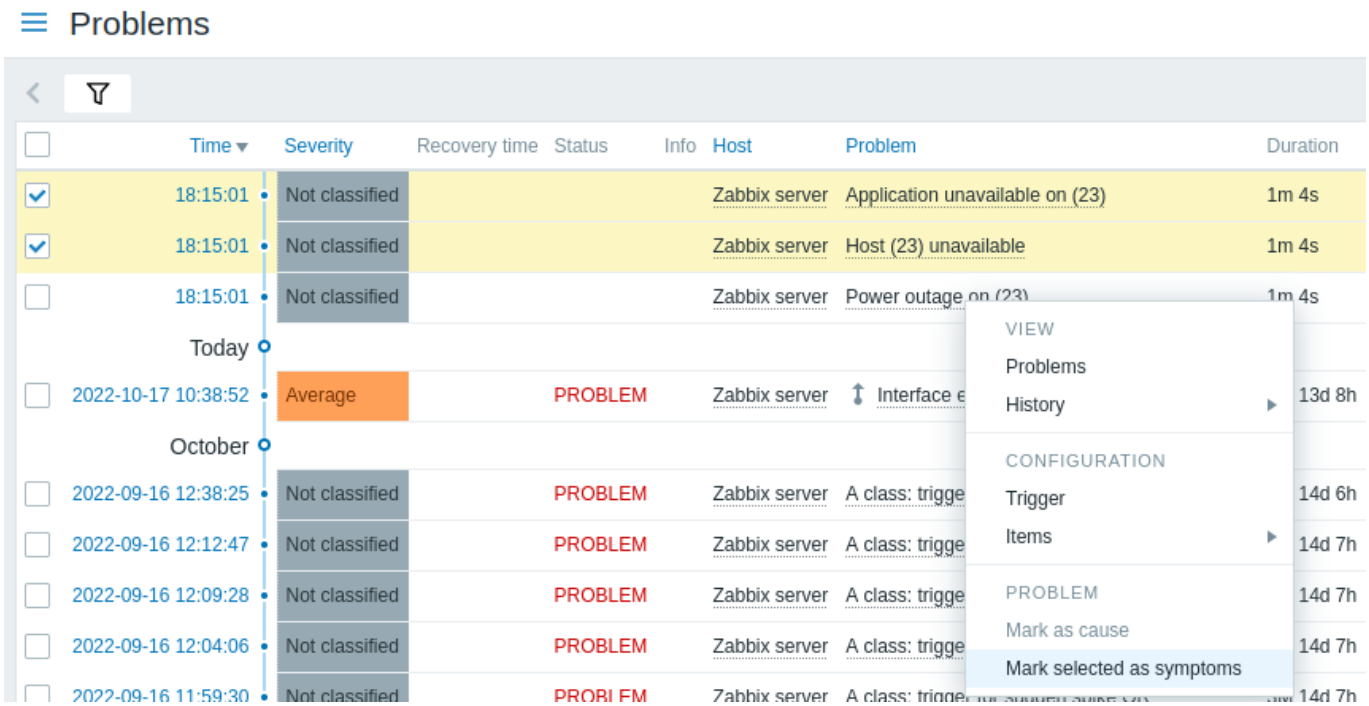
Only cause problems are counted in problem totals in maps, dashboard widgets such as *Problems by severity* or *Problem hosts*, etc. However, problem ranking does not affect services.

A symptom problem can be linked to only one cause problem. Symptom problems are not automatically resolved, if the cause problem is resolved or closed.

Configuration

To reclassify a problem as symptom problem, first select it in the list of problems. One or several problems can be selected.

Then go to the cause problem, and in its context menu click on the *Mark selected as symptoms* option.



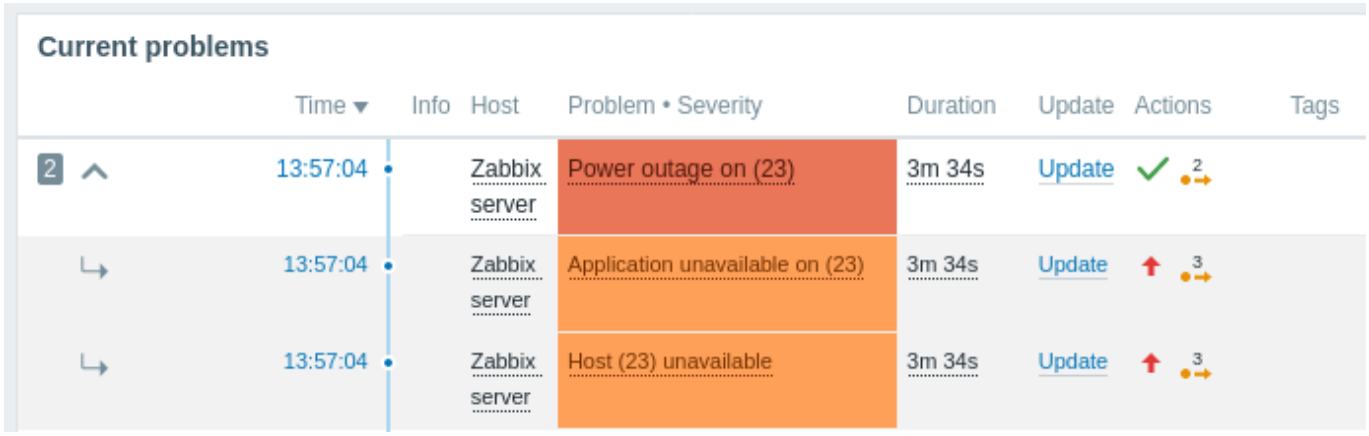
After that, the selected problems will be updated by the server to symptom problems of the cause problem.

While the status of the problem is being updated, it is displayed in one of two ways:

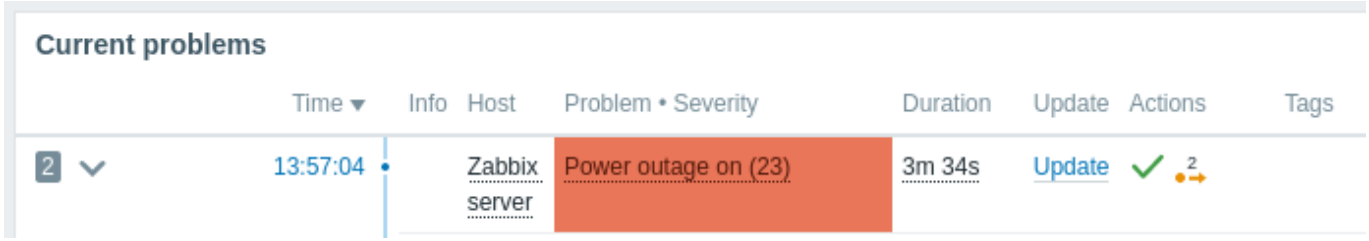
- A blinking "UPDATING" status is displayed in the Status column;
- A blinking or icon in the Info column (this is in effect if *Problems* only are selected in the filter and thus the Status column is not shown).

Display

Symptom problems are displayed below the cause problem and marked accordingly in *Monitoring -> Problems* (and the *Problems* dashboard widget) - with an icon, smaller font and different background.



In collapsed view, only the cause problem is seen; the existence of symptom problems is indicated by the number in the beginning of the line and the icon for expanding the view.



It is also possible to additionally display symptom problems in normal font and in their own line. For that select *Show symptoms* in the filter settings or the widget configuration.

Reverting to cause problem

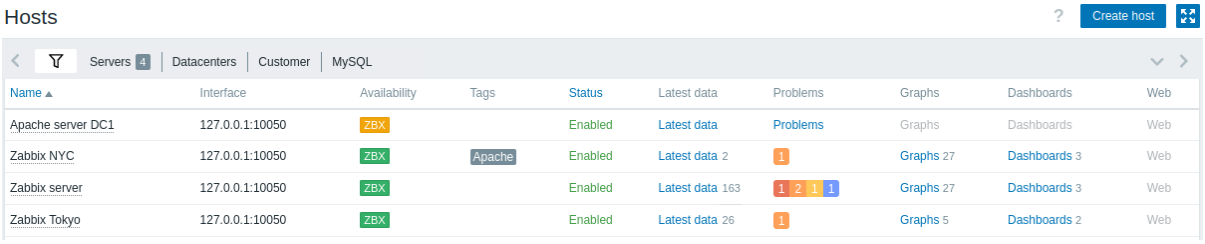
A symptom problem can be reverted back to a cause problem. To do that, either:

- click on the *Mark as cause* option in the context menu of the symptom problem;
- mark the *Convert to cause* option in to the problem update screen and click on *Update* (this option will also work if several problems are selected).

2 Hosts

Overview

The *Monitoring → Hosts* section displays a full list of monitored hosts with detailed information about host interface, availability, tags, current problems, status (enabled/disabled), and links to easily navigate to the host's latest data, problem history, graphs, dashboards and web scenarios.



Column	Description
Name	The visible host name. Clicking on the name brings up the host menu. An orange wrench icon after the name indicates that this host is in maintenance. Click on the column header to sort hosts by name in ascending or descending order.
Interface	The main interface of the host is displayed.

Column	Description
Availability	Host availability per configured interface is displayed. Availability icons represent host interface current status on Zabbix server. Therefore, if you disable a host in the frontend, its availability will update after Zabbix server has synchronized the configuration changes. Similarly, if you enable a host, its availability will update after Zabbix server has synchronized the configuration changes and polled the host. Availability icons represent only those interface types (Agent, SNMP, IPMI, JMX) that are configured. Hovering over the icon displays a pop-up with a list of all interfaces of the same type with details, status, and errors. For Agent interface, the pop-up displays interfaces (passive) and active checks. If a host has active checks only, the Agent interface icon is displayed even if the host does not have an Agent interface (passive) configured. The column is empty for hosts with no interfaces. The current status of all interfaces of one type is indicated by the icon color: **Green** - all interfaces are available; **Yellow** - at least one interface is not available, and at least one is available or unknown; **Red** - all interfaces are not available; **Gray** - at least one interface is unknown, but none are not available. For details on how Zabbix server determines the "Unknown" status, see Unknown interface status. **Active check availability.** Active checks also affect host interface availability if at least one active check is enabled on the host. Active check availability is counted towards the total Agent interface availability as described above. For example, if a host has an Agent interface (passive) that is available, but active checks are unknown, the total Agent interface availability is displayed as gray (unknown). **Note:** Since Zabbix 6.4.12, there are two exceptions for determining the total Agent interface availability: - if active checks are available, but at least one Agent interface (passive) is unknown while the host also has at least one item using this interface, the total Agent interface availability is displayed as gray (unknown); - if active checks are available and all Agent interfaces (passive) are unknown (and no items are using this interface), the total Agent interface availability is displayed as green (available). To determine active check availability, heartbeat messages are sent in the agent active check thread. The frequency of heartbeat messages is controlled by the `HeartbeatFrequency` parameter in Zabbix agent or agent 2 configuration (default 60 seconds, range 0-3600). Active checks are considered unavailable when the active check heartbeat is older than 2 x HeartbeatFrequency seconds. **Note:** Zabbix agents older than version 6.2.x do not send active check heartbeats, so the availability of their hosts remains unknown.
Tags	Tags of the host and all linked templates, with macros unresolved.
Status	Host status - *Enabled* or *Disabled*. Click on the column header to sort hosts by status in ascending or descending order.
Latest data	Clicking on the link will open the *Monitoring* **-** *Latest data* page with all the latest data collected from the host. The number of items with latest data is displayed in gray.
Problems	The number of open host problems sorted by severity. The color of the square indicates problem severity. The number on the square means the number of problems for the given severity. Clicking on the icon will open *Monitoring* **-** *Problems* page for the current host. If a host has no problems, a link to the Problems section for this host is displayed as text. Use the filter to select whether suppressed problems should be included (not included by default).
Graphs	Clicking on the link will display graphs configured for the host. The number of graphs is displayed in gray. If a host has no graphs, the link is disabled (gray text) and no number is displayed.
Dashboards	Clicking on the link will display dashboards configured for the host. The number of dashboards is displayed in gray. If a host has no dashboards, the link is disabled (gray text) and no number is displayed.

Column	Description
Web	Clicking on the link will display web scenarios configured for the host. The number of web scenarios is displayed in gray. If a host has no web scenarios, the link is disabled (gray text) and no number is displayed.

Buttons

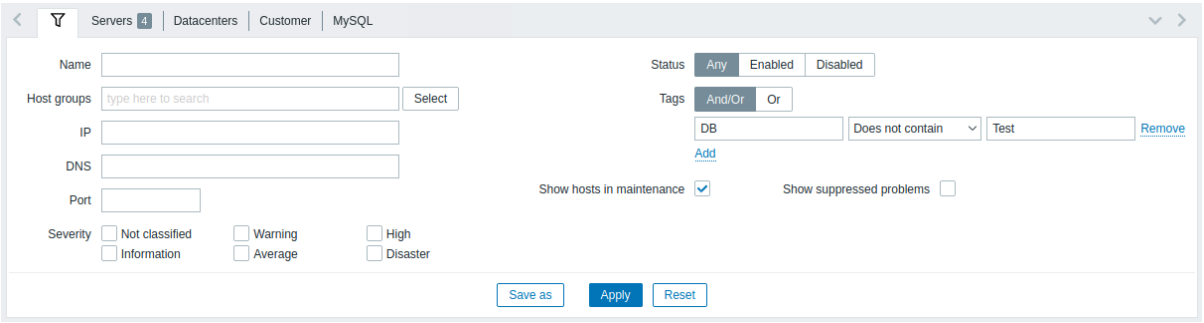
Create host allows to create a new host. This button is available for Admin and Super Admin users only.

View mode buttons being common for all sections are described on the Monitoring page.

Using filter

You can use the filter to display only the hosts you are interested in. For better search performance, data is searched with macros unresolved.

The filter is located above the table. It is possible to filter hosts by name, host group, IP or DNS, interface port, tags, problem severity, status (enabled/disabled/any); you can also select whether to display suppressed problems and hosts that are currently in maintenance.



Parameter	Description
Name	Filter by visible host name.
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
IP	Filter by IP address.
DNS	Filter by DNS name.
Port	Filter by port number.
Severity	Filter by problem severity. By default problems of all severities are displayed. Problems are displayed if not suppressed.
Status	Filter by host status.
Tags	Filter by host tag name and value. Hosts can be filtered by host-level tags as well as tags from all linked templates, including nested templates. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met
Show hosts in maintenance	Mark the checkbox to display hosts that are in maintenance (displayed by default).
Show suppressed problems	Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance or single problem suppression.

Saving filter

Favorite filter settings can be saved as tabs and then quickly accessed by clicking on the respective tab above the filter.

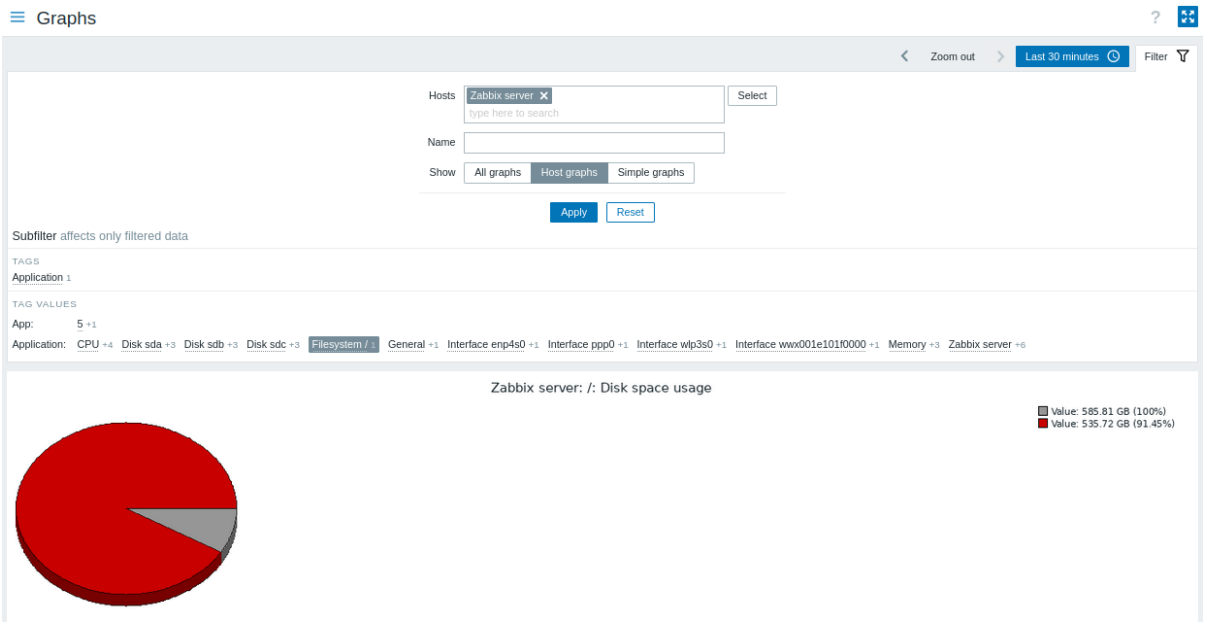
See more details about saving filters.

1 Graphs

Overview

Host graphs can be accessed from *Monitoring* → *Hosts* by clicking on Graphs for the respective host.

Any custom graph that has been configured for the host can be displayed, as well as any simple graph.



Graphs are sorted by:

- graph name (custom graphs)
- item name (simple graphs)

Graphs for disabled hosts are also accessible.

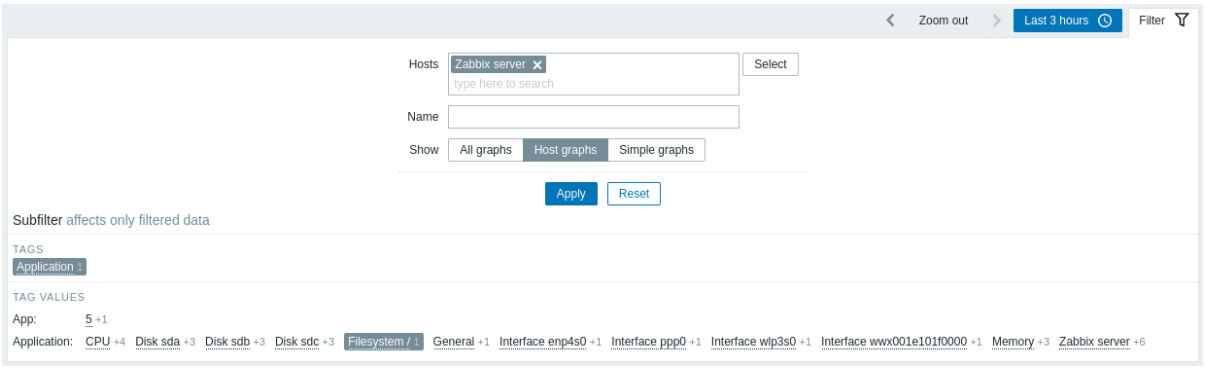
Time period selector

Take note of the time period selector above the graph. It allows selecting often required periods with one mouse click.

See also: Time period selector

Using filter

To view a specific graph, select it in the filter. The filter allows to specify the host, the graph name and the *Show* option (all/host graphs/simple graphs).



If no host is selected in the filter, no graphs are displayed.

Using subfilter

The subfilter allows to further modify the filtering from the main filter.

It contains clickable links for a quick access to related graphs. Graphs are related by common entity - tag name or value. When a tag name/value is clicked, it is highlighted with a gray background, and graphs are immediately filtered (no need to click *Apply* in the main filter). Clicking another tag name/value adds it to the filtered results. Clicking the tag name/value again removes the filtering.

Subfilters are generated based on the filtered data, which is limited to 1000 records. If you want to see more records in the subfilter, you need to increase the value of *Limit for search and filter results* parameter (in *Administration -> General -> GUI*).

Unlike the main filter, the subfilter is updated together with each table refresh request to always get up-to-date information of available filtering options and their counter numbers.

The number of entities displayed is limited to 100 horizontally. If there are more, a three-dot icon is displayed at the end; it is not clickable. Vertical lists (such as tags with their values) are limited to 20 entries. If there are more, a three-dot icon is displayed; it is not clickable.

A number next to each clickable entity indicates the number of graphs it has in the results of the main filter.

Once one entity is selected, the numbers with other available entities are displayed with a plus sign indicating how many graphs may be added to the current selection.

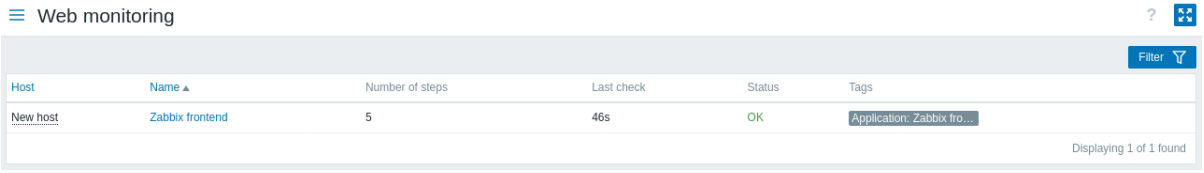
Buttons

View mode buttons, being common for all sections, are described on the Monitoring page.

2 Web scenarios

Overview

Host web scenario information can be accessed from *Monitoring → Hosts* by clicking on Web for the respective host.



Clicking on the host name brings up the host menu. Data of disabled hosts is also accessible. The name of a disabled host is listed in red.

The maximum number of scenarios displayed per page depends on the *Rows per page* user profile setting.

By default, only values that fall within the last 24 hours are displayed. This limit has been introduced with the aim of improving initial loading times for large pages of latest data. You can extend this time period by changing the value of *Max history display period* parameter in the *Administration → General → GUI* menu section.

The scenario name is link to more detailed statistics about it:



Using filter

The page shows a list of all web scenarios of the selected host. To view web scenarios for another host or host group without returning to the *Monitoring* → *Hosts* page, select that host or group in the filter. You may also filter scenarios based on tags.

Buttons

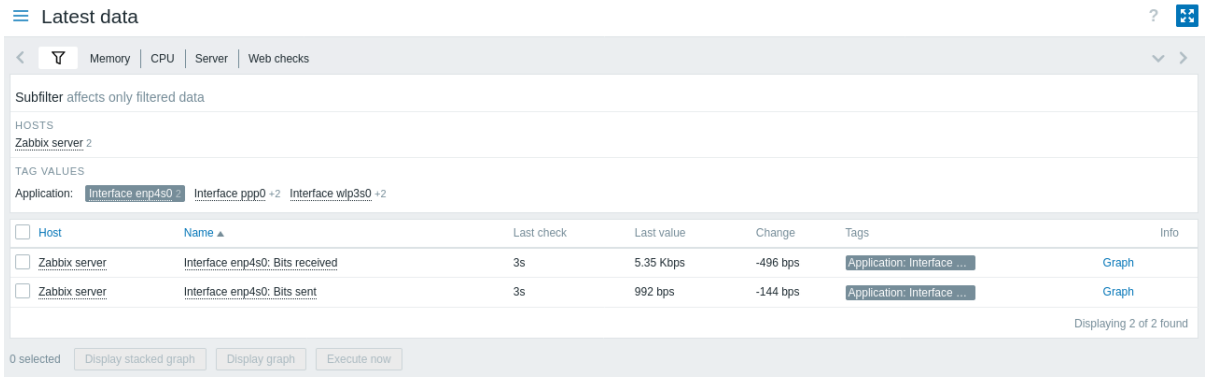
View mode buttons being common for all sections are described on the Monitoring page.

3 Latest data

Overview

In this section you can view the latest values gathered by items.

Graphs are also available for the item values.



This section contains:

- the filter (collapsed by default)
- the subfilter (never collapsed)
- the item list

Items are displayed with their name, time since the last check, **last value**, change amount, tags, and a link to a simple graph/history of item values.

Values in the *Last value* column are displayed with unit conversion and value mapping applied. To view raw data, hover over the value.

Tags in the item list are clickable. If you click on a tag, this tag becomes enabled in the subfilter. The item list now displays the items corresponding to this tag and any other previously selected tags in the subfilter. Note that once the items have been filtered in this way, tags in the list are no longer clickable. Further modification based on tags (e.g. remove, add another filter) must be done in the subfilter.

If an item has errors, for example, has become unsupported, an information icon will be displayed in the *Info* column. Hover over the icon for details.

An icon with a question mark is displayed next to the item name for all items that have a description. Hover over this icon to see a tooltip with the item description.

Clicking on the host name in the *Host* column brings up the host context menu.

Clicking on the item name in the *Item* column brings up the Item context menu.

If a host to which the item belongs is in maintenance, an orange wrench icon is displayed after the host's name.

Note: The name of a disabled host is displayed in red. Data of disabled hosts, including graphs and item value lists, is also accessible in *Latest data*.

By default, only values that fall within the last 24 hours are displayed. This limit has been introduced with the aim of improving initial loading times for large pages of the latest data. This time period can be extended by changing the value of the *Max history display period* parameter in Administration → General → GUI.

Attention:
For items with an update frequency of 1 day or more the change amount will never be displayed (with the default setting). Also in this case the last value will not be displayed at all if it was received more than 24 hours ago.

Buttons

View mode buttons being common for all sections are described on the Monitoring page.

Mass actions

Buttons below the list offer mass actions with one or several selected items:

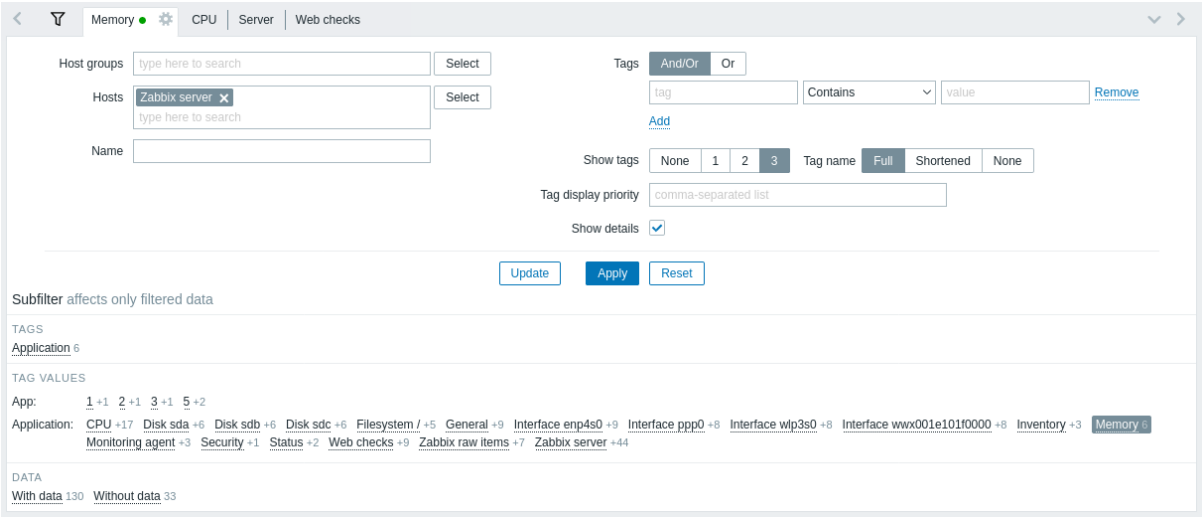
- *Display stacked graph* - display a stacked ad-hoc graph
- *Display graph* - display a simple ad-hoc graph
- *Execute now* - execute a check for new item values immediately. Supported for **passive** checks only (see more details). This option is available only for hosts with read-write access. Accessing this option for hosts with read-only permissions depends on the user role option called *Invoke "Execute now" on read-only hosts*.

To use these options, mark the checkboxes before the respective items, then click on the required button.

Using filter

You can use the filter to display only the items you are interested in. For better search performance, data is searched with macros unresolved.

The filter icon is located above the table and the subfilter. Click on it to expand the filter.



The filter allows to narrow the list by host group, host, item name, tag and other settings. Specifying a parent host group in the filter implicitly selects all nested host groups. See *Monitoring* -> *Problems* for details on filtering by tags.

Show details allows to extend the information displayed for the items. Such details as the refresh interval, history and trends settings, item type, and item errors (fine/unsupported) are displayed.

Saving filter

Favorite filter settings can be saved as tabs and then quickly accessed by clicking on the respective tab above the filter.

See more details about saving filters.

Using subfilter

The subfilter allows to further modify the filtering from the main filter.

It contains clickable links for a quick access to related items. Items are related by common entity - host, tag name or value, item state, or data status. When an entity is clicked, the entity is highlighted with a gray background, and items are immediately filtered (no need to click *Apply* in the main filter). Clicking another entity adds it to the filtered results. Clicking the entity again removes the filtering.

Subfilters are generated based on the filtered data, which is limited to 1000 records. If there are 20 hosts each having 100 items (so 2000 records in total), only half of the hosts will be visible in the subfilter. If you want to see more records in the subfilter, you need to increase the value of *Limit for search and filter results* parameter (in *Administration -> General -> GUI*).

Unlike the main filter, the subfilter is updated together with each table refresh request to always get up-to-date information of available filtering options and their counter numbers.

For each entity group (e.g. tags, hosts) up to 10 rows of entities are displayed. If there are more entities, this list can be expanded to a maximum of 1000 entries (the value of SUBFILTER_VALUES_PER_GROUP in frontend definitions) by clicking on a three-dot icon displayed at the end. Once expanded to the maximum, the list cannot be collapsed.

In the list of *Tag values* up to 10 rows of tag names are displayed. If there are more tag names with values, this list can be expanded to a maximum of 200 tag names by clicking on a three-dot icon displayed at the bottom. Once expanded to the maximum, the list cannot be collapsed.

For each tag name up to 10 rows of values are displayed (expandable to 1000 entries (the value of SUBFILTER_VALUES_PER_GROUP in frontend definitions)).

The host options in the subfilter are available only if no hosts or more than one host is selected in the main filter.

By default, items with and without data are displayed in the item list. If only one host is selected in the main filter, the subfilter offers the option to filter only items with data, only without data, or both for this host.

A number next to each clickable entity indicates the number of items it has in the results of the main filter. Entities without items are not displayed, unless they were selected in the subfilter before.

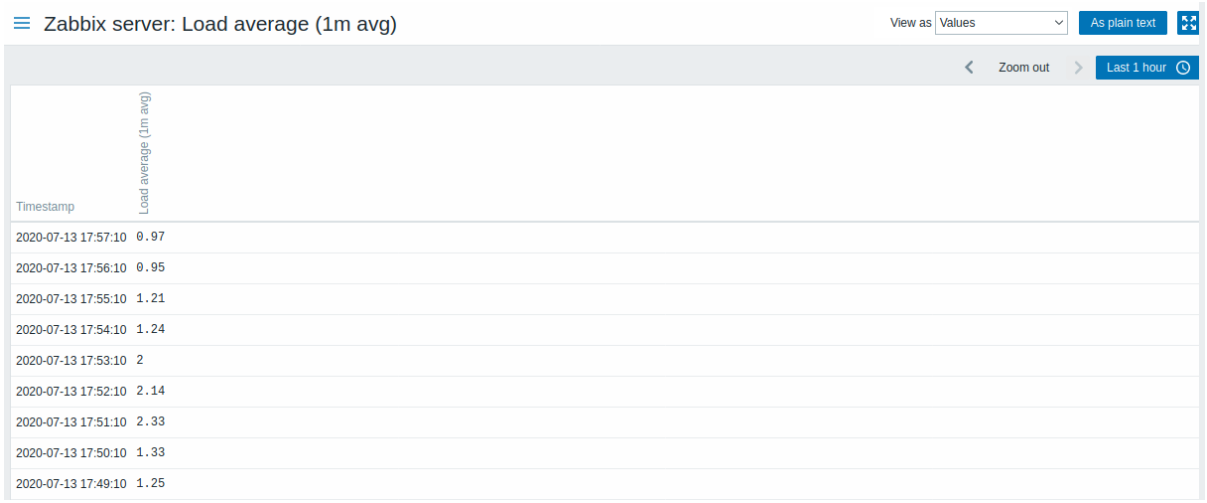
Once one entity is selected, the numbers with other available entities are displayed with a plus sign indicating how many items may be added to the current selection.

Graphs

Links to value history/simple graph

The last column in the latest value list offers:

- a **History** link (for all textual items) - leading to listings (*Values/500 latest values*) displaying the history of previous item values.
- a **Graph** link (for all numeric items) - leading to a simple graph. However, once the graph is displayed, a dropdown on the upper right offers a possibility to switch to *Values/500 latest values* as well.



The values displayed in this list are ”raw”, that is, no postprocessing is applied.

Note:
The total amount of values displayed is defined by the value of *Limit for search and filter results* parameter, set in Administration → General → GUI.

4 Maps

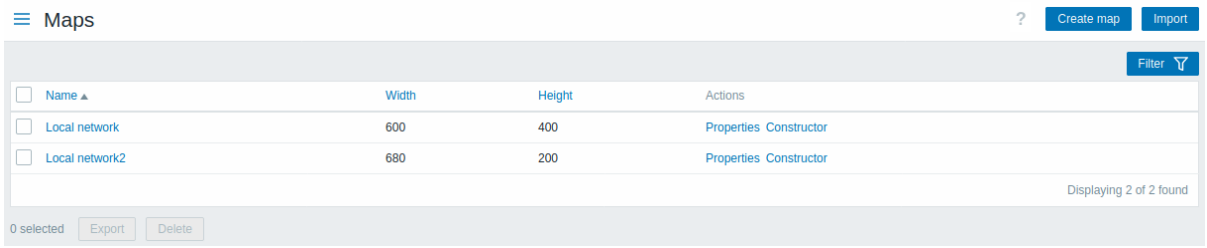
Overview

In the *Monitoring* → *Maps* section you can configure, manage and view network maps.

When you open this section, you will either see the last map you accessed or a listing of all maps you have access to.

All maps can be either public or private. Public maps are available to all users, while private maps are accessible only to their owner and the users the map is shared with.

Map listing



Displayed data:

Column	Description
Name	Name of the map. Click on the name to view the map.
Width	Map width is displayed.
Height	Map height is displayed.
Actions	Two actions are available: **Properties** - edit general map properties **Constructor** - access the grid for adding map elements

To configure a new map, click on the *Create map* button in the top right-hand corner. To import a map from a YAML, XML, or JSON file, click on the *Import* button in the top right-hand corner. The user who imports the map will be set as its owner.

Two buttons below the list offer some mass-editing options:

- *Export* - export the maps to a YAML, XML, or JSON file
- *Delete* - delete the maps

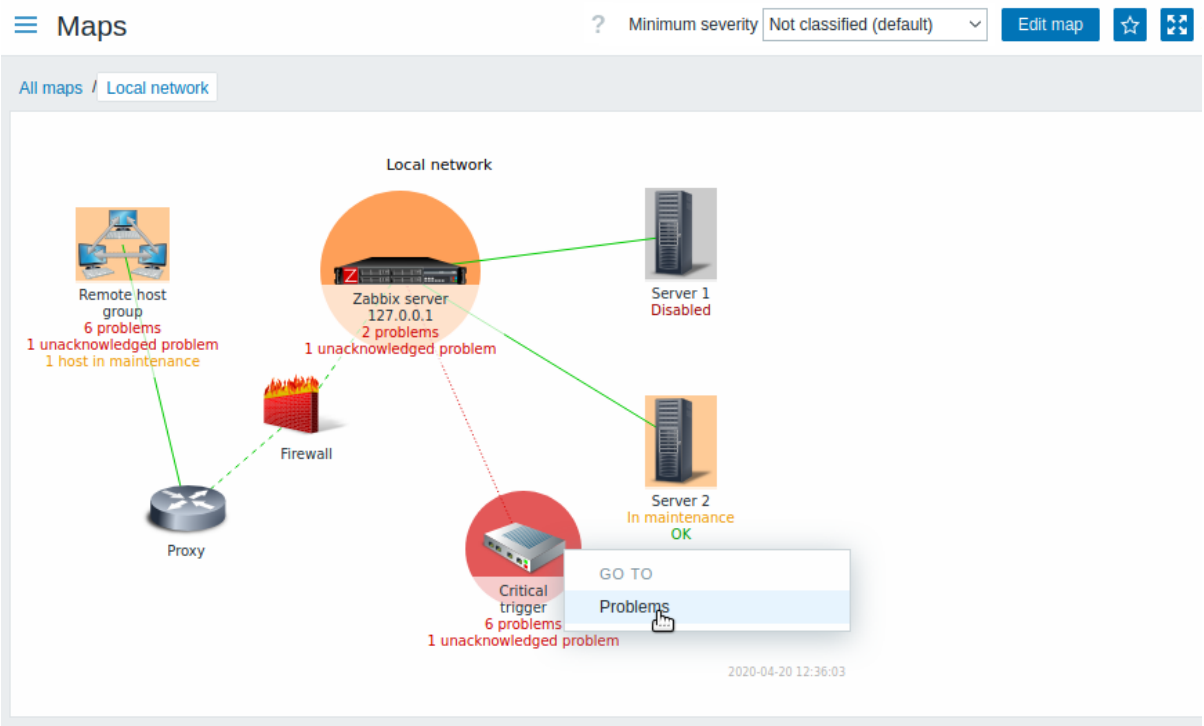
To use these options, mark the checkboxes before the respective maps, then click on the required button.

Using filter

You can use the filter to display only the maps you are interested in. For better search performance, data is searched with macros unresolved.

Viewing maps

To view a map, click on its name in the list of all maps.



You can use the drop-down in the map title bar to select the lowest severity level of the problem triggers to display. The severity marked as *default* is the level set in the map configuration. If the map contains a sub-map, navigating to the sub-map will retain the higher-level map severity (except if it is *Not classified*, in this case, it will not be passed to the sub-map).

Icon highlighting

If a map element is in problem status, it is highlighted with a round circle. The fill color of the circle corresponds to the severity color of the problem. Only problems on or above the selected severity level will be displayed with the element. If all problems are acknowledged, a thick green border around the circle is displayed.

Additionally:

- a host in maintenance is highlighted with an orange, filled square. Note that maintenance highlighting has priority over the problem severity highlighting, if the map element is host.
- a disabled (not-monitored) host is highlighted with a gray, filled square.

Highlighting is displayed if the *Icon highlighting* check-box is marked in map configuration.

Recent change markers

Inward pointing red triangles around an element indicate a recent trigger status change - one that's happened within the last 30 minutes. These triangles are shown if the *Mark elements on trigger status change* check-box is marked in map configuration.

Links

Clicking on a map element opens a menu with some available links. Clicking on the host name brings up the host menu.

Buttons

Buttons to the right offer the following options:

Edit map	Go to map constructor to edit the map content.
☆	Add map to the favorites widget in Dashboards.
★	The map is in the favorites widget in Dashboards. Click to remove map from the favorites widget.

View mode buttons being common for all sections are described on the Monitoring page.

Readable summary in maps

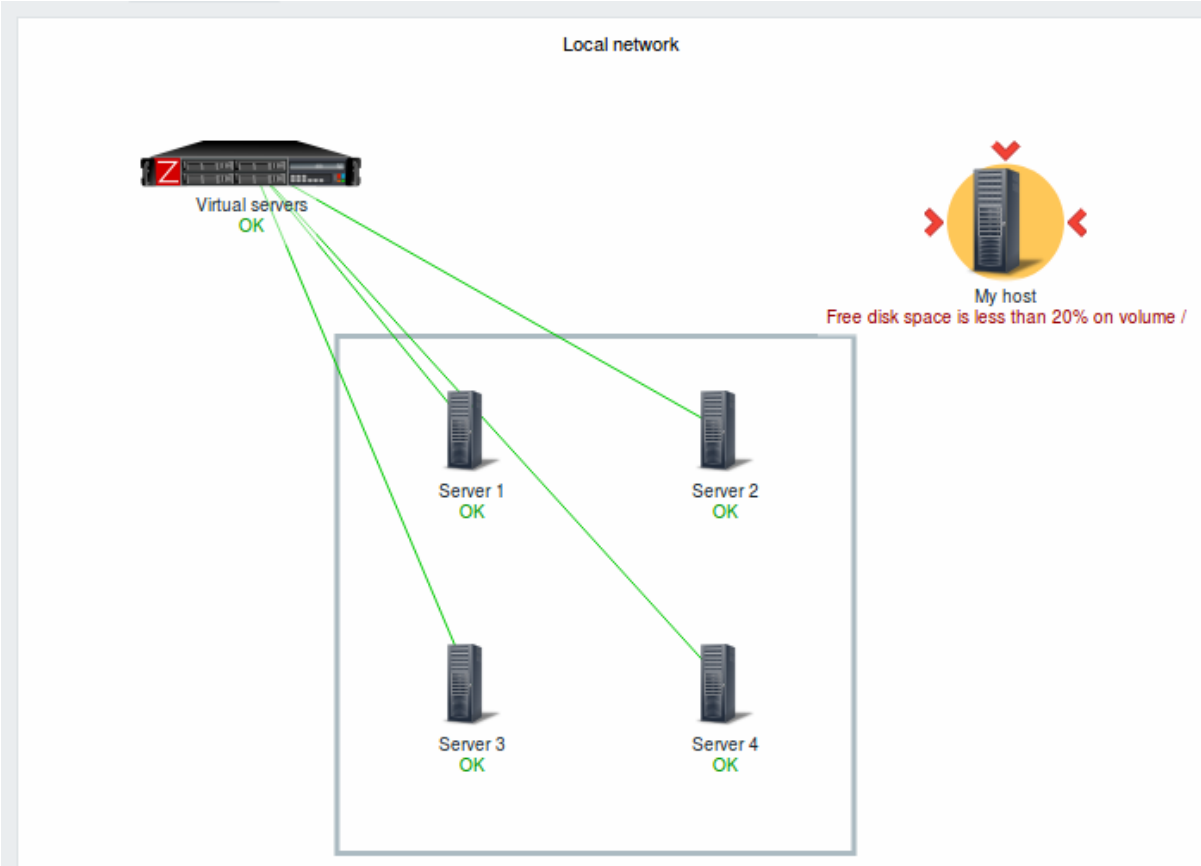
A hidden "aria-label" property is available allowing map information to be read with a screen reader. Both general map description and individual element description is available, in the following format:

- for map description: `<Map name>, <* of * items in problem state>, <* problems in total>.`
- for describing one element with one problem: `<Element type>, Status <Element status>, <Element name>, <Problem description>.`
- for describing one element with multiple problems: `<Element type>, Status <Element status>, <Element name>, <* problems>.`
- for describing one element without problems: `<Element type>, Status <Element status>, <Element name>.`

For example, this description is available:

```
'Local network, 1 of 6 elements in problem state, 1 problem in total. Host, Status problem, My host, Free
```

for the following map:



Referencing a network map

Network maps can be referenced by both `sysmapid` and `mapname` GET parameters. For example,

`http://zabbix/zabbix/zabbix.php?action=map.view&mapname=Local%20network`

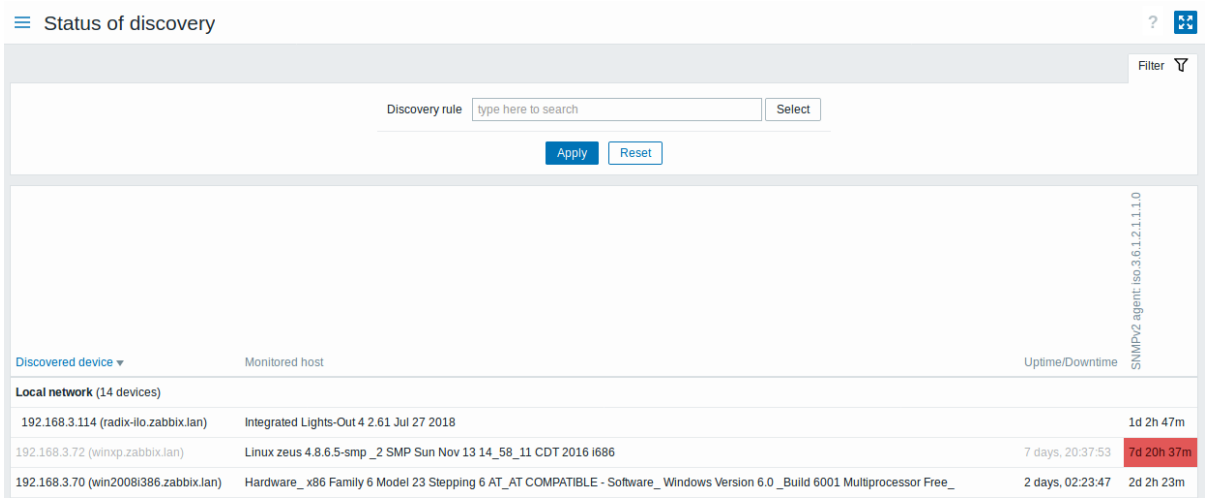
will open the map with that name (Local network).

If both `sysmapid` (map ID) and `mapname` (map name) are specified, `mapname` has higher priority.

5 Discovery

Overview

In the *Monitoring* → *Discovery* section results of network discovery are shown. Discovered devices are sorted by the discovery rule.



If a device is already monitored, the host name will be listed in the *Monitored host* column, and the duration of the device being discovered or lost after previous discovery is shown in the *Uptime/Downtime* column.

After that follow the columns showing the state of individual services for each discovered device (red cells show services that are down). Service uptime or downtime is included within the cell.

Attention:
Only those services that have been found on at least one device will have a column showing their state.

Buttons

View mode buttons being common for all sections are described on the Monitoring page.

Using filter

You can use the filter to display only the discovery rules you are interested in. For better search performance, data is searched with macros unresolved.

With nothing selected in the filter, all enabled discovery rules are displayed. To select a specific discovery rule for display, start typing its name in the filter. All matching enabled discovery rules will be listed for selection. More than one discovery rule can be selected.

3 Services

Overview

The Services menu is for the service monitoring functions of Zabbix.

1 Services

Overview

In this section you can see a high-level status of whole services that have been configured in Zabbix, based on your infrastructure.

A service may be a hierarchy consisting of several levels of other services, called ”child” services, which are attributes to the overall status of the service (see also an overview of the service monitoring functionality.)

The main categories of service status are *OK* or *Problem*, where the *Problem* status is expressed by the corresponding problem severity name and color.

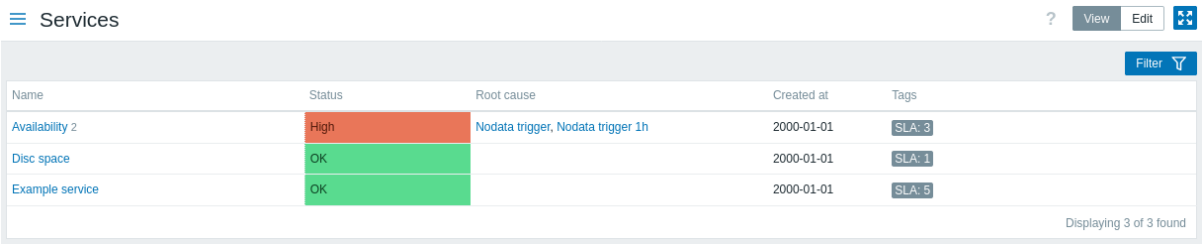
While the view mode allows to monitor services with their status and other details, you can also configure the service hierarchy in this section (add/edit services, child services) by switching to the edit mode.

To switch from the view to the edit mode (and back) click on the respective button in the upper right corner:

- View - view services
- Edit - add/edit services, and child services

Note that access to editing depends on user role settings.

Viewing services



A list of the existing services is displayed.

Displayed data:

Parameter	Description
Name	Service name. The service name is a link to service details. The number after the name indicates how many child services the service has.
Status	Service status: **OK** - no problems **(trigger color and severity)** - indicates a problem and its severity. If there are multiple problems, the color and severity of the problem with highest severity is displayed.
Root cause	Underlying problems that directly or indirectly affect the service status are listed. The same problems are listed as returned by the {SERVICE.ROOTCAUSE} macro. Click on the problem name to see more details about it in *Monitoring* → *Problems*. Problems that do not affect the service status are not in the list.
Created at	The time when the service was created is displayed.
Tags	Tagsof the service are displayed. Tags are used to identify a service in service actions and SLAs.

Buttons

View mode buttons being common for all sections are described on the Monitoring page.

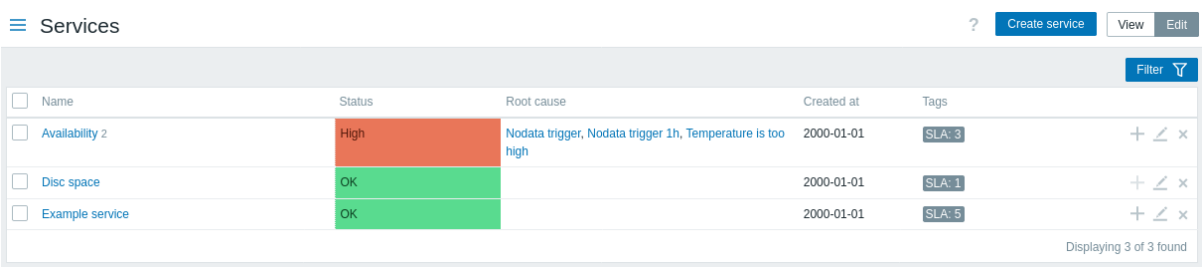
Using filter

You can use the filter to display only the services you are interested in.

Editing services

Click on the *Edit* button to access the edit mode. When in edit mode, the listing is complemented with checkboxes before the entries and also these additional options:

- - add a child service to this service
- - edit this service
- - delete this service

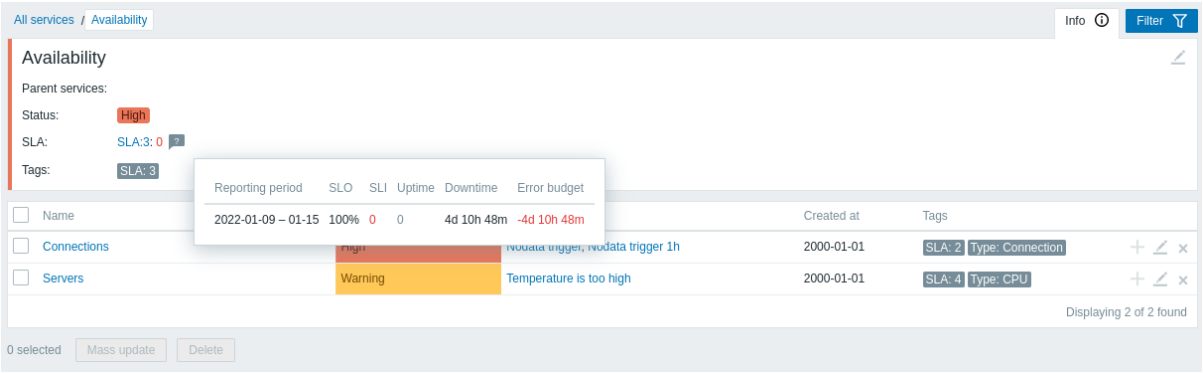


To configure a new service, click on the *Create service* button in the top right-hand corner.

Service details

To access service details, click on the service name. To return to the list of all services, click on *All services*.

Service details include the info box and the list of child services.



To access the info box, click on the *Info* tab. The info box contains the following entries:

- Names of parent services (if any)
- Current status of this service
- Current SLA(s) of this service, in the format `SLA name:service level indicator`. 'SLA name' is also a link to the SLA report for this service. If you position the mouse on the info box next to the service-level indicator (SLI), a pop-up info list is displayed with SLI details. The service-level indicator displays the current service level, in percentage.
- Service tags

The info box also contains a link to the service configuration.

To use the filter for child services, click on the *Filter* tab.

When in edit mode, the child service listing is complemented with additional editing options:

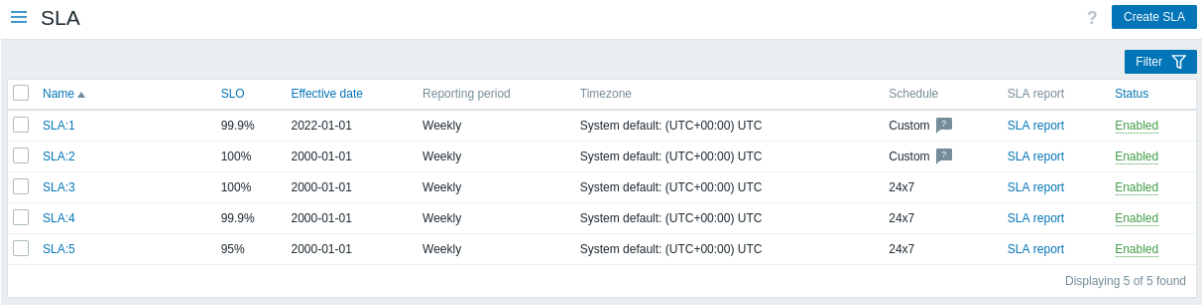
-  - add a child service to this service
-  - edit this service
-  - delete this service

2 SLA

Overview

This section allows to view and configure SLAs.

SLAs



A list of the configured SLAs is displayed. *Note* that only the SLAs related to services accessible to the user will be displayed (as read-only, unless *Manage SLA* is enabled for the user role).

Displayed data:

Parameter	Description
Name	The SLA name is displayed. The name is a link to SLA configuration.
SLO	The service level objective (SLO) is displayed.
Effective date	The date of starting SLA calculation is displayed.
Reporting period	The period used in the SLA report is displayed - *daily*, *weekly*, *monthly*, *quarterly*, or *annually*.
Time zone	The SLA time zone is displayed.

Parameter	Description
Schedule	The SLA schedule is displayed - 24x7 or custom.
SLA report	Click on the link to see the SLA report for this SLA.
Status	The SLA status is displayed - enabled or disabled.

3 SLA report

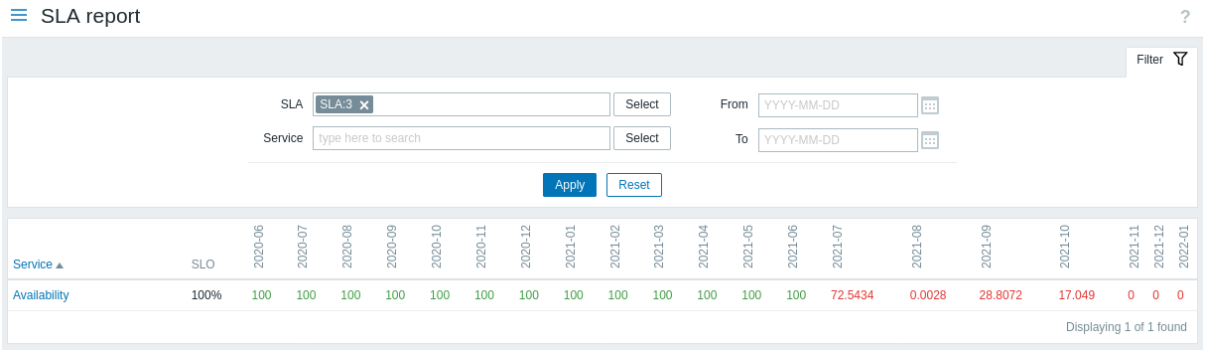
Overview

This section allows to view SLA reports, based on the criteria selected in the filter.

SLA reports can also be displayed as a dashboard widget.

Report

The filter allows to select the report based on the SLA name as well as the service name. It is also possible to limit the displayed period.

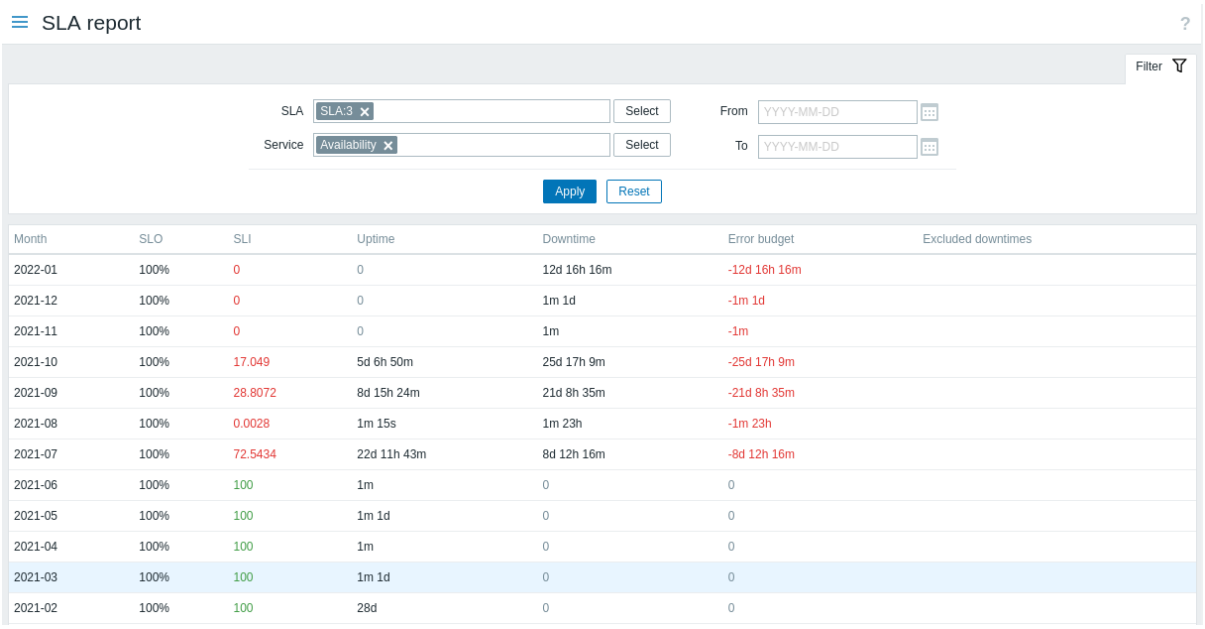


Each column (period) displays the SLI for that period. SLIs that are in breach of the set SLO are highlighted in red.

20 periods are displayed in the report. A maximum of 100 periods can be displayed, if both the *From* date and *To* date are specified.

Report details

If you click on the service name in the report, you can access another report that displays a more detailed view.



Note that negative problem duration does not affect SLA calculation or reporting.

4 Inventory

Overview

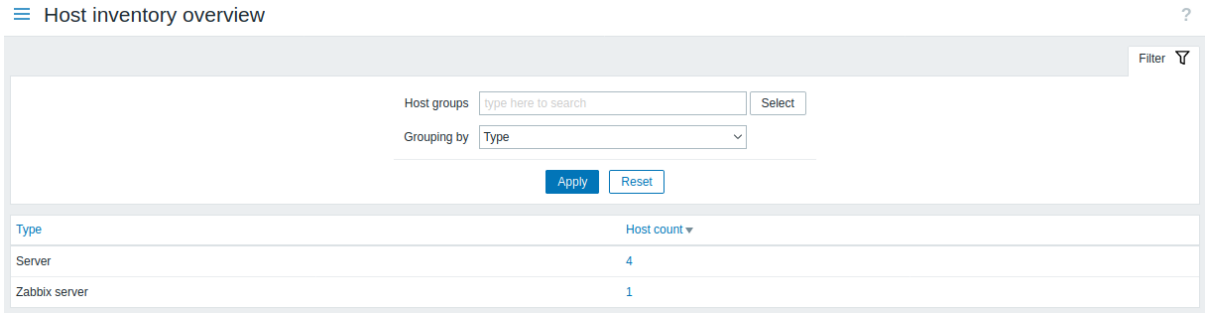
The Inventory menu features sections providing an overview of host inventory data by a chosen parameter as well as the ability to view host inventory details.

1 Overview

Overview

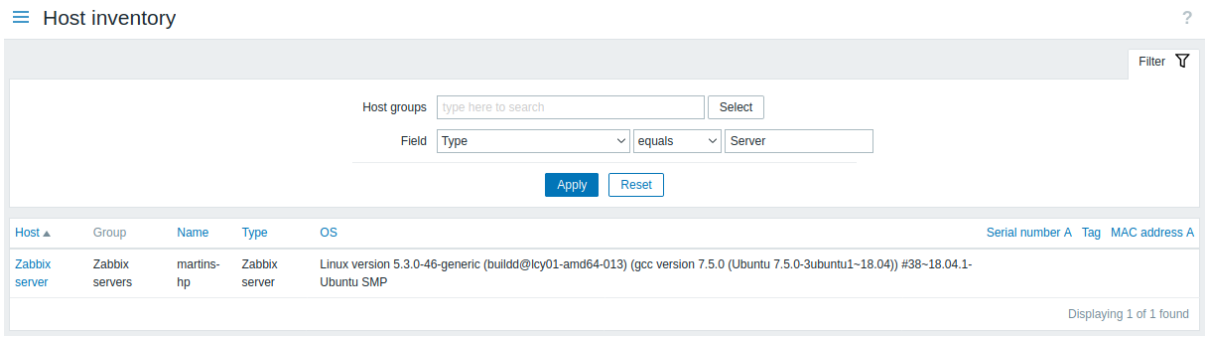
The *Inventory* → *Overview* section provides ways of having an overview of host inventory data.

For an overview to be displayed, choose host groups (or none) and the inventory field by which to display data. The number of hosts corresponding to each entry of the chosen field will be displayed.



The completeness of an overview depends on how much inventory information is maintained with the hosts.

Numbers in the *Host count* column are links; they lead to these hosts being filtered out in the *Host Inventories* table.

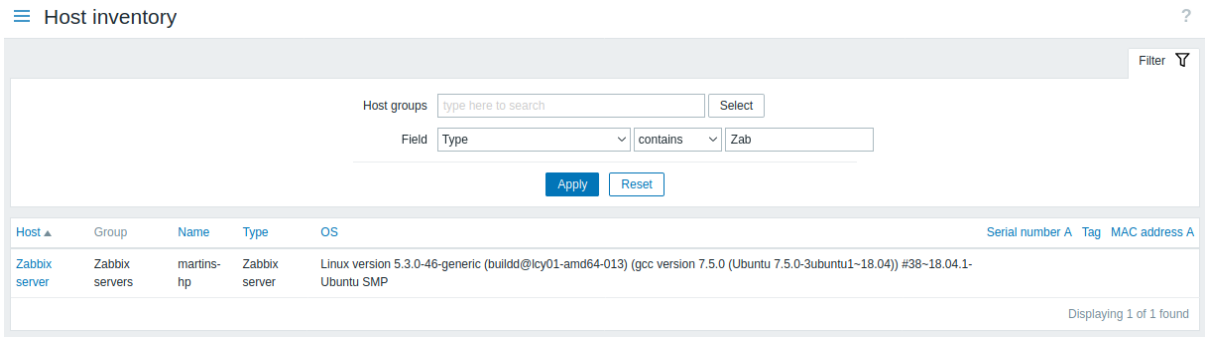


2 Hosts

Overview

In the *Inventory* → *Hosts* section inventory data of hosts are displayed.

You can filter the hosts by host group(s) and by any inventory field to display only the hosts you are interested in.

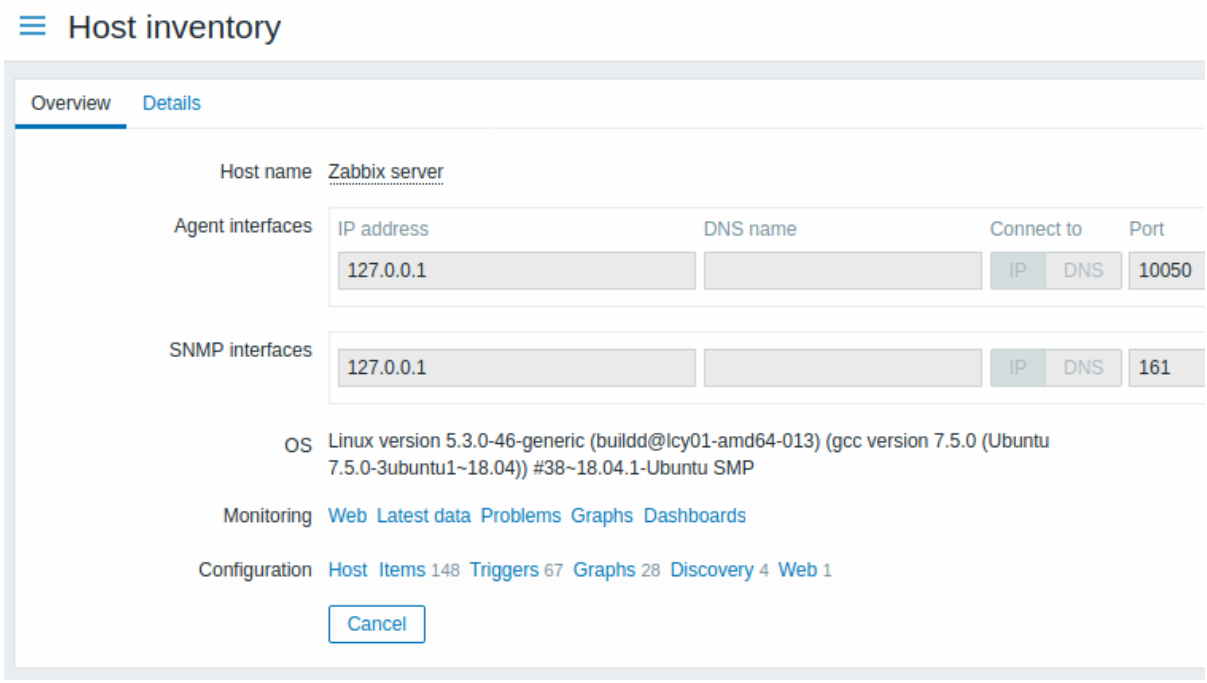


To display all host inventories, select no host group in the filter, clear the comparison field in the filter and press ”Filter”.

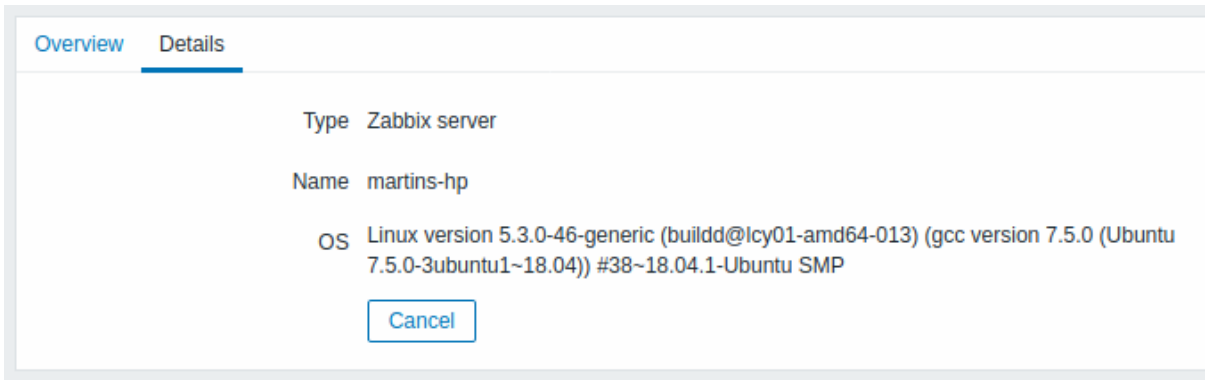
While only some key inventory fields are displayed in the table, you can also view all available inventory information for that host. To do that, click on the host name in the first column.

Inventory details

The **Overview** tab contains some general information about the host along with links to predefined scripts, latest monitoring data and host configuration options:



The **Details** tab contains all available inventory details for the host:



The completeness of inventory data depends on how much inventory information is maintained with the host. If no information is maintained, the *Details* tab is disabled.

5 Reports

Overview

The Reports menu features several sections that contain a variety of predefined and user-customizable reports focused on displaying an overview of such parameters as system information, triggers and gathered data.

1 System information

Overview

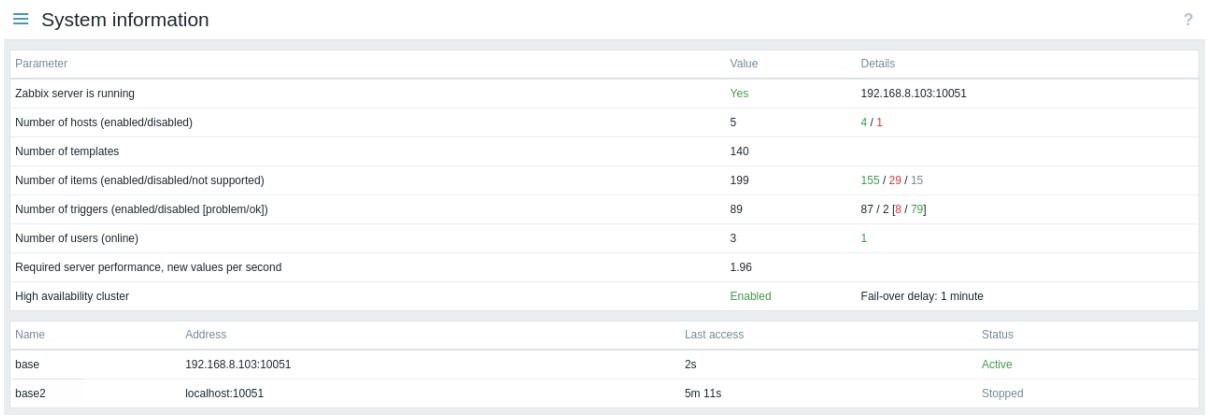
In *Reports* → *System information*, a summary of key Zabbix server and system data is displayed. System data is collected using internal items.

Note that in a high availability setup, it is possible to redirect the system information source (server instance). To do this, edit the *zabbix.conf.php* file - uncomment and set `$ZBX_SERVER` or both `$ZBX_SERVER` and `$ZBX_SERVER_PORT` to a server other than the one shown active. Note that when setting `$ZBX_SERVER` only, a default value (10051) for `$ZBX_SERVER_PORT` will be used.

With the high availability setup enabled, a separate block is displayed below the system stats with details of high availability nodes. This block is visible to Zabbix *Super Admin* users only.

System information is also available as a dashboard widget.

System stats



Displayed data:

Parameter	Value	Details
Zabbix server is running	Status of Zabbix server: **Yes** - server is running **No** - server is not running *Note:* To display the rest of the information the web, frontend needs the server to be running and there must be at least one trapper process started on the server (StartTrappers parameter in zabbix_server.conf file > 0).	Location and port of Zabbix server.
Number of hosts *Number of templates*	Total number of hosts configured is displayed. Total number of templates is displayed.	Number of monitored hosts/not monitored hosts.
Number of items	Total number of items is displayed.	Number of monitored/disabled/unsupported host-level items. Items on disabled hosts are counted as disabled.
Number of triggers	Total number of triggers is displayed.	Number of enabled/disabled host-level triggers; split of the enabled triggers according to "Problem"/"OK" states. Triggers listed under the "OK" state include triggers with the status "Unknown". Triggers dependent on disabled items or assigned to disabled hosts are counted as disabled.
Number of users *Required server performance, new values per second*	Total number of users configured is displayed. The expected number of new values processed by Zabbix server per second is displayed.	Number of users online. *Required server performance* is an estimate and can be useful as a guideline. For precise numbers of values processed, use the `zabbix[wcache,values,all]` internal item. Enabled items from monitored hosts are included in the calculation. Log items are counted as one value per item update interval. Regular interval values are counted; flexible and scheduling interval values are not. The calculation is not adjusted during a "nodata" maintenance period. Trapper items are not counted.
Database history tables upgraded	Database upgrade status: **No** - database history tables have not been upgraded	Support for the old numeric type is deprecated. Please upgrade to numeric values of extended range. This field is displayed if database upgrade to extended range for numeric (float) values has not been completed. See instructions for enabling an extended range of numeric (float) values.

Parameter	Value	Details
High availability cluster	Status of high availability cluster for Zabbix server: **disabled** - standalone server **enabled** - at least one high availability node exists	If enabled, the failover delay is displayed.

System information will also display an error message in the following conditions:

- The database used does not have the required character set or collation (UTF-8).
- The version of the database is below or above the supported range (available only to users with the *Super admin role* type).
- Housekeeping for TimescaleDB is incorrectly configured (history or trend tables contain compressed chunks, but *Override item history period* or *Override item trend period* options are disabled).

High availability nodes

If high availability cluster is enabled, then another block of data is displayed with the status of each high availability node.



Name	Address	Last access	Status
node-active	192.168.1.13:10051	12s	Active
node6	192.168.1.10:10053	1h 2m 40s	Unavailable
node7	192.168.1.11:10053	3m 40s	Unavailable
node4	192.168.1.8:10052	1h 34m 29s	Stopped
node5	192.168.1.9:10053	1h 9m 51s	Stopped
node8	192.168.1.12:10051	21m 16s	Stopped
node1	192.168.1.5:10051	17s	Standby
node2	192.168.1.6:10051	16s	Standby
node3	192.168.1.7:10052	16 2021-10-20 17:58:47	Standby

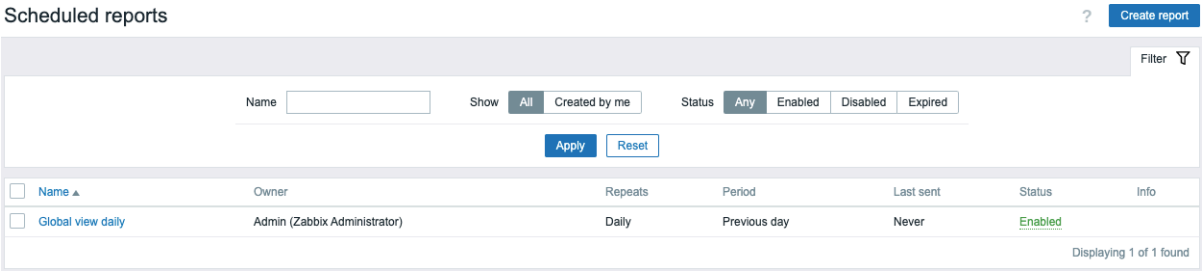
Displayed data:

Column	Description
Name	Node name, as defined in server configuration.
Address	Node IP address and port.
Last access	Time of node last access. Hovering over the cell shows the timestamp of last access in long format.
Status	Node status: **Active** - node is up and working **Unavailable** - node hasn't been seen for more than failover delay (you may want to find out why) **Stopped** - node has been stopped or couldn't start (you may want to start it or delete it) **Standby** - node is up and waiting

2 Scheduled reports

Overview

In *Reports* → *Scheduled reports*, users with sufficient permissions can configure scheduled generation of PDF versions of the dashboards, which will be sent by email to specified recipients.



The opening screen displays information about scheduled reports, which can be filtered out for easy navigation - see Using filter section below.

Displayed data:

Column	Description
Name	Name of the report. Clicking it opens the report configuration form.
Owner	User who created the report.
Repeats	Report generation frequency (daily/weekly/monthly/yearly).
Period	Period for which the report is prepared.
Last sent	The date and time when the latest report has been sent.
Status	Current status of the report (enabled/disabled/expired). Users with sufficient permissions can change the status by clicking it - from ”Enabled” to ”Disabled” (and back); from ”Expired” to ”Disabled” (and back). For users with insufficient rights, the status is not clickable.
Info	Displays informative icons: A red icon indicates that report generation has failed; hovering over it will display a tooltip with the error information. A yellow icon indicates that a report was generated, but sending to some (or all) recipients has failed or that a report is expired; hovering over it will display a tooltip with additional information.

Using filter

You may use the filter to narrow down the list of reports. For better search performance, data is searched with macros unresolved.

The following filtering options are available:

- *Name* - partial name match is allowed
- *Report owner* - created by current user or all reports
- *Status* - select between ”Any” (show all reports), ”Enabled”, ”Disabled”, or ”Expired”

The filter is located above the *Scheduled reports* bar. It can be opened and collapsed by clicking the *Filter* tab in the upper right corner.

Mass update

Sometimes you may want to delete or change the status of a number of reports at once. Instead of opening each individual report for editing, you may use the mass update function for that.

To mass-update some reports, do the following:

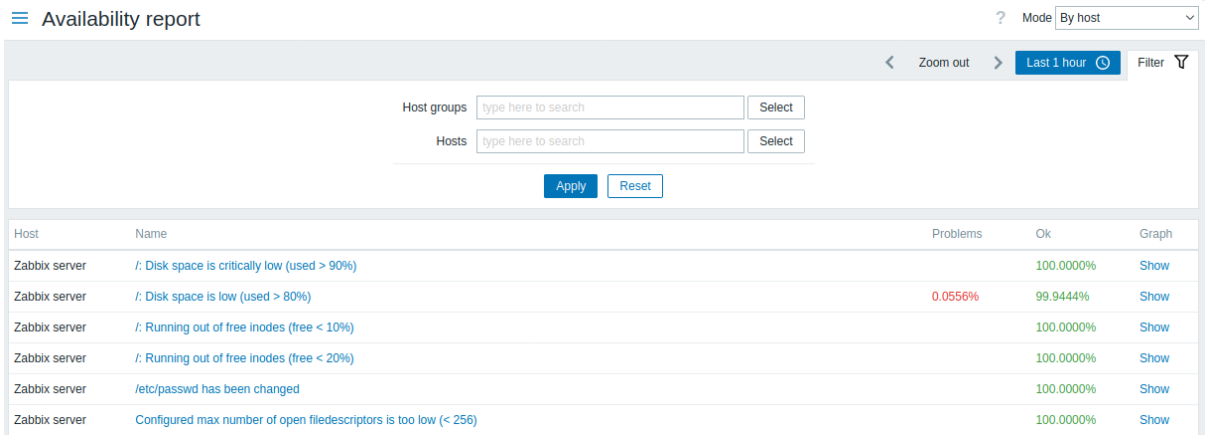
- Mark the checkboxes of the reports to update in the list
- Click the required button below the list to make the changes (*Enable*, *Disable*, or *Delete*)

3 Availability report

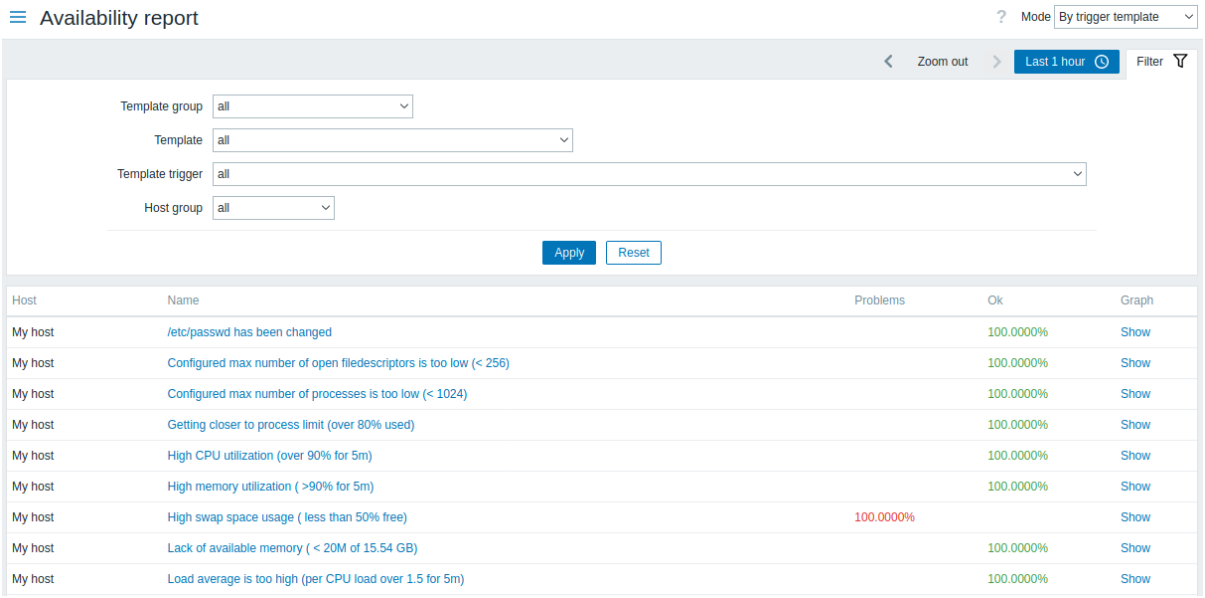
Overview

In *Reports* → *Availability report* you can see what proportion of time each trigger has been in problem/ok state. The percentage of time for each state is displayed.

Thus it is easy to determine the availability situation of various elements on your system.



From the drop-down in the upper right corner, you can choose the selection mode - whether to display triggers by hosts or by triggers belonging to a template.



The name of the trigger is a link to the latest events of that trigger.

Using filter

The filter can help narrow down the number of hosts and/or triggers displayed. For better search performance, data is searched with macros unresolved.

The filter is located below the *Availability report* bar. It can be opened and collapsed by clicking on the *Filter* tab on the right.

Filtering by trigger template

In the *By trigger template* mode results can be filtered by one or several parameters listed below.

Parameter	Description
Template group	Filter hosts by triggers that are inherited from templates belonging to the selected template group.
Template	Filter hosts by triggers that are inherited from the selected template, including nested templates. If a nested template has its own triggers, those triggers will not be displayed.
Template trigger	Filter hosts by the selected trigger. Other triggers of the filtered hosts will not be displayed.
Host group	Filter hosts belonging to the selected host group.

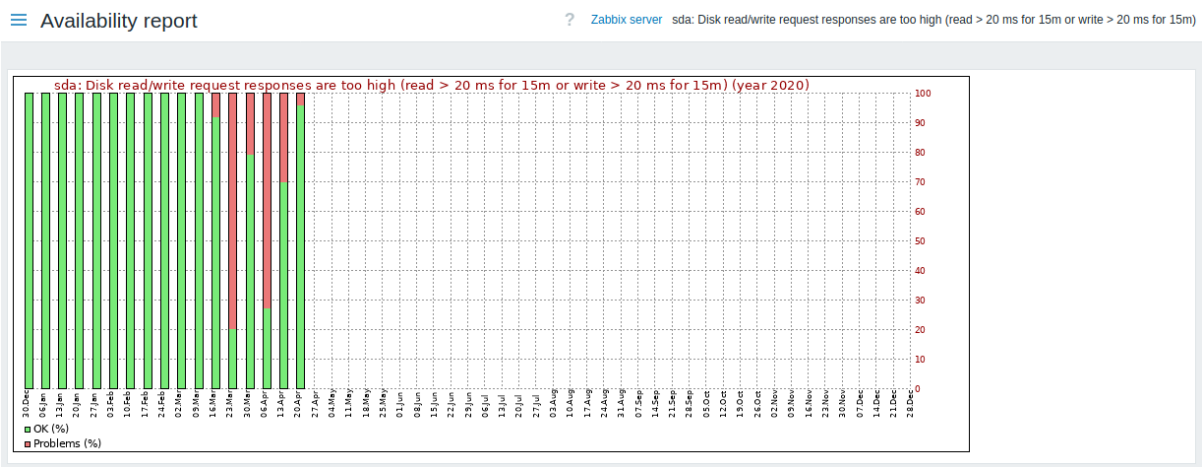
Filtering by host

In the *By host* mode results can be filtered by a host or by the host group. Specifying a parent host group implicitly selects all nested host groups.

Time period selector

The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

Clicking on *Show* in the Graph column displays a bar graph where availability information is displayed in bar format each bar representing a past week of the current year.

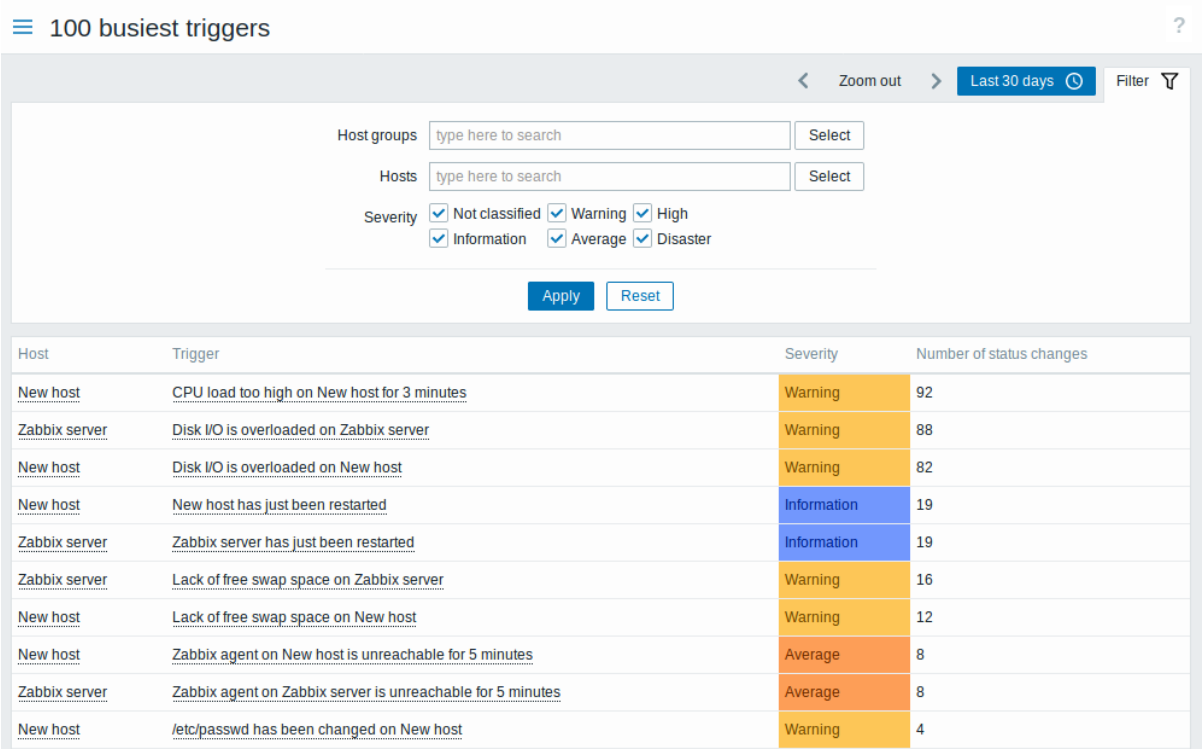


The green part of a bar stands for OK time and red for problem time.

4 Triggers top 100

Overview

In *Reports* → *Triggers top 100* you can see the triggers that have changed their state most often within the period of evaluation, sorted by the number of status changes.



Host	Trigger	Severity	Number of status changes
New host	CPU load too high on New host for 3 minutes	Warning	92
Zabbix server	Disk I/O is overloaded on Zabbix server	Warning	88
New host	Disk I/O is overloaded on New host	Warning	82
New host	New host has just been restarted	Information	19
Zabbix server	Zabbix server has just been restarted	Information	19
Zabbix server	Lack of free swap space on Zabbix server	Warning	16
New host	Lack of free swap space on New host	Warning	12
New host	Zabbix agent on New host is unreachable for 5 minutes	Average	8
Zabbix server	Zabbix agent on Zabbix server is unreachable for 5 minutes	Average	8
New host	/etc/passwd has been changed on New host	Warning	4

Both host and trigger column entries are links that offer some useful options:

- for host - clicking on the host name brings up the host menu
- for trigger - clicking on the trigger name brings up links to latest events, the trigger configuration form, and a simple graph

Using filter

You may use the filter to display triggers by host group, host, or trigger severity. Specifying a parent host group implicitly selects all nested host groups. For better search performance, data is searched with macros unresolved.

The filter is located below the *100 busiest triggers* bar. It can be opened and collapsed by clicking on the *Filter* tab on the left.

Time period selector

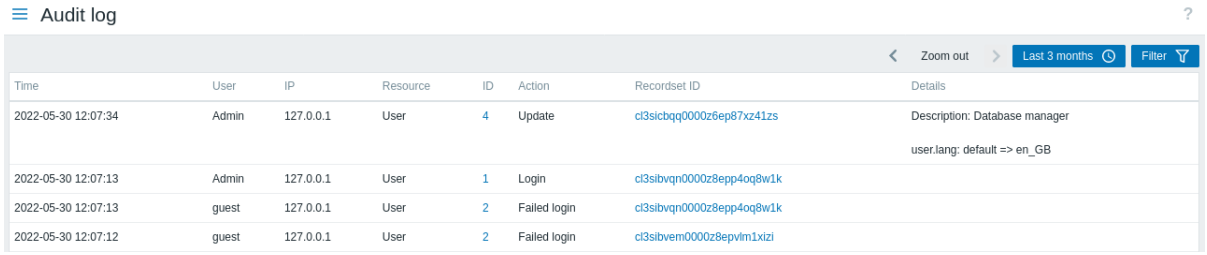
The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

5 Audit log

Overview

In the *Reports* → *Audit log* section, the records of user and system activity can be viewed.

Note:
For audit records to be collected and displayed, the *Enable audit logging* checkbox has to be marked in the *Administration* → *Audit log* section. Without this setting enabled, the history of activities will not be recorded in the database and will not be shown in the audit log.

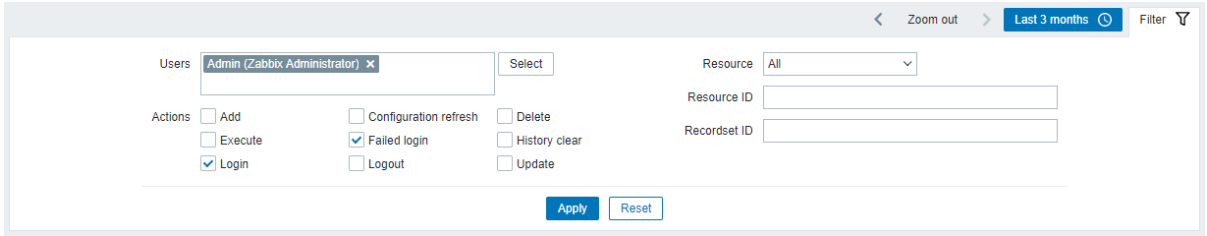


Audit log displays the following data:

Column	Description
Time	Timestamp of the audit record.
User	User who performed the activity.
IP	IP from which the activity was initiated.
Resource	Type of the affected resource (*All*, *API token*, *Action*, *Authentication*, *Autoregistration*, etc.).
ID	ID of the affected resource. Clicking on the hyperlink will result in filtering audit log records by this resource ID.
Action	Type of the activity (*Add*, *Configuration refresh*, *Delete*, *Execute*, *Failed login*, *History clear*, *Login*, *Logout*, *Update*).
Recordset ID	Shared ID for all audit log records created as a result of the same operation. For example, when linking a template to a host, a separate audit log record is created for each inherited template entity (item, trigger, etc.) - all these records will have the same *Recordset ID*. Clicking on the hyperlink will result in filtering audit log records by this *Recordset ID*.
Details	Description of the resource and detailed information about the performed activity. If a record contains more than two rows, an additional link *Details* will be displayed. Click on this link to view the full list of changes.

Using filter

The filter is located below the *Audit log* bar. It can be opened and collapsed by clicking on the *Filter* tab in the upper right corner.



You may use the filter to narrow down the records by user, affected resource, resource ID and performed operation (*Recordset ID*). Depending on the resource, one or more specific actions can be selected in the filter.

For better search performance, all data is searched with macros unresolved.

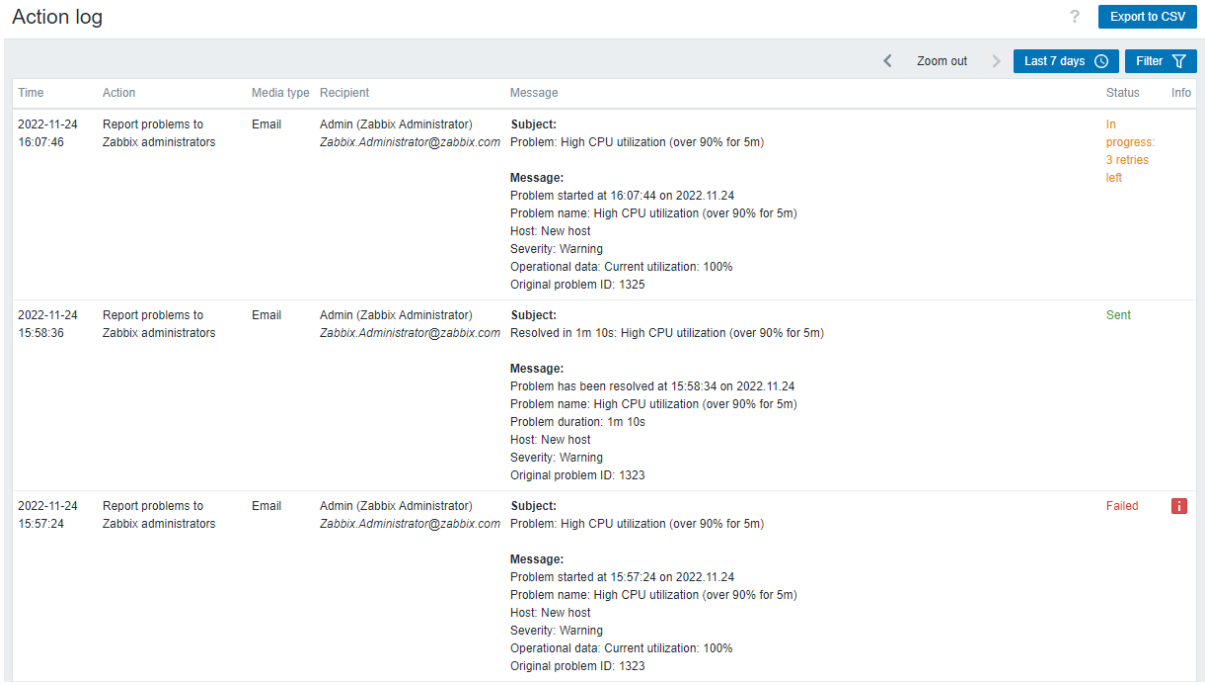
Time period selector

The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

6 Action log

Overview

In the *Reports* → *Action log* section users can view details of operations (notifications, remote commands) executed within an action.



Displayed data:

Column	Description
Time	Timestamp of the operation.
Action	Name of the action causing operations.
Media type	Media type (e.g. Email, Jira, etc.) used for sending a notification. For operations that executed remote commands, this column will be empty.
Recipient	Information about the notification recipient - username, name and surname (in parentheses), and additional information depending on the media type (email, username, etc.). For operations that executed remote commands, this column will be empty.
Message	The content of the message/remote command. A remote command is separated from the target host with a colon symbol: `<host>:<command>`. For example, if the remote command was executed on Zabbix server, then the information will have the following format: `Zabbix server:<command>`.
Status	Operation status: *In progress* - operation for sending a notification is in progress (the remaining number of times the server will try to send the notification is also displayed) *Sent* - notification has been sent *Executed* - remote command has been executed *Failed* - operation has not been completed
Info	Error information (if any) regarding the operation execution.

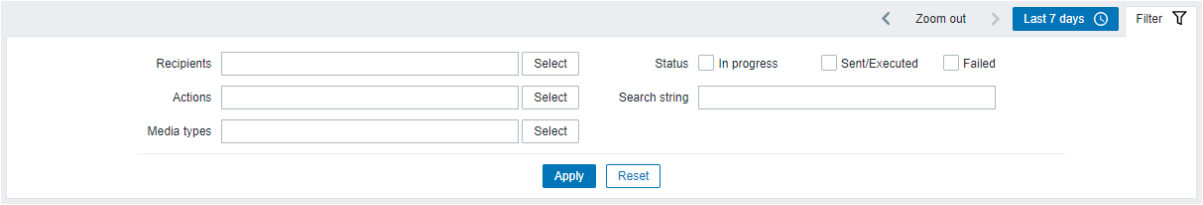
Buttons

The button at the top right corner of the page offers the following option:

Export to CSV	Export action log records from all pages to a CSV file. If a filter is applied, only the filtered records will be exported. In the exported CSV file the columns ”Recipient” and ”Message” are divided into several columns - ”Recipient’s Zabbix username”, ”Recipient’s name”, ”Recipient’s surname”, ”Recipient”, and ”Subject”, ”Message”, ”Command”.

Using filter

The filter is located below the *Action log* bar. It can be opened and collapsed by clicking on the *Filter* tab at the top right corner of the page.



You may use the filter to narrow down the records by notification recipients, actions, media types, status, or by the message/remote command content (*Search string*). For better search performance, data is searched with macros unresolved.

Time period selector

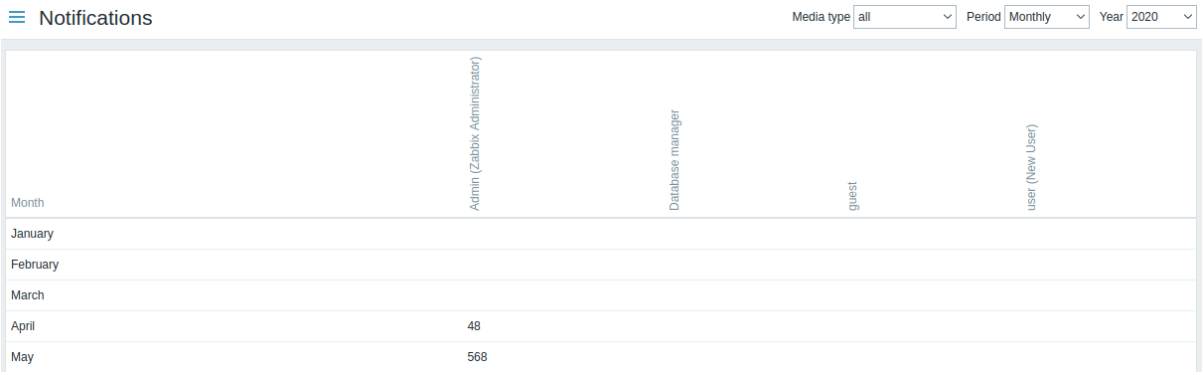
The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

7 Notifications

Overview

In the *Reports* → *Notifications* section a report on the number of notifications sent to each user is displayed.

From the dropdowns in the top right-hand corner you can choose the media type (or all), period (data for each day/week/month/year) and year for the notifications sent.



Each column displays totals per one system user.

6 Data collection

Overview

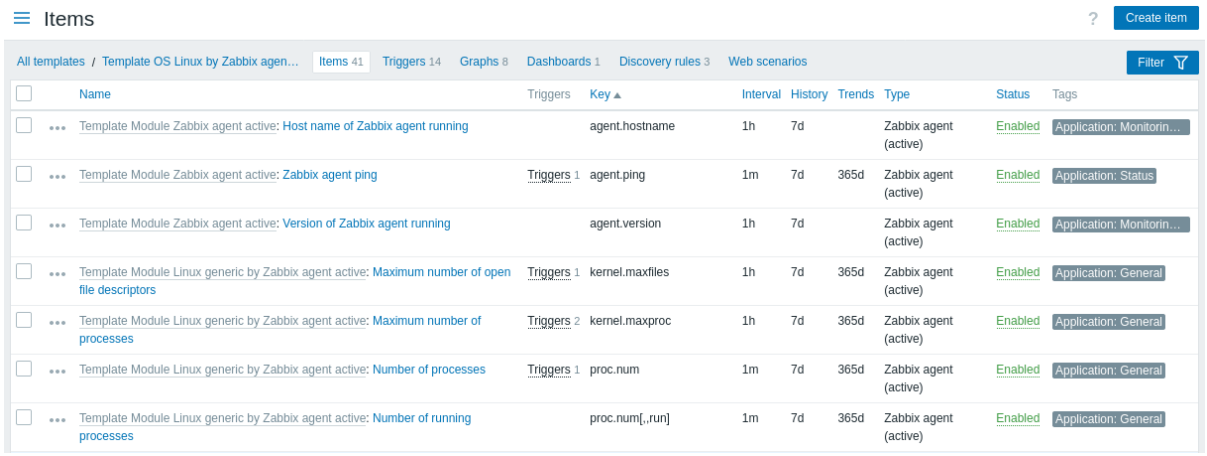
This menu features sections that are related to configuring data collection.

1 Items

Overview

The item list for a template can be accessed from *Data collection* → *Templates* by clicking on *Items* for the respective template.

A list of existing items is displayed.



Displayed data:

Column	Description
Item menu	Click on the three-dot icon to open the item context menu for this specific item.
Template	Template the item belongs to. This column is displayed only if multiple templates are selected in the filter.
Name	Name of the item displayed as a blue link to item details. Clicking on the item name link opens the item configuration form. If the item is inherited from another template, the template name is displayed before the item name, as a gray link. Clicking on the template link will open the item list on that template level.
Triggers	Moving the mouse over Triggers will display an infobox displaying the triggers associated with the item. The number of the triggers is displayed in gray.
Key	Item key is displayed.
Interval	Frequency of the check is displayed.
History	How many days item data history will be kept is displayed.
Trends	How many days item trends history will be kept is displayed.
Type	Item type is displayed (Zabbix agent, SNMP agent, simple check, etc).
Status	Item status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it - from Enabled to Disabled (and back).
Tags	Item tags are displayed. Up to three tags (name:value pairs) can be displayed. If there are more tags, a "..." link is displayed that allows to see all tags on mouseover.

To configure a new item, click on the *Create item* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

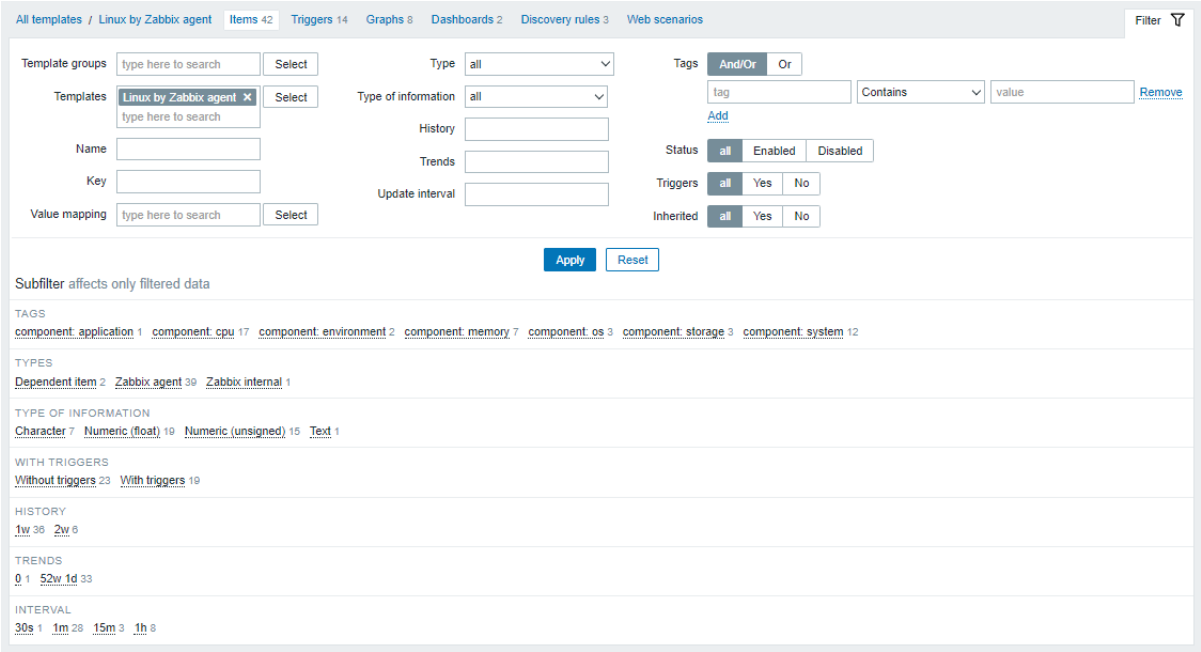
- *Enable* - change item status to *Enabled*.
- *Disable* - change item status to *Disabled*.
- *Copy* - copy the items to other hosts or templates.
- *Mass update* - update several properties for a number of items at once.
- *Delete* - delete the items.

To use these options, mark the checkboxes before the respective items, then click on the required button.

Using filter

The item list may contain a lot of items. By using the filter, you can filter out some of them to quickly locate the items you're looking for. For better search performance, data is searched with macros unresolved.

The *Filter* icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria.

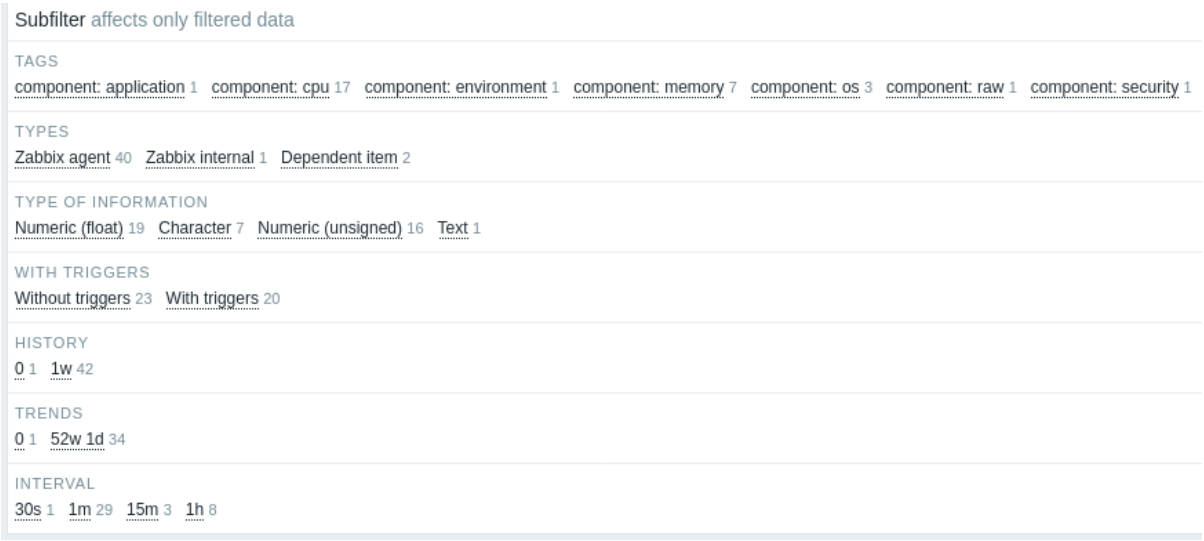


Parameter	Description
Template groups	Filter by one or more template groups. Specifying a parent template group implicitly selects all nested groups.
Templates	Filter by one or more templates.
Name	Filter by item name.
Key	Filter by item key.
Value mapping	Filter by the value map used. This parameter is not displayed if the *Templates* option is empty.
Type	Filter by item type (Zabbix agent, SNMP agent, etc.).
Type of information	Filter by type of information (Numeric unsigned, float, etc.).
History	Filter by how long item history is kept.
Trends	Filter by how long item trends are kept.
Update interval	Filter by item update interval.
Tags	Specify tags to limit the number of items displayed. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met
Status	Filter by item status - *Enabled* or *Disabled*.
Triggers	Filter items with (or without) triggers.
Inherited	Filter items inherited (or not inherited) from linked templates.

Using subfilter

The subfilter allows to further modify the filtering from the main filter.

It contains clickable links for a quick access to related items. Items are related by common entity - tag, item type, item update interval, etc. When an entity is clicked, the entity is highlighted with a gray background, and items are immediately filtered (no need to click *Apply* in the main filter). Clicking another entity adds it to the filtered results. Clicking the entity again removes the filtering.



Subfilters are generated based on the filtered data, which is limited to 1000 records. If you want to see more records in the subfilter, you need to increase the value of *Limit for search and filter results* parameter (in *Administration -> General -> GUI*).

Unlike the main filter, the subfilter is updated with each table refresh request to always have up-to-date information of available filtering options and their counter numbers.

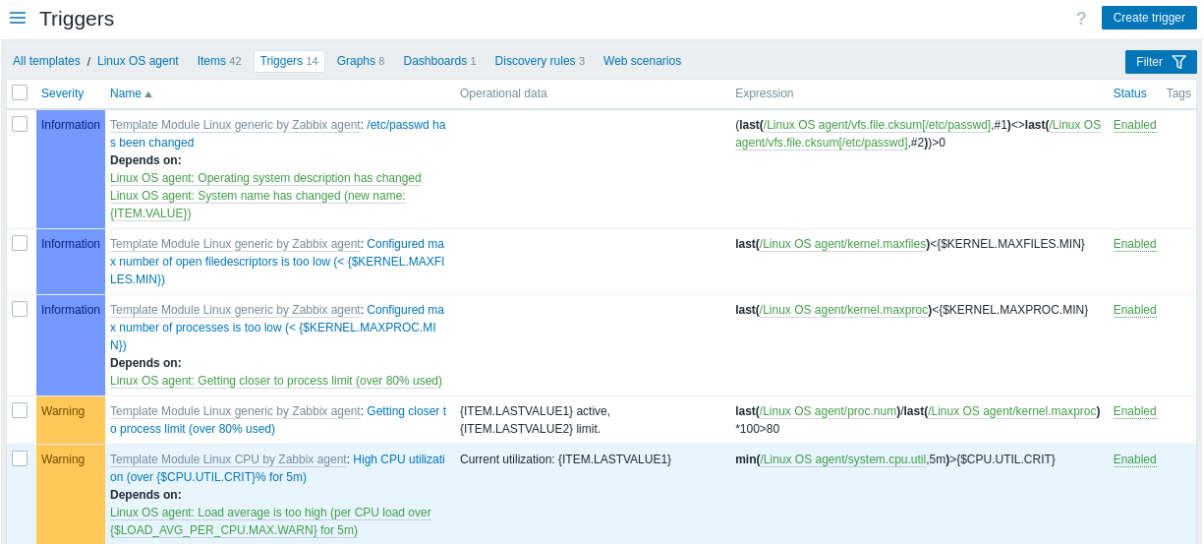
The number of entities displayed is limited to 100 horizontally. If there are more, a three-dot icon is displayed at the end; it is not clickable.

A number next to each clickable entity indicates the number of items grouped in it (based on the results of the main filter). When an entity is clicked, the numbers with other available entities are displayed with a plus sign indicating how many items may be added to the current selection. Entities without items are not displayed unless selected in the subfilter before.

2 Triggers

Overview

The trigger list for a template can be accessed from *Data collection → Templates* by clicking on *Triggers* for the respective template.



Displayed data:

Column	Description
Severity	Severity of the trigger is displayed by both name and cell background color.
Template	Template the trigger belongs to. This column is displayed only if multiple templates are selected in the filter.

Column	Description
Name	Name of the trigger displayed as a blue link to trigger details. Clicking on the trigger name link opens the trigger configuration form. If the trigger is inherited from another template, the template name is displayed before the trigger name, as a gray link. Clicking on the template link will open the trigger list on that template level.
Operational data	Operational data definition of the trigger, containing arbitrary strings and macros that will resolve dynamically in *Monitoring* → *Problems*.
Expression	Trigger expression is displayed. The template-item part of the expression is displayed as a link, leading to the item configuration form.
Status	Trigger status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it - from Enabled to Disabled (and back).
Tags	If a trigger contains tags, tag name and value are displayed in this column.

To configure a new trigger, click on the *Create trigger* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

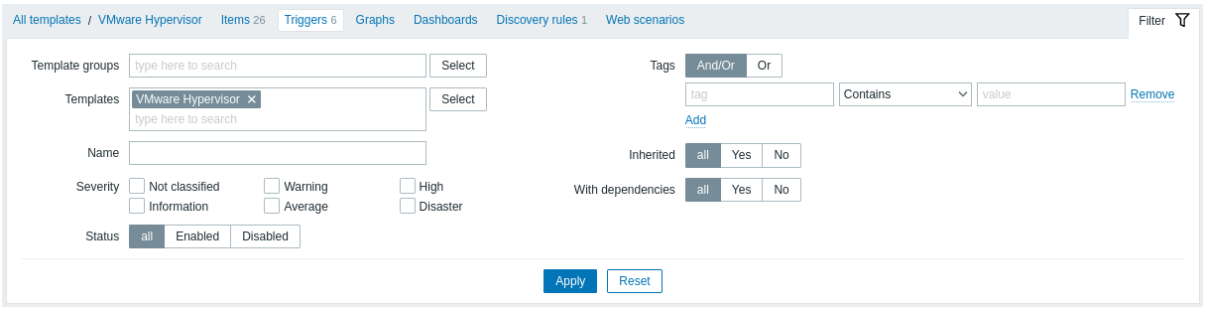
- *Enable* - change trigger status to *Enabled*
- *Disable* - change trigger status to *Disabled*
- *Copy* - copy the triggers to other hosts or templates
- *Mass update* - update several properties for a number of triggers at once
- *Delete* - delete the triggers

To use these options, mark the checkboxes before the respective triggers, then click on the required button.

Using filter

You can use the filter to display only the triggers you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria.



Parameter	Description
Template groups	Filter by one or more template groups. Specifying a parent template group implicitly selects all nested groups.
Templates	Filter by one or more templates. If template groups are already selected above, template selection is limited to those groups.
Name	Filter by trigger name.
Severity	Select to filter by one or several trigger severities.
Status	Filter by trigger status.

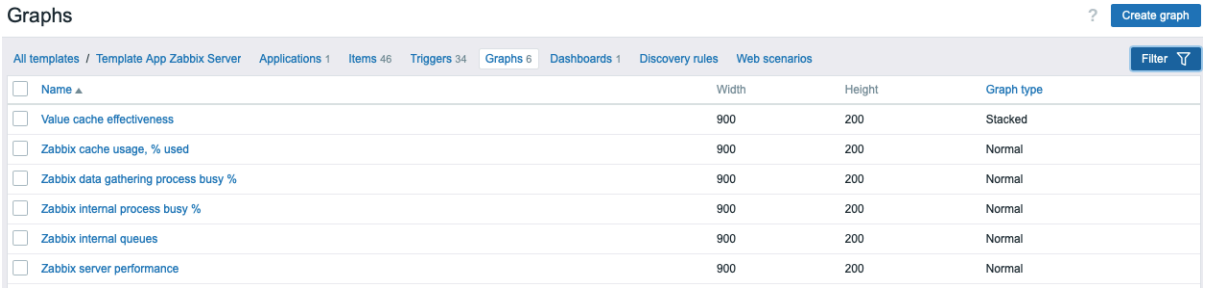
Parameter	Description
Tags	Filter by trigger tag name and value. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met Macros and macro functions are supported in tag name and tag value fields.
Inherited	Filter triggers inherited (or not inherited) from linked templates.
With dependencies	Filter triggers with (or without) dependencies.

3 Graphs

Overview

The custom graph list for a template can be accessed from *Data collection* → *Templates* by clicking on *Graphs* for the respective template.

A list of existing graphs is displayed.



Displayed data:

Column	Description
Template	Template the graph belongs to. This column is displayed only if multiple templates are selected in the filter.
Name	Name of the custom graph, displayed as a blue link to graph details. Clicking on the graph name link opens the graph configuration form. If the graph is inherited from another template, the template name is displayed before the graph name, as a gray link. Clicking on the template link will open the graph list on that template level.
Width	Graph width is displayed.
Height	Graph height is displayed.
Graph type	Graph type is displayed - *Normal*, *Stacked*, *Pie* or *Exploded*.

To configure a new graph, click on the *Create graph* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Copy* - copy the graphs to other hosts or templates
- *Delete* - delete the graphs

To use these options, mark the checkboxes before the respective graphs, then click on the required button.

Using filter

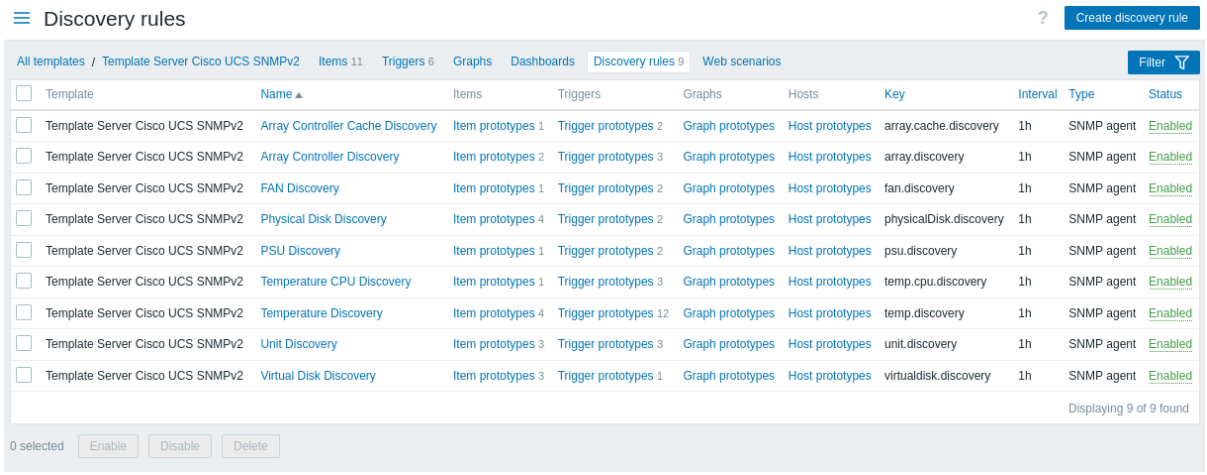
You can filter graphs by template group and template. For better search performance, data is searched with macros unresolved.

4 Discovery rules

Overview

The list of low-level discovery rules for a template can be accessed from *Data collection* → *Templates* by clicking on *Discovery* for the respective template.

A list of existing low-level discovery rules is displayed. It is also possible to see all discovery rules independently of the template, or all discovery rules of a specific template group by changing the filter settings.



Displayed data:

Column	Description
Template	The template discovery rule belongs to.
Name	Name of the rule, displayed as a blue link. Clicking on the rule name opens the low-level discovery rule configuration form. If the discovery rule is inherited from another template, the template name is displayed before the rule name, as a gray link. Clicking on the template link will open the discovery rule list on that template level.
Items	A link to the list of item prototypes is displayed. The number of existing item prototypes is displayed in gray.
Triggers	A link to the list of trigger prototypes is displayed. The number of existing trigger prototypes is displayed in gray.
Graphs	A link to the list of graph prototypes displayed. The number of existing graph prototypes is displayed in gray.
Hosts	A link to the list of host prototypes displayed. The number of existing host prototypes is displayed in gray.
Key	The item key used for discovery is displayed.
Interval	The frequency of performing discovery is displayed.
Type	The item type used for discovery is displayed (Zabbix agent, SNMP agent, etc).
Status	Discovery rule status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it - from Enabled to Disabled (and back).

To configure a new low-level discovery rule, click on the *Create discovery rule* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

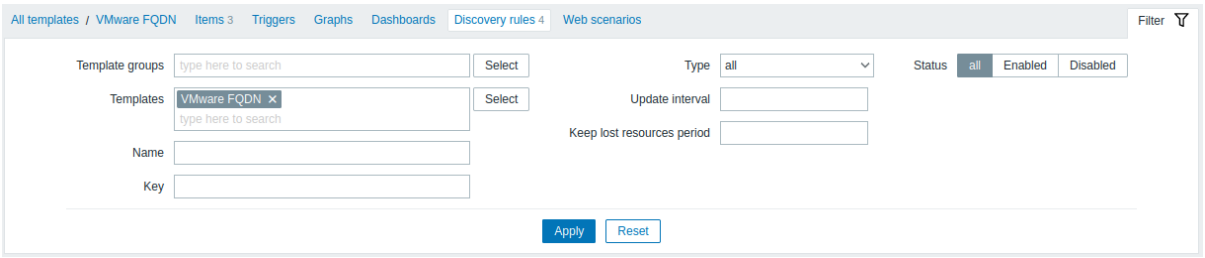
- *Enable* - change the low-level discovery rule status to *Enabled*
- *Disable* - change the low-level discovery rule status to *Disabled*
- *Delete* - delete the low-level discovery rules

To use these options, mark the checkboxes before the respective discovery rules, then click on the required button.

Using filter

You can use the filter to display only the discovery rules you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria such as template, discovery rule name, item key, item type, etc.



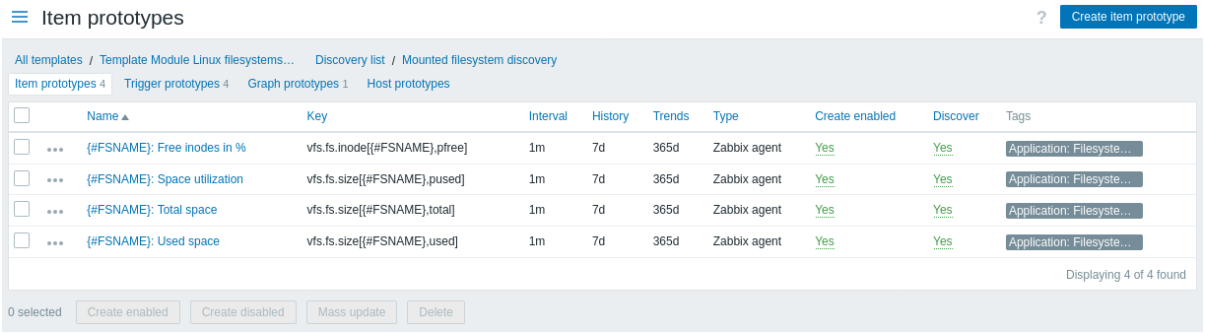
Parameter	Description
Template groups	Filter by one or more template groups. Specifying a parent template group implicitly selects all nested groups.
Templates	Filter by one or more templates.
Name	Filter by discovery rule name.
Key	Filter by discovery item key.
Type	Filter by discovery item type.
Update interval	Filter by update interval. Not available for Zabbix trapper and dependent items.
Keep lost resources period	Filter by Keep lost resources period.
Status	Filter by discovery rule status (All/Enabled/Disabled).

1 Item prototypes

Overview

In this section the configured item prototypes of a low-level discovery rule on the template are displayed.

If the template is linked to the host, item prototypes will become the basis of creating real host items during low-level discovery.



Displayed data:

Column	Description
Name	Name of the item prototype, displayed as a blue link. Clicking on the name opens the item prototype configuration form. If the item prototype belongs to a linked template, the template name is displayed before the item name, as a gray link. Clicking on the template link will open the item prototype list on the linked template level.
Key	Key of the item prototype is displayed.
Interval	Frequency of the check is displayed.
History	How many days to keep item data history is displayed.

Column	Description
Trends	How many days to keep item trends history is displayed.
Type	Type of the item prototype is displayed (Zabbix agent, SNMP agent, simple check, etc).
Create enabled	Create the item based on this prototype as: **Yes** - enabled **No** - disabled. You can switch between 'Yes' and 'No' by clicking on them.
Discover	Discover the item based on this prototype: **Yes** - discover **No** - do not discover. You can switch between 'Yes' and 'No' by clicking on them.
Tags	Tags of the item prototype is displayed.

To configure a new item prototype, click on the *Create item prototype* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Create enabled* - create these items as *Enabled*
- *Create disabled* - create these items as *Disabled*
- *Mass update* - mass update these item prototypes
- *Delete* - delete these item prototypes

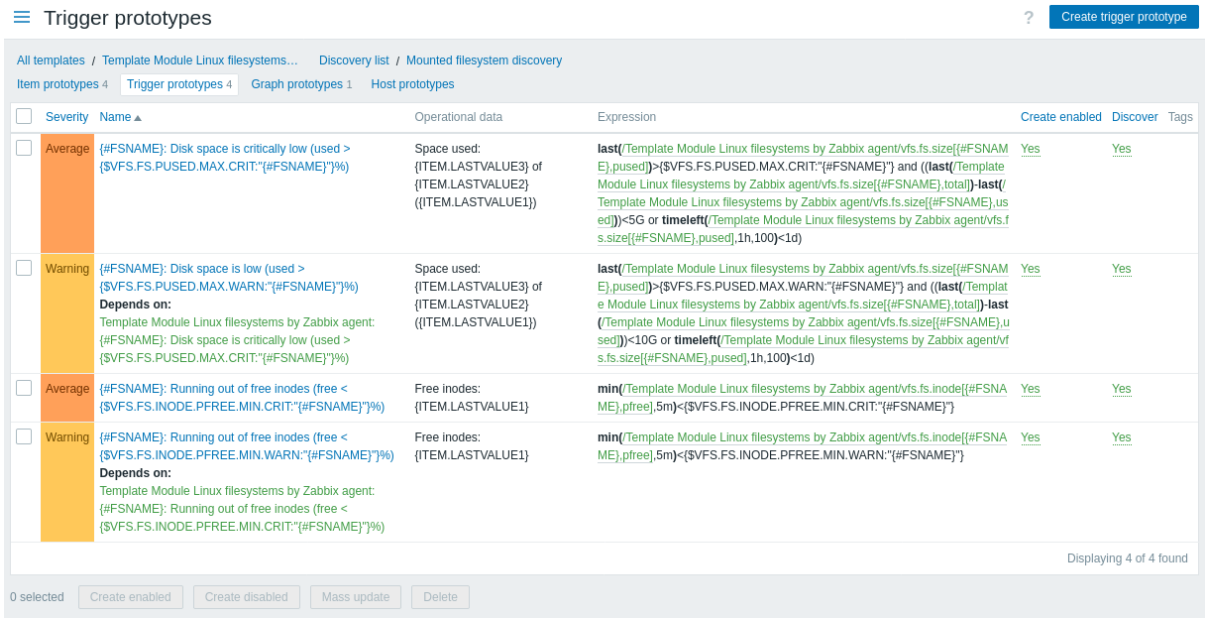
To use these options, mark the checkboxes before the respective item prototypes, then click on the required button.

2 Trigger prototypes

Overview

In this section the configured trigger prototypes of a low-level discovery rule on the template are displayed.

If the template is linked to the host, trigger prototypes will become the basis of creating real host triggers during low-level discovery.



Displayed data:

Column	Description
Name	Name of the trigger prototype, displayed as a blue link. Clicking on the name opens the trigger prototype configuration form. If the trigger prototype belongs to a linked template, the template name is displayed before the trigger name, as a gray link. Clicking on the template link will open the trigger prototype list on the linked template level.
Operational data	Format of the operational data of the trigger is displayed, containing arbitrary strings and macros that will resolve dynamically in *Monitoring → Problems*.

Column	Description
Create enabled	Create the trigger based on this prototype as: **Yes** - enabled **No** - disabled. You can switch between 'Yes' and 'No' by clicking on them.
Discover	Discover the trigger based on this prototype: **Yes** - discover **No** - do not discover. You can switch between 'Yes' and 'No' by clicking on them.
Tags	Tags of the trigger prototype are displayed.

To configure a new trigger prototype, click on the *Create trigger prototype* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Create enabled* - create these triggers as *Enabled*
- *Create disabled* - create these triggers as *Disabled*
- *Mass update* - mass update these trigger prototypes
- *Delete* - delete these trigger prototypes

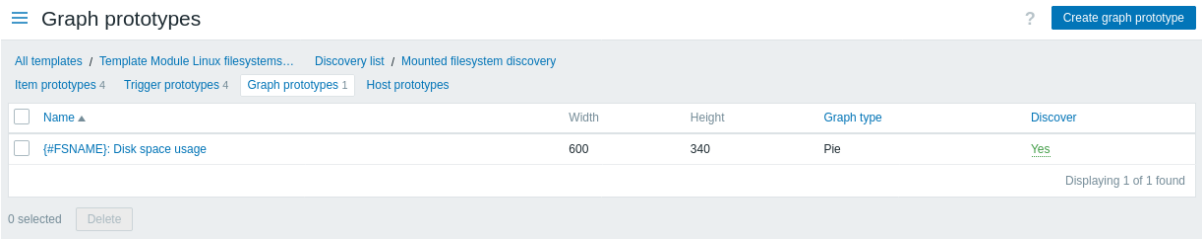
To use these options, mark the checkboxes before the respective trigger prototypes, then click on the required button.

3 Graph prototypes

Overview

In this section the configured graph prototypes of a low-level discovery rule on the template are displayed.

If the template is linked to the host, graph prototypes will become the basis of creating real host graphs during low-level discovery.



Displayed data:

Column	Description
Name	Name of the graph prototype, displayed as a blue link. Clicking on the name opens the graph prototype configuration form. If the graph prototype belongs to a linked template, the template name is displayed before the graph name, as a gray link. Clicking on the template link will open the graph prototype list on the linked template level.
Width	Width of the graph prototype is displayed.
Height	Height of the graph prototype is displayed.
Type	Type of the graph prototype is displayed - *Normal*, *Stacked*, *Pie* or *Exploded*.
Discover	Discover the graph based on this prototype: **Yes** - discover **No** - do not discover. You can switch between 'Yes' and 'No' by clicking on them.

To configure a new graph prototype, click on the *Create graph prototype* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Delete* - delete these graph prototypes

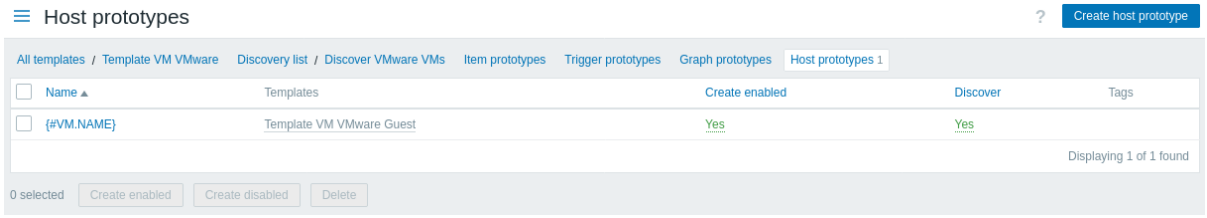
To use these options, mark the checkboxes before the respective graph prototypes, then click on the required button.

4 Host prototypes

Overview

In this section the configured host prototypes of a low-level discovery rule on the template are displayed.

If the template is linked to the host, host prototypes will become the basis of creating real hosts during low-level discovery.



Displayed data:

Column	Description
Name	Name of the host prototype, displayed as a blue link. Clicking on the name opens the host prototype configuration form. If the host prototype belongs to a linked template, the template name is displayed before the host name, as a gray link. Clicking on the template link will open the host prototype list on the linked template level.
Templates	Templates of the host prototype are displayed.
Create enabled	Create the host based on this prototype as: **Yes** - enabled **No** - disabled. You can switch between 'Yes' and 'No' by clicking on them.
Discover	Discover the host based on this prototype: **Yes** - discover **No** - do not discover. You can switch between 'Yes' and 'No' by clicking on them.
Tags	Tags of the host prototype are displayed.

To configure a new host prototype, click on the *Create host prototype* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Create enabled* - create these hosts as *Enabled*
- *Create disabled* - create these hosts as *Disabled*
- *Delete* - delete these host prototypes

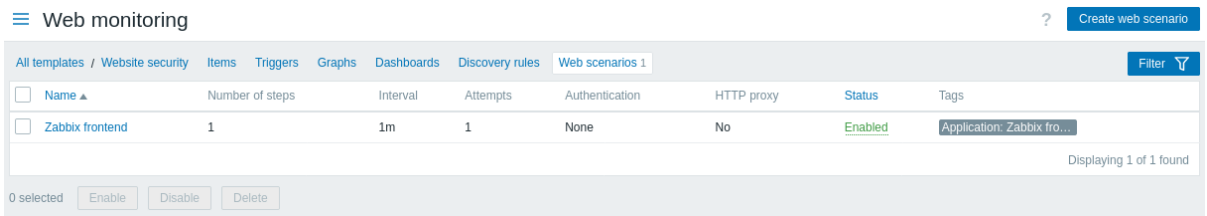
To use these options, mark the checkboxes before the respective host prototypes, then click on the required button.

5 Web scenarios

Overview

The web scenario list for a template can be accessed from *Data collection* → *Templates* by clicking on *Web* for the respective template.

A list of existing web scenarios is displayed.



Displayed data:

Column	Description
Name	Name of the web scenario. Clicking on the web scenario name opens the web scenario configuration form. If the web scenario is inherited from another template, the template name is displayed before the web scenario name, as a gray link. Clicking on the template link will open the web scenarios list on that template level.
Number of steps	The number of steps the scenario contains.
Update interval	How often the scenario is performed.
Attempts	How many attempts for executing web scenario steps are performed.
Authentication	Authentication method is displayed - Basic, NTLM on None.
HTTP proxy	Displays HTTP proxy or 'No' if not used.
Status	Web scenario status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.
Tags	Web scenario tags are displayed. Up to three tags (name:value pairs) can be displayed. If there are more tags, a "..." link is displayed that allows to see all tags on mouseover.

To configure a new web scenario, click on the *Create web scenario* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

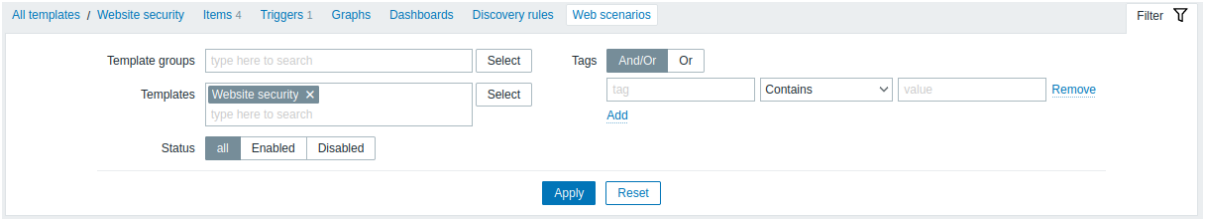
- *Enable* - change the scenario status to *Enabled*
- *Disable* - change the scenario status to *Disabled*
- *Delete* - delete the web scenarios

To use these options, mark the checkboxes before the respective web scenarios, then click on the required button.

Using filter

You can use the filter to display only the scenarios you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is available above the list of web scenarios. If you click on it, a filter becomes available where you can filter scenarios by template group, template, status and tags.

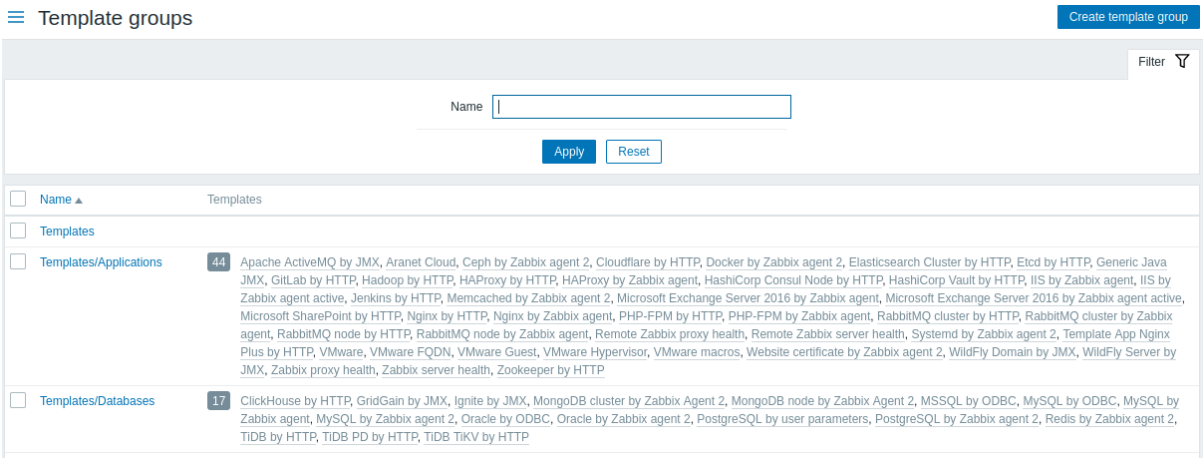


1 Template groups

Overview

In the *Data collection* → *Templates groups* section users can configure and maintain template groups.

A listing of existing template groups with their details is displayed. You can search and filter template groups by name.



Displayed data:

Column	Description
Name	Name of the template group. Clicking on the group name opens the group configuration form.
Templates	Number of templates in the group (displayed in gray) followed by the list of group members. Clicking on a template name will open the template configuration form. Clicking on the number opens the list of templates in this group.

Mass editing options

To delete several template groups at once, mark the checkboxes before the respective groups, then click on the Delete button below the list.

Using filter

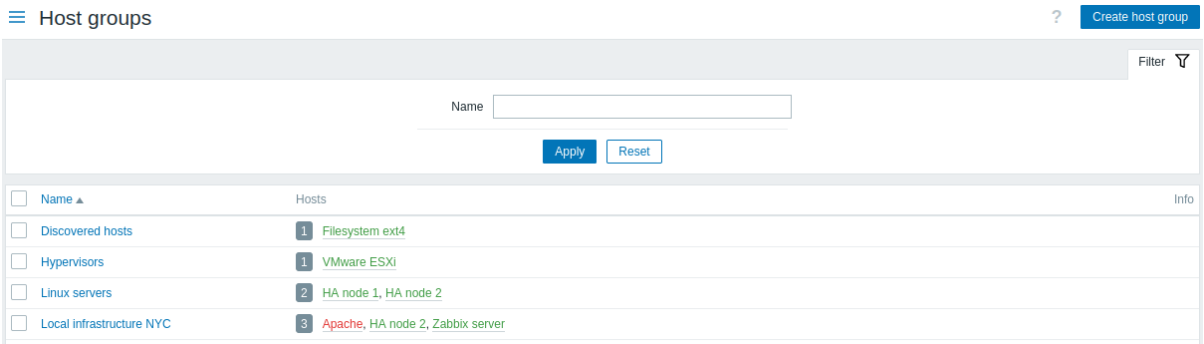
You can use the filter to display only the template groups you are interested in. For better search performance, data is searched with macros unresolved.

2 Host groups

Overview

In the *Data collection* → *Host groups* section users can configure and maintain host groups.

A listing of existing host groups with their details is displayed. You can search and filter host groups by name.



Displayed data:

Column	Description
Name	Name of the host group. Clicking on the group name opens the group configuration form.
Hosts	Number of hosts in the group (displayed in gray) followed by the list of group members. Clicking on a host name will open the host configuration form. Clicking on the number will, in the whole listing of hosts, filter out those that belong to the group.
Info	Error information (if any) regarding the host group is displayed.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Enable hosts* - change the status of all hosts in the group to "Monitored".
- *Disable hosts* - change the status of all hosts in the group to "Not monitored".
- *Delete* - delete the host groups. Note that deleting a host group only deletes the logical group, not the hosts in the group. It is not possible to delete a host group that is the only group for some hosts.

To use these options, mark the checkboxes before the respective host groups, then click on the required button.

Using filter

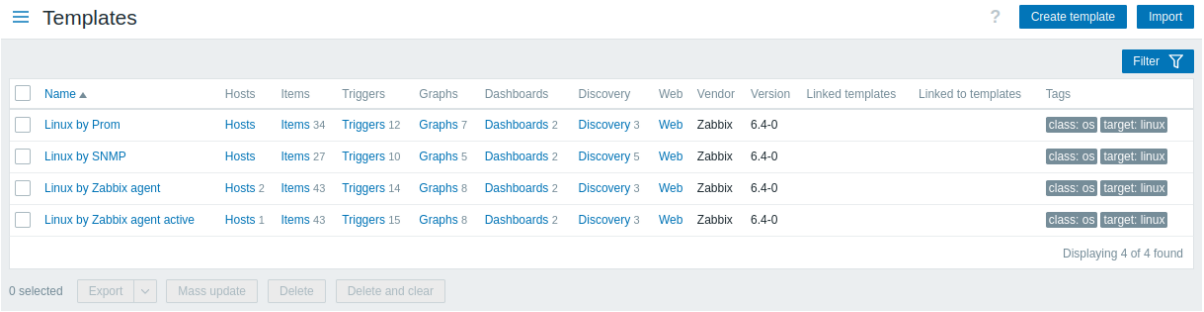
You can use the filter to display only the host groups you are interested in. For better search performance, data is searched with macros unresolved.

3 Templates

Overview

In the *Data collection* → *Templates* section users can configure and maintain templates.

A listing of existing templates with their details is displayed.



Displayed data:

Column	Description
Name	Name of the template. Clicking on the template name opens the template configuration form.
Hosts	Number of editable hosts to which the template is linked; read-only hosts are not included. Clicking on *Hosts* will open the host list with only those hosts filtered that are linked to the template.
Entities (Items, Triggers, Graphs, Dashboards, Discovery, Web)	Number of the respective entities in the template (displayed in gray). Clicking on the entity name will, in the whole listing of that entity, filter out those that belong to the template.
Linked templates	Templates that are linked to the template.
Linked to templates	Templates that the template is linked to. Since Zabbix 5.0.3, this column no longer includes hosts.
Vendor, *Version*	Template vendor and version; displayed if the template configuration contains such information, and only for out-of-the-box templates, imported templates, or templates modified through the Template API. For out-of-the-box templates, version is displayed as follows: major version of Zabbix, delimiter ("-"), revision number (increased with each new version of the template, and reset with each major version of Zabbix). For example, 6.4-0, 6.4-3, 7.0-0, 7.0-3.
Tags	Tags of the template, with macros unresolved.

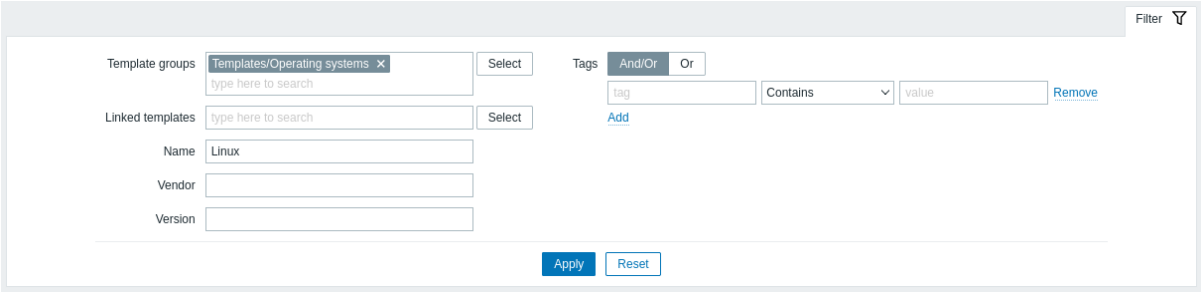
To configure a new template, click on the *Create template* button in the top right-hand corner.

To import a template from a YAML, XML, or JSON file, click on the *Import* button in the top right-hand corner.

Using filter

You can use the filter to display only the templates you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is available below *Create template* and *Import* buttons. If you click on it, a filter becomes available where you can filter templates by template group, linked templates, name and tags.



Parameter	Description
Template groups	Filter by one or more template groups. Specifying a parent template group implicitly selects all nested groups.
Linked templates	Filter by directly linked templates.
Name	Filter by template name.
Vendor	Filter by template vendor.
Version	Filter by template version.
Tags	Filter by template tag name and value. Filtering is possible only by template-level tags (not inherited ones). It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names; **Equals** - include the specified tag names and values (case-sensitive); **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive); **Does not exist** - exclude the specified tag names; **Does not equal** - exclude the specified tag names and values (case-sensitive); **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive). There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition; **Or** - enough if one condition is met.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Export* - export the template to a YAML, XML or JSON file;
- *Mass update* - update several properties for a number of templates at once;
- *Delete* - delete the template while leaving its linked entities (items, triggers etc.) with the hosts;
- *Delete and clear* - delete the template and its linked entities from the hosts.

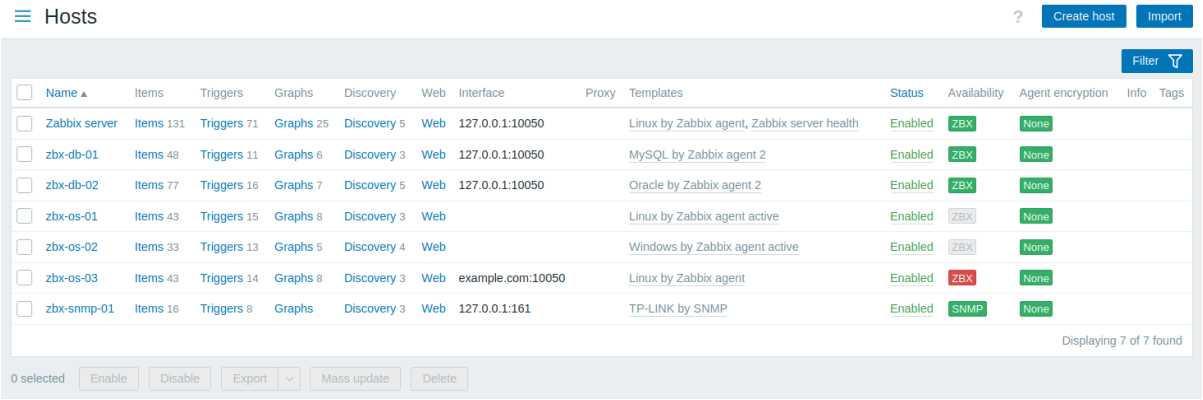
To use these options, mark the checkboxes before the respective templates, then click on the required button.

4 Hosts

Overview

In the *Data collection* → *Hosts* section users can configure and maintain hosts.

A listing of existing hosts with their details is displayed.



Displayed data:

Column	Description
Name	Name of the host. Clicking on the host name opens the host configuration form.
Entities (Items, Triggers, Graphs, Discovery, Web)	Clicking on the entity name will display items, triggers etc. of the host. The number of the respective entities is displayed in gray.
Interface	The main interface of the host is displayed.
Proxy	Proxy name is displayed, if the host is monitored by a proxy. This column is only displayed if the *Monitored by* filter option is set to 'Any' or 'Proxy'.
Templates	The templates linked to the host are displayed. If other templates are contained in the linked template, those are displayed in parentheses, separated by a comma. Clicking on a template name will open its configuration form.
Status	Host status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it. An orange wrench icon before the host status indicates that this host is in maintenance. Maintenance details are displayed when the mouse pointer is positioned on the icon.

Column	Description
Availability	Host availability per configured interface is displayed. Availability icons represent host interface current status on Zabbix server. Therefore, if you disable a host in the frontend, its availability will update after Zabbix server has synchronized the configuration changes. Similarly, if you enable a host, its availability will update after Zabbix server has synchronized the configuration changes and polled the host. Availability icons represent only those interface types (Agent, SNMP, IPMI, JMX) that are configured. Hovering over the icon displays a pop-up with a list of all interfaces of the same type with details, status, and errors. For Agent interface, the pop-up displays interfaces (passive) and active checks. If a host has active checks only, the Agent interface icon is displayed even if the host does not have an Agent interface (passive) configured. The column is empty for hosts with no interfaces. The current status of all interfaces of one type is indicated by the icon color: **Green** - all interfaces are available; **Yellow** - at least one interface is not available, and at least one is available or unknown; **Red** - all interfaces are not available; **Gray** - at least one interface is unknown, but none are not available. For details on how Zabbix server determines the "Unknown" status, see Unknown interface status. **Active check availability.** Active checks also affect host interface availability if at least one active check is enabled on the host. Active check availability is counted towards the total Agent interface availability as described above. For example, if a host has an Agent interface (passive) that is available, but active checks are unknown, the total Agent interface availability is displayed as gray (unknown). **Note:** Since Zabbix 6.4.12, there are two exceptions for determining the total Agent interface availability: - if active checks are available, but at least one Agent interface (passive) is unknown while the host also has at least one item using this interface, the total Agent interface availability is displayed as gray (unknown); - if active checks are available and all Agent interfaces (passive) are unknown (and no items are using this interface), the total Agent interface availability is displayed as green (available). To determine active check availability, heartbeat messages are sent in the agent active check thread. The frequency of heartbeat messages is controlled by the `HeartbeatFrequency` parameter in Zabbix agent or agent 2 configuration (default 60 seconds, range 0-3600). Active checks are considered unavailable when the active check heartbeat is older than 2 x HeartbeatFrequency seconds. **Note:** Zabbix agents older than version 6.2.x do not send active check heartbeats, so the availability of their hosts remains unknown.
Agent encryption	Encryption status for connections to the host is displayed: **None** - no encryption; **PSK** - using pre-shared key; **Cert** - using certificate.
Info	Error information (if any) regarding the host is displayed.
Tags	Tags of the host with macros unresolved.

To configure a new host, click on the *Create host* button in the top right-hand corner. To import a host from a YAML, XML, or JSON file, click on the *Import* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass editing options:

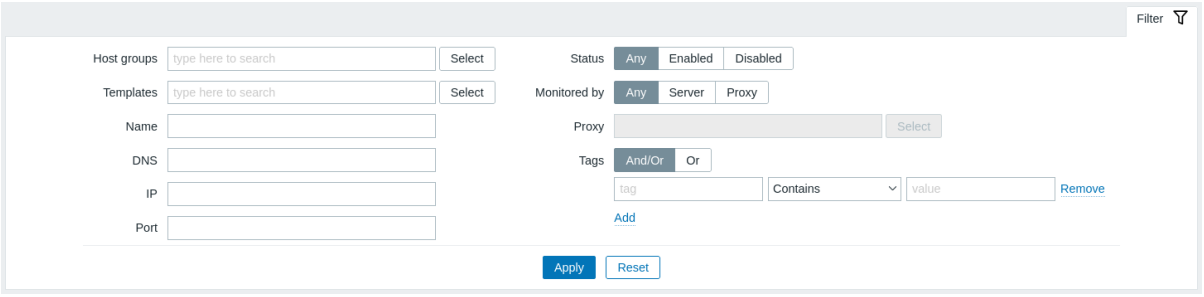
- *Enable* - change host status to *Monitored*;
- *Disable* - change host status to *Not monitored*;
- *Export* - export the hosts to a YAML, XML or JSON file;
- *Mass update* - update several properties for a number of hosts at once;
- *Delete* - delete the hosts.

To use these options, mark the checkboxes before the respective hosts, then click on the required button.

Using filter

You can use the filter to display only the hosts you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria.



Parameter	Description
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
Templates	Filter by linked templates.
Name	Filter by visible host name.
DNS	Filter by DNS name.
IP	Filter by IP address.
Port	Filter by port number.
Status	Filter by host status.
Monitored by	Filter hosts that are monitored by server only, proxy only or both.
Proxy	Filter hosts that are monitored by the proxy specified here.
Tags	Filter by host tag name and value. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names; **Equals** - include the specified tag names and values (case-sensitive); **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive); **Does not exist** - exclude the specified tag names; **Does not equal** - exclude the specified tag names and values (case-sensitive); **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive). There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition; **Or** - enough if one condition is met.

Unknown interface status

Zabbix server displays the ”Unknown” status for a host interface (Agent, SNMP, IPMI, JMX) in the following cases:

- Host is disabled.
- Host is set to be monitored by proxy, different proxy, or server if it was previously monitored by proxy.
- Host is monitored by a proxy that appears to be offline (no updates received from the proxy during the maximum heartbeat interval - 1 hour).
- All host items with that interface type are disabled.
- No pollers for that interface type have been configured (for example, the `StartPollers` or `StartSNMPPollers` server configuration parameter is set to 0).

Interface availability is set to ”Unknown” after Zabbix server configuration cache synchronization.

Interface availability (Available/Not available) on hosts monitored by proxies is restored after proxy configuration cache synchronization.

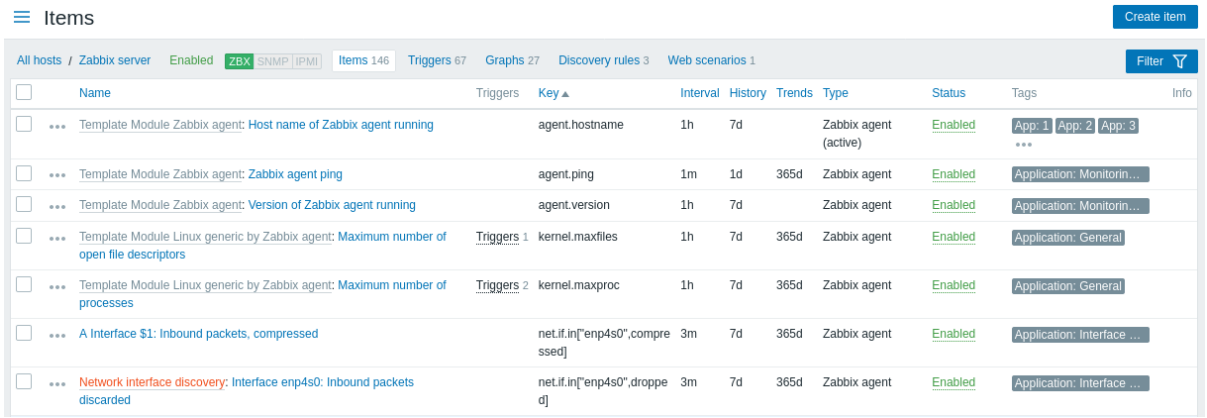
For details on host interface unreachability, see Unreachable/unavailable host interface settings.

1 Items

Overview

The item list for a host can be accessed from *Data collection* → *Hosts* by clicking on *Items* for the respective host.

A list of existing items is displayed.



Displayed data:

Column	Description
Item context menu	Click on the three-dot icon to open the item context menu.
Host	Host of the item. This column is displayed only if multiple hosts are selected in the filter.
Name	Name of the item displayed as a blue link to item details. Clicking on the item name link opens the item configuration form. If the host item belongs to a template, the template name is displayed before the item name as a gray link. Clicking on the template link will open the item list on the template level. If the item has been created from an item prototype, its name is preceded by the low-level discovery rule name, in orange. Clicking on the discovery rule name will open the item prototype list.
Triggers	Moving the mouse over Triggers will display an infobox displaying the triggers associated with the item. The number of the triggers is displayed in gray.
Key	Item key is displayed.
Interval	Frequency of the check is displayed. *Note* that passive items can also be checked immediately by pushing the *Execute now* button.
History	How many days item data history will be kept is displayed.
Trends	How many days item trends history will be kept is displayed.
Type	Item type is displayed (Zabbix agent, SNMP agent, simple check, etc).
Status	Item status is displayed - *Enabled*, *Disabled* or *Not supported*. You can change the status by clicking on it - from Enabled to Disabled (and back); from Not supported to Disabled (and back).
Tags	Item tags are displayed. Up to three tags (name:value pairs) can be displayed. If there are more tags, a "..." link is displayed that allows to see all tags on mouseover.
Info	If the item is working correctly, no icon is displayed in this column. In case of errors, a square icon with the letter "i" is displayed. Hover over the icon to see a tooltip with the error description.

To configure a new item, click on the *Create item* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Enable* - change item status to *Enabled*
- *Disable* - change item status to *Disabled*

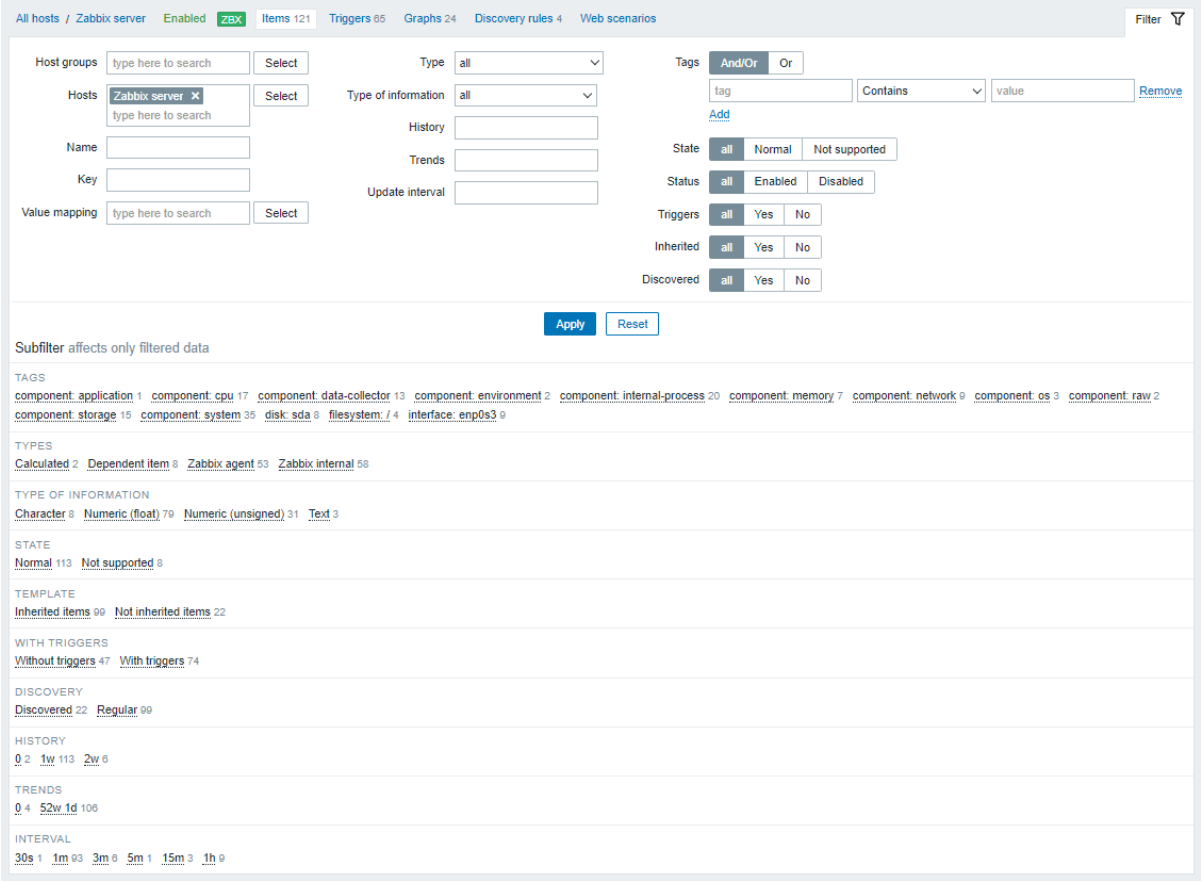
- *Execute now* - execute a check for new item values immediately. Supported for **passive** checks only (see more details). Note that when checking for values immediately, configuration cache is not updated, thus the values will not reflect very recent changes to item configuration.
- *Clear history and trends* - delete history and trend data for items.
- *Copy* - copy the items to other hosts or templates.
- *Mass update* - update several properties for a number of items at once.
- *Delete* - delete the items.

To use these options, mark the checkboxes before the respective items, then click on the required button.

Using filter

You can use the filter to display only the items you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria.



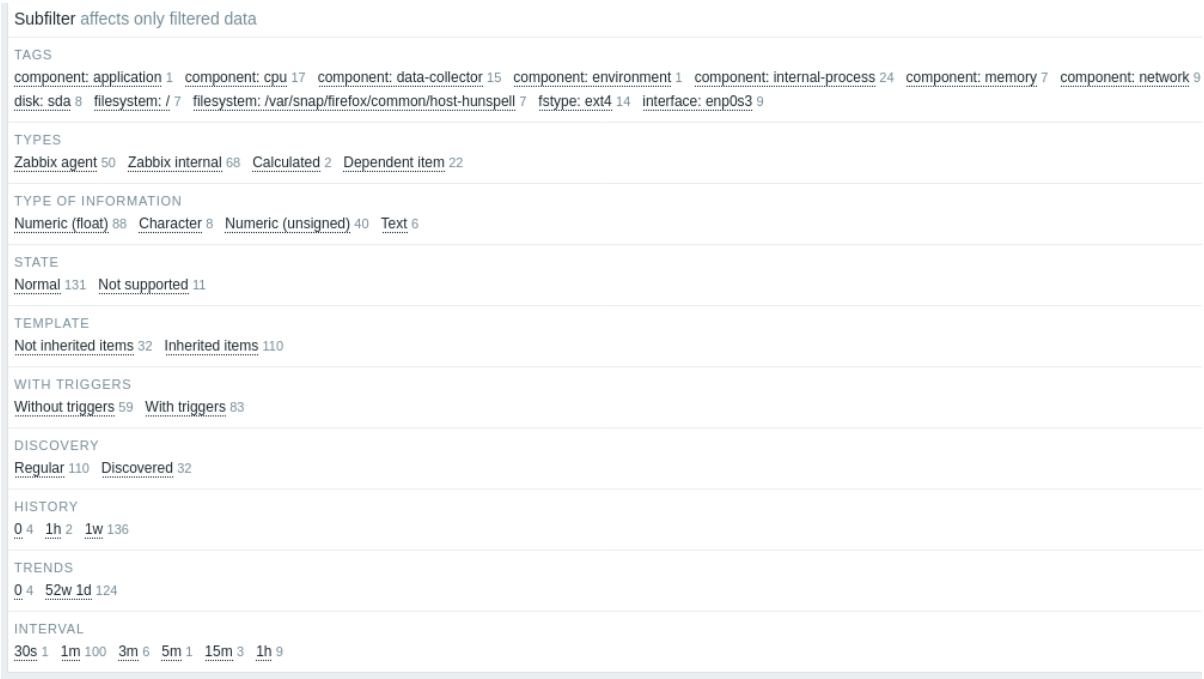
Parameter	Description
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups. Host groups containing templates only cannot be selected.
Hosts	Filter by one or more hosts.
Name	Filter by item name.
Key	Filter by item key.
Value mapping	Filter by the value map used. This parameter is not displayed if the *Hosts* option is empty.
Type	Filter by item type (Zabbix agent, SNMP agent, etc.).
Type of information	Filter by type of information (Numeric unsigned, float, etc.).
History	Filter by how long item history is kept.
Trends	Filter by how long item trends are kept.
Update interval	Filter by item update interval.

Parameter	Description
Tags	Specify tags to limit the number of items displayed. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met
State	Filter by item state - *Normal* or *Not supported*.
Status	Filter by item status - *Enabled* or *Disabled*.
Triggers	Filter items with (or without) triggers.
Inherited	Filter items inherited (or not inherited) from a template.
Discovery	Filter items discovered (or not discovered) by low-level discovery.

Using subfilter

The subfilter allows to further modify the filtering from the main filter.

It contains clickable links for a quick access to related items. Items are related by common entity - tag, item type, item state, item update interval, etc. When an entity is clicked, the entity is highlighted with a gray background, and items are immediately filtered (no need to click *Apply* in the main filter). Clicking another entity adds it to the filtered results. Clicking the entity again removes the filtering.



Subfilters are generated based on the filtered data, which is limited to 1000 records. If you want to see more records in the subfilter, you need to increase the value of *Limit for search and filter results* parameter (in *Administration -> General -> GUI*).

Unlike the main filter, the subfilter is updated with each table refresh request to always have up-to-date information of available filtering options and their counter numbers.

The number of entities displayed is limited to 100 horizontally. If there are more, a three-dot icon is displayed at the end; it is not clickable.

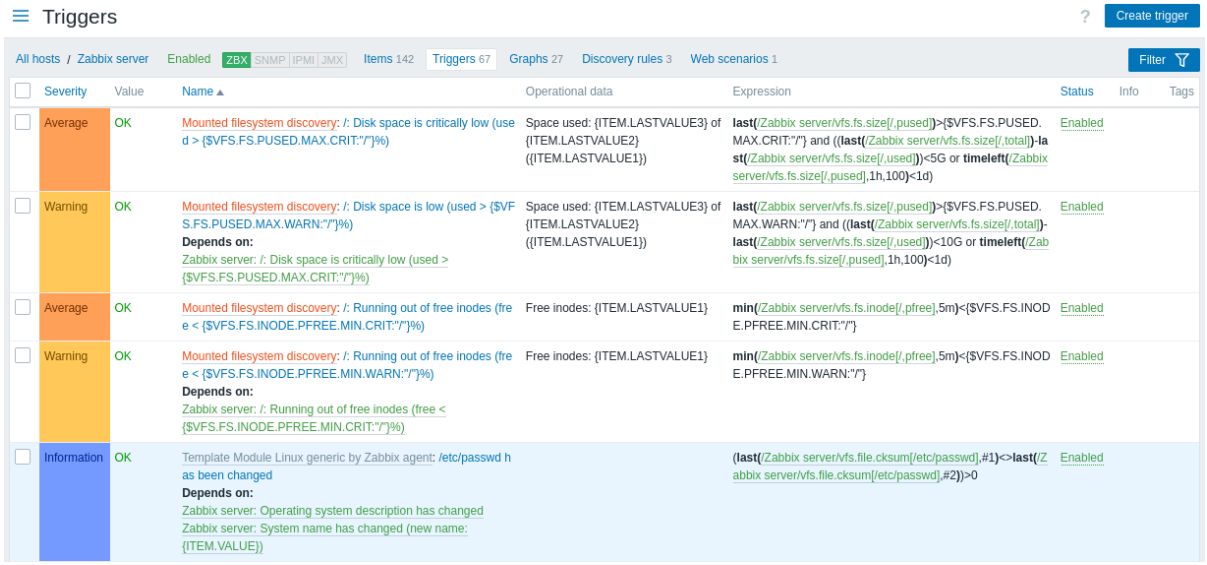
A number next to each clickable entity indicates the number of items grouped in it (based on the results of the main filter). When an entity is clicked, the numbers with other available entities are displayed with a plus sign indicating how many items may be

added to the current selection. Entities without items are not displayed unless selected in the subfilter before.

2 Triggers

Overview

The trigger list for a host can be accessed from *Data collection* → *Hosts* by clicking on *Triggers* for the respective host.



Displayed data:

Column	Description
Severity	Severity of the trigger is displayed by both name and cell background color.
Value	Trigger value is displayed: **OK** - the trigger is in the OK state **PROBLEM** - the trigger is in the Problem state
Host	Host of the trigger. This column is displayed only if multiple hosts are selected in the filter.
Name	Name of the trigger, displayed as a blue link to trigger details. Clicking on the trigger name link opens the trigger configuration form. If the host trigger belongs to a template, the template name is displayed before the trigger name, as a gray link. Clicking on the template link will open the trigger list on the template level. If the trigger has been created from a trigger prototype, its name is preceded by the low-level discovery rule name, in orange. Clicking on the discovery rule name will open the trigger prototype list.
Operational data	Operational data definition of the trigger, containing arbitrary strings and macros that will resolve dynamically in *Monitoring* → *Problems*.
Expression	Trigger expression is displayed. The host-item part of the expression is displayed as a link, leading to the item configuration form.
Status	Trigger status is displayed - *Enabled*, *Disabled* or *Unknown*. By clicking on the status you can change it - from Enabled to Disabled (and back); from Unknown to Disabled (and back). Problems of a disabled trigger are no longer displayed in the frontend, but are not deleted.
Info	If everything is working correctly, no icon is displayed in this column. In case of errors, a square icon with the letter "i" is displayed. Hover over the icon to see a tooltip with the error description.
Tags	If a trigger contains tags, tag name and value are displayed in this column.

To configure a new trigger, click on the *Create trigger* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Enable* - change trigger status to *Enabled*.
- *Disable* - change trigger status to *Disabled*.
- *Copy* - copy the triggers to other hosts or templates.

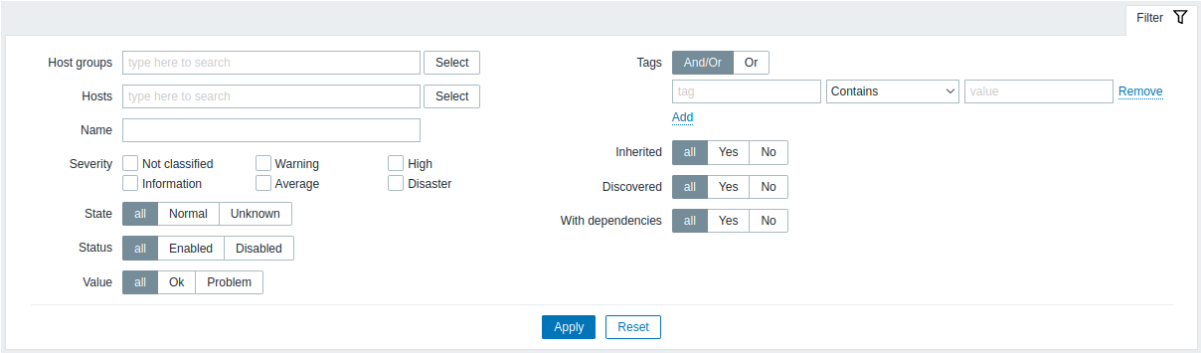
- *Mass update* - update several properties for a number of triggers at once.
- *Delete* - delete the triggers.

To use these options, mark the checkboxes before the respective triggers, then click on the required button.

Using filter

You can use the filter to display only the triggers you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* icon is available at the top right corner. Clicking on it will open a filter where you can specify the desired filtering criteria.



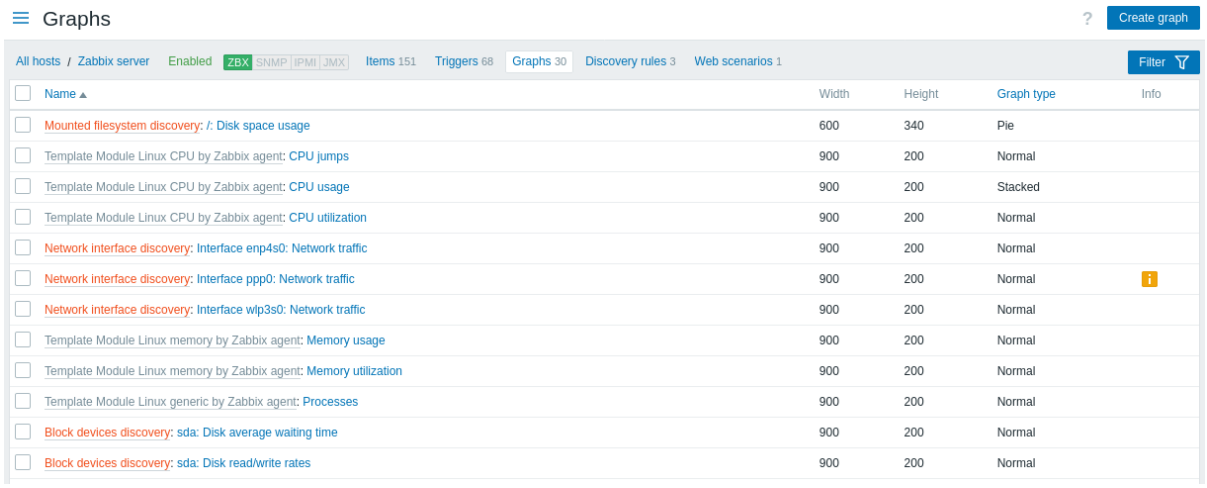
Parameter	Description
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups. Host groups containing templates only cannot be selected.
Hosts	Filter by one or more hosts. If host groups are already selected above, host selection is limited to those groups.
Name	Filter by trigger name.
Severity	Select to filter by one or several trigger severities.
State	Filter by trigger state.
Status	Filter by trigger status.
Value	Filter by trigger value.
Tags	Filter by trigger tag name and value. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified tag names **Equals** - include the specified tag names and values (case-sensitive) **Contains** - include the specified tag names where the tag values contain the entered string (substring match, case-insensitive) **Does not exist** - exclude the specified tag names **Does not equal** - exclude the specified tag names and values (case-sensitive) **Does not contain** - exclude the specified tag names where the tag values contain the entered string (substring match, case-insensitive) There are two calculation types for conditions: **And/Or** - all conditions must be met, conditions having the same tag name will be grouped by the Or condition **Or** - enough if one condition is met Macros and macro functions are supported both in tag name and tag value fields.
Inherited	Filter triggers inherited (or not inherited) from a template.
Discovered	Filter triggers discovered (or not discovered) by low-level discovery.
With dependencies	Filter triggers with (or without) dependencies.

3 Graphs

Overview

The custom graph list for a host can be accessed from *Data collection* → *Hosts* by clicking on *Graphs* for the respective host.

A list of existing graphs is displayed.



Displayed data:

Column	Description
Name	Name of the custom graph, displayed as a blue link to graph details. Clicking on the graph name link opens the graph configuration form. If the host graph belongs to a template, the template name is displayed before the graph name, as a gray link. Clicking on the template link will open the graph list on the template level. If the graph has been created from a graph prototype, its name is preceded by the low-level discovery rule name, in orange. Clicking on the discovery rule name will open the graph prototype list.
Width	Graph width is displayed.
Height	Graph height is displayed.
Graph type	Graph type is displayed - *Normal*, *Stacked*, *Pie* or *Exploded*.
Info	If the graph is working correctly, no icon is displayed in this column. In case of errors, a square icon with the letter "i" is displayed. Hover over the icon to see a tooltip with the error description.

To configure a new graph, click on the *Create graph* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Copy* - copy the graphs to other hosts or templates
- *Delete* - delete the graphs

To use these options, mark the checkboxes before the respective graphs, then click on the required button.

Using filter

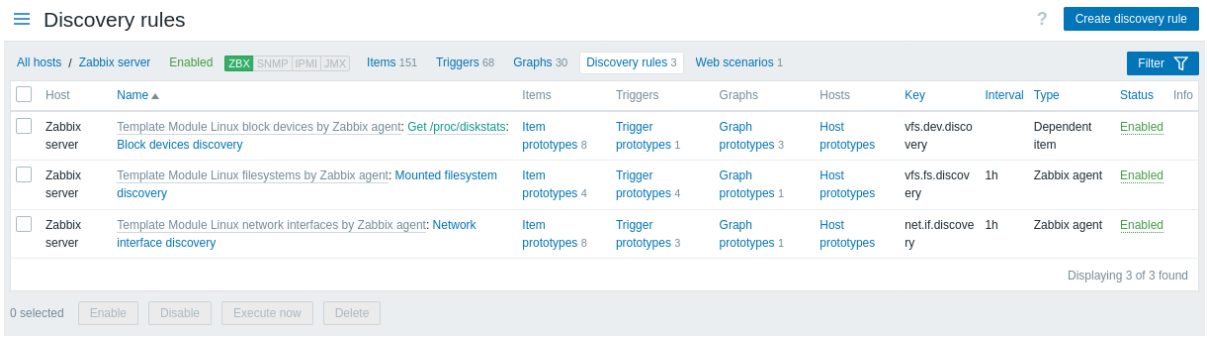
You can filter graphs by host group and host. For better search performance, data is searched with macros unresolved.

4 Discovery rules

Overview

The list of low-level discovery rules for a host can be accessed from *Data collection* → *Hosts* by clicking on *Discovery* for the respective host.

A list of existing low-level discovery rules is displayed. It is also possible to see all discovery rules independently of the host, or all discovery rules of a specific host group by changing the filter settings.



Displayed data:

Column	Description
Host	The visible host name is displayed. In the absence of a visible host name, the technical host name is displayed.
Name	Name of the rule, displayed as a blue link. Clicking on the rule name opens the low-level discovery rule configuration form. If the discovery rule belongs to a template, the template name is displayed before the rule name, as a gray link. Clicking on the template link will open the rule list on the template level.
Items	A link to the list of item prototypes is displayed. The number of existing item prototypes is displayed in gray.
Triggers	A link to the list of trigger prototypes is displayed. The number of existing trigger prototypes is displayed in gray.
Graphs	A link to the list of graph prototypes is displayed. The number of existing graph prototypes is displayed in gray.
Hosts	A link to the list of host prototypes is displayed. The number of existing host prototypes is displayed in gray.
Key	The item key used for discovery is displayed.
Interval	The frequency of performing discovery is displayed. *Note* that discovery can also be performed immediately by pushing the *Execute now* button below the list.
Type	The item type used for discovery is displayed (Zabbix agent, SNMP agent, etc).
Status	Discovery rule status is displayed - *Enabled*, *Disabled* or *Not supported*. By clicking on the status you can change it - from Enabled to Disabled (and back); from Not supported to Disabled (and back).
Info	If everything is fine, no icon is displayed in this column. In case of errors, a square icon with the letter "i" is displayed. Hover over the icon to see a tooltip with the error description.

To configure a new low-level discovery rule, click on the *Create discovery rule* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

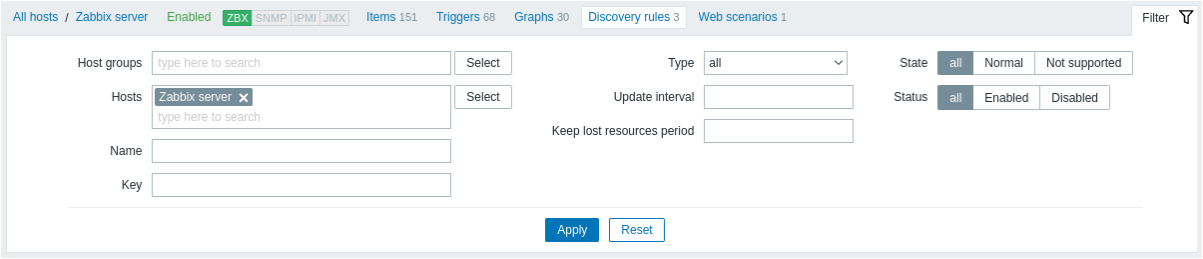
- *Enable* - change the low-level discovery rule status to *Enabled*.
- *Disable* - change the low-level discovery rule status to *Disabled*.
- *Execute now* - perform discovery based on the discovery rules immediately. See more details. Note that when performing discovery immediately, the configuration cache is not updated, thus the result will not reflect very recent changes to discovery rule configuration.
- *Delete* - delete the low-level discovery rules.

To use these options, mark the checkboxes before the respective discovery rules, then click on the required button.

Using filter

You can use the filter to display only the discovery rules you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is available above the list of discovery rules. If you click on it, a filter becomes available where you can filter discovery rules by host group, host, name, item key, item type, and other parameters.

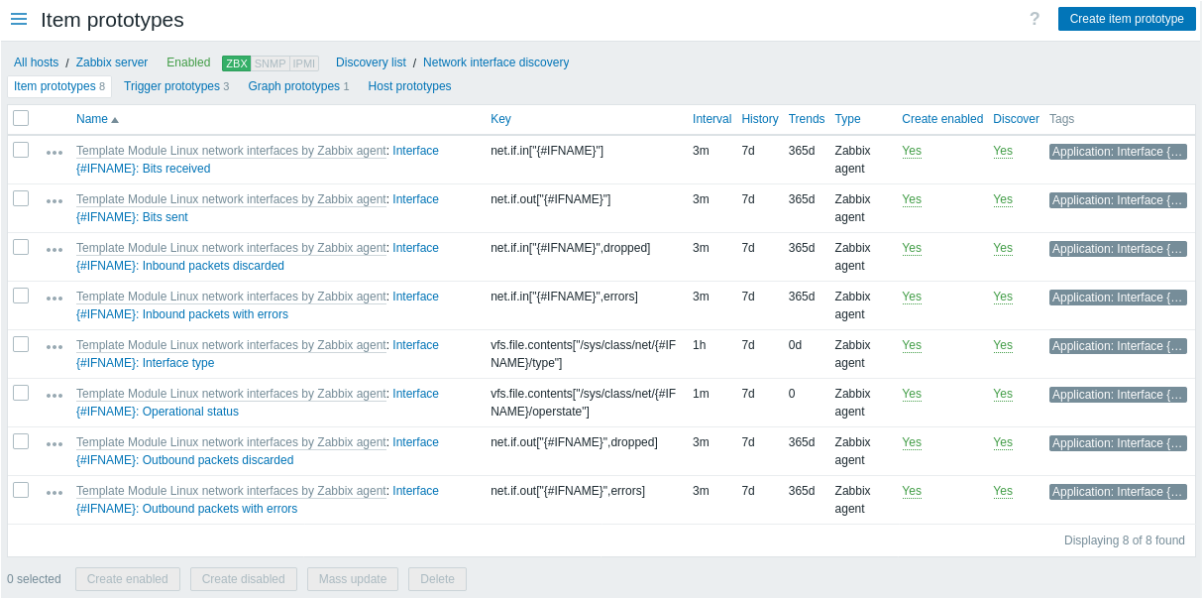


Parameter	Description
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
Hosts	Filter by one or more hosts.
Name	Filter by discovery rule name.
Key	Filter by discovery item key.
Type	Filter by discovery item type.
Update interval	Filter by update interval. Not available for Zabbix trapper and dependent items.
Keep lost resources period	Filter by Keep lost resources period.
SNMP OID	Filter by SNMP OID. Only available if *SNMP agent* is selected as type.
State	Filter by discovery rule state (All/Normal/Not supported).
Status	Filter by discovery rule status (All/Enabled/Disabled).

1 Item prototypes

Overview

In this section the item prototypes of a low-level discovery rule on the host are displayed. Item prototypes are the basis of real host items that are created during low-level discovery.



Displayed data:

Column	Description
Name	Name of the item prototype, displayed as a blue link. Clicking on the name opens the item prototype configuration form. If the item prototype belongs to a template, the template name is displayed before the rule name, as a gray link. Clicking on the template link will open the item prototype list on the template level.

Column	Description
Key	Key of the item prototype is displayed.
Interval	Frequency of the check is displayed.
History	How many days to keep item data history is displayed.
Trends	How many days to keep item trends history is displayed.
Type	Type of the item prototype is displayed (Zabbix agent, SNMP agent, simple check, etc).
Create enabled	Create the item based on this prototype as: **Yes** - enabled **No** - disabled. You can switch between 'Yes' and 'No' by clicking on them.
Discover	Discover the item based on this prototype: **Yes** - discover **No** - do not discover. You can switch between 'Yes' and 'No' by clicking on them.
Tags	Tags of the item prototype are displayed.

To configure a new item prototype, click on the *Create item prototype* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

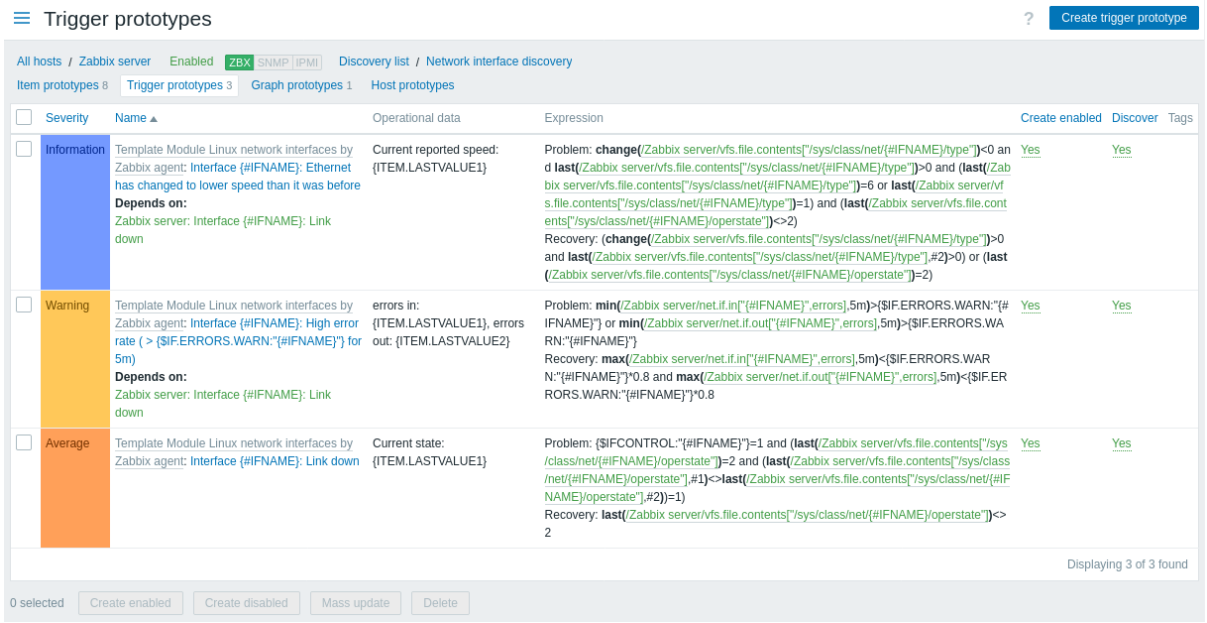
- *Create enabled* - create these items as *Enabled*
- *Create disabled* - create these items as *Disabled*
- *Mass update* - mass update these item prototypes
- *Delete* - delete these item prototypes

To use these options, mark the checkboxes before the respective item prototypes, then click on the required button.

2 Trigger prototypes

Overview

In this section the trigger prototypes of a low-level discovery rule on the host are displayed. Trigger prototypes are the basis of real host triggers that are created during low-level discovery.



Displayed data:

Column	Description
Name	Name of the trigger prototype, displayed as a blue link. Clicking on the name opens the trigger prototype configuration form. If the trigger prototype belongs to a linked template, the template name is displayed before the trigger name, as a gray link. Clicking on the template link will open the trigger prototype list on the linked template level.

Column	Description
Operational data	Format of the operational data of the trigger is displayed, containing arbitrary strings and macros that will resolve dynamically in *Monitoring* → *Problems*.
Create enabled	Create the trigger based on this prototype as: **Yes** - enabled **No** - disabled. You can switch between 'Yes' and 'No' by clicking on them.
Discover	Discover the trigger based on this prototype: **Yes** - discover **No** - do not discover. You can switch between 'Yes' and 'No' by clicking on them.
Tags	Tags of the trigger prototype are displayed.

To configure a new trigger prototype, click on the *Create trigger prototype* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

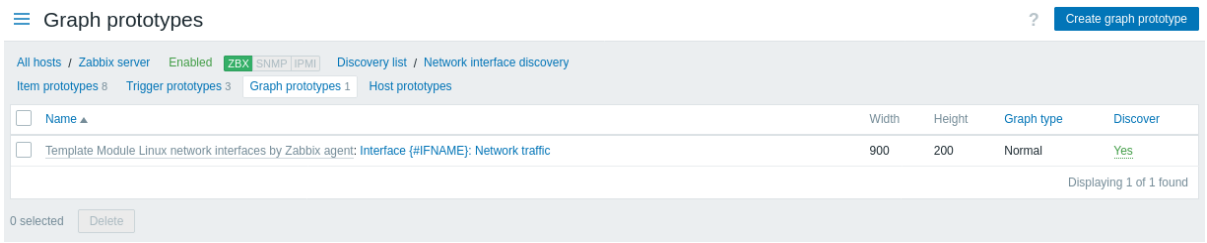
- *Create enabled* - create these triggers as *Enabled*
- *Create disabled* - create these triggers as *Disabled*
- *Mass update* - mass update these trigger prototypes
- *Delete* - delete these trigger prototypes

To use these options, mark the checkboxes before the respective trigger prototypes, then click on the required button.

3 Graph prototypes

Overview

In this section the graph prototypes of a low-level discovery rule on the host are displayed. Graph prototypes are the basis of real host graphs that are created during low-level discovery.



Displayed data:

Column	Description
Name	Name of the graph prototype, displayed as a blue link. Clicking on the name opens the graph prototype configuration form. If the graph prototype belongs to a linked template, the template name is displayed before the graph name, as a gray link. Clicking on the template link will open the graph prototype list on the linked template level.
Width	Width of the graph prototype is displayed.
Height	Height of the graph prototype is displayed.
Type	Type of the graph prototype is displayed - *Normal*, *Stacked*, *Pie* or *Exploded*.
Discover	Discover the graph based on this prototype: **Yes** - discover **No** - do not discover. You can switch between 'Yes' and 'No' by clicking on them.

To configure a new graph prototype, click on the *Create graph prototype* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

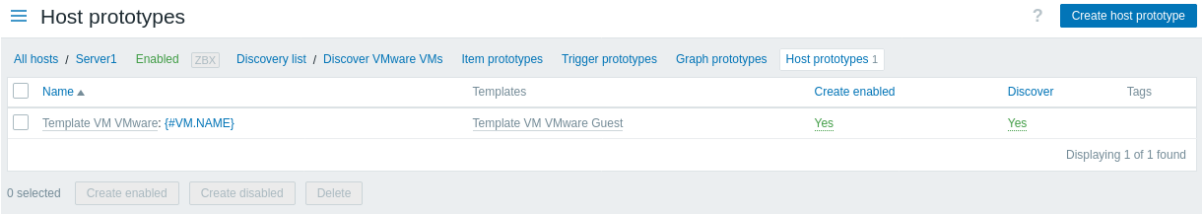
- *Delete* - delete these graph prototypes

To use these options, mark the checkboxes before the respective graph prototypes, then click on the required button.

4 Host prototypes

Overview

In this section the host prototypes of a low-level discovery rule on the host are displayed. Host prototypes are the basis of real hosts that are created during low-level discovery.



Displayed data:

Column	Description
Name	Name of the host prototype, displayed as a blue link. Clicking on the name opens the host prototype configuration form. If the host prototype belongs to a linked template, the template name is displayed before the host name, as a gray link. Clicking on the template link will open the host prototype list on the linked template level.
Templates	Templates of the host prototype are displayed.
Create enabled	Create the host based on this prototype as: **Yes** - enabled **No** - disabled. You can switch between 'Yes' and 'No' by clicking on them.
Discover	Discover the host based on this prototype: **Yes** - discover **No** - do not discover. You can switch between 'Yes' and 'No' by clicking on them.
Tags	Tags of the host prototype are displayed.

To configure a new host prototype, click on the *Create host prototype* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Create enabled* - create these hosts as *Enabled*
- *Create disabled* - create these hosts as *Disabled*
- *Delete* - delete these host prototypes

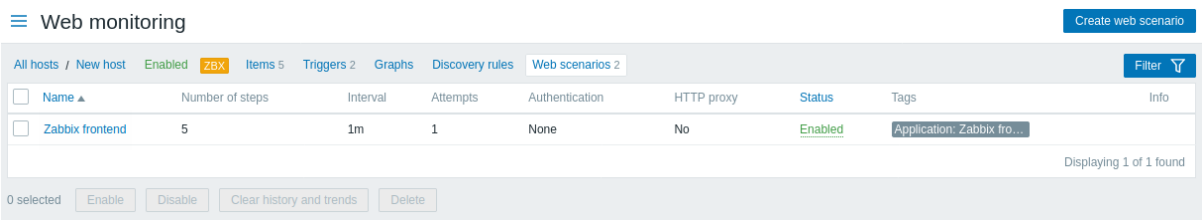
To use these options, mark the checkboxes before the respective host prototypes, then click on the required button.

5 Web scenarios

Overview

The web scenario list for a host can be accessed from *Data collection* → *Hosts* by clicking on *Web* for the respective host.

A list of existing web scenarios is displayed.



Displayed data:

Column	Description
Name	Name of the web scenario. Clicking on the web scenario name opens the web scenario configuration form. If the host web scenario belongs to a template, the template name is displayed before the web scenario name as a gray link. Clicking on the template link will open the web scenario list on the template level.
Number of steps	The number of steps the scenario contains.
Update interval	How often the scenario is performed.
Attempts	How many attempts for executing web scenario steps are performed.
Authentication	Authentication method is displayed - Basic, NTLM, or None.
HTTP proxy	Displays HTTP proxy or 'No' if not used.
Status	Web scenario status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.
Tags	Web scenario tags are displayed. Up to three tags (name:value pairs) can be displayed. If there are more tags, a "..." link is displayed that allows to see all tags on mouseover.
Info	If everything is working correctly, no icon is displayed in this column. In case of errors, a square icon with the letter "i" is displayed. Hover over the icon to see a tooltip with the error description.

To configure a new web scenario, click on the *Create web scenario* button at the top right corner.

Mass editing options

Buttons below the list offer some mass-editing options:

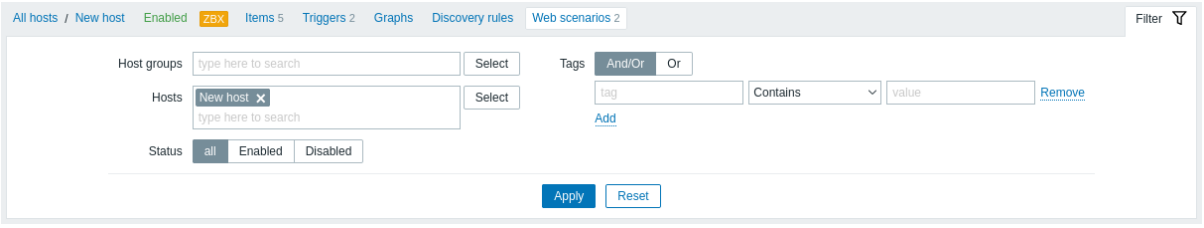
- *Enable* - change the scenario status to *Enabled*
- *Disable* - change the scenario status to *Disabled*
- *Clear history and trends* - clear history and trend data for the scenarios
- *Delete* - delete the web scenarios

To use these options, mark the checkboxes before the respective web scenarios, then click on the required button.

Using filter

You can use the filter to display only the scenarios you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is available above the list of web scenarios. If you click on it, a filter becomes available where you can filter scenarios by host group, host, status and tags.

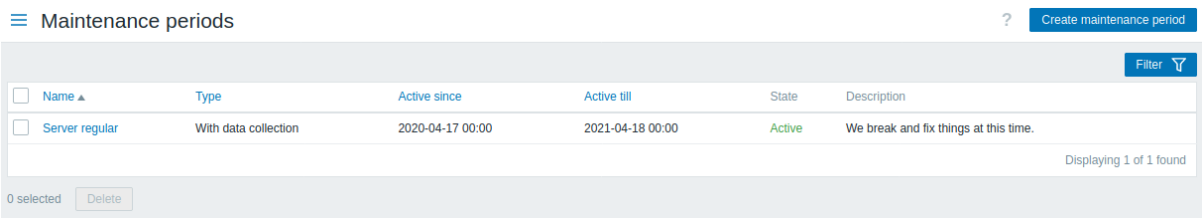


5 Maintenance

Overview

In the *Data collection* → *Maintenance* section users can configure and maintain maintenance periods for hosts.

A listing of existing maintenance periods with their details is displayed.



Displayed data:

Column	Description
Name	Name of the maintenance period. Clicking on the maintenance period name opens the maintenance period configuration form.
Type	The type of maintenance is displayed: *With data collection* or *No data collection*
Active since	The date and time when executing maintenance periods becomes active. Note: This time does not activate a maintenance period; maintenance periods need to be set separately.
Active till	The date and time when executing maintenance periods stops being active.
State	The state of the maintenance period: **Approaching** - will become active soon **Active** - is active **Expired** - is not active any more
Description	Description of the maintenance period is displayed.

To configure a new maintenance period, click on the *Create maintenance period* button in the top right-hand corner.

Mass editing options

A button below the list offers one mass-editing option:

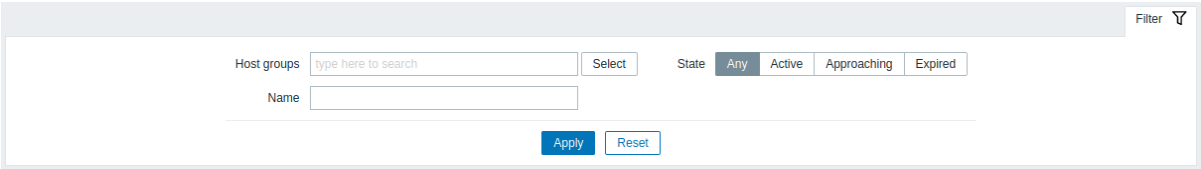
- *Delete* - delete the maintenance periods

To use this option, mark the checkboxes before the respective maintenance periods and click on *Delete*.

Using filter

You can use the filter to display only the maintenance periods you are interested in. For better search performance, data is searched with macros unresolved.

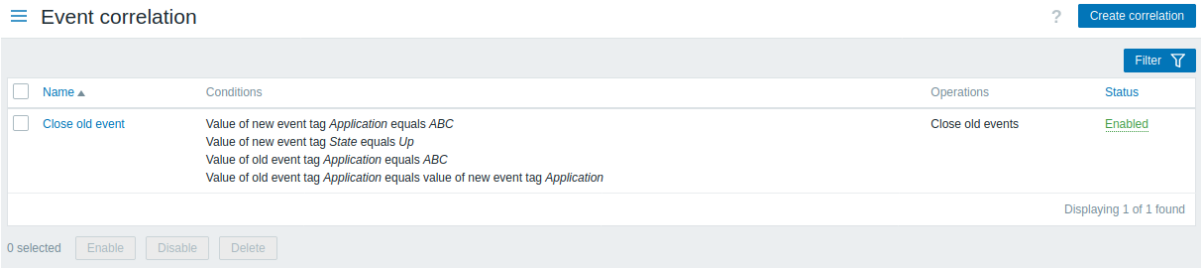
The *Filter* link is available above the list of maintenance periods. If you click on it, a filter becomes available where you can filter maintenance periods by host group, name and state.



6 Event correlation

Overview

In the *Data collection* → *Event correlation* section users can configure and maintain global correlation rules for Zabbix events.



Displayed data:

Column	Description
Name	Name of the correlation rule. Clicking on the correlation rule name opens the rule configuration form.
Conditions	Correlation rule conditions are displayed.
Operations	Correlation rule operations are displayed.
Status	Correlation rule status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.

To configure a new correlation rule, click on the *Create correlation* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

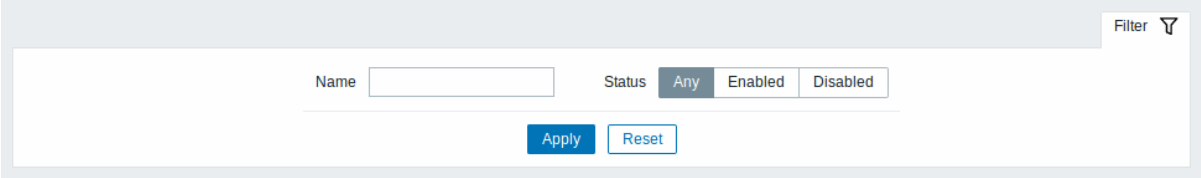
- *Enable* - change the correlation rule status to *Enabled*
- *Disable* - change the correlation rule status to *Disabled*
- *Delete* - delete the correlation rules

To use these options, mark the checkboxes before the respective correlation rules, then click on the required button.

Using filter

You can use the filter to display only the correlation rules you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is available above the list of correlation rules. If you click on it, a filter becomes available where you can filter correlation rules by name and status.

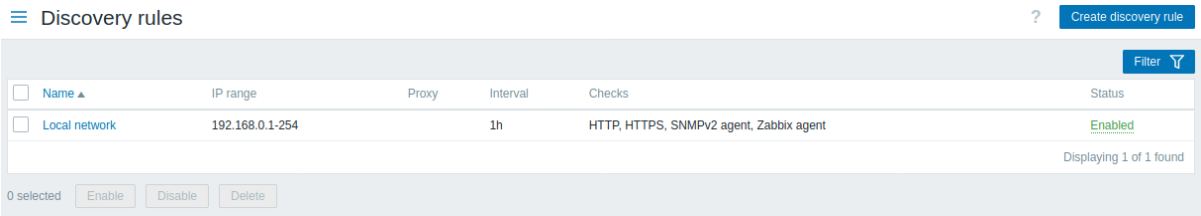


7 Discovery

Overview

In the *Data collection* → *Discovery* section users can configure and maintain discovery rules.

A listing of existing discovery rules with their details is displayed.



Displayed data:

Column	Description
Name	Name of the discovery rule. Clicking on the discovery rule name opens the discovery rule configuration form.
IP range	The range of IP addresses to use for network scanning is displayed.
Proxy	The proxy name is displayed, if discovery is performed by the proxy.
Interval	The frequency of performing discovery displayed.
Checks	The types of checks used for discovery are displayed.
Status	Action status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.

To configure a new discovery rule, click on the *Create discovery rule* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

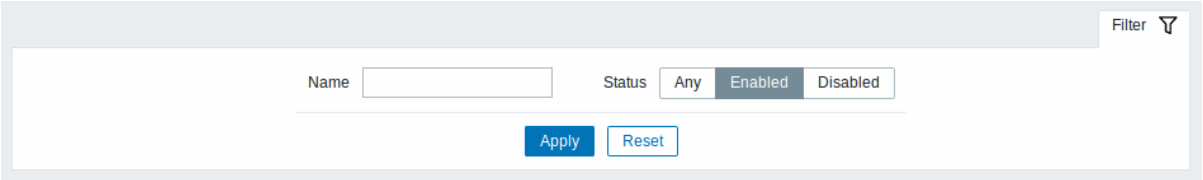
- *Enable* - change the discovery rule status to *Enabled*
- *Disable* - change the discovery rule status to *Disabled*
- *Delete* - delete the discovery rules

To use these options, mark the checkboxes before the respective discovery rules, then click on the required button.

Using filter

You can use the filter to display only the discovery rules you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is available above the list of discovery rules. If you click on it, a filter becomes available where you can filter discovery rules by name and status.



7 Alerts

Overview

This menu features sections that are related to configuring alerts in Zabbix.

1 Actions

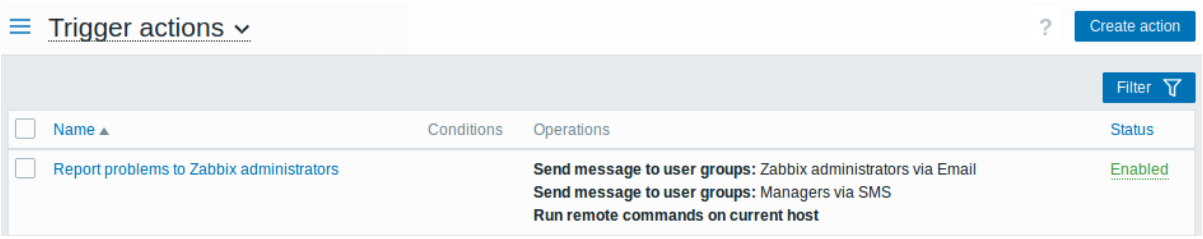
Overview

In the *Alerts* → *Actions* section users can configure and maintain actions.

The actions displayed are actions assigned to the selected event source (trigger, services, discovery, autoregistration, internal actions).

Actions are grouped into subsections by event source (trigger, service, discovery, autoregistration, internal actions). The list of available subsections appears upon clicking on *Actions* in the *Alerts* menu section. It is also possible to switch between subsections by using the title dropdown in the top left corner.

After selecting a subsection, a list of existing actions with their details will be displayed.



Displayed data:

Column	Description
Name	Name of the action. Clicking on the action name opens the action configuration form.
Conditions	Action conditions are displayed.
Operations	Action operations are displayed. Since Zabbix 2.2, the operation list also displays the media type (email, SMS or script) used for notification as well as the name and surname (in parentheses after the username) of a notification recipient. Action operation can both be a notification or a remote command depending on the selected type of operation.
Status	Action status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it. See the Escalations section for more details as to what happens if an action is disabled during an escalation in progress.

To configure a new action, click on the *Create action* button in the top right-hand corner.

For users without Super admin rights actions are displayed according to the permission settings. That means in some cases a user without Super admin rights isn't able to view the complete action list because of certain permission restrictions. An action is displayed to the user without Super admin rights if the following conditions are fulfilled:

- The user has read-write access to host groups, hosts, templates, and triggers in action conditions
- The user has read-write access to host groups, hosts, and templates in action operations, recovery operations, and update operations
- The user has read access to user groups and users in action operations, recovery operations, and update operations

Mass editing options

Buttons below the list offer some mass-editing options:

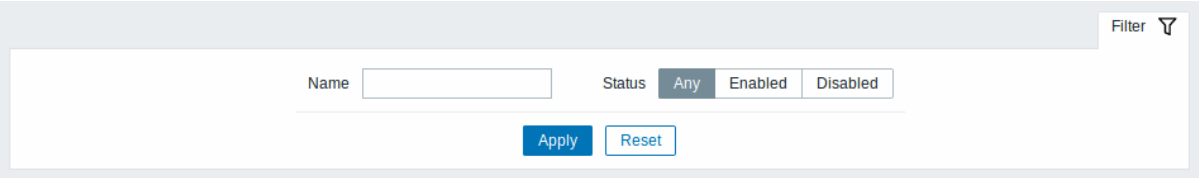
- *Enable* - change the action status to *Enabled*
- *Disable* - change the action status to *Disabled*
- *Delete* - delete the actions

To use these options, mark the checkboxes before the respective actions, then click on the required button.

Using filter

You can use the filter to display only the actions you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is available above the list of actions. If you click on it, a filter becomes available where you can filter actions by name and status.



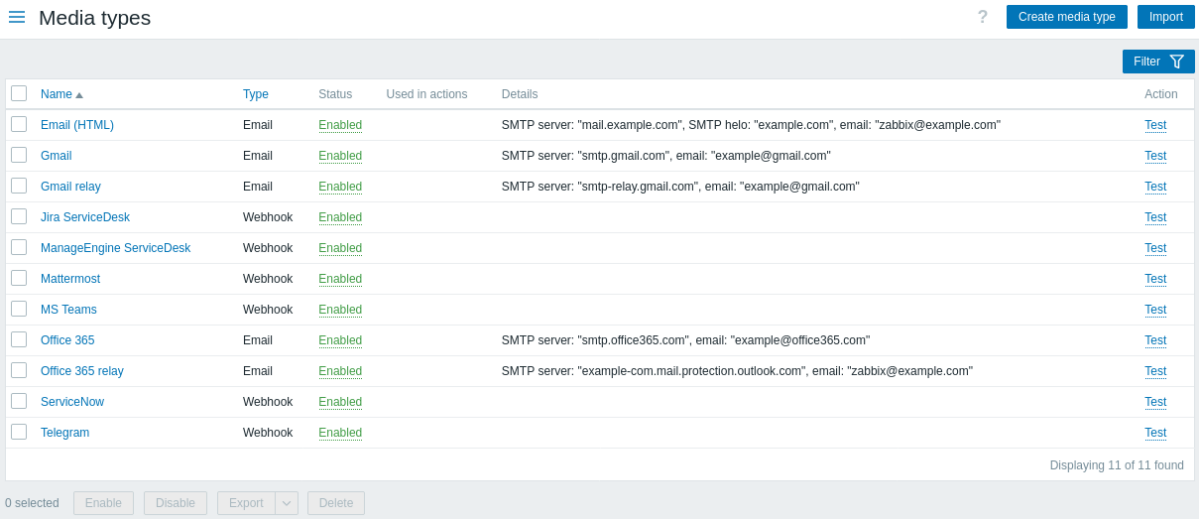
2 Media types

Overview

In the *Alerts → Media types* section users can configure and maintain media type information.

Media type information contains general instructions for using a medium as delivery channel for notifications. Specific details, such as the individual email addresses to send a notification to are kept with individual users.

A listing of existing media types with their details is displayed.



Displayed data:

Column	Description
Name	Name of the media type. Clicking on the name opens the media type configuration form.
Type	Type of the media (email, SMS, etc) is displayed.
Status	Media type status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.
Used in actions	All actions where the media type is used directly (selected in the *Send only to* dropdown) are displayed. Clicking on the action name opens the action configuration form.
Details	Detailed information of the media type is displayed.
Actions	The following action is available: **Test** - click to open a testing form where you can enter media type parameters (e.g. a recipient address with test subject and body) and send a test message to verify that the configured media type works. See also: Media type testing for Email, Webhook, or Script.

To configure a new media type, click on the *Create media type* button in the top right-hand corner.

To import a media type from XML, click on the *Import* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

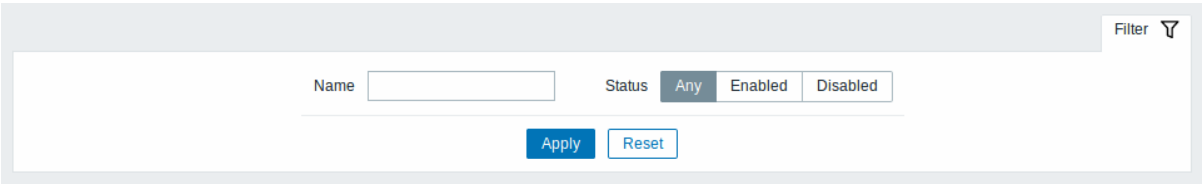
- *Enable* - change the media type status to *Enabled*
- *Disable* - change the media type status to *Disabled*
- *Export* - export the media types to a YAML, XML or JSON file
- *Delete* - delete the media types

To use these options, mark the checkboxes before the respective media types, then click on the required button.

Using filter

You can use the filter to display only the media types you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is available above the list of media types. If you click on it, a filter becomes available where you can filter media types by name and status.



3 Scripts

Overview

In the *Alerts* → *Scripts* section user-defined global scripts can be configured and maintained.

Global scripts, depending on the configured scope and also user permissions, are available for execution:

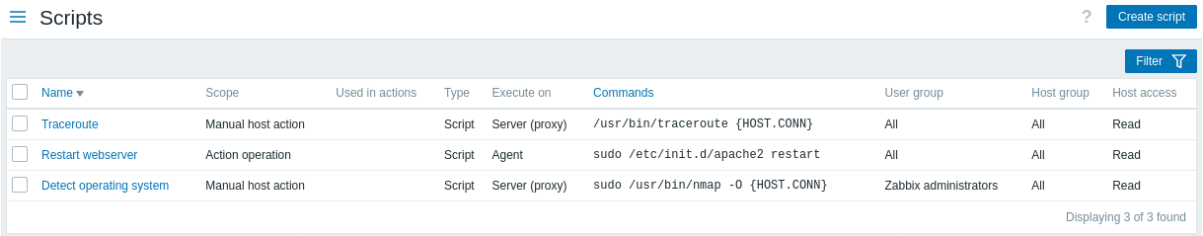
- from the host menu in various frontend locations (*Dashboard*, *Problems*, *Latest data*, *Maps*, etc.)
- from the event menu
- can be run as an action operation

The scripts are executed on Zabbix agent, Zabbix server (proxy) or Zabbix server only. See also Command execution.

Both on Zabbix agent and Zabbix proxy remote scripts are disabled by default. They can be enabled by:

- For remote commands executed on Zabbix agent:
  - adding an AllowKey=system.run[<command>,*] parameter for each allowed command in agent configuration, * stands for wait and nowait mode;
- For remote commands executed on Zabbix proxy:
  - **Warning: It is not required to enable remote commands on Zabbix proxy if remote commands are executed on Zabbix agent that is monitored by Zabbix proxy.** If, however, it is required to execute remote commands on Zabbix proxy, set *EnableRemoteCommands* parameter to '1' in the proxy configuration.

A listing of existing scripts with their details is displayed.



Displayed data:

Column	Description
Name	Name of the script. Clicking on the script name opens the script configuration form.
Scope	Scope of the script - action operation, manual host action or manual event action. This setting determines where the script is available.

Column	Description
Used in actions	Actions where the script is used are displayed.
Type	Script type is displayed - *URL*, *Webhook*, *Script*, *SSH*, *Telnet* or *IPMI* command.
Execute on	It is displayed whether the script will be executed on Zabbix agent, Zabbix server (proxy) or Zabbix server only.
Commands	All commands to be executed within the script are displayed.
User group	The user group that the script is available to is displayed (or *All* for all user groups).
Host group	The host group that the script is available for is displayed (or *All* for all host groups).
Host access	The permission level for the host group is displayed - *Read* or *Write*. Only users with the required permission level will have access to executing the script.

To configure a new script, click on the *Create script* button in the top right-hand corner.

Mass editing options

A button below the list offers one mass-editing option:

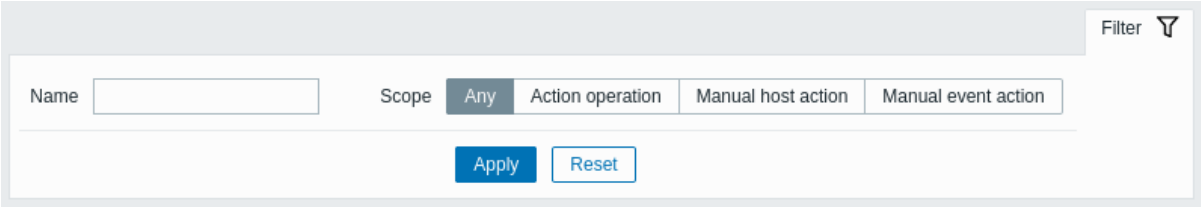
- *Delete* - delete the scripts

To use this option, mark the checkboxes before the respective scripts and click on *Delete*.

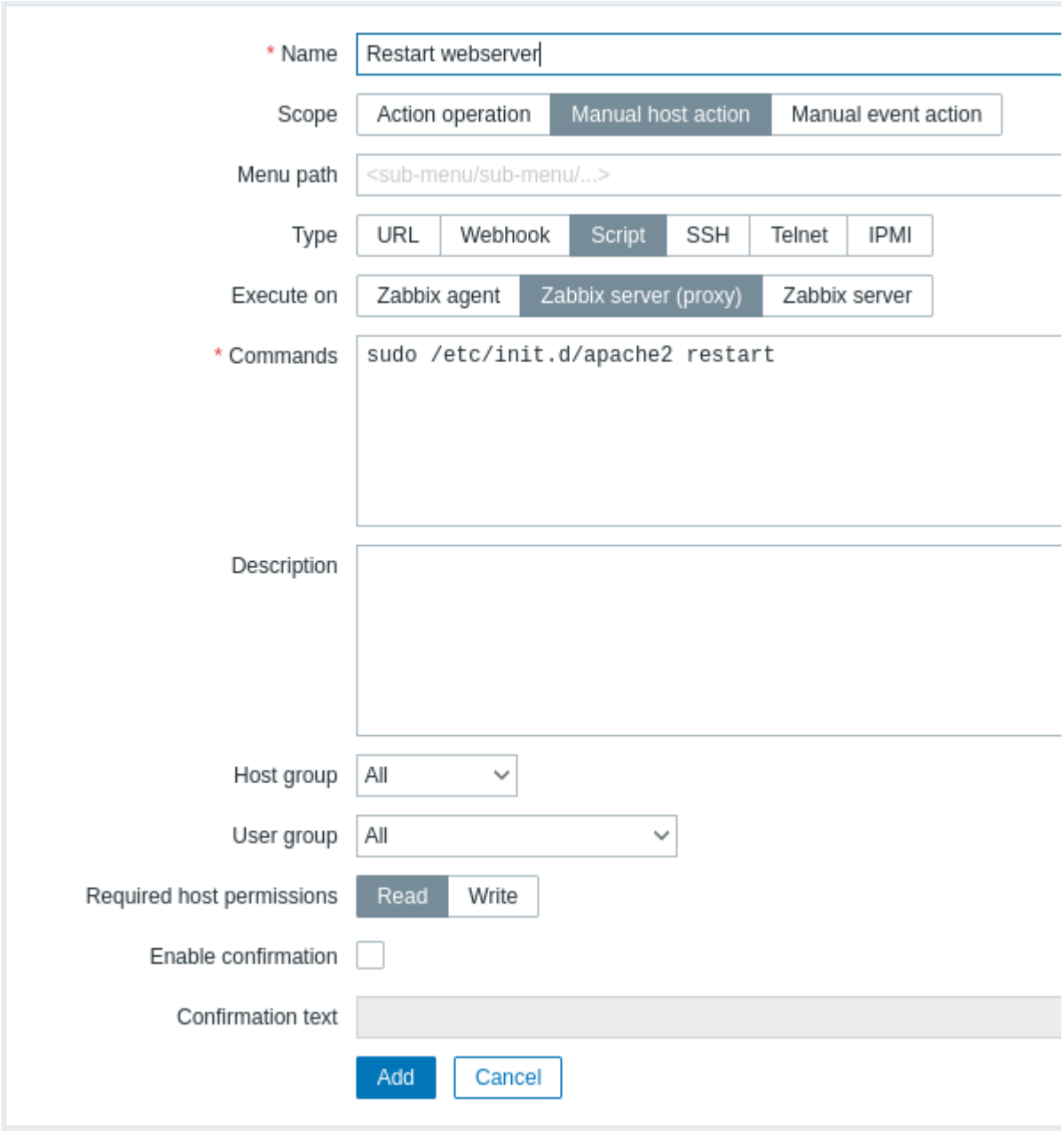
Using filter

You can use the filter to display only the scripts you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is available above the list of scripts. If you click on it, a filter becomes available where you can filter scripts by name and scope.



Configuring a global script



Script attributes:

Parameter	Description
Name	Unique name of the script. E.g. `Clear /tmp filesystem`
Scope	Scope of the script - action operation, manual host action or manual event action. This setting determines where the script can be used - in remote commands of action operations, from the host menu or from the event menu respectively. Setting the scope to 'Action operation' makes the script available for all users with access to *Alerts* → *Actions*. If a script is actually used in an action, its scope cannot be changed away from 'action operation'. **Macro support** The scope affects the range of available macros. For example, user-related macros ({USER.*}) are supported in scripts to allow passing information about the user that launched the script. However, they are not supported if the script scope is action operation, as action operations are executed automatically. To find out which macros are supported, do a search for 'Trigger-based notifications and commands/Trigger-based commands', 'Manual host action scripts' and 'Manual event action scripts' in the supported macro table. Note that if a macro may resolve to a value with spaces (for example, host name), don't forget to quote as needed.

Parameter	Description
Menu path	The desired menu path to the script. For example, `Default` or `Default/`, will display the script in the respective directory. Menus can be nested, e.g. `Main menu/Sub menu1/Sub menu2`. When accessing scripts through the host/event menu in monitoring sections, they will be organized according to the given directories. This field is displayed only if 'Manual host action' or 'Manual event action' is selected as *Scope*.
Type	Click the respective button to select script type: **URL**, **Webhook**, **Script**, **SSH**, **Telnet** or **IPMI** command. The type **URL** is available only when 'Manual host action' or 'Manual event action' is selected as *Scope*.
Script type: URL	
URL	Specify the URL for quick access from the host menu or event menu. Macros and custom user macros are supported. Macro support depends on the scope of the script (see *Scope* above). Macro values must not be URL-encoded.
Open in new window	Determines whether the URL should be opened in a new or the same browser tab.
Script type: Webhook	
Parameters	Specify the webhook variables as attribute-value pairs. See also: Webhook media configuration. Macros and custom user macros are supported in parameter values. Macro support depends on the scope of the script (see *Scope* above).
Script	Enter the JavaScript code in the block that appears when clicking in the parameter field (or on the view/edit button next to it). Macro support depends on the scope of the script (see *Scope* above). See also: Webhook media configuration, Additional Javascript objects.
Timeout	JavaScript execution timeout (1-60s, default 30s). Time suffixes are supported, e.g. 30s, 1m.
Script type: Script	
Execute on	Click the respective button to execute the shell script on: **Zabbix agent** - the script will be executed by Zabbix agent (if the system.run item is allowed) on the host **Zabbix server (proxy)** - the script will be executed by Zabbix server or proxy (if enabled by EnableRemoteCommands) - depending on whether the host is monitored by server or proxy **Zabbix server** - the script will be executed by Zabbix server only
Commands	Enter full path to the commands to be executed within the script. Macro support depends on the scope of the script (see *Scope* above). Custom user macros are supported.
Script type: SSH	
Authentication method	Select authentication method - password or public key.
Username	Enter the username.
Password	Enter the password. This field is available if 'Password' is selected as the authentication method.
Public key file	Enter the path to the public key file. This field is available if 'Public key' is selected as the authentication method.
Private key file	Enter the path to the private key file. This field is available if 'Public key' is selected as the authentication method.
Passphrase	Enter the passphrase. This field is available if 'Public key' is selected as the authentication method.
Port	Enter the port.
Commands	Enter the commands. Macro support depends on the scope of the script (see *Scope* above). Custom user macros are supported.
Script type: Telnet	
Username	Enter the username.
Password	Enter the password.
Port	Enter the port.
Commands	Enter the commands. Macro support depends on the scope of the script (see *Scope* above). Custom user macros are supported.
Script type: IPMI	

Parameter	Description
Command	Enter the IPMI command. Macro support depends on the scope of the script (see *Scope* above). Custom user macros are supported.
Description	Enter a description for the script.
Host group	Select the host group that the script will be available for (or *All* for all host groups).
User group	Select the user group that the script will be available to (or *All* for all user groups). This field is displayed only if 'Manual host action' or 'Manual event action' is selected as *Scope*.
Required host permissions	Select the permission level for the host group - *Read* or *Write*. Only users with the required permission level will have access to executing the script. This field is displayed only if 'Manual host action' or 'Manual event action' is selected as *Scope*.
Enable confirmation	Mark the checkbox to display a confirmation message before executing the script. This feature might be especially useful with potentially dangerous operations (like a reboot script) or ones that might take a long time. This option is displayed only if 'Manual host action' or 'Manual event action' is selected as *Scope*.
Confirmation text	Enter a custom confirmation text for the confirmation popup enabled with the checkbox above (for example, *Remote system will be rebooted. Are you sure?*). To see how the text will look like, click on *Test confirmation* next to the field. Macros and custom user macros are supported. *Note:* the macros will not be expanded when testing the confirmation message. This field is displayed only if 'Manual host action' or 'Manual event action' is selected as *Scope*.

Script execution and result

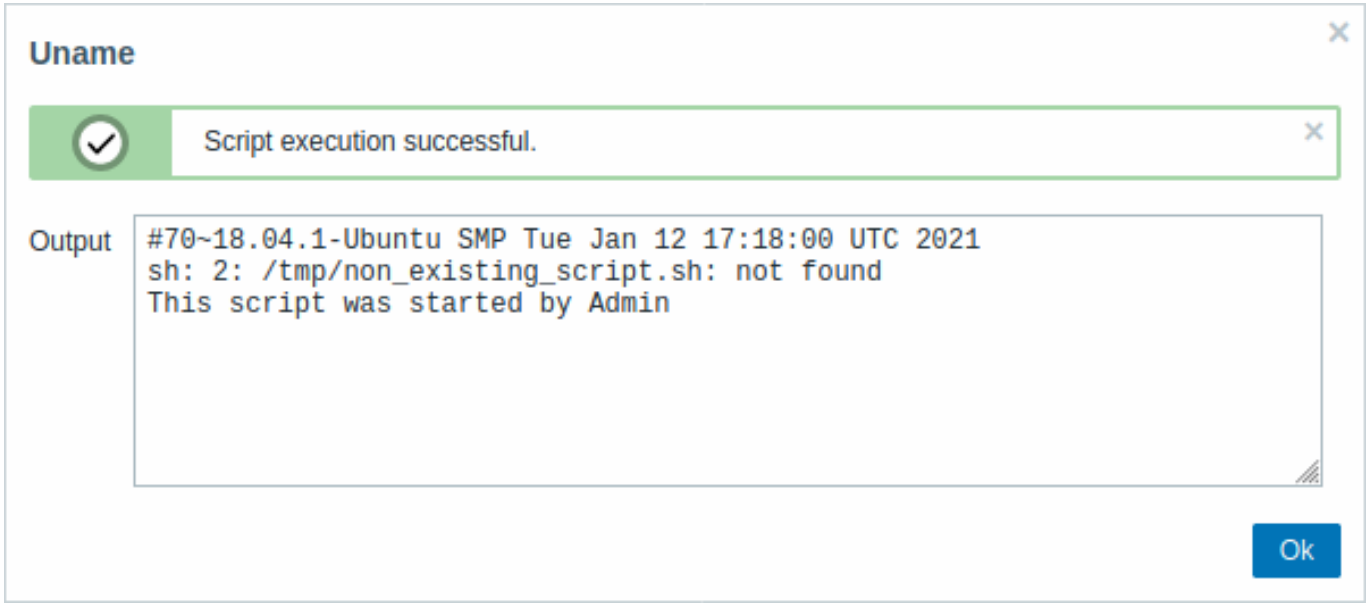
Scripts run by Zabbix server are executed in the order described in the *Command execution* page (including exit code checking). The script result will be displayed in a pop-up window that will appear after the script is run.

The return value of the script is a standard output together with a standard error.

The return value is limited to 16MB (including trailing whitespace that is truncated); database limits also apply. When data has to pass through Zabbix proxy, it must be stored in the database, thus subjecting it to the same database limits.

See an example of a script and the result window below:

```
uname -v
/tmp/non_existing_script.sh
echo "This script was started by {USER.USERNAME}"
```



The script result does not display the script itself.

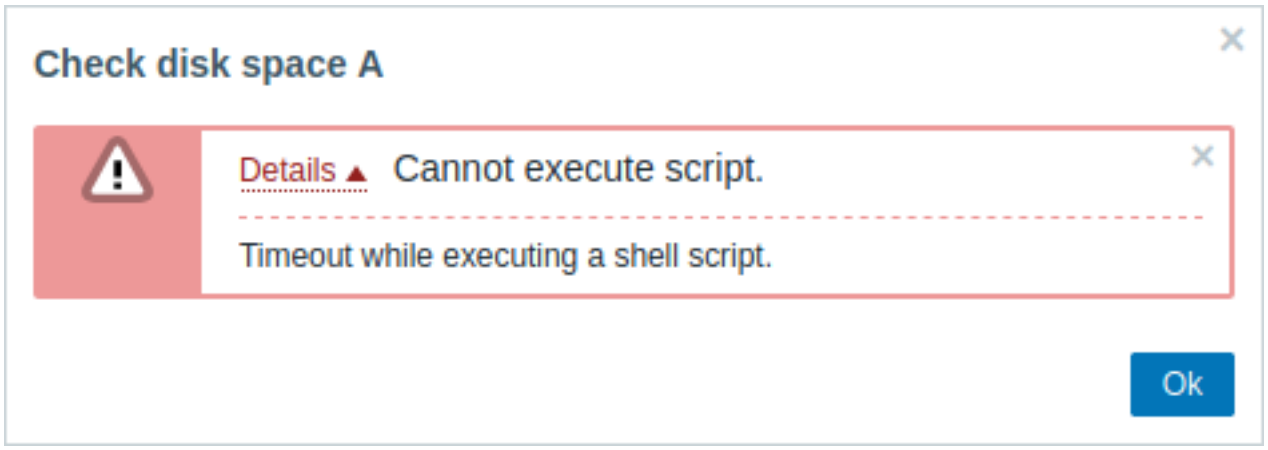
Script timeout

Zabbix agent

You may encounter a situation when a timeout occurs while executing a script.

See an example of a script running on Zabbix agent and the result window below:

```
sleep 5
df -h
```



The error message, in this case, is the following:

```
Timeout while executing a shell script.
```

In order to avoid such situations, it is advised to optimize the script itself instead of adjusting the Timeout parameter to a corresponding value (in the example above, > '5') in Zabbix agent configuration and Zabbix server configuration.

In case the Timeout parameter is changed in Zabbix agent configuration, the following error message will appear:

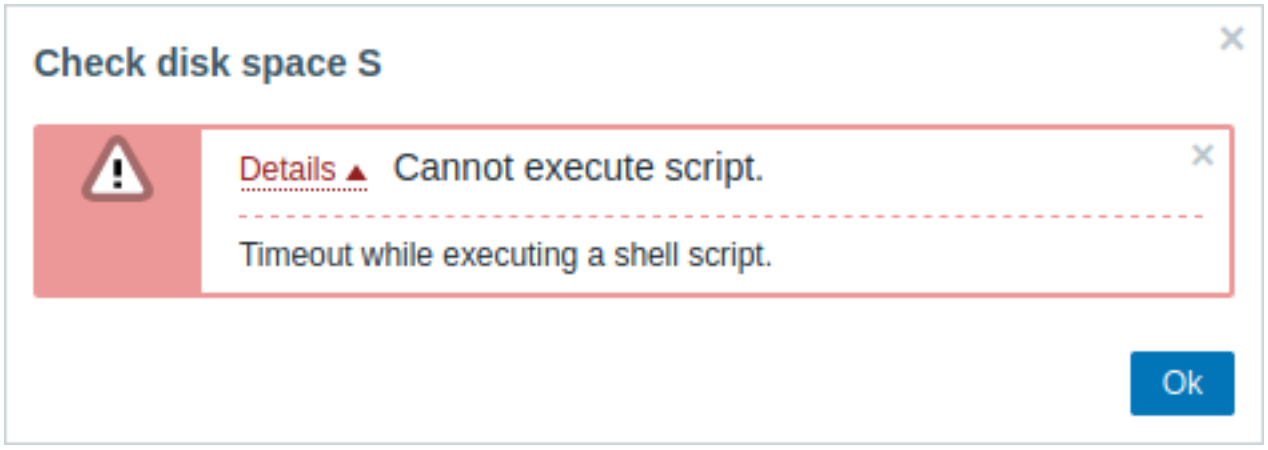
```
Get value from agent failed: ZBX_TCP_READ() timed out.
```

It means that modification has been made in Zabbix agent configuration but it is required to modify Timeout setting in Zabbix server configuration as well.

Zabbix server/proxy

See an example of a script running on Zabbix server and the result window below:

```
sleep 11
df -h
```



It is also advised to optimize the script itself (instead of adjusting TrapperTimeout parameter to a corresponding value (in our case, > '11') by modifying the Zabbix server configuration).

8 Users

Overview

This menu features sections that are related to configuring users in Zabbix. This menu is available to SuperAdmin user type users only.

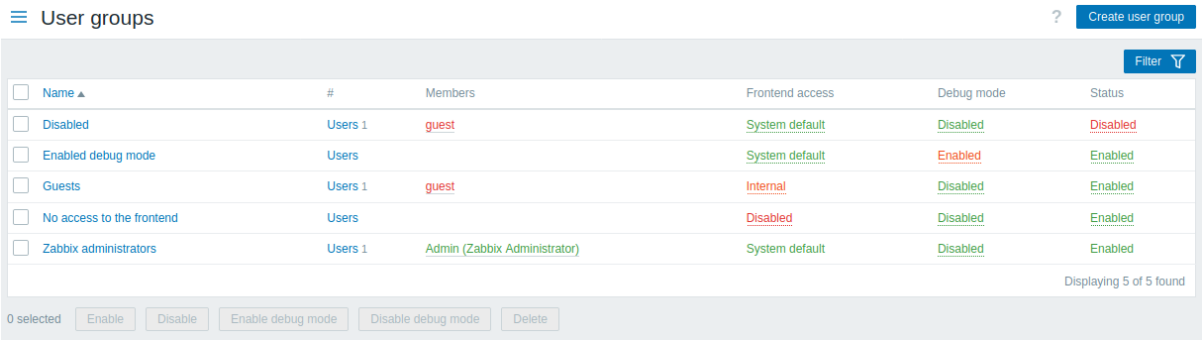
1 User groups

Overview

In the *Users* → *User groups* section user groups of the system are maintained.

User groups

A listing of existing user groups with their details is displayed.



Displayed data:

Column	Description
Name	Name of the user group. Clicking on the user group name opens the user group configuration form.
#	The number of users in the group. Clicking on *Users* will display the respective users filtered out in the user list.
Members	Usernames of individual users in the user group (with name and surname in parentheses). Clicking on the username will open the user configuration form. Users from disabled groups are displayed in red.
Frontend access	Frontend access level is displayed: **System default** - users are authenticated by Zabbix, LDAP or HTTP (depending on the authentication method set globally); **Internal** - users are authenticated by Zabbix; ignored if HTTP authentication is the global default; **LDAP** - users are authenticated by LDAP; ignored if HTTP authentication is the global default; **Disabled** - access to Zabbix frontend is forbidden for this group. By clicking on the current level, you can change it.
Debug mode	Debug mode status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.
Status	User group status is displayed - *Enabled* or *Disabled*. By clicking on the status you can change it.

To configure a new user group, click on the *Create user group* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

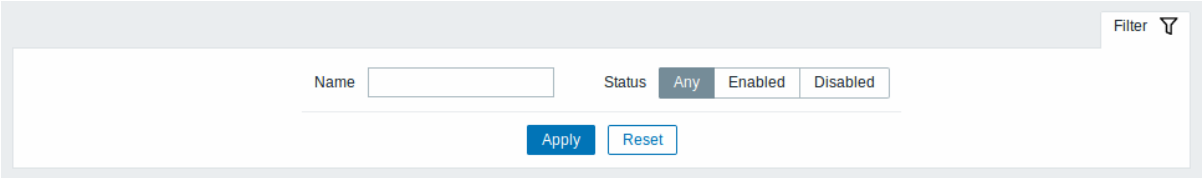
- *Enable* - change the user group status to *Enabled*
- *Disable* - change the user group status to *Disabled*
- *Enable debug mode* - enable debug mode for the user groups
- *Disable debug mode* - disable debug mode for the user groups
- *Delete* - delete the user groups

To use these options, mark the checkboxes before the respective user groups, then click on the required button.

Using filter

You can use the filter to display only the user groups you are interested in. For better search performance, data is searched with macros unresolved.

The *Filter* link is available above the list of user groups. If you click on it, a filter becomes available where you can filter user groups by name and status.



2 User roles

Overview

In the *Users* → *User roles* section you may create user roles.

User roles allow to create fine-grained permissions based on the initially selected user type (*User*, *Admin*, *Super admin*).

Upon selecting a user type, all available permissions for this user type are granted (checked by default).

Permissions can only be revoked from the subset that is available for the user type; they cannot be extended beyond what is available for the user type.

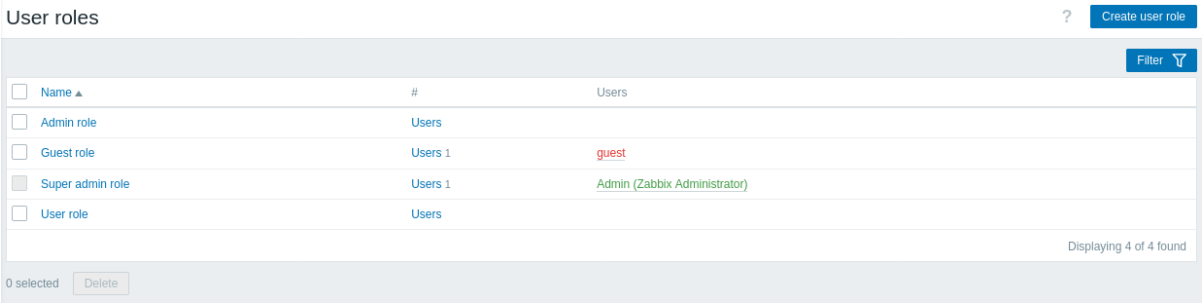
Checkboxes for unavailable permissions are grayed out; users will not be able to access the element even by entering a direct URL to this element into the browser.

User roles can be assigned to system users. Each user may have only one role assigned.

Default user roles

By default, Zabbix is configured with four user roles, which have a pre-defined set of permissions:

- Guest role
- User role
- Admin role
- Super admin role

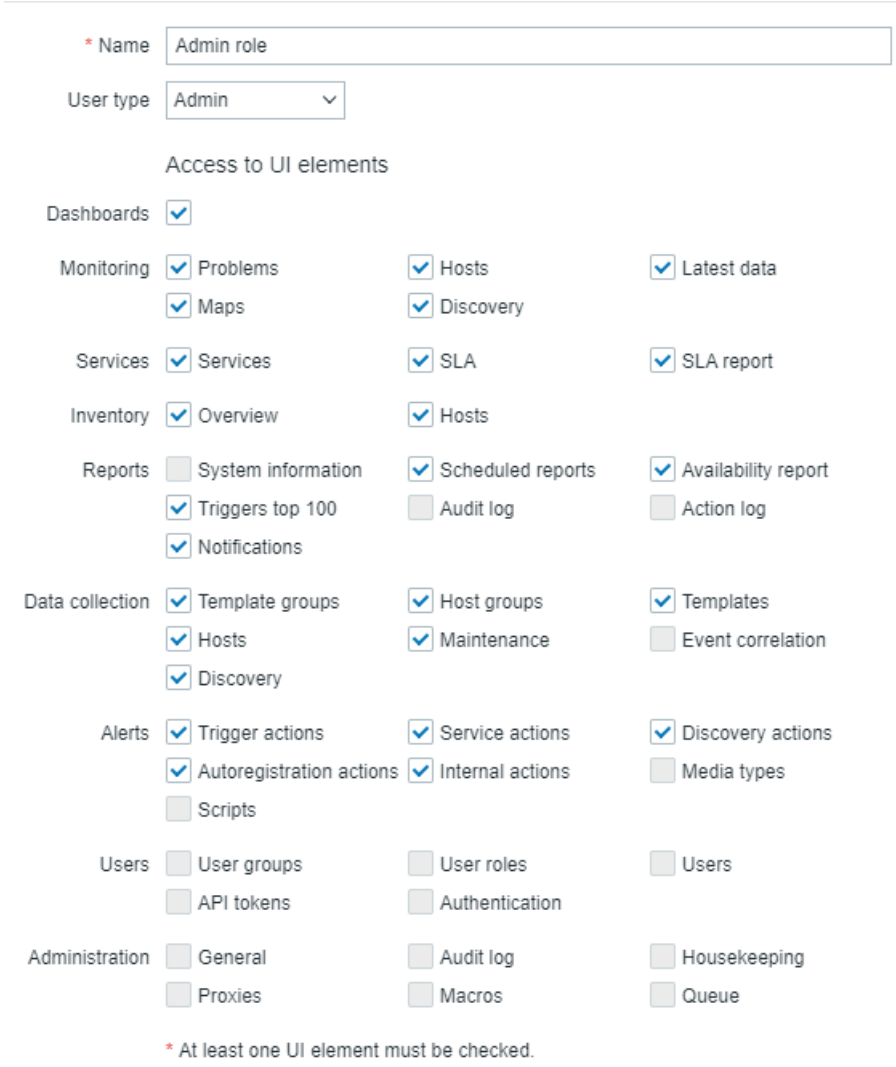


These are based on the main user types in Zabbix. The list of all users assigned the respective role is displayed. The users included in disabled groups are stated in red. The *Guest role* is a user-type role with the only permissions to view some frontend sections.

Note:
The default *Super admin role* cannot be modified or deleted, because at least one Super admin user with unlimited privileges must exist in Zabbix. Users of type *Super admin* can modify settings of their own role, but not the user type.

Configuration

To create a new role, click on the *Create user role* button at the top right corner. To update an existing role, click on the role name to open the configuration form.



Available permissions are displayed. To revoke a certain permission, unmark its checkbox.

Available permissions along with the defaults for each pre-configured user role in Zabbix are described below.

Default permissions

Access to UI elements

The default access to menu sections depends on the user type. See the Permissions page for details.

Access to other options

Parameter	Description	Default user roles			
		Super admin role	*Admin role*	*User role*	*Guest role*
Default access to new UI elements	Enable/disable access to the custom UI elements. Modules, if present, will be listed below.	Yes	Yes	Yes	Yes
Access to services					

Read-write access to services	Select read-write access to services: **None** - no access at all **All** - access to all services is read-write **Service list** - select services for read-write access The read-write access, if granted, takes precedence over the read-only access settings and is dynamically inherited by the child services.	All	All	None	None
Read-write access to services with tag	Specify tag name and, optionally, value to additionally grant read-write access to services matching the tag. This option is available if 'Service list' is selected in the *Read-write access to services* parameter. The read-write access, if granted, takes precedence over the read-only access settings and is dynamically inherited by the child services.				
Read-only access to services	Select read-only access to services: **None** - no access at all **All** - access to all services is read-only **Service list** - select services for read-only access The read-only access does not take precedence over the read-write access and is dynamically inherited by the child services.			All	All
Read-only access to services with tag	Specify tag name and, optionally, value to additionally grant read-only access to services matching the tag. This option is available if 'Service list' is selected in the *Read-only access to services* parameter. The read-only access does not take precedence over the read-write access and is dynamically inherited by the child services.				
Access to modules					
<Module name>	Allow/deny access to a specific module. Only enabled modules are shown in this section. It is not possible to grant or restrict access to a module that is currently disabled.	Yes	Yes	Yes	Yes
Default access to new modules	Enable/disable access to modules that may be added in the future.				
Access to API					
Enabled	Enable/disable access to API.	Yes	Yes	Yes	No
API methods	Select *Allow list* to allow only specified API methods or *Deny list* to restrict only specified API methods. In the search field, start typing the method name, then select the method from the auto-complete list. You can also press the Select button and select methods from the full list available for this user type. Note that if certain action from the Access to actions block is unchecked, users will not be able to use API methods related to this action. Wildcards are supported. Examples: `dashboard.*` (all methods of 'dashboard.' API service) `*` (any method), `*.export` (methods with '.export' name from all API services). If no methods have been specified the Allow/Deny list rule will be ignored.				

Access to actions					
Create and edit dashboards	Clearing this checkbox will also revoke the rights to use `.create`, `.update` and `.delete` API methods for the corresponding elements.	Yes	Yes	Yes	No
Create and edit maps					
Create and edit maintenance				No	
Add problem comments	Clearing this checkbox will also revoke the rights to perform corresponding action via `event.acknowledge` API method.			Yes	
Change severity					
Acknowledge problems					
Suppress problems					
Close problems					
Execute scripts	Clearing this checkbox will also revoke the rights to use the `script.execute` API method.				
Manage API tokens	Clearing this checkbox will also revoke the rights to use all `token.` API methods.				
Manage scheduled reports	Clearing this checkbox will also revoke the rights to use all `report.` API methods.			No	
Manage SLA	Enable/disable the rights to manage SLA.				
Invoke "Execute now" on read-only hosts	Allow to use the "Execute now" option in latest data for items of read-only hosts.			Yes	
Change problem ranking	Allow to change the problem ranking from cause to symptom, and vice versa.				
Default access to new actions	Enable/disable access to new actions.				

See also:

- Configuring a user

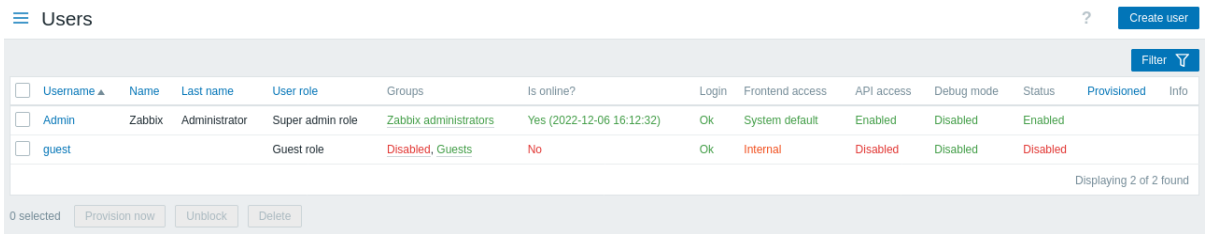
3 Users

Overview

In the *Users* → *Users* section users of the system are maintained.

Users

A listing of existing users with their details is displayed.



Displayed data:

Column	Description
Username	Username for logging into Zabbix. Clicking on the username opens the user configuration form.
Name	First name of the user.
Last name	Second name of the user.

Column	Description
User role	User role is displayed.
Groups	Groups that the user is a member of are listed. Clicking on the user group name opens the user group configuration form. Disabled groups are displayed in red.
Is online?	The on-line status of the user is displayed - *Yes* or *No*. The time of last user activity is displayed in parentheses.
Login	The login status of the user is displayed - *Ok* or *Blocked*. A user can become temporarily blocked upon exceeding the number of unsuccessful login attempts set in the *Administration → General → Other* section (five by default). By clicking on *Blocked* you can unblock the user.
Frontend access	Frontend access level is displayed - *System default*, *Internal*, *LDAP*, or *Disabled*, depending on the one set for the whole user group.
API access	API access status is displayed - *Enabled* or *Disabled*, depending on the one set for the user role.
Debug mode	Debug mode status is displayed - *Enabled* or *Disabled*, depending on the one set for the whole user group.
Status	User status is displayed - *Enabled* or *Disabled*, depending on the one set for the whole user group.
Provisioned	The date when the user was last provisioned is displayed. Used for users created by JIT provisioning from LDAP/SAML.
Info	Information about errors is displayed. A yellow warning is displayed for users without user groups. A red warning is displayed for users without roles, and for users without roles and user groups.

To configure a new user, click on the *Create user* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

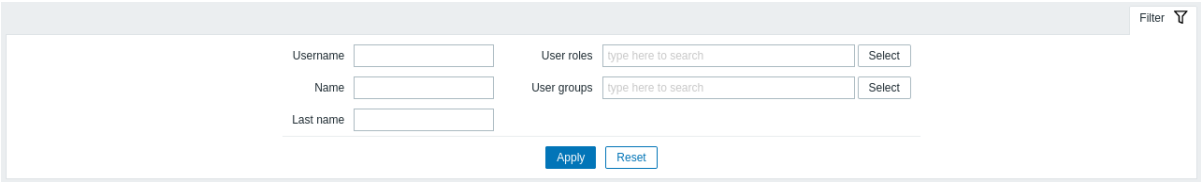
- *Provision now* - update user information from LDAP (this option is only enabled if an LDAP user is selected)
- *Unblock* - re-enable system access to blocked users
- *Delete* - delete the users

To use these options, mark the check-boxes before the respective users, then click on the required button.

Using filter

You can use the filter to display only the users you are interested in. For better search performance, data is searched with macros unresolved.

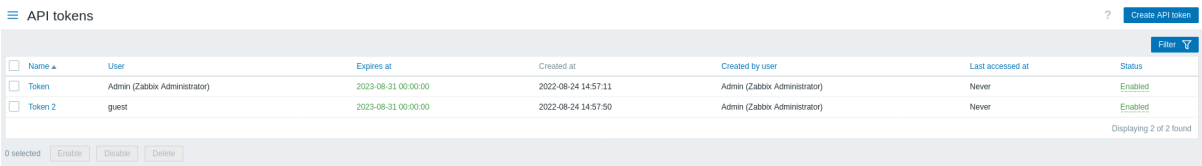
The *Filter* link is available above the list of users. If you click on it, a filter becomes available where you can filter users by username, name, last name, user role and user group.



4 API tokens

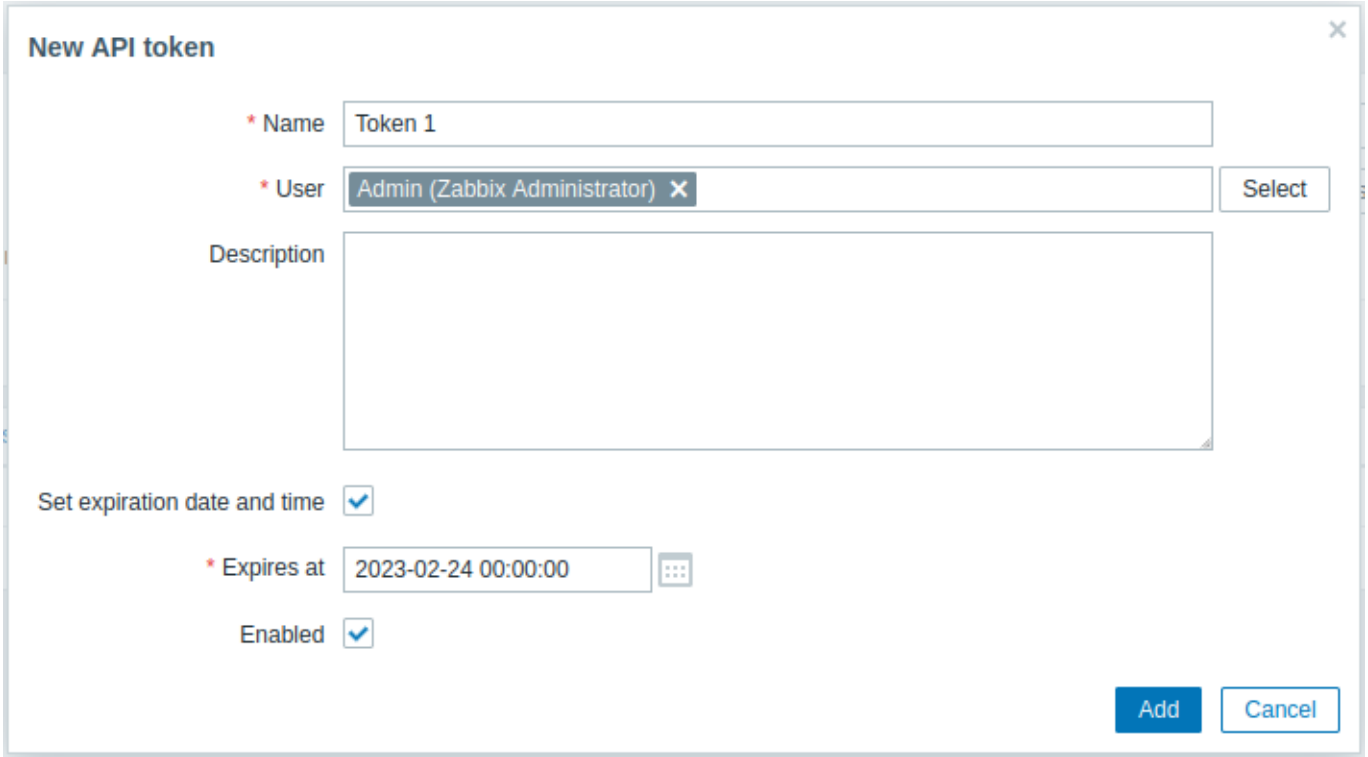
Overview

This section allows to create and manage API tokens.



You may filter API tokens by name, users to whom the tokens are assigned, expiry date, users that created tokens, or status (enabled/disabled). Click on the token status in the list to quickly enable/disable a token. You may also mass enable/disable tokens by selecting them in the list and then clicking on the *Enable/Disable* buttons below the list.

To create a new token, press *Create API token* button at the top right corner, then fill out the required fields in the token configuration screen:



Parameter	Description
Name	Token's visible name.
User	User the token should be assigned to. To quickly select a user, start typing the username, first or last name, then select the required user from the auto-complete list. Alternatively, you can press the Select button and select a user from the full user list. A token can be assigned only to one user.
Description	Optional token description.
Set expiration date and time	Unmark this checkbox if a token should not have an expiry date.
Expiry date	Click on the calendar icon to select token expiry date or enter the date manually in a format YYYY-MM-DD hh:mm:ss
Enabled	Unmark this checkbox if you need to create a token in a disabled state.

Press Add to create a token. On the next screen, copy and save in a safe place *Auth token* value **before closing the page**, then press Close. The token will appear in the list.

Warning:
Auth token value cannot be viewed again later. It is only available immediately after creating a token. If you lose a saved token you will have to regenerate it and doing so will create a new authorization string.

Click on the token name to edit the name, description, expiry date settings, or token status. Note that it is not possible to change to which user the token is assigned. Press *Update* button to save changes. If a token has been lost or exposed, you may press *Regenerate* button to generate new token value. A confirmation dialog box will appear, asking you to confirm this operation since after proceeding the previously generated token will become invalid.

Users without access to the *Administration* menu section can see and modify details of tokens assigned to them in the *User profile* → *API tokens* section only if *Manage API tokens* is allowed in their user role permissions.

5 Authentication

Overview

The *Users* → *Authentication* section allows to specify the user authentication method for Zabbix and internal password requirements.

The available authentication methods are internal, HTTP, LDAP, and SAML authentication.

Default authentication

By default, Zabbix uses **internal** Zabbix authentication for all users.

It is possible to change the default authentication method to LDAP system-wide. To do so, navigate to the *LDAP* tab and configure LDAP parameters, then return to the *Authentication* tab and switch the *Default authentication* selector to LDAP.

Note that the authentication method can be fine-tuned on the user group level. Even if LDAP authentication is set globally, some user groups can still be authenticated by Zabbix. These groups must have frontend access set to Internal.

It is also possible to enable LDAP authentication only for specific user groups, if internal authentication is used globally. In this case LDAP authentication details can be specified and used for specific user groups whose frontend access must then be set to LDAP. If a user is included into at least one user group with LDAP authentication, this user will not be able to use the internal authentication method.

HTTP and SAML 2.0 authentication methods can be used in addition to the default authentication method.

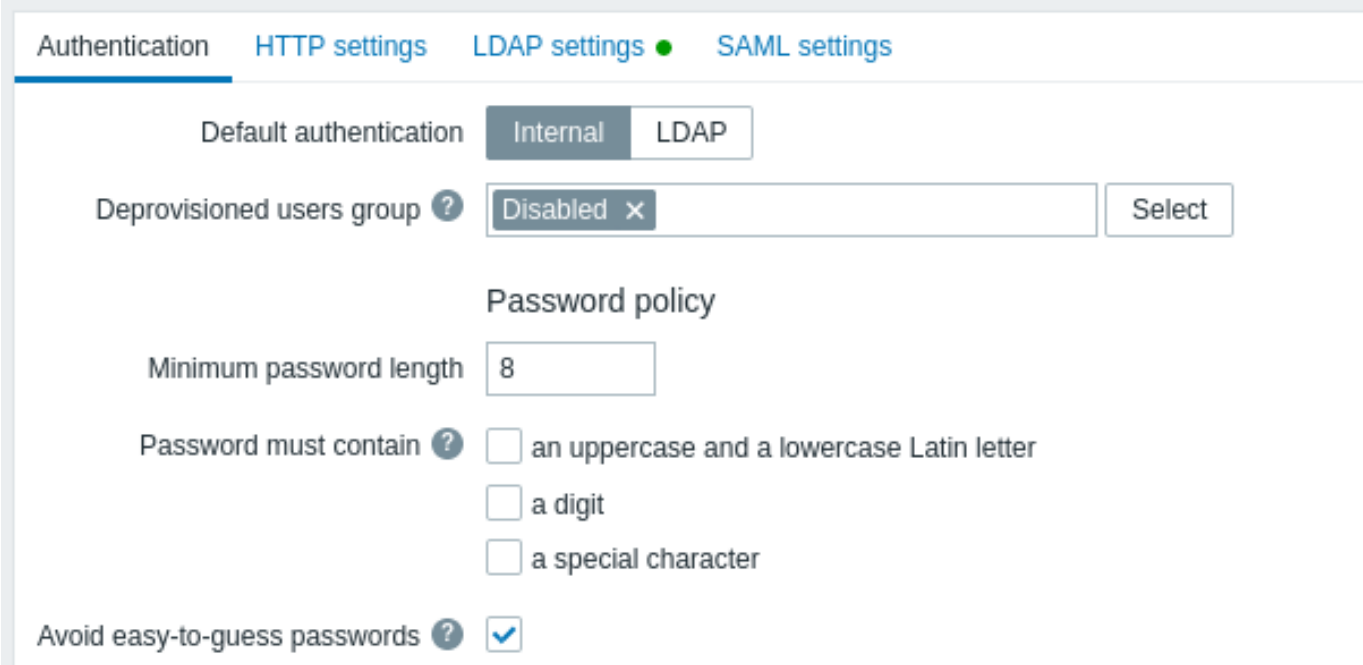
Zabbix supports just-in-time (JIT) provisioning that allows to create user accounts in Zabbix the first time an external user authenticates and provision these user accounts. JIT provisioning is supported for LDAP and SAML.

See also:

- HTTP authentication
- LDAP authentication
- SAML authentication

Configuration

The *Authentication* tab allows to set the default authentication method, specify a group for deprovisioned users and set password complexity requirements for Zabbix users.



Configuration parameters:

Parameter	Description
Default authentication	Select the default authentication method for Zabbix - *Internal* or *LDAP*.
Deprovisioned users group	Specify a user group for deprovisioned users. This setting is required only for JIT provisioning, regarding users that were created in Zabbix from LDAP or SAML systems, but no longer need to be provisioned. A disabled user group must be specified.
Minimum password length	By default, the minimum password length is set to 8. Supported range: 1-70. Note that passwords longer than 72 characters will be truncated.
Password must contain	Mark one or several checkboxes to require usage of specified characters in a password: - an uppercase and a lowercase Latin letter - a digit - a special character Hover over the question mark to see a hint with the list of characters for each option.

Parameter	Description
Avoid easy-to-guess passwords	If marked, a password will be checked against the following requirements: - must not contain user's name, surname, or username - must not be one of the common or context-specific passwords. The list of common and context-specific passwords is generated automatically from the list of NCSC "Top 100k passwords", the list of SecLists "Top 1M passwords" and the list of Zabbix context-specific passwords. Internal users will not be allowed to set passwords included in this list as such passwords are considered weak due to their common use.

Changes in password complexity requirements will not affect existing user passwords, but if an existing user chooses to change a password, the new password will have to meet current requirements. A hint with the list of requirements will be displayed next to the *Password* field in the user profile and in the user configuration form accessible from the *Users* → *Users* menu.

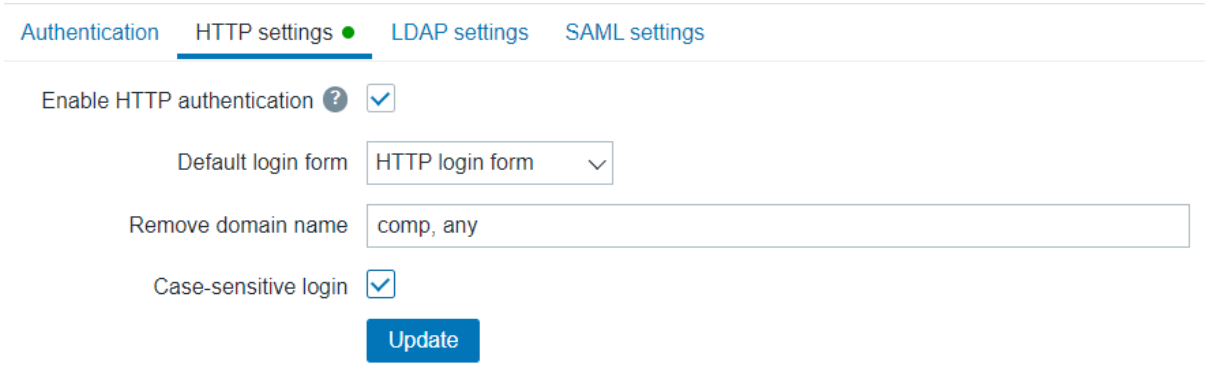
1 HTTP

Overview

HTTP or web server-based authentication (for example: BasicAuthentication, NTLM/Kerberos) can be used to check user names and passwords. Note that a user must exist in Zabbix as well, however its Zabbix password will not be used.

Attention:
Be careful! Make sure that web server authentication is configured and works properly before switching it on.

Configuration



Configuration parameters:

Parameter	Description
Enable HTTP authentication	Mark the checkbox to enable HTTP authentication. Hovering the mouse over will bring up a hint box warning that in the case of web server authentication, all users (even with frontend access set to LDAP/Internal) will be authenticated by the web server, not by Zabbix.
Default login form	Specify whether to direct non-authenticated users to: **Zabbix login form** - standard Zabbix login page. **HTTP login form** - HTTP login page. It is recommended to enable web-server based authentication for the `index_http.php` page only. If *Default login form* is set to 'HTTP login page' the user will be logged in automatically if web server authentication module will set valid user login in the `$_SERVER` variable. Supported `$_SERVER` keys are `PHP_AUTH_USER`, `REMOTE_USER`, `AUTH_USER`.
Remove domain name	A comma-delimited list of domain names that should be removed from the username. E.g. `comp,any` - if username is 'Admin@any', 'comp\Admin', user will be logged in as 'Admin'; if username is 'notacompany\Admin', login will be denied.

Parameter	Description
Case sensitive login	Unmark the checkbox to disable case-sensitive login (enabled by default) for usernames. E.g. disable case-sensitive login and log in with, for example, 'ADMIN' user even if the Zabbix user is 'Admin'. *Note* that with case-sensitive login disabled the login will be denied if multiple users exist in Zabbix database with similar usernames (e.g. Admin, admin).

Note:
For internal users who are unable to log in using HTTP credentials (with HTTP login form set as default) leading to the 401 error, you may want to add a `ErrorDocument 401 /index.php?form=default` line to basic authentication directives, which will redirect to the regular Zabbix login form.

2 LDAP

Overview

External LDAP authentication can be used to check user names and passwords.

Zabbix LDAP authentication works at least with Microsoft Active Directory and OpenLDAP.

If only LDAP sign-in is configured, then the user must also exist in Zabbix, however, its Zabbix password will not be used. If authentication is successful, then Zabbix will match a local username with the username attribute returned by LDAP.

User provisioning

It is possible to configure JIT (just-in-time) **user provisioning** for LDAP users. In this case, it is not required that a user already exists in Zabbix. The user account can be created when the user logs into Zabbix for the first time.

When an LDAP user enters their LDAP login and password, Zabbix checks the *default* LDAP server if this user exists. If the user exists and does not have an account in Zabbix yet, a new user is created in Zabbix and the user is able to log in.

Attention:
If JIT provisioning is enabled, a user group for deprovisioned users must be specified in the *Authentication* tab.

JIT provisioning also allows to update provisioned user accounts based on changes in LDAP. For example, if a user is moved from one LDAP group to another, the user will also be moved from one group to another in Zabbix; if a user is removed from an LDAP group, the user will also be removed from the group in Zabbix and, if not belonging to any other group, added to the user group for deprovisioned users. Note that provisioned user accounts are updated based on the configured provisioning period or when the user logs into Zabbix.

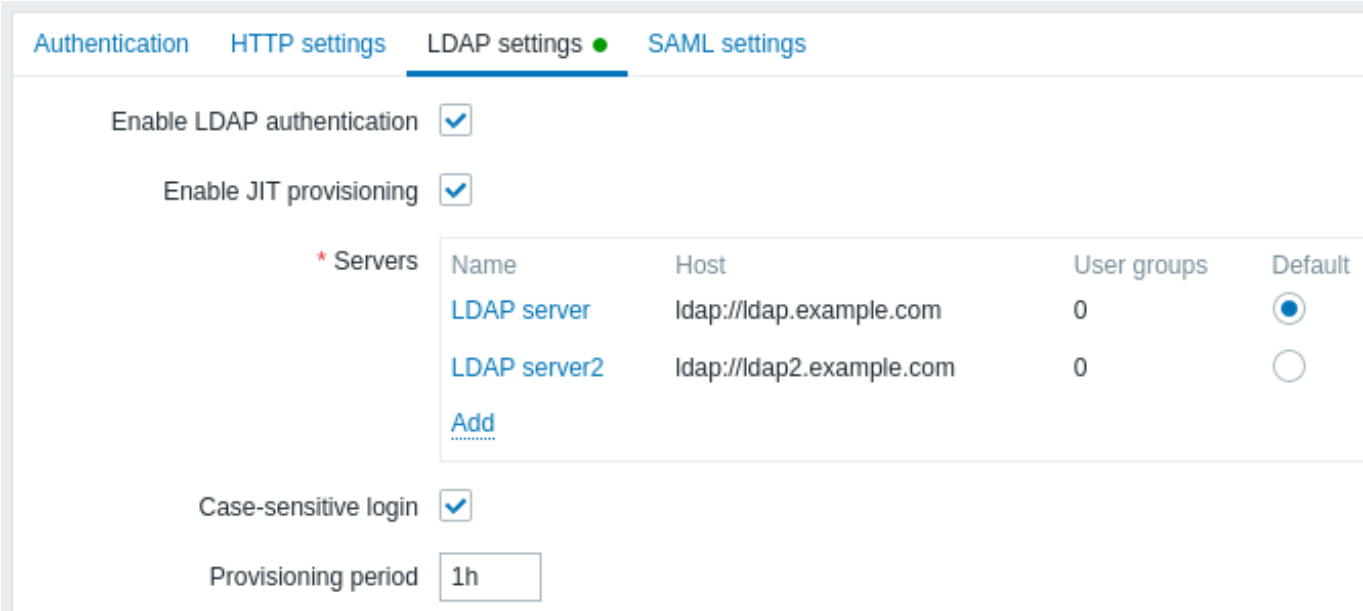
LDAP JIT provisioning is available only when LDAP is configured to use "anonymous" or "special user" for binding. For direct user binding, provisioning will be made only for user login action, because logging in user password is used for such type of binding.

Multiple servers

Several LDAP servers can be defined, if necessary. For example, a different server can be used to authenticate a different user group. Once LDAP servers are configured, in user group configuration it becomes possible to select the required LDAP server for the respective user group.

If a user is in multiple user groups and multiple LDAP servers, the first server in the list of LDAP servers sorted by name in ascending order will be used for authentication.

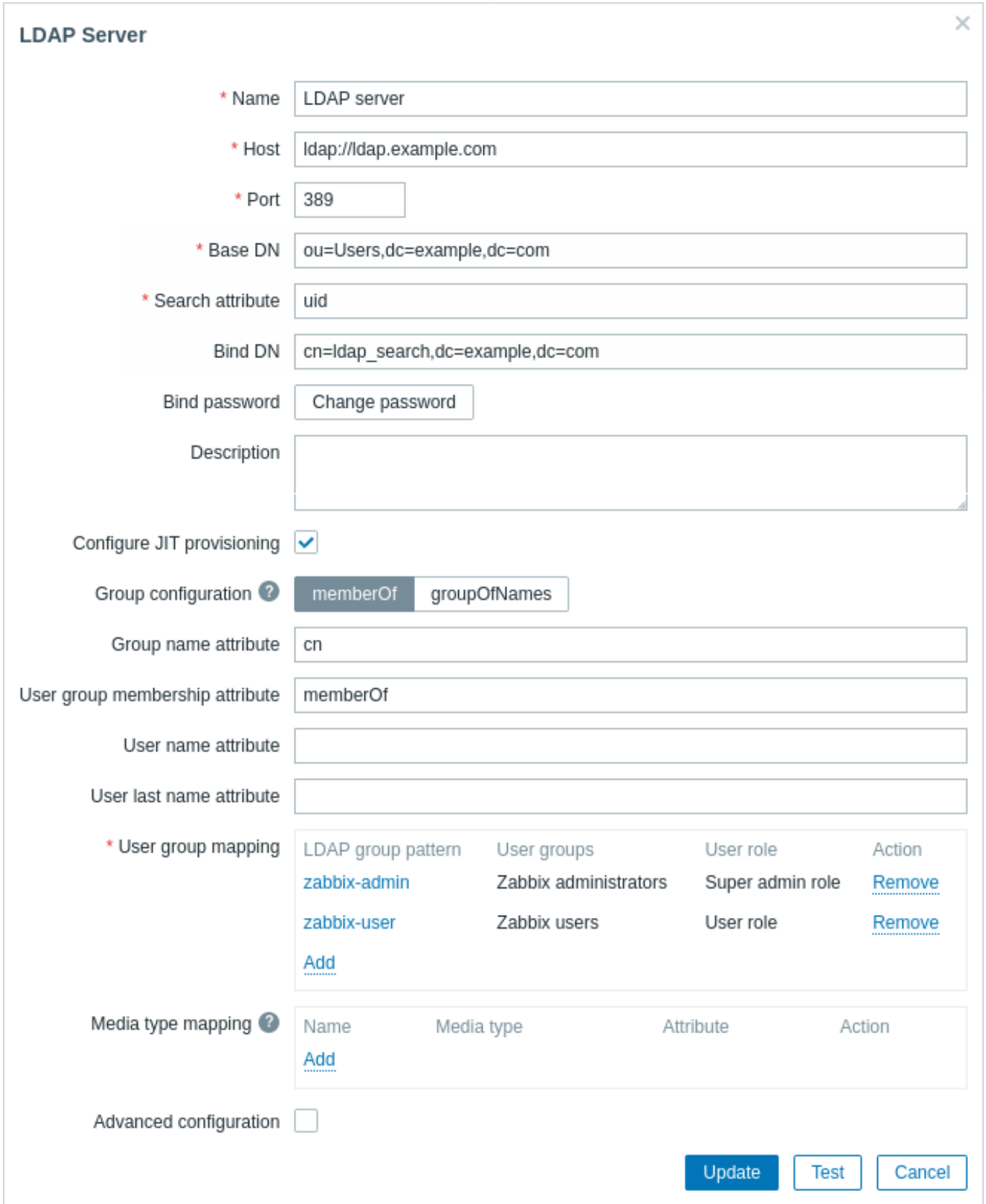
Configuration



Configuration parameters:

Parameter	Description
Enable LDAP authentication	Mark the checkbox to enable LDAP authentication.
Enable JIT provisioning	Mark the checkbox to enable JIT provisioning.
Servers	Click on *Add* to configure an LDAP server (see LDAP server configuration below).
Case-sensitive login	Unmark the checkbox to disable case-sensitive login (enabled by default) for usernames. E.g. disable case-sensitive login and log in with, for example, 'ADMIN' user even if the Zabbix user is 'Admin'. *Note* that with case-sensitive login disabled the login will be denied if multiple users exist in Zabbix database with similar usernames (e.g. Admin, admin).
Provisioning period	Set the provisioning period, i.e. how often user provisioning is performed.

LDAP server configuration



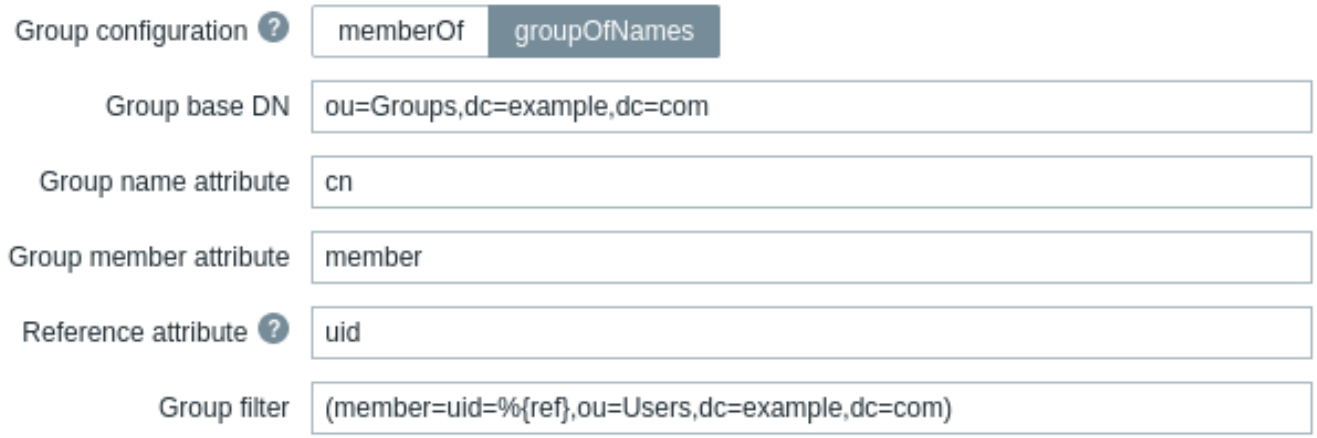
LDAP server configuration parameters:

Parameter	Description
Name	Name of the LDAP server in Zabbix configuration.
Host	Hostname, IP or URI of the LDAP server. Examples: ldap.example.com, 127.0.0.1, ldap://ldap.example.com For secure LDAP server, use *ldaps* protocol and hostname. Example: ldaps://ldap.example.com With OpenLDAP 2.x.x and later, a full LDAP URI of the form ldap://hostname:port or ldaps://hostname:port may be used.

Parameter	Description
Port	Port of the LDAP server. Default is 389. For secure LDAP connection port number is normally 636. Not used when using full LDAP URIs.
Base DN	Base path to user accounts in LDAP server: ou=Users,ou=system (for OpenLDAP), DC=company,DC=com (for Microsoft Active Directory) uid=%{user},dc=example,dc=com (for direct user binding, see a note below)
Search attribute	LDAP account attribute used for search: uid (for OpenLDAP), sAMAccountName (for Microsoft Active Directory)
Bind DN	LDAP account for binding and searching over the LDAP server, examples: uid=ldap_search,ou=system (for OpenLDAP), CN=ldap_search,OU=user_group,DC=company,DC=com (for Microsoft Active Directory) Anonymous binding is also supported. Note that anonymous binding potentially opens up domain configuration to unauthorized users (information about users, computers, servers, groups, services, etc.). For security reasons, disable anonymous binds on LDAP hosts and use authenticated access instead.
Bind password	LDAP password of the account for binding and searching over the LDAP server.
Description	Description of the LDAP server.
Configure JIT provisioning	Mark this checkbox to show options related to JIT provisioning.
Group configuration	Select the group configuration method: **memberOf** - by searching users and their group membership attribute **groupOfNames** - by searching groups through the member attribute Note that memberOf is preferable as it is faster; use groupOfNames if your LDAP server does not support `memberOf` or group filtering is required.
Group name attribute	Specify the attribute to get the group name from all objects in the `memberOf` attribute (see the *User group membership attribute* field) The group name is necessary for user group mapping.
User group membership attribute	Specify the attribute that contains information about the groups that the user belongs to (e.g. `memberOf`). For example, the memberOf attribute may hold information like this: `memberOf=cn=zabbix-admin,ou=Groups,dc=example,dc=com` This field is available only for the memberOf method.
User name attribute	Specify the attribute that contains the user's first name.
User last name attribute	Specify the attribute that contains the user's last name.
User group mapping	Map an LDAP user group pattern to Zabbix user group and user role. This is required to determine what user group/role the provisioned user will get in Zabbix. Click on *Add* to add a mapping. The *LDAP group pattern* field supports wildcards. The group name must match an existing group. If an LDAP user matches several Zabbix user groups, the user becomes a member of all of them. If a user matches several Zabbix user roles, the user will get the one with the highest permission level among them.
Media type mapping	Map the user's LDAP media attributes (e.g. email) to Zabbix user media for sending notifications.
Advanced configuration	Mark this checkbox to show advanced configuration options (see below).
StartTLS	Mark the checkbox to use the StartTLS operation when connecting to LDAP server. The connection will fall if the server doesn't support StartTLS. StartTLS cannot be used with servers that use the *ldaps* protocol.
Search filter	Define a custom string when authenticating a user in LDAP. The following placeholders are supported: `%{attr}` - search attribute name (uid, sAMAccountName) `%{user}` - user username value to authenticate For example, to carry out a case-sensitive search within the case-insensitive LDAP or Microsoft Active Directory environment, the string can be defined as follows: `(%{attr}:caseExactMatch:=%{user})`. If the filter is not customized, LDAP will use the default: `(%{attr}=%{user})`.

Note:
To configure an LDAP server for **direct user binding**, append an attribute uid=%{user} to the *Base DN* parameter (for example,*uid=%{user},dc=example,dc=com*) and leave *BindDN* and *Bind password* parameters empty. When authenticating, a placeholder %{user} will be replaced by the username entered during login.

The following fields are specific to "groupOfNames" as the *Group configuration* method:



Parameter	Description
Group base DN	Base path to the groups in LDAP server.
Group name attribute	Specify the attribute to get the group name in the specified base path to groups. The group name is necessary for user group mapping.
Group member attribute	Specify the attribute that contains information about the members of the group in LDAP (e.g. `member`).
Reference attribute	Specify the reference attribute for the group filter (see the *Group filter* field). Then use `%{ref}` in the group filter to get values for the attribute specified here.
Group filter	Specify the filter to retrieve the group that the user is member of. For example, `(member=uid=%{ref},ou=Users,dc=example,dc=com)` will match "User1" if the member attribute of the group is `uid=User1,ou=Users,dc=example,dc=com` and will return the group that "User1" is a member of.

Warning:
In case of trouble with certificates, to make a secure LDAP connection (ldaps) work you may need to add a `TLS_REQCERT allow` line to the /etc/openldap/ldap.conf configuration file. It may decrease the security of connection to the LDAP catalog.

Note:
It is recommended to create a separate LDAP account (*Bind DN*) to perform binding and searching over the LDAP server with minimal privileges in the LDAP instead of using real user accounts (used for logging in the Zabbix frontend).
Such an approach provides more security and does not require changing the *Bind password* when the user changes his own password in the LDAP server.
In the table above it's the *ldap_search* account name.

Testing access

The *Test* button allows to test user access:

Parameter	Description
Login	LDAP user name to test (prefilled with the current user name from Zabbix frontend). This user name must exist in the LDAP server. Zabbix will not activate LDAP authentication if it is unable to authenticate the test user.
User password	LDAP user password to test.

3 SAML

Overview

SAML 2.0 authentication can be used to sign in to Zabbix.

If only SAML sign-in is configured, then the user must also exist in Zabbix, however, its Zabbix password will not be used. If authentication is successful, then Zabbix will match a local username with the username attribute returned by SAML.

User provisioning

It is possible to configure JIT (just-in-time) **user provisioning** for SAML users. In this case, it is not required that a user already exists in Zabbix. The user account can be created when the user logs into Zabbix for the first time.

> **Attention:**
> If JIT provisioning is enabled, a user group for deprovisioned users must be specified in the *Authentication* tab.

On top of JIT provisioning it is also possible to enable and configure SCIM (System for Cross-domain Identity Management) provisioning - *continuous* user account management for those users that have been created by user provisioning. SCIM provisioning requires a Zabbix API token (with Super admin permissions) for authentication into Zabbix.

For example, if a user is moved from one SAML group to another, the user will also be moved from one group to another in Zabbix; if a user is removed from a SAML group, the user will also be removed from the group in Zabbix and, if not belonging to any other group, added to the user group for deprovisioned users.

If SCIM is enabled and configured, a SAML user will be provisioned at the moment the user logs into Zabbix and continuously updated based on changes in SAML. Already existing SAML users will not be provisioned, and only provisioned users will be updated. Note that only valid media will be added to a user when the user is provisioned or updated.

If SCIM is not enabled, a SAML user will be provisioned (and later updated) at the moment the user logs into Zabbix.

> **Note:**
> If SAML authentication is enabled, users will be able to choose between logging in locally or via SAML single sign-on. If JIT provisioning is used, then only single sign-on is possible.

Setting up identity provider

In order to work with Zabbix, a SAML identity provider - (onelogin.com, auth0.com, okta.com, etc.) needs to be configured in the following way:

- *Assertion Consumer URL* should be set to `<path_to_zabbix_ui>/index_sso.php?acs`
- *Single Logout URL* should be set to `<path_to_zabbix_ui>/index_sso.php?sls`

`<path_to_zabbix_ui>` examples: `https://example.com/zabbix/ui`, `http://another.example.com/zabbix`, `http://<any_public_ip_address>/zabbix`

Setting up Zabbix

> **Attention:**
> It is required to install php-openssl if you want to use SAML authentication in the frontend.

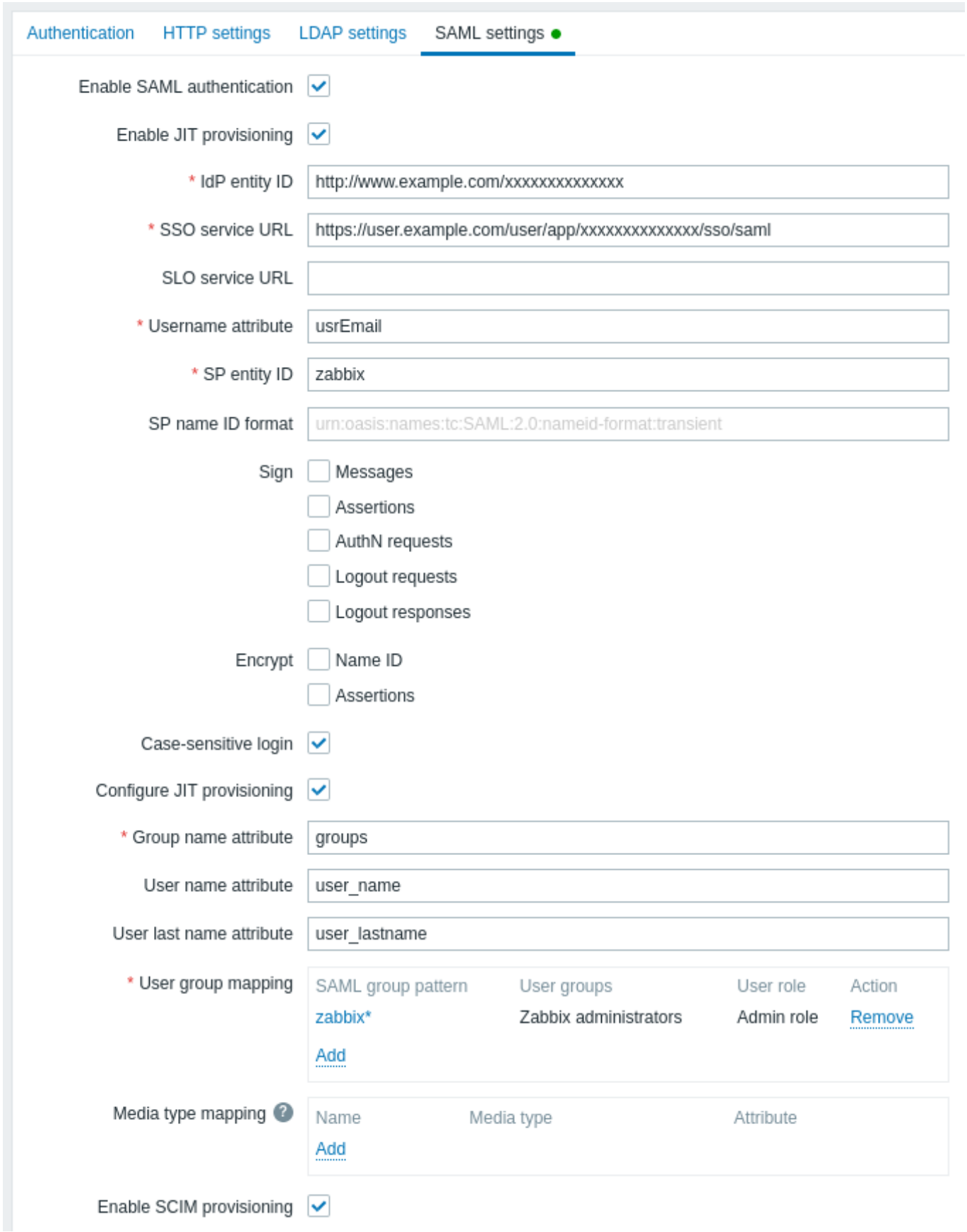
To use SAML authentication Zabbix should be configured in the following way:

1. Private key and certificate should be stored in the *ui/conf/certs/*, unless custom paths are provided in zabbix.conf.php.

By default, Zabbix will look in the following locations:

- ui/conf/certs/sp.key - SP private key file
- ui/conf/certs/sp.crt - SP cert file
- ui/conf/certs/idp.crt - IDP cert file

2. All of the most important settings can be configured in the Zabbix frontend. However, it is possible to specify additional settings in the configuration file.



Configuration parameters, available in the Zabbix frontend:

Parameter	Description
Enable SAML authentication	Mark the checkbox to enable SAML authentication.
Enable JIT provisioning	Mark the checkbox to enable JIT user provisioning.
IDP entity ID	The unique entity identifier within the SAML identity provider.

Parameter	Description
SSO service URL	The URL users will be redirected to when logging in.
SLO service URL	The URL users will be redirected to when logging out. If left empty, the SLO service will not be used.
Username attribute	SAML attribute to be used as a username when logging into Zabbix. The list of supported values is determined by the identity provider. Examples: uid userprincipalname samaccountname username userusername urn:oid:0.9.2342.19200300.100.1.1 urn:oid:1.3.6.1.4.1.5923.1.1.1.13 urn:oid:0.9.2342.19200300.100.1.44
SP entity ID	The unique service provider identifier (if not matching, the operation will be rejected). It is possible to specify a URL or any string of data.
SP name ID format	Defines which name identifier format should be used. Examples: urn:oasis:names:tc:SAML:2.0:nameid-format:persistent urn:oasis:names:tc:SAML:2.0:nameid-format:transient urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos urn:oasis:names:tc:SAML:2.0:nameid-format:entity
Sign	Mark the checkboxes to select entities for which SAML signature should be enabled: *Messages* *Assertions* *AuthN requests* *Logout requests* *Logout responses*
Encrypt	Mark the checkboxes to select entities for which SAML encryption should be enabled: *Name ID* *Assertions*
Case-sensitive login	Mark the checkbox to enable case-sensitive login (disabled by default) for usernames. E.g. disable case-sensitive login and log in with, for example, 'ADMIN' user even if the Zabbix user is 'Admin'. *Note* that with case-sensitive login disabled the login will be denied if multiple users exist in Zabbix database with similar usernames (e.g. Admin, admin).
Configure JIT provisioning	Mark this checkbox to show options related to JIT user provisioning.
Group name attribute	Specify the group name attribute for JIT user provisioning.
User name attribute	Specify the user name attribute for JIT user provisioning.
User last name attribute	Specify the user last name attribute for JIT user provisioning.
User group mapping	Map a SAML user group pattern to Zabbix user group and user role. This is required to determine what user group/role the provisioned user will get in Zabbix. Click on *Add* to add a mapping. The *SAML group pattern* field supports wildcards. The group name must match an existing group. If a SAML user matches several Zabbix user groups, the user becomes a member of all of them. If a user matches several Zabbix user roles, the user will get the highest permission level among them.
Media type mapping	Map the user's SAML media attributes (e.g. email) to Zabbix user media for sending notifications.
Enable SCIM provisioning	Mark this checkbox to enable SCIM 2.0 provisioning.

See examples of configuring SAML identity providers for sign-in and user provisioning into Zabbix with:

- Microsoft Azure AD
- Okta
- Onelogin

Notes on SCIM provisioning

For SCIM provisioning specify the path to the Zabbix frontend and append api_scim.php to it, on the identity provider side, i.e.:

`https://<your-zabbix-url>/zabbix/api_scim.php`

User attributes that are used in Zabbix (username, user name, user lastname and media attributes) need to be added as custom attributes and, if necessary, external namespace should be the same as user schema: `urn:ietf:params:scim:schemas:core:2.0:User`.

Advanced settings

Additional SAML parameters can be configured in the Zabbix frontend configuration file (*zabbix.conf.php*):

- $SSO['SP_KEY'] = '<path to the SP private key file>';
- $SSO['SP_CERT'] = '<path to the SP cert file>';
- $SSO['IDP_CERT'] = '<path to the IDP cert file>';
- $SSO['SETTINGS']

Note:
Zabbix uses OneLogin's SAML PHP Toolkit library (version 3.4.1). The structure of $SSO['SETTINGS'] section should be similar to the structure used by the library. For the description of configuration options, see official library documentation.

Only the following options can be set as part of $SSO['SETTINGS']:

- *strict*
- *baseurl*
- *compress*
- *contactPerson*
- *organization*
- *sp* (only options specified in this list)
  - *attributeConsumingService*
  - *x509certNew*
- *idp* (only options specified in this list)
  - *singleLogoutService* (only one option)
    - *responseUrl*
  - *certFingerprint*
  - *certFingerprintAlgorithm*
  - *x509certMulti*
- *security* (only options specified in this list)
  - *signMetadata*
  - *wantNameId*
  - *requestedAuthnContext*
  - *requestedAuthnContextComparison*
  - *wantXMLValidation*
  - *relaxDestinationValidation*
  - *destinationStrictlyMatches*
  - *rejectUnsolicitedResponsesWithInResponseTo*
  - *signatureAlgorithm*
  - *digestAlgorithm*
  - *lowercaseUrlencoding*

All other options will be taken from the database and cannot be overridden. The *debug* option will be ignored.

In addition, if Zabbix UI is behind a proxy or a load balancer, the custom *use_proxy_headers* option can be used:

- *false* (default) - ignore the option;
- *true* - use X-Forwarded-* HTTP headers for building the base URL.

If using a load balancer to connect to Zabbix instance, where the load balancer uses TLS/SSL and Zabbix does not, you must indicate 'baseurl', 'strict' and 'use_proxy_headers' parameters as follows:

```
$SSO['SETTINGS']=['strict' => false, 'baseurl' => "https://zabbix.example.com/zabbix/", 'use_proxy_headers
```

Configuration example:

```
$SSO['SETTINGS'] = [
    'security' => [
        'signatureAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#rsa-sha384'
        'digestAlgorithm' => 'http://www.w3.org/2001/04/xmldsig-more#sha384',
        // ...
    ],
```

```
    // ...
];
```

9 Administration

Overview

The Administration menu is for administrative functions of Zabbix. This menu is available to SuperAdmin user type users only.

1 General

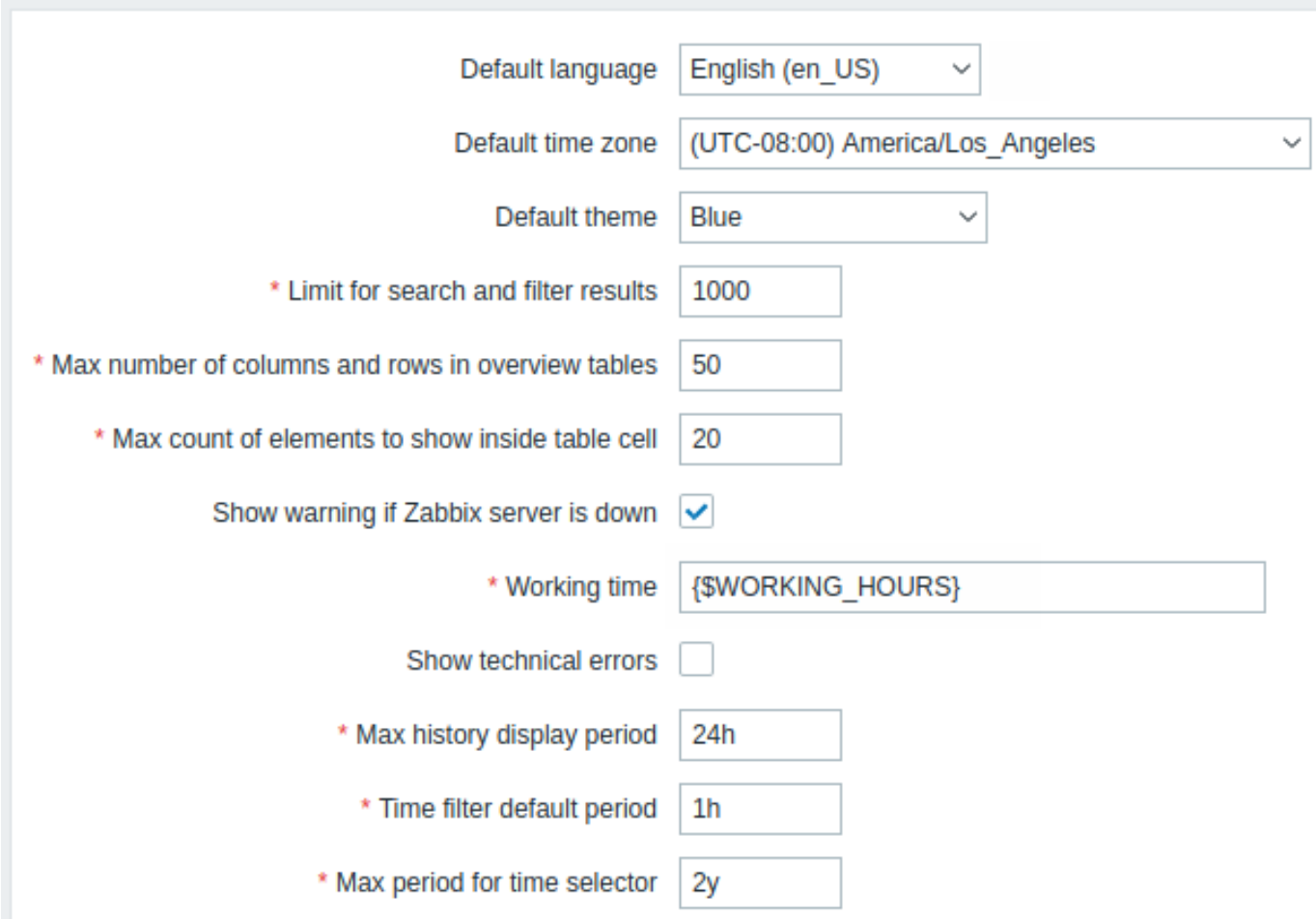
Overview

The *Administration* → *General* section contains a number of subsections for setting frontend-related defaults and customizing Zabbix.

The list of available subsections appears upon pressing on *General* in the *Administration* menu section. It is also possible to switch between subsections by using the title dropdown in the top left corner.

1 GUI

This section provides customization of several frontend-related defaults.



Configuration parameters:

Parameter	Description
Default language	Default language for users who have not specified a language in their profiles and guest users. For more information, see Installation of additional frontend languages.
Default time zone	Default time zone for users who have not specified a time zone in their profiles and guest users.
Default theme	Default theme for users who have not specified a theme in their profiles and guest users.

Parameter	Description
Limit for search and filter results	Maximum amount of elements (rows) that will be displayed in a web-interface list, for example, in *Data collection → Hosts*. *Note*: If set to, for example, '50', only the first 50 elements will be displayed in all affected frontend lists. If some list contains more than fifty elements, the indication of that will be the '+' sign in *"Displaying 1 to 50 of **50+** found"*. Also, if filtering is used and still there are more than 50 matches, only the first 50 will be displayed. Note that increasing the value of this parameter may lead to decreased performance and increased memory consumption on the web server side.
Max number of columns and rows in overview tables	Maximum number of columns and rows to display in Data overview and Trigger overview dashboard widgets. The same limit applies to both columns and rows. If more rows and/or columns than shown exist, the system will display a warning at the bottom of the table: "Not all results are displayed. Please provide more specific search criteria."
Max count of elements to show inside table cell	For entries that are displayed in a single table cell, no more than configured here will be shown.
Show warning if Zabbix server is down	This parameter enables a warning message to be displayed in a browser window if the Zabbix server cannot be reached (possibly down). The message remains visible even if the user scrolls down the page. When hovered over, the message is temporarily hidden to reveal the contents underneath it. This parameter is supported since Zabbix 2.0.1.
Working time	This system-wide parameter defines working hours. In graphs, working time is displayed as a white background and non-working time is displayed as gray. See Time period specification page for description of the time format. User macros are supported (since Zabbix 3.4.0).
Show technical errors	If checked, all registered users will be able to see technical errors (PHP/SQL). If unchecked, the information is only available to Zabbix Super Admins and users belonging to the user groups with enabled debug mode.
Max history display period	Maximum time period for which to display historical data in *Monitoring* subsections: *Latest data*, *Web*, and in the *Data overview* dashboard widget. Allowed range: 24 hours (default) - 1 week. Time suffixes, e.g. 1w (one week), 36h (36 hours), are supported.
Time filter default period	Time period to be used in graphs and dashboards by default. Allowed range: 1 minute - 10 years (default: 1 hour). Time suffixes, e.g. 10m (ten minutes), 5w (five weeks), are supported. Note: when a user changes the time period while viewing a graph, this time period is stored as user preference, replacing the global default or a previous user selection.
Max period for time selector	Maximum available time period for graphs and dashboards. Users will not be able to visualize older data. Allowed range: 1 year - 10 years (default: 2 years). Time suffixes, e.g. 1y (one year), 365w (365 weeks), are supported.

2 Autoregistration

In this section, you can configure the encryption level for active agent autoregistration.



Parameters marked with an asterisk are mandatory.

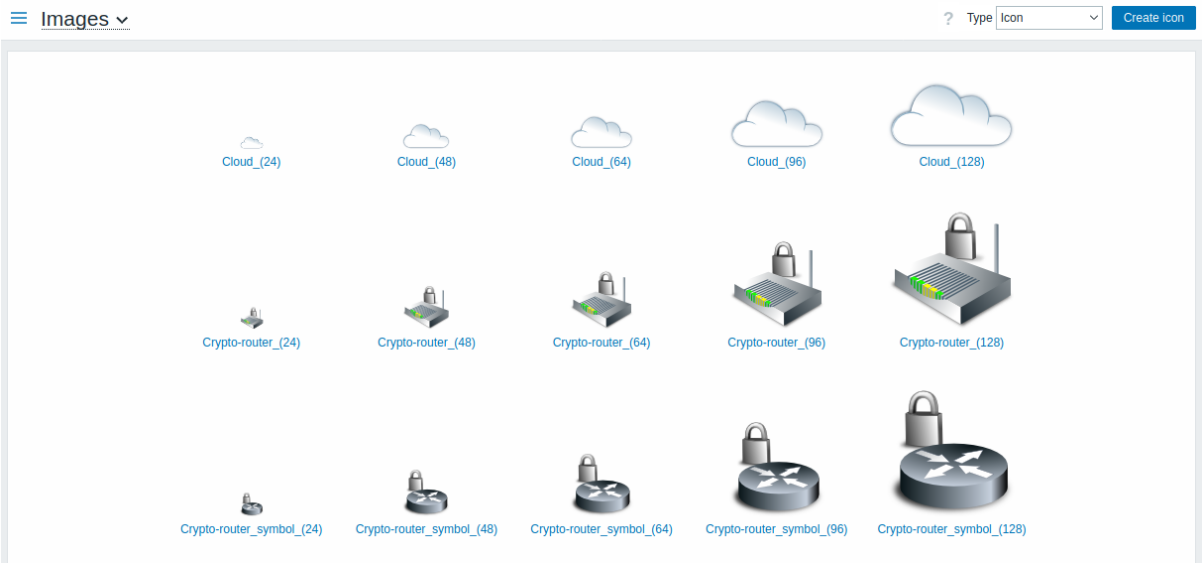
Configuration parameters:

Parameter	Description
Encryption level	Select one or both options for encryption level: **No encryption** - unencrypted connections are allowed **PSK** - TLS encrypted connections with a pre-shared key are allowed
PSK identity	Enter the pre-shared key identity string. This field is only available if 'PSK' is selected as *Encryption level*. Do not put sensitive information in the PSK identity, it is transmitted unencrypted over the network to inform a receiver which PSK to use.
PSK	Enter the pre-shared key (an even number of hexadecimal characters). Maximum length: 512 hex-digits (256-byte PSK) if Zabbix uses GnuTLS or OpenSSL library, 64 hex-digits (32-byte PSK) if Zabbix uses mbed TLS (PolarSSL) library. Example: 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952 This field is only available if 'PSK' is selected as *Encryption level*.

See also: Secure autoregistration

3 Images

The Images section displays all the images available in Zabbix. Images are stored in the database.



The *Type* dropdown allows you to switch between icon and background images:

- Icons are used to display network map elements
- Backgrounds are used as background images of network maps

Adding image

You can add your own image by clicking on the *Create icon* or *Create background* button in the top right corner.

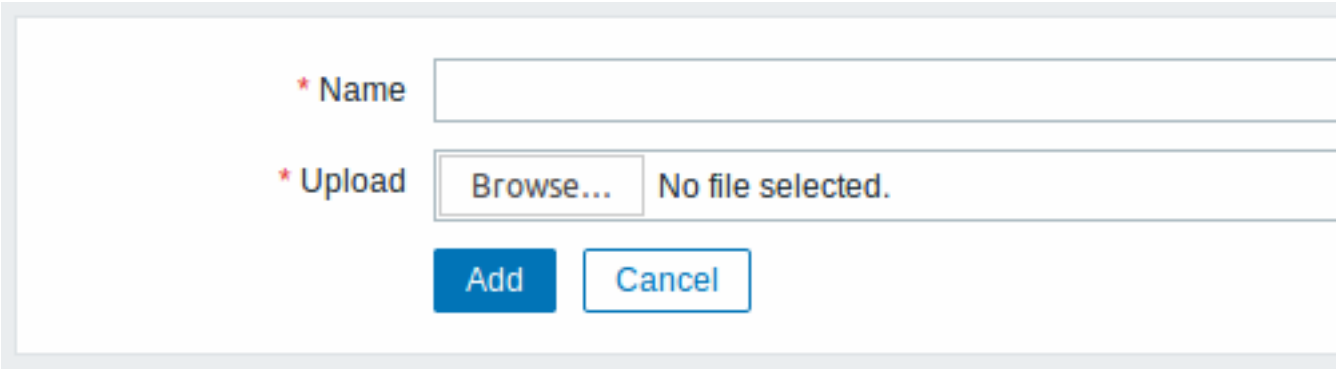


Image attributes:

Parameter	Description
Name	Unique name of an image.

Parameter	Description
Upload	Select the file (PNG, JPEG, GIF) from a local system to be uploaded to Zabbix. *Note* that it may be possible to upload other formats that will be converted to PNG during upload. GD library is used for image processing, therefore formats that are supported depend on the library version used (2.0.28 or higher is required by Zabbix).

Note:
Maximum size of the upload file is limited by the value of ZBX_MAX_IMAGE_SIZE that is 1024x1024 bytes or 1 MB.

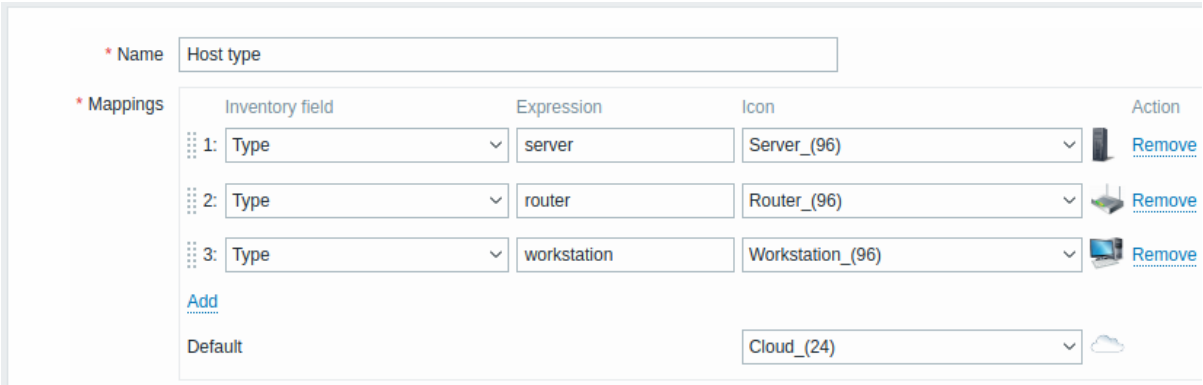
The upload of an image may fail if the image size is close to 1 MB and the `max_allowed_packet` MySQL configuration parameter is at a default of 1MB. In this case, increase the max_allowed_packet parameter.

4 Icon mapping

This section allows creating the mapping of certain hosts with certain icons. Host inventory field information is used to create the mapping.

The mappings can then be used in network map configuration to assign appropriate icons to matching hosts automatically.

To create a new icon map, click on *Create icon map* in the top right corner.



Configuration parameters:

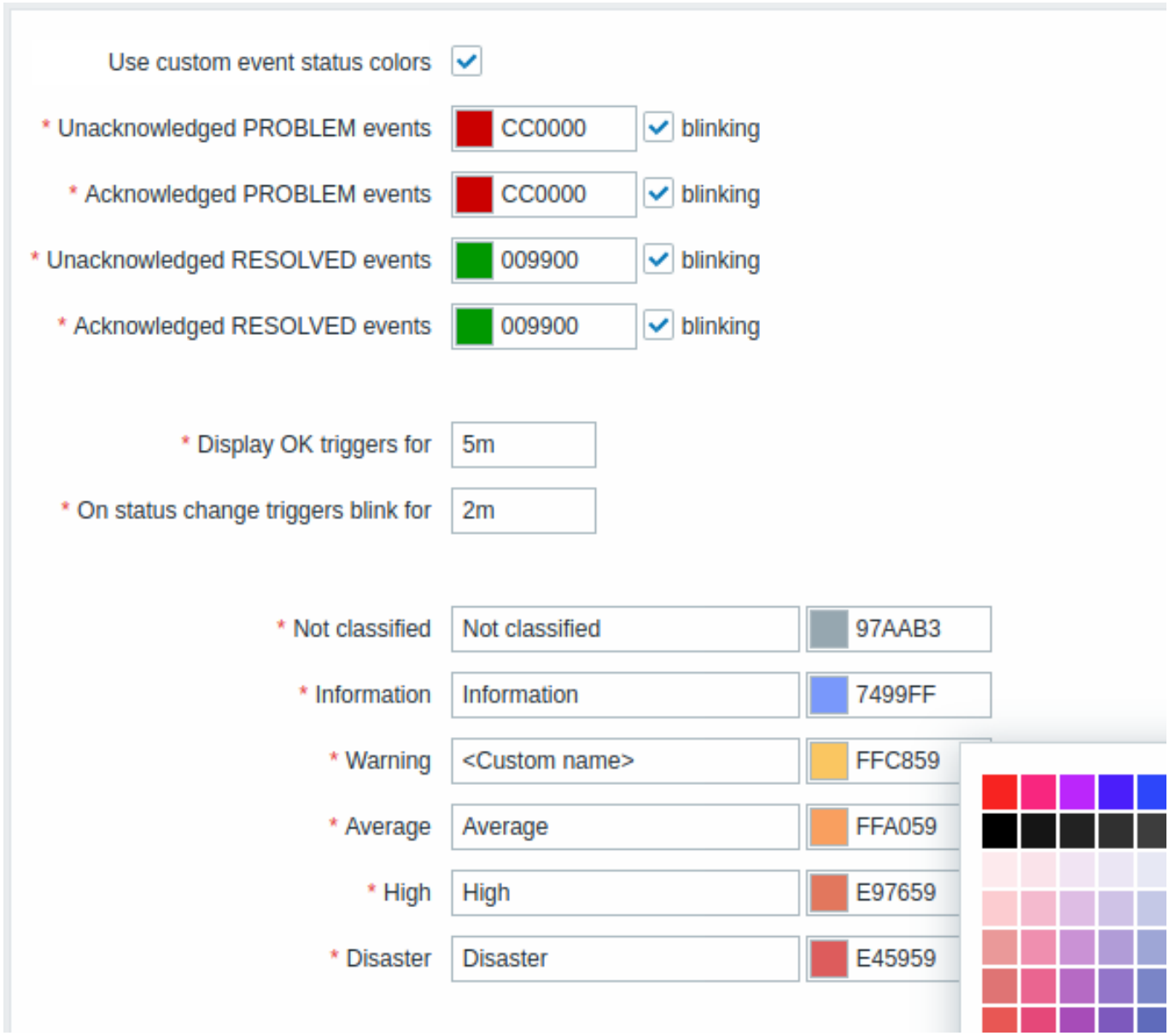
Parameter	Description
Name	Unique name of icon map.
Mappings	A list of mappings. The order of mappings determines which one will have priority. You can move mappings up and down the list with drag-and-drop.
Inventory field	Host inventory field that will be looked into to seek a match.
Expression	Regular expression describing the match.
Icon	Icon to use if a match for the expression is found.
Default	Default icon to use.

5 Regular expressions

This section allows creating custom regular expressions that can be used in several places in the frontend. See Regular expressions section for details.

6 Trigger displaying options

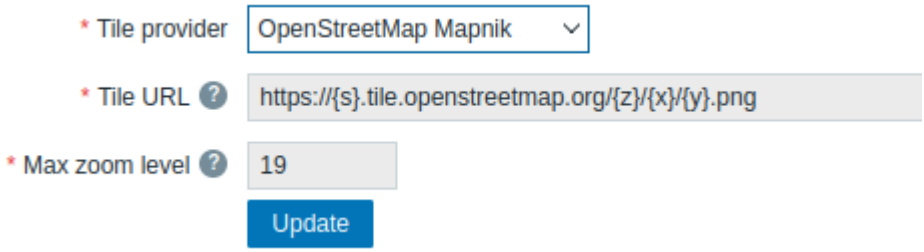
This section allows customizing how trigger status is displayed in the frontend and trigger severity names and colors.



Parameter	Description
Use custom event status colors	Checking this parameter turns on the customization of colors for acknowledged/unacknowledged problems.
Unacknowledged PROBLEM events, *Acknowledged PROBLEM events*, *Unacknowledged RESOLVED events*, *Acknowledged RESOLVED events*	Enter new color code or click on the color to select a new one from the provided palette. If *blinking* checkbox is marked, triggers will blink for some time upon the status change to become more visible.
Display OK triggers for	Time period for displaying OK triggers. Allowed range: 0 - 24 hours. Time suffixes, e.g. 5m, 2h, 1d, are supported.
On status change triggers blink for	Length of trigger blinking. Allowed range: 0 - 24 hours. Time suffixes, e.g. 5m, 2h, 1d, are supported.
Not classified, *Information*, *Warning*, *Average*, *High*, *Disaster*	Custom severity names and/or colors to display instead of system default. Enter new color code or click on the color to select a new one from the provided palette. Note that custom severity names entered here will be used in all locales. If you need to translate them to other languages for certain users, see Customizing trigger severities page.

7 Geographical maps

This section allows selecting geographical map tile service provider and configuring service provider settings for the Geomap dashboard widget. To provide visualization using the geographical maps, Zabbix uses open-source JavaScript interactive maps library Leaflet. Please note that Zabbix has no control over the quality of images provided by third-party tile providers, including the predefined tile providers.



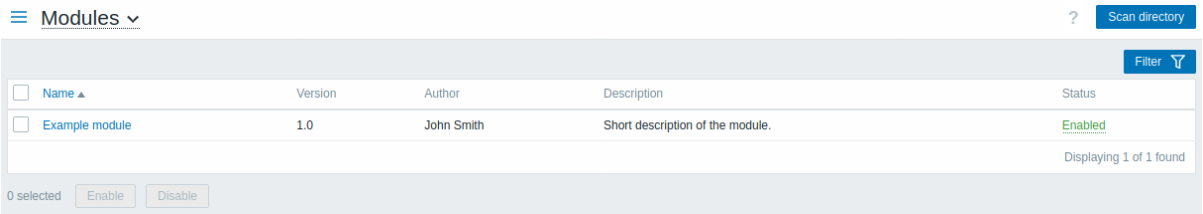
Parameter	Description
Tile provider	Select one of the available tile service providers or select *Other* to add another tile provider or self-hosted tiles (see Using a custom tile service provider).
Tile URL	The URL template for loading and displaying the tile layer on geographical maps. This field is editable only if *Tile provider* is set to *Other*. The following placeholders are supported: *{s}* represents one of the available subdomains; *{z}* represents zoom level parameter in the URL; *{x}* and *{y}* represent tile coordinates; *{r}* can be used to add "@2x" to the URL to load retina tiles. Example: `https://{s}.example.com/{z}/{x}/{y}{r}.png`
Attribution text	Tile provider attribution text to be displayed in a small text box on the map. This field is visible only if *Tile provider* is set to *Other*.
Max zoom level	Maximum zoom level of the map. This field is editable only if *Tile provider* is set to *Other*.

Using a custom tile service provider

The Geomap widget is capable to load raster tile images from a custom self-hosted or a third-party tile provider service. To use a custom third-party tile provider service or a self-hosted tile folder or server, select *Other* in the *Tile provider* field and specify the custom URL in the *Tile URL* field using proper placeholders.

8 Modules

This section allows to administer custom frontend modules.

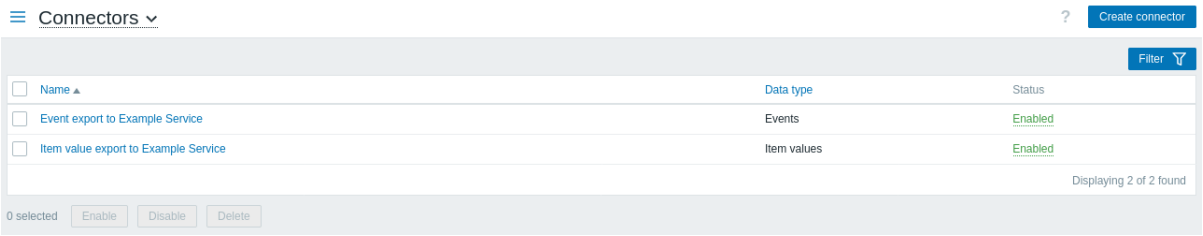


Click on *Scan directory* to register/unregister any custom modules. Registered modules will appear in the list, along with their details. Unregistered modules will be removed from the list.

You may filter modules by name or status (enabled/disabled). Click on the module status in the list to enable/disable a module. You may also mass enable/disable modules by selecting them in the list and then clicking on the *Enable/Disable* buttons below the list.

9 Connectors

This section allows to configure connectors for Zabbix data streaming to external systems over HTTP.

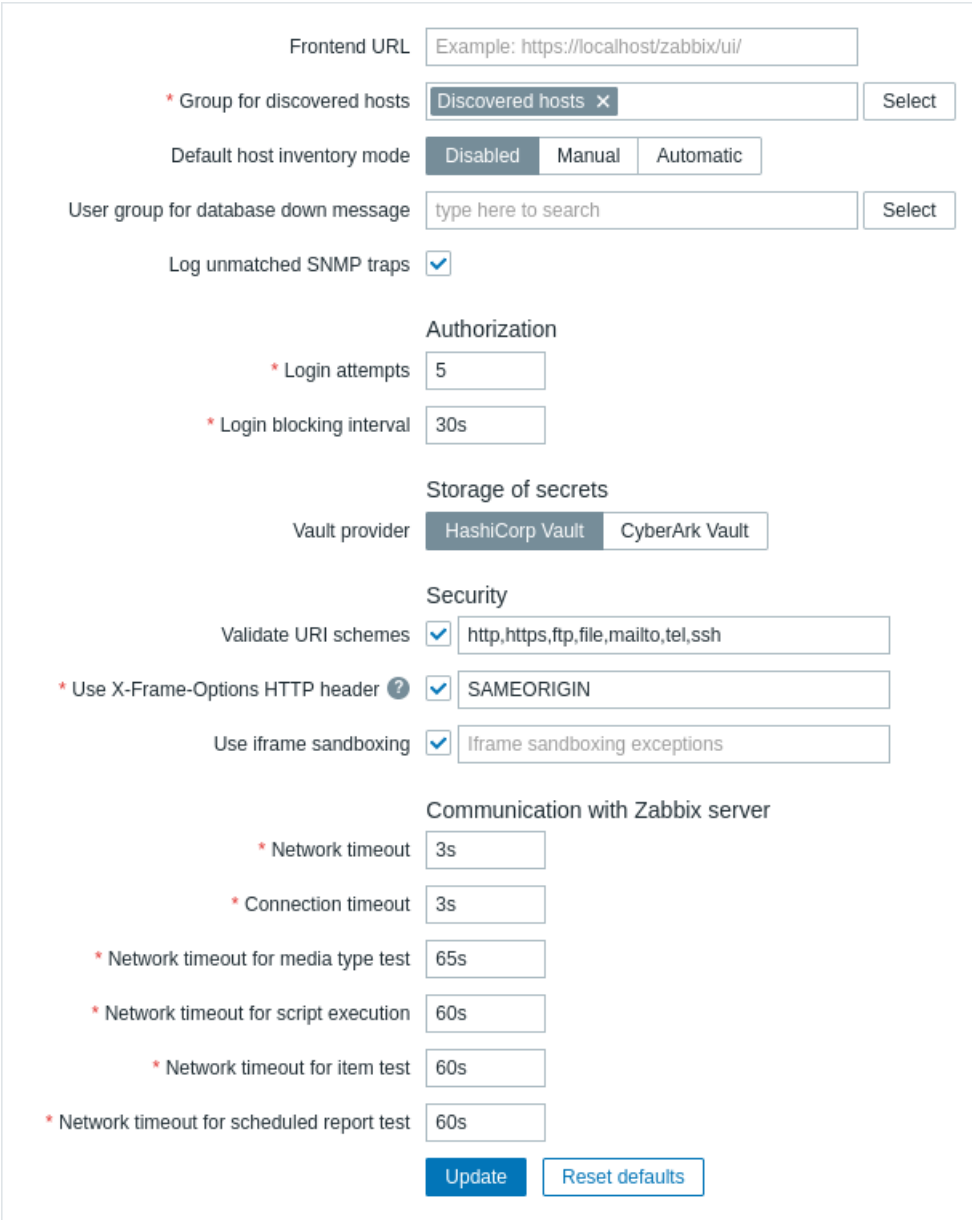


Click on *Create connector* to configure a new connector.

You may filter connectors by name or status (enabled/disabled). Click on the connector status in the list to enable/disable a connector. You may also mass enable/disable connectors by selecting them in the list and then clicking on the *Enable/Disable* buttons below the list.

10 Other parameters

This section allows configuring miscellaneous other frontend parameters.



Parameter	Description
Frontend URL	URL to Zabbix web interface. This parameter is used by Zabbix web service for communication with frontend and should be specified to enable scheduled reports.
Group for discovered hosts	Hosts discovered by network discovery and agent autoregistration will be automatically placed in the host group, selected here.

Parameter	Description
Default host inventory mode	Default mode for host inventory. It will be followed whenever a new host or host prototype is created by server or frontend unless overridden during host discovery/autoregistration by the *Set host inventory mode* operation.
User group for database down message	User group for sending alarm message or 'None'. Zabbix server depends on the availability of the backend database. It cannot work without a database. If the database is down, selected users can be notified by Zabbix. Notifications will be sent to the user group set here using all configured user media entries. Zabbix server will not stop; it will wait until the database is back again to continue processing. Notification consists of the following content: `[MySQL|PostgreSQL|Oracle] database <DB Name> [on <DB Host>:<DB Port>] is not available: <error message depending on the type of DBMS (database)>` <DB Host> is not added to the message if it is defined as an empty value and <DB Port> is not added if it is the default value ("0"). The alert manager (a special Zabbix server process) tries to establish a new connection to the database every 10 seconds. If the database is still down the alert manager repeats sending alerts, but not more often than every 15 minutes.
Log unmatched SNMP traps	Log SNMP trap if no corresponding SNMP interfaces have been found.

Authorization

Parameter	Description
Login attempts	Number of unsuccessful login attempts before the possibility to log in gets blocked.
Login blocking interval	Period of time for which logging in will be prohibited when *Login attempts* limit is exceeded.

Storage of secrets

Vault provider parameter allows selecting secret management software for storing user macro values. Supported options:

- *HashiCorp Vault* (default)
- *CyberArk Vault*

See also: Storage of secrets.

Security

Parameter	Description
Validate URI schemes	Unmark this checkbox to disable URI scheme validation (enabled by default). If marked, you can specify a comma-separated list of allowed URI schemes (default: http,https,ftp,file,mailto,tel,ssh). Applies to all fields in the frontend where URIs are used (for example, map element URLs).
Use X-Frame-Options HTTP header	Unmark this checkbox to disable the HTTP X-Frame-options header (not recommended). If marked, you can specify the value of the HTTP X-Frame-options header. Supported values: **SAMEORIGIN** (default) or **'self'** (must be single-quoted) - the page can only be displayed in a frame on the same origin as the page itself; **DENY** or **'none'** (must be single-quoted) - the page cannot be displayed in a frame, regardless of the site attempting to do so; **a string of space-separated hostnames**; adding **'self'** (must be single-quoted) to the list allows the page to be displayed in a frame on the same origin as the page itself. Note that using **'self'** or **'none'** without single quotes will result in them being regarded as hostnames.
Use iframe sandboxing	Unmark this checkbox to disable putting the retrieved URL content into sandbox (not recommended). If marked, you can specify the iframe sandboxing exceptions; unspecified restrictions will still be applied. If this field is empty, all sandbox attribute restrictions apply. For more information, see the description of the `sandbox` attribute.

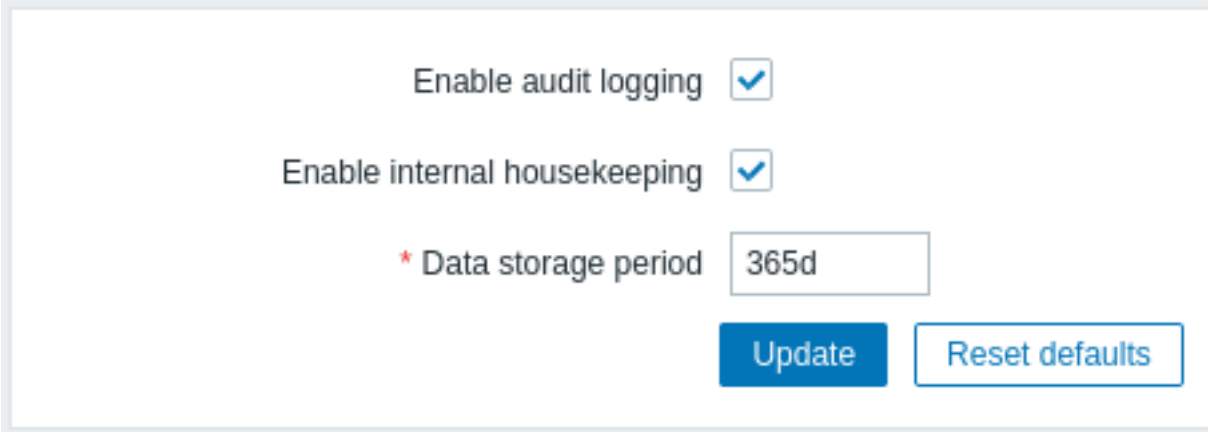
Communication with Zabbix server

Parameter	Description
Network timeout	How many seconds to wait before closing an idle socket (if a connection to Zabbix server has been established earlier, but frontend can not finish read/send data operation during this time, the connection will be dropped). Allowed range: 1 - 300s (default: 3s).
Connection timeout	How many seconds to wait before stopping an attempt to connect to Zabbix server. Allowed range: 1 - 30s (default: 3s).
Network timeout for media type test	How many seconds to wait for a response when testing a media type. Allowed range: 1 - 300s (default: 65s).
Network timeout for script execution	How many seconds to wait for a response when executing a script. Allowed range: 1 - 300s (default: 60s).
Network timeout for item test	How many seconds to wait for returned data when testing an item. Allowed range: 1 - 300s (default: 60s).
Network timeout for scheduled report test	How many seconds to wait for returned data when testing a scheduled report. Allowed range: 1 - 300s (default: 60s).

2 Audit log

Overview

This section allows configuring audit log settings.



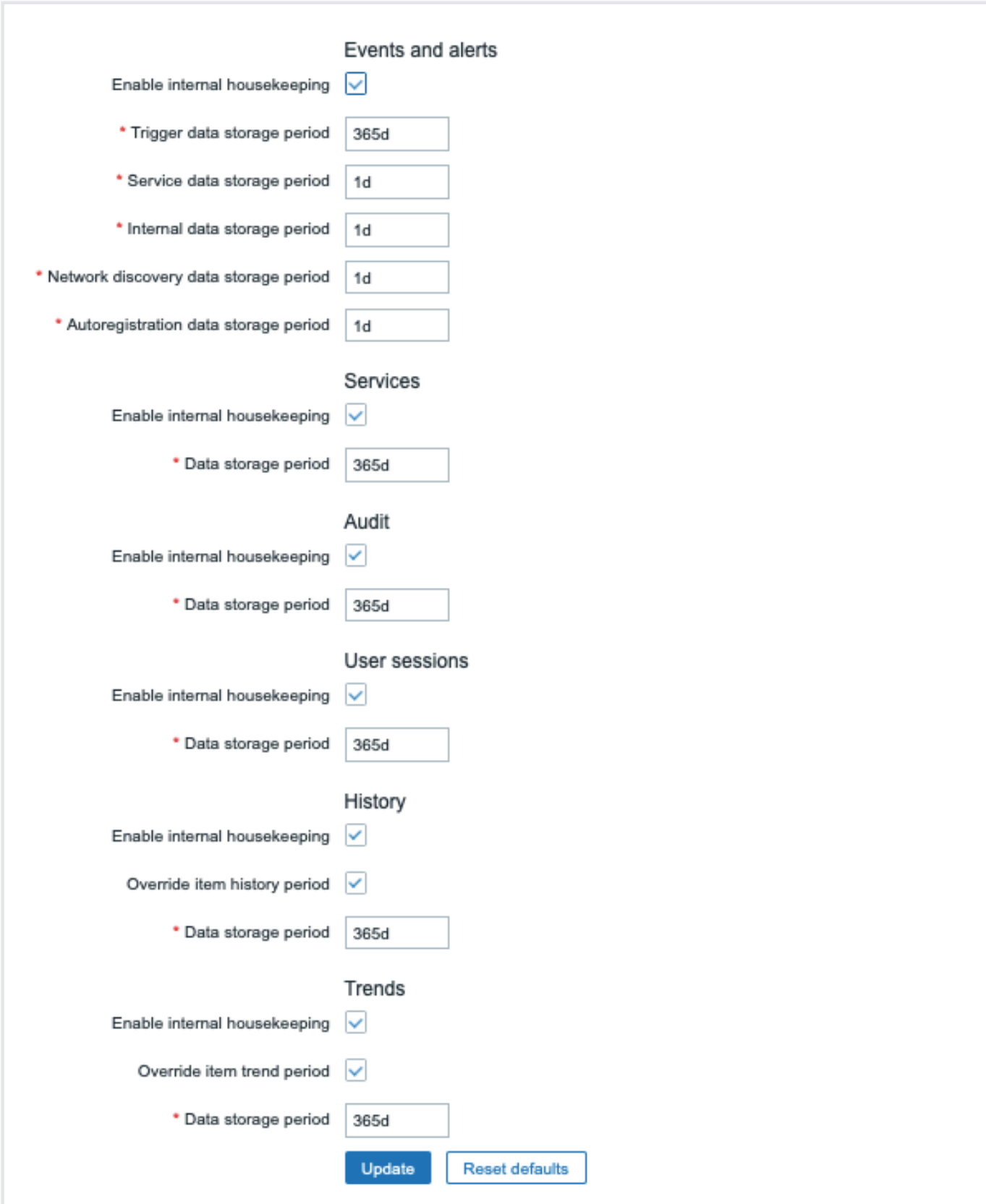
The following parameters are available:

Parameter	Description
Enable audit logging	Enable/disable audit logging. Marked by default.
Enable internal housekeeping	Enable/disable internal housekeeping for audit. Marked by default.
Data storage period	Amount of days audit records should be kept for before being removed by the housekeeper. Mandatory if housekeeping is enabled. Default: 365 days.

3 Housekeeping

Overview

The housekeeper is a periodical process, executed by Zabbix server. The process removes outdated information and information deleted by user.



In this section housekeeping tasks can be enabled or disabled on a per-task basis separately for: events and alerts/IT services/user sessions/history/trends. Audit housekeeping settings are available in a separate menu section.

If housekeeping is enabled, it is possible to set for how many days data records will be kept before being removed by the housekeeper.

Deleting an item/trigger will also delete problems generated by that item/trigger.

Also, an event will only be deleted by the housekeeper if it is not associated with a problem in any way. This means that if an event is either a problem or recovery event, it will not be deleted until the related problem record is removed. The housekeeper will delete problems first and events after, to avoid potential problems with stale events or problem records.

For history and trends an additional option is available: *Override item history period* and *Override item trend period*. This option allows to globally set for how many days item history/trends will be kept (1 hour to 25 years; or "0"), in this case overriding the values set for individual items in *History storage period/Trend storage period* fields in item configuration. Note that the storage period will not be overridden for items that have configuration option *Do not keep history* and/or *Do not keep trends* enabled.

It is possible to override the history/trend storage period even if internal housekeeping is disabled. Thus, when using an external housekeeper, the history storage period could be set using the history *Data storage period* field.

Attention:
If using TimescaleDB, in order to take full advantage of TimescaleDB automatic partitioning of history and trends tables, *Override item history period* and *Override item trend period* options must be enabled as well as *Enable internal housekeeping* option for history and trends. Otherwise, data kept in these tables will still be stored in partitions, however, the housekeeper will not drop outdated partitions, and warnings about incorrect configuration will be displayed. When dropping of outdated partitions is enabled, Zabbix server and frontend will no longer keep track of deleted items, and history for deleted items will be cleared when an outdated partition is deleted.

Time suffixes are supported in the period fields, e.g. 1d (one day), 1w (one week). The minimum is 1 day (1 hour for history), the maximum - 25 years.

The *Reset defaults* button allows to revert any changes made.

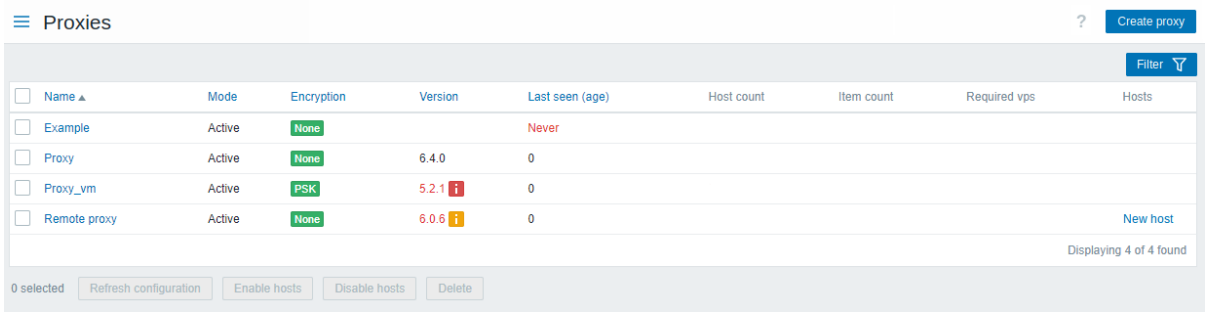
4 Proxies

Overview

In the *Administration* → *Proxies* section proxies for distributed monitoring can be configured in the Zabbix frontend.

Proxies

A listing of existing proxies with their details is displayed.



Displayed data:

Column	Description
Name	Name of the proxy. Clicking on the proxy name opens the proxy configuration form.
Mode	Proxy mode is displayed - *Active* or *Passive*.
Encryption	Encryption status for connections from the proxy is displayed: **None** - no encryption; **PSK** - using pre-shared key; **Cert** - using certificate.
Version	Proxy version (three digit version number). If proxy is outdated or unsupported, version number is highlighted (red) and info status icon (yellow or red) is displayed. Hover over the icon for details.
Last seen (age)	The time when the proxy was last seen by the server is displayed.
Host count	The number of enabled hosts assigned to the proxy is displayed.
Item count	The number of enabled items on enabled hosts assigned to the proxy is displayed.
Required vps	Required proxy performance is displayed (the number of values that need to be collected per second).
Hosts	All hosts monitored by the proxy are listed. Clicking on the host name opens the host configuration form.

To configure a new proxy, click on the *Create proxy* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

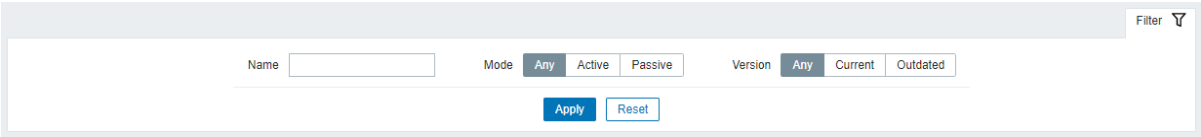
- *Refresh configuration* - refresh configuration of the proxies;
- *Enable hosts* - change the status of hosts monitored by the proxy to *Monitored*;
- *Disable hosts* - change the status of hosts monitored by the proxy to *Not monitored*;
- *Delete* - delete the proxies.

To use these options, mark the checkboxes before the respective proxies, then click on the required button.

Using filter

You can use the filter to display only the proxies you are interested in. For better search performance, data is searched with macros unresolved.

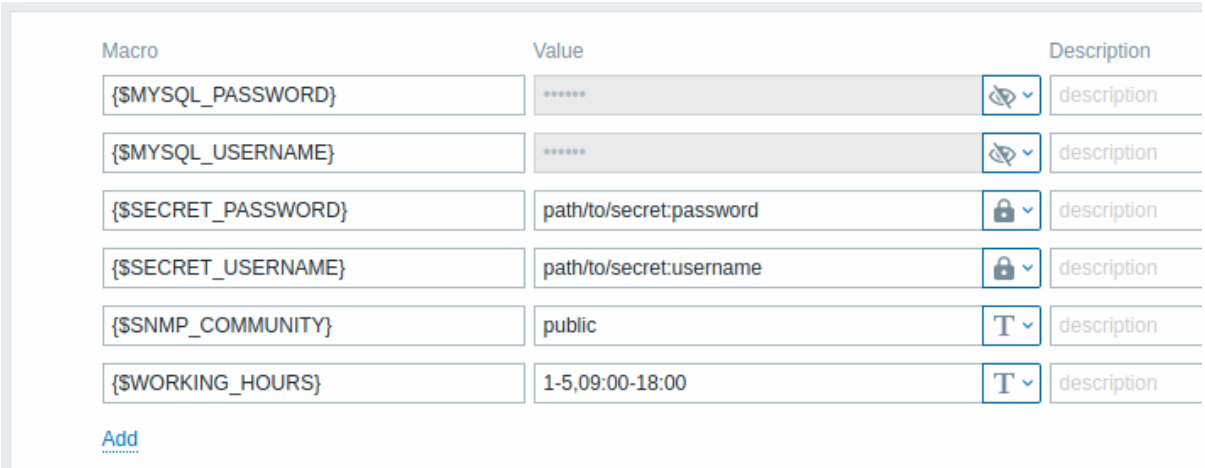
The *Filter* link is available above the list of proxies. If you click on it, a filter becomes available where you can filter proxies by name, mode and version. Note that the filter option *Outdated* displays both outdated (partially supported) and unsupported proxies.



5 Macros

Overview

This section allows to define system-wide user macros as name-value pairs. Note that macro values can be kept as plain text, secret text or Vault secret. Adding a description is also supported.



6 Queue

Overview

In the *Administration* → *Queue* section items that are waiting to be updated are displayed.

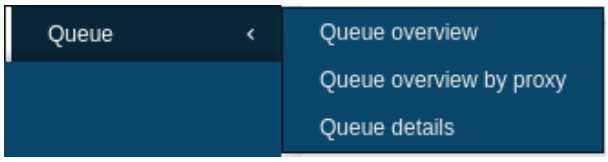
Ideally, when you open this section it should all be "green" meaning no items in the queue. If all items are updated without delay, there are none waiting. However, due to lacking server performance, some items may get delayed and the information is displayed in this section. For more details, see the Queue section.

Note:
The queue is available only if Zabbix server is running. Items are not counted in the queue if the item interface becomes unavailable due to connection problems or agent not working properly.

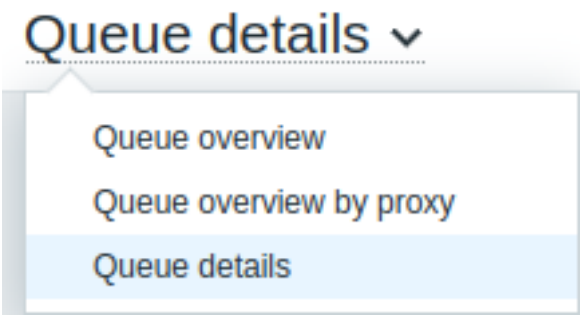
The *Administration* → *Queue* section contains the following pages:

- Queue overview — displays queue by item type;
- Queue overview by proxy — displays queue by proxy;
- Queue details — displays a list of delayed items.

The list of available pages appears upon pressing on *Queue* in the *Administration* menu section. It is also possible to switch between pages by using a title dropdown in the top left corner.



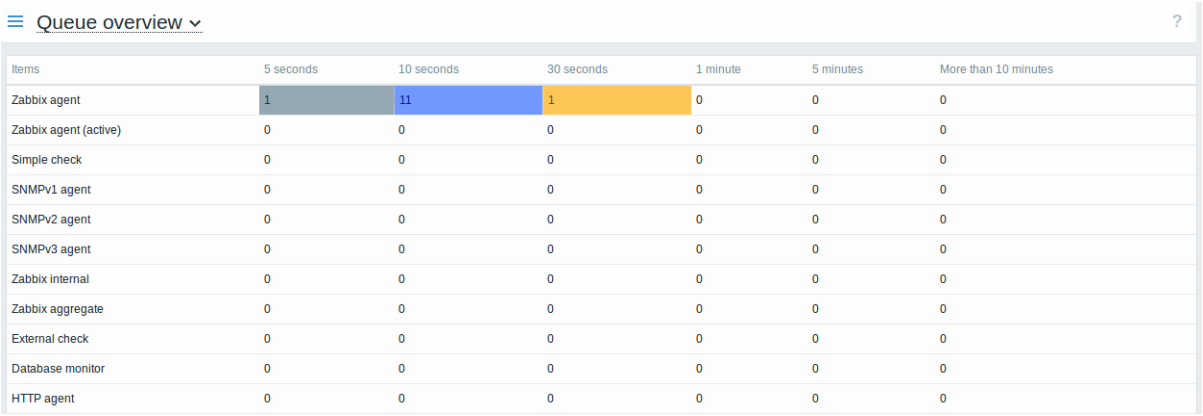
Third-level menu.



Title dropdown.

Overview by item type

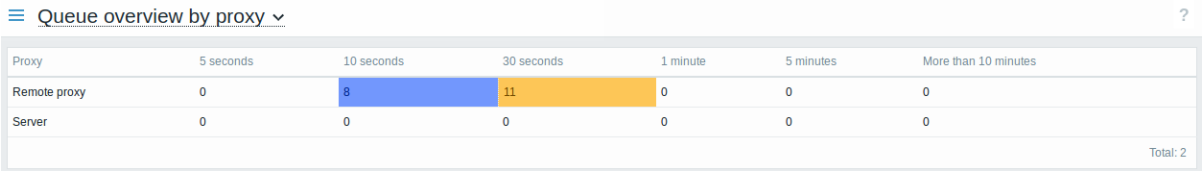
In this screen it is easy to locate if the problem is related to one or several item types.



Each line contains an item type. Each column shows the number of waiting items - waiting for 5-10 seconds/10-30 seconds/30-60 seconds/1-5 minutes/5-10 minutes or over 10 minutes respectively.

Overview by proxy

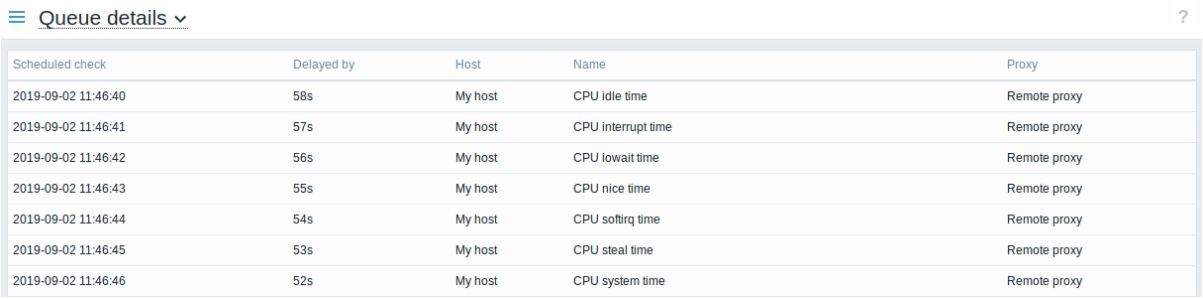
In this screen it is easy to locate if the problem is related to one of the proxies or the server.



Each line contains a proxy, with the server last in the list. Each column shows the number of waiting items - waiting for 5-10 seconds/10-30 seconds/30-60 seconds/1-5 minutes/5-10 minutes or over 10 minutes respectively.

List of waiting items

In this screen, each waiting item is listed.



Displayed data:

Column	Description
Scheduled check	The time when the check was due is displayed.

Column	Description
Delayed by	The length of the delay is displayed.
Host	Host of the item is displayed.
Name	Name of the waiting item is displayed.
Proxy	The proxy name is displayed, if the host is monitored by proxy.

Possible error messages

You may encounter a situation when no data is displayed and the following error message appears:



Error message in this case is the following:

`Cannot display item queue. Permission denied`

This happens when the PHP configuration parameters in the *zabbix.conf.php* file - `$ZBX_SERVER` or both `$ZBX_SERVER` and `$ZBX_SERVER_PORT` - point to an existing Zabbix server that uses a different database.

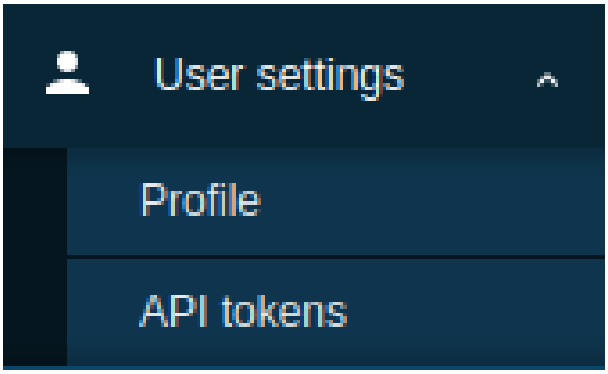
3 User settings

Overview

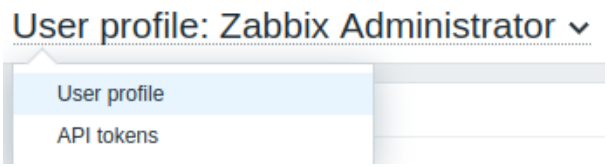
Depending on user role permissions, the *User settings* section may contain the following pages:

- *Profile* or *User profile* - for customizing certain Zabbix frontend features.
- *API tokens* - for managing API tokens assigned to the current user.

The list of available pages appears upon pressing on the user icon near the bottom of the Zabbix menu (not available for a guest user). It is also possible to switch between pages by using a title dropdown in the top left corner.



Third-level menu.

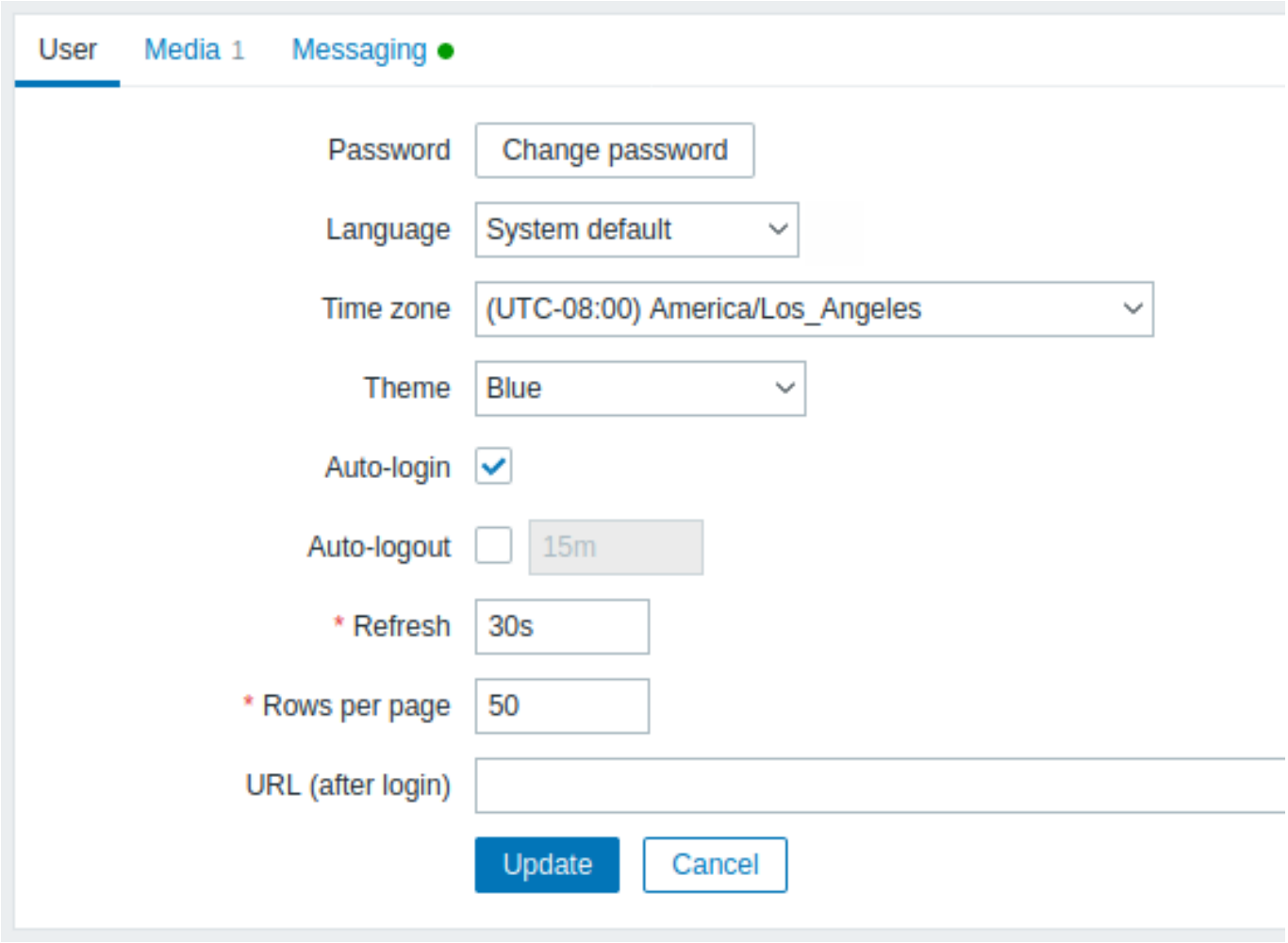


Title dropdown.

User profile

The **User profile** section provides options to set custom interface language, color theme, number of rows displayed in the lists, etc. The changes made here will be applied to the current user only.

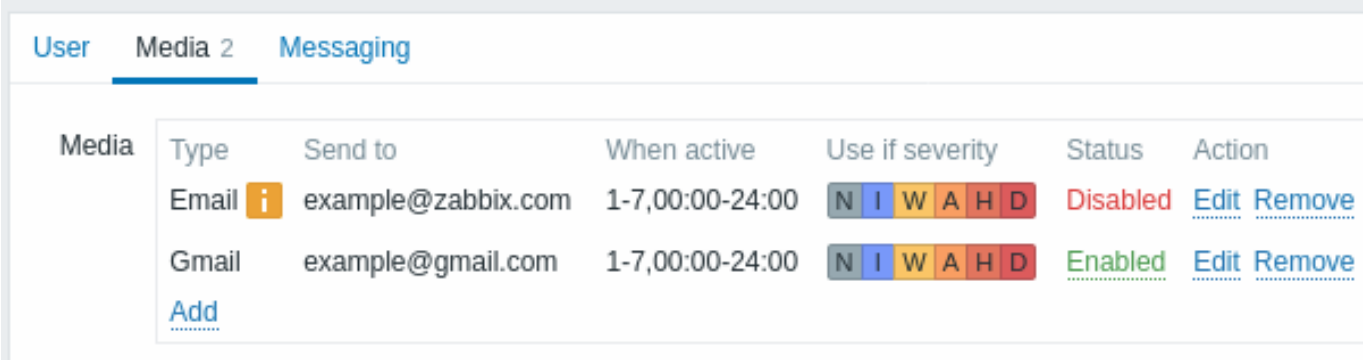
The **User** tab allows you to set various user preferences.



Parameter	Description
Password	Click on the *Change password* button to open three fields: *Old password*, *New password*, *New password (once again)*. On a successful password change, the user will be logged out of all active sessions. Note that the password can only be changed for users using Zabbix internal authentication.
Language	Select the interface language of your choice or select **System default** to use default system settings. For more information, see Installation of additional frontend languages.
Time zone	Select the time zone to override global time zone on user level or select **System default** to use global time zone settings.
Theme	Select a color theme specifically for your profile: **System default** - use default system settings **Blue** - standard blue theme **Dark** - alternative dark theme **High-contrast light** - light theme with high contrast **High-contrast dark** - dark theme with high contrast
Auto-login	Mark this checkbox to make Zabbix remember you and log you in automatically for 30 days. Browser cookies are used for this.
Auto-logout	With this checkbox marked you will be logged out automatically, after the set amount of seconds (minimum 90 seconds, maximum 1 day). Time suffixes are supported, for example: 90s, 5m, 2h, 1d. Note that this option will not work: * When Monitoring menu pages perform background information refreshes. In case pages refreshing data in a specific time interval (dashboards, graphs, latest data, etc.) are left open session lifetime is extended, respectively disabling auto-logout feature. * If logging in with the *Remember me for 30 days* option checked. *Auto-logout* can also accept "0", meaning that auto-logout feature becomes disabled after profile settings update.

Parameter	Description
Refresh	Set how often the information on the *Monitoring* menu pages will be refreshed (minimum 0 seconds, maximum 1 hour). Time suffixes are supported, for example: 30s, 90s, 1m, 1h.
Rows per page	You can set how many rows will be displayed per page in the lists. Fewer rows (and fewer records to display) mean faster loading times.
URL (after login)	You can set a specific URL to be displayed after the login. Instead of the default *Dashboards* it can be, for example, the URL of *Monitoring* → *Triggers*.

The **Media** tab allows you to specify the media details for the user, such as media types and addresses to use and when to use them to deliver notifications.



If the media type has been disabled:

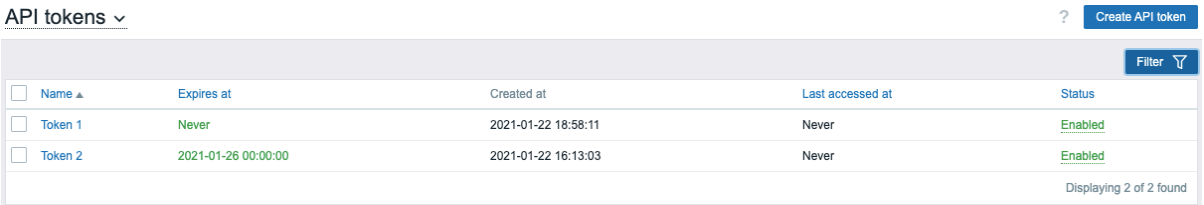
- A yellow info icon is displayed after the name.
- *Disabled* is displayed in the Status column.

Note:
Only admin level users (*Admin* and *Super admin*) can change their own media details.

The **Messaging** tab allows you to set global notifications.

API tokens

API tokens section allows to view tokens assigned to the user, edit token details and create new tokens. This section is only available to a user if *Manage API tokens* action is allowed in the user role settings.



You may filter API tokens by name, expiry date, or status (*Enabled*/*Disabled*). Click on the token status in the list to quickly enable/disable a token. You may also mass enable/disable tokens by selecting them in the list and then clicking on the *Enable*/*Disable* buttons below the list.

Attention:
Users cannot view *Auth token* value of the tokens assigned to them in Zabbix. *Auth token* value is displayed only once - immediately after creating a token. If it has been lost, the token has to be regenerated.

1 Global notifications

Overview

Global notifications are a way of displaying issues that are currently happening right on the screen you're at in Zabbix frontend.

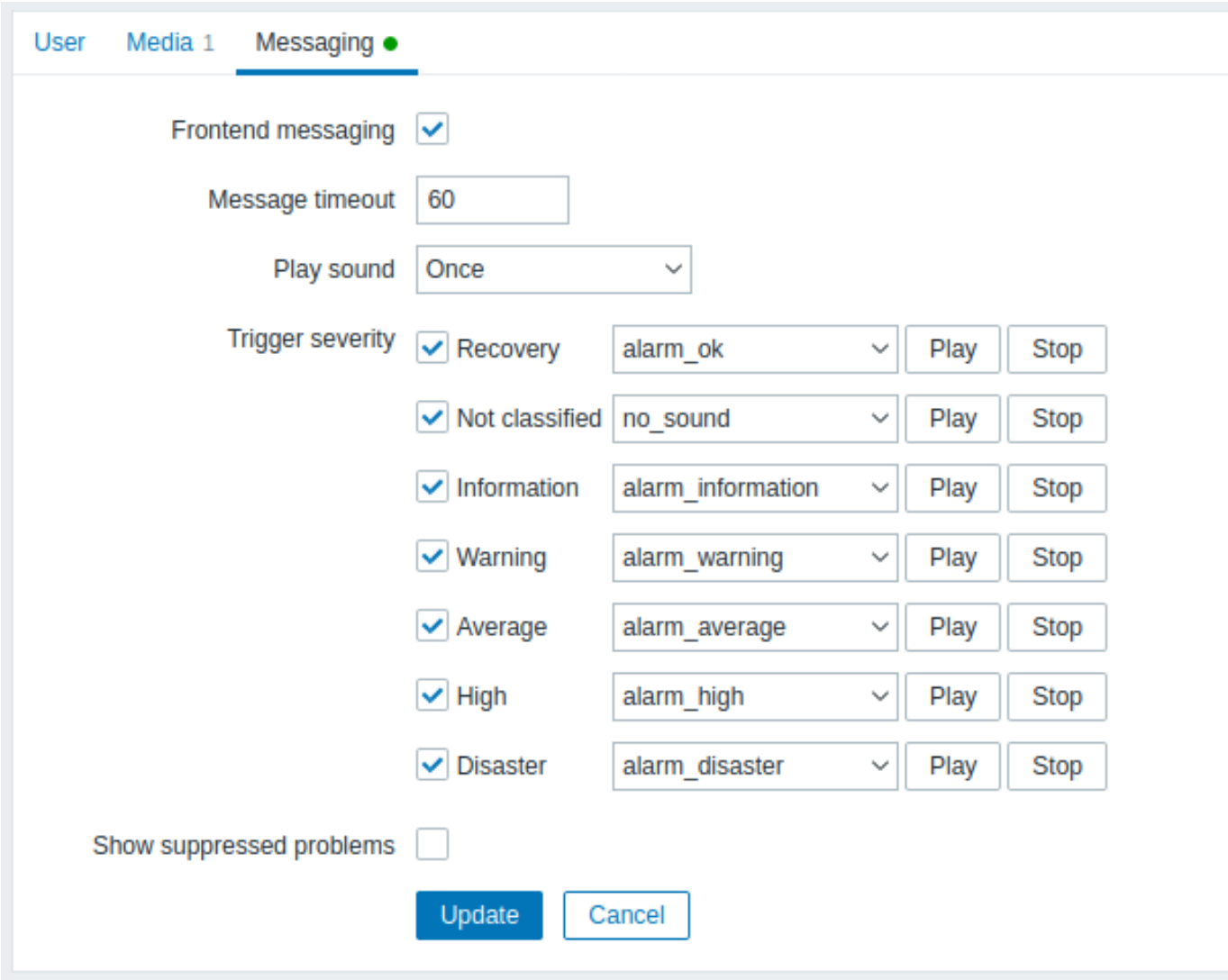
Without global notifications, working in some other location than *Problems* or the *Dashboard* would not show any information regarding issues that are currently happening. Global notifications will display this information regardless of where you are.

Global notifications involve both showing a message and playing a sound.

Attention:
The auto play of sounds may be disabled in recent browser versions by default. In this case, you need to change this setting manually.

Configuration

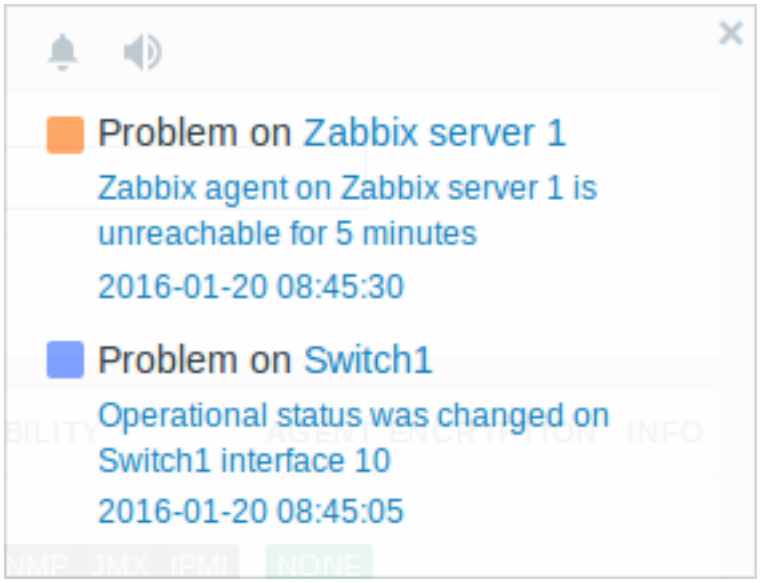
Global notifications can be enabled per user in the *Messaging* tab of profile configuration.



Parameter	Description
Frontend messaging	Mark the checkbox to enable global notifications.
Message timeout	You can set for how long the message will be displayed. By default, messages will stay on screen for 60 seconds. Time suffixes are supported, e.g. 30s, 5m, 2h, 1d.
Play sound	You can set how long the sound will be played. **Once** - sound is played once and fully. **10 seconds** - sound is repeated for 10 seconds. **Message timeout** - sound is repeated while the message is visible.
Trigger severity	You can set the trigger severities that global notifications and sounds will be activated for. You can also select the sounds appropriate for various severities. If no severity is marked then no messages will be displayed at all. Also, recovery messages will only be displayed for those severities that are marked. So if you mark *Recovery* and *Disaster*, global notifications will be displayed for the problems and the recoveries of disaster severity triggers.
Show suppressed problems	Mark the checkbox to display notifications for problems which would otherwise be suppressed (not shown) because of host maintenance.

Global messages displayed

As the messages arrive, they are displayed in a floating section on the right hand side. This section can be repositioned freely by dragging the section header.



For this section, several controls are available:

- **Snooze** button silences the currently active alarm sound;
- **Mute/Unmute** button switches between playing and not playing the alarm sounds at all.

2 Sound in browsers

Overview

Sound is used in global notifications.

For the sounds to be played in Zabbix frontend, *Frontend messaging* must be enabled in the user profile *Messaging* tab, with all trigger severities checked, and sounds should also be enabled in the global notification pop-up window.

If for some reasons audio cannot be played on the device, the button in the global notification pop-up window will permanently remain in the "mute" state and the message "Cannot support notification audio for this device." will be displayed upon hovering over the button.

Sounds, including the default audio clips, are supported in MP3 format only.

The sounds of Zabbix frontend have been successfully tested in recent Firefox/Opera browsers on Linux and in Chrome, Firefox, Microsoft Edge, and Opera browsers on Windows.

Attention:
The auto play of sounds may be disabled in recent browser versions by default. In this case, you need to change this setting manually.

4 Global search

It is possible to search Zabbix frontend for hosts, host groups, templates and template groups.

The search input box is located below the Zabbix logo in the menu. The search can be started by pressing *Enter* or clicking on the search icon.



If there is a host that contains the entered string in any part of the name, a dropdown will appear, listing all such hosts (with the matching part highlighted in orange). The dropdown will also list a host if that host's visible name is a match to the technical name entered as a search string; the matching host will be listed, but without any highlighting.

Searchable attributes

Hosts can be searched by the following properties:

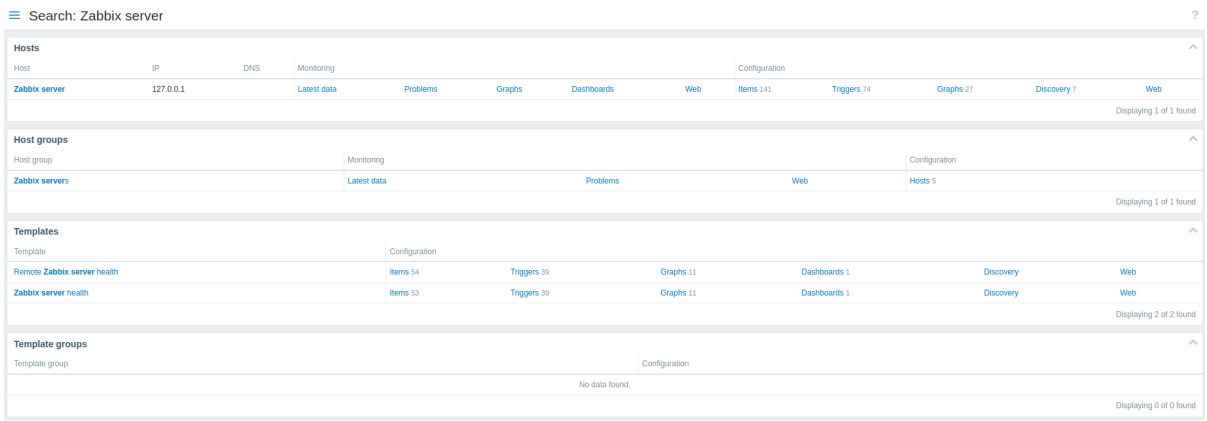
- Host name
- Visible name
- IP address
- DNS name

Templates can be searched by name or visible name. If you search by a name that is different from the visible name (of a template/host), in the search results it is displayed below the visible name in parentheses.

Host and template groups can be searched by name. Specifying a parent group implicitly selects all nested groups.

Search results

Search results consist of four separate blocks for hosts, host groups, templates and template groups.



It is possible to collapse/expand each individual block. The entry count is displayed at the bottom of each block, for example, *Displaying 13 of 13 found*. Total entries displayed within one block are limited to 100.

Each entry provides links to monitoring and configuration data. See the full list of links.

For all configuration data (such as items, triggers, graphs) the amount of entities found is displayed by a number next to the entity name, in gray. **Note** that if there are zero entities, no number is displayed.

Enabled hosts are displayed in blue, disabled hosts in red.

Links available

For each entry the following links are available:

- Hosts
  - Monitoring
    - Latest data
    - Problems
    - Graphs

        * Host dashboards
        * Web scenarios
    - Configuration
        * Items
        * Triggers
        * Graphs
        * Discovery rules
        * Web scenarios
- Host groups
    - Monitoring
        * Latest data
        * Problems
        * Web scenarios
    - Configuration
        * Hosts
- Templates
    - Configuration
        * Items
        * Triggers
        * Graphs
        * Template dashboards
        * Discovery rules
        * Web scenarios
- Template groups
    - Configuration
        * Templates

5 Frontend maintenance mode

Overview

Zabbix web frontend can be temporarily disabled in order to prohibit access to it. This can be useful for protecting the Zabbix database from any changes initiated by users, thus protecting the integrity of database.

Zabbix database can be stopped and maintenance tasks can be performed while Zabbix frontend is in maintenance mode.

Users from defined IP addresses will be able to work with the frontend normally during maintenance mode.

Configuration

In order to enable maintenance mode, the `maintenance.inc.php` file (located in /conf of Zabbix HTML document directory on the web server) must be modified to uncomment the following lines:

```
// Maintenance mode.
define('ZBX_DENY_GUI_ACCESS', 1);

// Array of IP addresses, which are allowed to connect to frontend (optional).
$ZBX_GUI_ACCESS_IP_RANGE = array('127.0.0.1');

// Message shown on warning screen (optional).
$ZBX_GUI_ACCESS_MESSAGE = 'We are upgrading MySQL database till 15:00. Stay tuned...';
```

Note:
Mostly the `maintenance.inc.php` file is located in `/conf` of Zabbix HTML document directory on the web server. However, the location of the directory may differ depending on the operating system and a web server it uses.
For example, the location for:

- SUSE and RedHat is `/etc/zabbix/web/maintenance.inc.php`.
- Debian-based systems is `/usr/share/zabbix/conf/`.

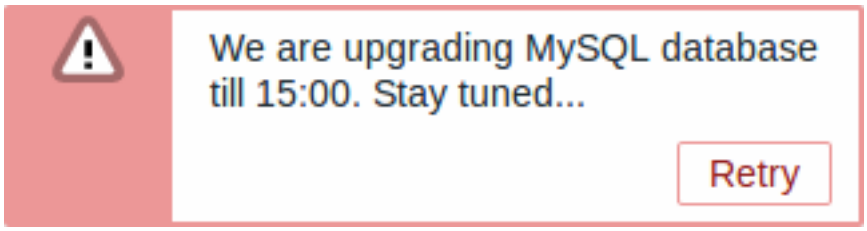
See also Copying PHP files.

Parameter	Details
ZBX_DENY_GUI_ACCESS	Enable maintenance mode: 1 – maintenance mode is enabled, disabled otherwise
ZBX_GUI_ACCESS_IP_RANGE	Array of IP addresses, which are allowed to connect to frontend (optional). For example: `array('192.168.1.1', '192.168.1.2')`
ZBX_GUI_ACCESS_MESSAGE	A message you can enter to inform users about the maintenance (optional).

Note that the location of the `/conf` directory will vary based on the operating system and web server.

Display

The following screen will be displayed when trying to access the Zabbix frontend while in maintenance mode. The screen is refreshed every 30 seconds in order to return to a normal state without user intervention when the maintenance is over.



IP addresses defined in ZBX_GUI_ACCESS_IP_RANGE will be able to access the frontend as always.

6 Page parameters

Overview

Most Zabbix web interface pages support various HTTP GET parameters that control what will be displayed. They may be passed by specifying parameter=value pairs after the URL, separated from the URL by a question mark (?) and from each other by ampersands (&).

Monitoring → Problems

The following parameters are supported:

- `show` - filter option ”Show”: 1 - recent problems, 2 - all, 3 - in problem state
- `name` - filter option ”Problem”: freeform string
- `severities` - filter option ”Severity”: array of selected severities in a format ’severities[*]=*’ (replace * with severity level): 0 - not classified, 1 - information, 2 - warning, 3 - average, 4 - high, 5 - disaster
- `inventory` - filter option ”Host inventory”: array of inventory fields: [field], [value]
- `evaltype` - filter option ”Tags”, tag filtering strategy: 0 - And/Or, 2 - Or
- `tags` - filter option ”Tags”: array of defined tags: [tag], [operator], [value]
- `show_tags` - filter option ”Show tags”: 0 - none, 1 - one, 2 - two, 3 - three
- `tag_name_format` - filter option ”Tag name”: 0 - full name, 1 - shortened, 2 - none
- `tag_priority` - filter option ”Tag display priority”: comma-separated string of tag display priority
- `show_suppressed` - filter option ”Show suppressed problems”: should be ’show_suppressed=1’ to show
- `unacknowledged` - filter option ”Show unacknowledged only”: should be ’unacknowledged=1’ to show
- `compact_view` - filter option ”Compact view”: should be ’compact_view=1’ to show
- `highlight_row` - filter option ”Highlight whole row” (use problem color as background color for every problem row): should be ’1’ to highlight; can be set only when ’compact_view’ is set
- `filter_name` - filter properties option ”Name”: freeform string
- `filter_show_counter` - filter properties option ”Show number of records”: 1 - show, 0 - do not show
- `filter_custom_time` - filter properties option ”Set custom time period”: 1 - set, 0 - do not set
- `sort` - sort column: clock, host, severity, name
- `sortorder` - sort order or results: DESC - descending, ASC - ascending
- `age_state` - filter option ”Age less than”: should be ’age_state=1’ to enable ’age’. Is used only when ’show’ equals 3.
- `age` - filter option ”Age less than”: days
- `groupids` - filter option ”Host groups”: array of host groups IDs
- `hostids` - filter option ”Hosts”: array of host IDs
- `triggerids` - filter option ”Triggers”: array of trigger IDs
- `show_timeline` - filter option ”Show timeline”: should be ’show_timeline=1’ to show

- `details` - filter option "Show details": should be 'details=1' to show
- `from` - date range start, can be 'relative' (e.g.: now-1m). Is used only when 'filter_custom_time' equals 1.
- `to` - date range end, can be 'relative' (e.g.: now-1m). Is used only when 'filter_custom_time' equals 1.

Kiosk mode

The kiosk mode in supported frontend pages can be activated using URL parameters. For example, in dashboards:

- `/zabbix.php?action=dashboard.view&kiosk=1` - activate kiosk mode
- `/zabbix.php?action=dashboard.view&kiosk=0` - activate normal mode

Slideshow

It is possible to activate a slideshow in the dashboard:

- `/zabbix.php?action=dashboard.view&slideshow=1` - activate slideshow

7 Definitions

Overview

While many things in the frontend can be configured using the frontend itself, some customizations are currently only possible by editing a definitions file.

This file is `defines.inc.php` located in /include of the Zabbix HTML document directory.

Parameters

Parameters in this file that could be of interest to users:

- ZBX_MIN_PERIOD

Minimum graph period, in seconds. One minute by default.

- GRAPH_YAXIS_SIDE_DEFAULT

Default location of Y axis in simple graphs and default value for drop down box when adding items to custom graphs. Possible values: 0 - left, 1 - right.

Default: 0

- ZBX_SESSION_NAME (available since 4.0.0)

String used as the name of the Zabbix frontend session cookie.

Default: zbx_sessionid

- ZBX_DATA_CACHE_TTL (available since 5.2.0)

TTL timeout in seconds used to invalidate data cache of Vault response. Set 0 to disable Vault response caching.

Default: 60

- SUBFILTER_VALUES_PER_GROUP (available since 6.0.5)

Number of subfilter values per group (For example, in the latest data subfilter).

Default: 1000

8 Creating your own theme

Overview

By default, Zabbix provides a number of predefined themes. You may follow the step-by-step procedure provided here in order to create your own. Feel free to share the result of your work with Zabbix community if you created something nice.

Step 1

To define your own theme you'll need to create a CSS file and save it in the `assets/styles/` folder (for example, *custom-theme.css*). You can either copy the files from a different theme and create your theme based on it or start from scratch.

Step 2

Add your theme to the list of themes returned by the APP::getThemes() method. You can do this by overriding the ZBase::getThemes() method in the APP class. This can be done by adding the following code before the closing brace in *include/classes/core/APP.php*:

```
public static function getThemes() {
    return array_merge(parent::getThemes(), [
        'custom-theme' => _('Custom theme')
    ]);
}
```

Attention:
Note that the name you specify within the first pair of quotes must match the name of the theme file without extension.

To add multiple themes, just list them under the first theme, for example:

```
public static function getThemes() {
    return array_merge(parent::getThemes(), [
        'custom-theme' => _('Custom theme'),
        'anothertheme' => _('Another theme'),
        'onemoretheme' => _('One more theme')
    ]);
}
```

Note that every theme except the last one must have a trailing comma.

Note:
To change graph colors, the entry must be added in the *graph_theme* database table.

Step 3

Activate the new theme.

In Zabbix frontend, you may either set this theme to be the default one or change your theme in the user profile.

Enjoy the new look and feel!

9 Debug mode

Overview

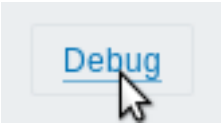
Debug mode may be used to diagnose performance problems with frontend pages.

Configuration

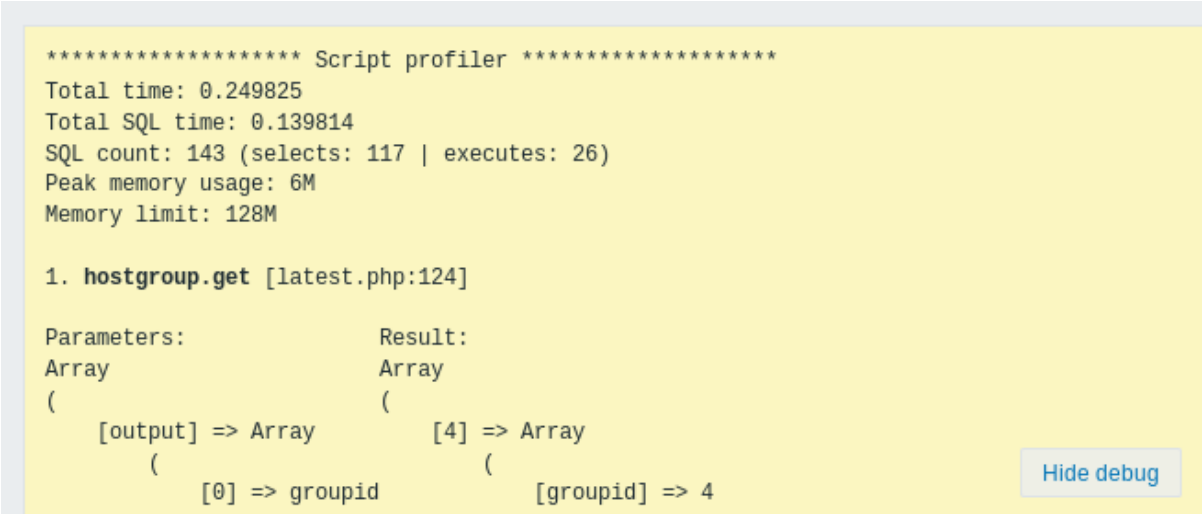
Debug mode can be activated for individual users who belong to a user group:

- when configuring a user group;
- when viewing configured user groups.

When *Debug mode* is enabled for a user group, its users will see a *Debug* button in the lower right corner of the browser window:



Clicking on the *Debug* button opens a new window below the page contents which contains the SQL statistics of the page, along with a list of API calls and individual SQL statements:



In case of performance problems with the page, this window may be used to search for the root cause of the problem.

Warning:
Enabled *Debug mode* negatively affects frontend performance.

10 Cookies used by Zabbix

Overview

This page provides a list of cookies used by Zabbix.

Name	Description	Values	Expires/Max-Age	HttpOnly[1]	Secure[2]
ZBX_SESSIONID	Zabbix frontend session data, stored as JSON encoded by Base64		Session (expires when the browsing session ends)	+	+ (only if HTTPS is enabled on a web server)
tab	Active tab number; this cookie is only used on pages with multiple tabs (e.g. *Host*, *Trigger* or *Action* configuration page) and is created, when a user navigates from a primary tab to another tab (such as *Tags* or *Dependencies* tab). 0 is used for the primary tab.	Example: 1	Session (expires when the browsing session ends)	-	-
browserwarning_ignore	Whether a warning about using an outdated browser should be ignored.	yes	Session (expires when the browsing session ends)	-	-
system-message-ok	A message to show as soon as page is reloaded.	Plain text message	Session (expires when the browsing session ends) or as soon as page is reloaded	+	-
system-message-error	An error message to show as soon as page is reloaded.	Plain text message	Session (expires when the browsing session ends) or as soon as page is reloaded	+	-

Note:
Forcing 'HttpOnly' flag on Zabbix cookies by a webserver directive is not supported.

Footnotes

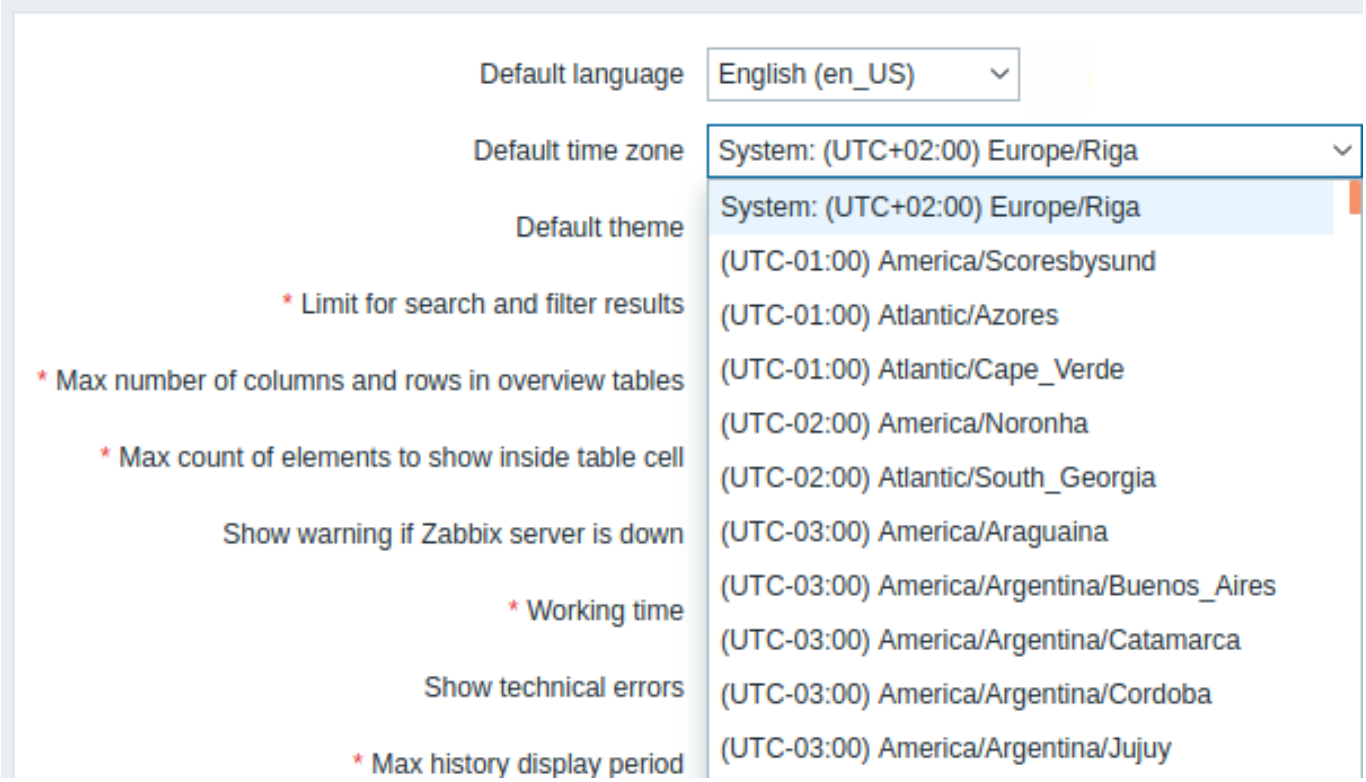
[1] When `HttpOnly` is 'true' the cookie will be made accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript. This setting can effectively help to reduce identity theft through XSS attacks (although it is not supported by all browsers).

[2] `Secure` indicates that the cookie should only be transmitted over a secure HTTPS connection from the client. When set to 'true', the cookie will only be set if a secure connection exists.

11 Time zones

Overview

The frontend time zone can be set globally in the frontend and adjusted for individual users.



If *System* is selected, the web server time zone will be used for the frontend (including the value of 'date.timezone' of php.ini, if set), while Zabbix server will use the time zone of the machine it is running on.

Note:
Zabbix server will only use the specified global/user time zone when expanding macros in notifications (e.g. {EVENT.TIME} can expand to a different time zone per user) and for the time limit when notifications are sent (see "When active" setting in user media configuration).

Configuration

The global time zone:

- can be set manually when installing the frontend
- can be modified in *Administration* → *General* → *GUI*

User-level time zone:

- can be set when configuring/updating a user
- can be set by each user in their user profile

See also: Aligning time zones when using scheduling intervals.

12 Resetting password

Overview This section describes the steps for resetting user passwords in Zabbix.

Steps Turn to your Zabbix administrator if you have forgotten your Zabbix password and cannot log in.

A Super administrator user can change passwords for all users in the user configuration form.

If a Super administrator has forgotten their password and cannot log in, the following SQL query must be run to apply the default password to the Super admin user (replace 'Admin' with the appropriate Super admin username):

```
UPDATE users SET passwd = '$2a$10$ZXIvHAEP2ZM.dLXTm6uPHOMVlARXX7cqjbhM6Fn0cANzkCQBWpMrS' WHERE username =
```

After running this query, the user password will be set to *zabbix*. Make sure to change the default password on the first login.

19 API

Overview The Zabbix API allows you to programmatically retrieve and modify configuration of Zabbix and provides access to historical data. It is widely used to:

- Create new applications to work with Zabbix;
- Integrate Zabbix into a third-party software;
- Automate routine tasks.

The Zabbix API is an HTTP-based API, and it is shipped as a part of the web frontend. It uses the JSON-RPC 2.0 protocol, which means two things:

- The API consists of a set of separate methods.
- Requests and responses between the clients and the API are encoded using the JSON format.

More information about the protocol and JSON can be found in the JSON-RPC 2.0 specification and the JSON format homepage.

For more information about integrating Zabbix functionality into your Python applications, see the zabbix_utils Python library for Zabbix API.

Structure The API consists of a number of methods that are nominally grouped into separate APIs. Each of the methods performs one specific task. For example, the `host.create` method belongs to the *host* API and is used to create new hosts. Historically, APIs are sometimes referred to as "classes".

> **Note:**
> Most APIs contain at least four methods: `get`, `create`, `update` and `delete` for retrieving, creating, updating and deleting data respectively, but some APIs may provide a totally different set of methods.

Performing requests Once you have set up the frontend, you can use remote HTTP requests to call the API. To do that, you need to send HTTP POST requests to the `api_jsonrpc.php` file located in the frontend directory. For example, if your Zabbix frontend is installed under `https://example.com/zabbix`, an HTTP request to call the `apiinfo.version` method may look like this:

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"apiinfo.version","params":{},"id":1}'
```

The request must have the `Content-Type` header set to one of these values: `application/json-rpc`, `application/json` or `application/jsonrequest`.

The request object contains the following properties:

- `jsonrpc` - the version of the JSON-RPC protocol used by the API (Zabbix API implements JSON-RPC version 2.0);
- `method` - the API method being called;
- `params` - the parameters that will be passed to the API method;
- `id` - an arbitrary identifier of the request.

If the request is correct, the response returned by the API should look like this:

```
{
    "jsonrpc": "2.0",
    "result": "6.4.0",
```

```
    "id": 1
}
```

The response object, in turn, contains the following properties:

- `jsonrpc` - the version of the JSON-RPC protocol;
- `result` - the data returned by the method;
- `id` - an identifier of the corresponding request.

Example workflow The following section will walk you through some examples of usage in a greater detail.

Authentication To access any data in Zabbix, you need to either:

- use an existing API token (created in Zabbix frontend or using the Token API);
- use an authentication token obtained with the user.login method.

For example, if you wanted to obtain a new authentication token by logging in as a standard *Admin* user, then a JSON request would look like this:

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"user.login","params":{"username":"Admin","password":"zabbix"},"id":1}
```

If you provided the credentials correctly, the response returned by the API should contain the user authentication token:

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

Authorization methods By Authorization header

All API requests require an authentication or an API token. You can provide the credentials by using the Authorization header in the request:

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer 0424bd59b807674191e7d77572075f33'
```

Attention:
If you are using Apache, you may need to change the default Apache configuration in `/etc/apache2/apache2.conf` by adding the following line:


```
SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
```

Otherwise, Apache might not send the Authorization header in the request.

Since Zabbix 6.4.21, the Authorization header is supported in cross-origin requests (CORS).

By `auth` property

An API request can be authorized by the `auth` property.

Attention:
Note that the `auth` property is deprecated. It will be removed in the future releases.

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"host.get","params":{"output":["hostid"]},"auth":"0424bd59b807674191e7
```

By Zabbix cookie

A *"zbx_session"* cookie is used to authorize an API request from Zabbix UI performed using JavaScript (from a module or a custom widget).

Retrieving hosts Now you have a valid user authentication token that can be used to access the data in Zabbix. For example, you can use the host.get method to retrieve the IDs, host names and interfaces of all the configured hosts:

Request:

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data @data.json
```

Note:
`data.json` is a file that contains a JSON query. Instead of a file, you can pass the query in the `--data` argument.

data.json

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "hostid",
            "host"
        ],
        "selectInterfaces": [
            "interfaceid",
            "ip"
        ]
    },
    "id": 2
}
```

The response object will contain the requested data about the hosts:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "interfaces": [
                {
                    "interfaceid": "1",
                    "ip": "127.0.0.1"
                }
            ]
        }
    ],
    "id": 2
}
```

Note:
For performance reasons it is always recommended to list the object properties you want to retrieve. Thus, you will avoid retrieving everything.

Creating a new item Now, create a new item on the host "Zabbix server" using the data you have obtained from the previous `host.get` request. This can be done using the item.create method:

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"item.create","params":{"name":"Free disk space on /home/joe/","key_":
```

A successful response will contain the ID of the newly created item, which can be used to reference the item in the following requests:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 3
}
```

Note:
The `item.create` method as well as other *create methods* can also accept arrays of objects and create multiple items with one API call.

Creating multiple triggers Thus, if *create methods* accept arrays, you can add multiple triggers, for example, this one:

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"trigger.create","params":[{"description":"Processor load is too high
```

The successful response will contain the IDs of the newly created triggers:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17369",
            "17370"
        ]
    },
    "id": 4
}
```

Updating an item Enable an item by setting its status to "0":

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"item.update","params":{"itemid":"10092","status":0},"id":5}'
```

The successful response will contain the ID of the updated item:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "10092"
        ]
    },
    "id": 5
}
```

Note:
The `item.update` method as well as other *update methods* can also accept arrays of objects and update multiple items with one API call.

Updating multiple triggers Enable multiple triggers by setting their status to "0":

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"trigger.update","params":[{"triggerid":"13938","status":0},{"triggeri
```

The successful response will contain the IDs of the updated triggers:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938",
            "13939"
        ]
    },
    "id": 6
}
```

Note:
This is the preferred method of updating. Some API methods, such as the `host.massupdate` allow to write a simpler code. However, it is not recommended to use these methods as they will be removed in the future releases.

Error handling Up to the present moment, everything you have tried has worked fine. But what would happen if you tried making an incorrect call to the API? Try to create another host by calling host.create but omitting the mandatory `groups` parameter:

```
curl --request POST \
  --url 'https://example.com/zabbix/api_jsonrpc.php' \
  --header 'Authorization: Bearer ${AUTHORIZATION_TOKEN}' \
  --header 'Content-Type: application/json-rpc' \
  --data '{"jsonrpc":"2.0","method":"host.create","params":{"host":"Linux server","interfaces":[{"type":1,
```

The response will then contain an error message:

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32602,
        "message": "Invalid params.",
        "data": "No groups for host \"Linux server\"."
    },
    "id": 7
}
```

If an error has occurred, instead of the `result` property, the response object will contain the `error` property with the following data:

- `code` - an error code;
- `message` - a short error summary;
- `data` - a more detailed error message.

Errors can occur in various cases, such as, using incorrect input values, a session timeout or trying to access non-existing objects. Your application should be able to gracefully handle these kinds of errors.

API versions To simplify API versioning, since Zabbix 2.0.4, the version of the API matches the version of Zabbix itself. You can use the apiinfo.version method to find out the version of the API you are working with. This can be useful for adjusting your application to use version-specific features.

Zabbix guarantees feature backward compatibility inside a major version. When making backward incompatible changes between major releases, Zabbix usually leaves the old features as deprecated in the next release, and only removes them in the release after that. Occasionally, Zabbix may remove features between major releases without providing any backward compatibility. It is important that you never rely on any deprecated features and migrate to newer alternatives as soon as possible.

Note:
You can follow all the changes made to the API in the API changelog.

Further reading Now, you have enough knowledge to start working with the Zabbix API, however, do not stop here. For further reading you are advised to have a look at the list of available APIs.

Method reference

This section provides an overview of the functions provided by the Zabbix API and will help you find your way around the available classes and methods.

Monitoring The Zabbix API allows you to access history and other data gathered during monitoring.

Dashboards

Manage dashboards and make scheduled reports based on them.

Dashboard API | Template dashboard API | Report API

High availability cluster

Retrieve a list of server nodes and their status.

High availability cluster API

History

Retrieve historical values gathered by Zabbix monitoring processes for presentation or further processing.

History API

Trends

Retrieve trend values calculated by Zabbix server for presentation or further processing.

Trend API

Events

Retrieve events generated by triggers, network discovery and other Zabbix systems for more flexible situation management or third-party tool integration.

Event API

Problems

Retrieve problems according to the given parameters.

Problem API

Maps

Configure maps to create detailed dynamic representations of your IT infrastructure.

Map API

Tasks

Interact with Zabbix server task manager, creating tasks and retrieving response.

Task API

Services The Zabbix API allows you to access data gathered during service monitoring.

Service Level Agreement

Define Service Level Objectives (SLO), retrieve detailed Service Level Indicators (SLI) information about service performance.

SLA API

Services

Manage services for service-level monitoring and retrieve detailed SLA information about any service.

Service API

Data collection The Zabbix API allows you to manage the configuration of your monitoring system.

Hosts and host groups

Manage host groups, hosts and everything related to them, including host interfaces, host macros and maintenance periods.

Host API | Host group API | Host interface API | User macro API | Value map API | Maintenance API

Items

Define items to monitor.

Item API

Triggers

Configure triggers to notify you about problems in your system. Manage trigger dependencies.

Trigger API

Graphs

Edit graphs or separate graph items for better presentation of the gathered data.

Graph API | Graph item API

Templates and template groups

Manage templates and link them to hosts or other templates.

Template API | Template group API | Value map API

Low-level discovery

Configure low-level discovery rules as well as item, trigger and graph prototypes to monitor dynamic entities.

LLD rule API | Item prototype API | Trigger prototype API | Graph prototype API | Host prototype API

Event correlation

Create custom event correlation rules.

Correlation API

Network discovery

Manage network-level discovery rules to automatically find and monitor new hosts. Gain full access to information about discovered services and hosts.

Discovery rule API | Discovery check API | Discovered host API | Discovered service API

Export and import

Export and import Zabbix configuration data for configuration backups, migration or large-scale configuration updates.

Configuration API

Web monitoring

Configure web scenarios to monitor your web applications and services.

Web scenario API

Alerts The Zabbix API allows you to manage the actions and alerts of your monitoring system.

Actions and alerts

Define actions and operations to notify users about certain events or automatically execute remote commands. Gain access to information about generated alerts and their receivers.

Action API | Alert API

Media types

Configure media types and multiple ways users will receive alerts.

Media type API

Scripts

Configure and execute scripts to help you with your daily tasks.

Script API

Users The Zabbix API allows you to manage users of your monitoring system.

Users and user groups

Add users that will have access to Zabbix, assign them to user groups and grant permissions. Make roles for granular management of user rights.

User API | User group API | User directory API | User role API

API Tokens

Manage authorization tokens.

Token API

Authentication

Change authentication configuration options.

Authentication API

Administration With the Zabbix API you can change administration settings of your monitoring system.

General

Change certain global configuration options.

Autoregistration API | Icon map API | Image API | Settings API | Regular expression API | Module API | Connector API

Audit log

Track configuration changes each user has done.

Audit log API

Housekeeping

Configure housekeeping.

Housekeeping API

Proxies

Manage the proxies used in your distributed monitoring setup.

Proxy API

Macros

Manage macros.

User macro API

API information Retrieve the version of the Zabbix API so that your application could use version-specific features.

API info API

Action

This class is designed to work with actions.

Object references:

- Action
- Action condition
- Action operation

Available methods:

- action.create - create new actions
- action.delete - delete actions

- action.get - retrieve actions
- action.update - update actions

> Action object

The following objects are directly related to the `action` API.

Action

The action object has the following properties.

Property	Type	Description
actionid	string	ID of the action. Property behavior: - *read-only* - *required* for update operations
esc_period	string	Default operation step duration. Must be at least 60 seconds. Accepts seconds, time unit with suffix, or a user macro. Property behavior: - *supported* if `eventsource` is set to ”event created by a trigger”, ”internal event”, or ”event created on service status update”
eventsource	integer	Type of events that the action will handle. Refer to the event `source` property for a list of supported event types. Property behavior: - *constant* - *required* for create operations
name	string	Name of the action. Property behavior: - *required* for create operations
status	integer	Whether the action is enabled or disabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
pause_symptoms	integer	Whether to pause escalation if event is a symptom event. Possible values: 0 - Don’t pause escalation for symptom problems; 1 - *(default)* Pause escalation for symptom problems. Property behavior: - *supported* if `eventsource` is set to ”event created by a trigger”
pause_suppressed	integer	Whether to pause escalation during maintenance periods or not. Possible values: 0 - Don’t pause escalation; 1 - *(default)* Pause escalation. Property behavior: - *supported* if `eventsource` is set to ”event created by a trigger”

Property	Type	Description
notify_if_canceled	integer	Whether to notify when escalation is canceled. Possible values: 0 - Don't notify when escalation is canceled; 1 - *(default)* Notify when escalation is canceled. Property behavior: - *supported* if `eventsource` is set to "event created by a trigger"

Action operation

The action operation object defines an operation that will be performed when an action is executed. It has the following properties.

Property	Type	Description
operationtype	integer	Type of operation. Possible values: 0 - send message; 1 - global script; 2 - add host; 3 - remove host; 4 - add to host group; 5 - remove from host group; 6 - link to template; 7 - unlink from template; 8 - enable host; 9 - disable host; 10 - set host inventory mode. Possible values if `eventsource` of Action object is set to "event created by a trigger" or "event created on service status update": 0 - "send message"; 1 - "global script". Possible values if `eventsource` of Action object is set to "internal event": 0 - "send message". Property behavior: - *required*
esc_period	string	Duration of an escalation step in seconds. Must be greater than 60 seconds. Accepts seconds, time unit with suffix, or a user macro. If set to 0 or 0s, the default action escalation period will be used. Default: 0s. Property behavior: - *supported* if `eventsource` of Action object is set to "event created by a trigger", "internal event", or "event created on service status update"
esc_step_from	integer	Step to start escalation from. Default: 1. Property behavior: - *supported* if `eventsource` of Action object is set to "event created by a trigger", "internal event", or "event created on service status update"

Property	Type	Description
esc_step_to	integer	Step to end escalation at. Default: 1. Property behavior: - *supported* if `eventsource` of Action object is set to ”event created by a trigger”, ”internal event”, or ”event created on service status update”
evaltype	integer	Operation condition evaluation method. Possible values: 0 - *(default)* AND / OR; 1 - AND; 2 - OR.
opcommand	object	Global script to execute. The global script must have the `scriptid` property defined. Property behavior: - *required* if `operationtype` is set to ”global script”
opcommand_grp	array	Host groups to run global scripts on. The host groups must have the `groupid` property defined. Property behavior: - *required* if `operationtype` is set to ”global script” and `opcommand_hst` is not set
opcommand_hst	array	Host to run global scripts on. The hosts must have the `hostid` property defined. Property behavior: - *required* if `operationtype` is set to ”global script” and `opcommand_grp` is not set
opconditions	array	Operation conditions used for trigger actions. The operation condition object is described in detail below.
opgroup	array	Host groups to add hosts to. The host groups must have the `groupid` property defined. Property behavior: - *required* if `operationtype` is set to ”add to host group” or ”remove from host group”
opmessage	object	Object containing the data about the message sent by the operation. The operation message object is described in detail below. Property behavior: - *required* if `operationtype` is set to ”send message”
opmessage_grp	array	User groups to send messages to. The user groups must have the `usrgrpid` property defined. Property behavior: - *required* if `operationtype` is set to ”send message” and `opmessage_usr` is not set

Property	Type	Description
opmessage_usr	array	Users to send messages to. The users must have the `userid` property defined. Property behavior: - *required* if `operationtype` is set to ”send message” and `opmessage_grp` is not set
optemplate	array	Templates to link the hosts to. The templates must have the `templateid` property defined. Property behavior: - *required* if `operationtype` is set to ”link to template” or ”unlink from template”
opinventory	object	Inventory mode set host to. The inventory must have the `inventory_mode` property defined. Property behavior: - *required* if `operationtype` is set to ”set host inventory mode”

Action operation message

The operation message object contains data about the message that will be sent by the operation. It has the following properties.

Property	Type	Description
default_msg	integer	Whether to use the default action message text and subject. Possible values: 0 - use the data from the operation; 1 - *(default)* use the data from the media type.
mediatypeid	string	ID of the media type that will be used to send the message. Property behavior: - *supported* if `operationtype` of Action operation object, Action recovery operation object, or Action update operation object is set to ”send message”, or if `operationtype` of Action update operation object is set to ”notify all involved”
message	string	Operation message text. Property behavior: - *supported* if `default_msg` is set to ”use the data from the operation”
subject	string	Operation message subject. Property behavior: - *supported* if `default_msg` is set to ”use the data from the operation”

Action operation condition

The action operation condition object defines a condition that must be met to perform the current operation. It has the following properties.

Property	Type	Description
conditiontype	integer	Type of condition. Possible values: 14 - event acknowledged. Property behavior: - *required*
value	string	Value to compare with. Property behavior: - *required*
operator	integer	Condition operator. Possible values: 0 - *(default)* =

The following operators and values are supported for each operation condition type.

Condition	Condition name	Supported operators	Expected value
14	Event acknowledged	=	Whether the event is acknowledged. Possible values: 0 - not acknowledged; 1 - acknowledged.

Action recovery operation

The action recovery operation object defines an operation that will be performed when a problem is resolved. Recovery operations are possible **only** for trigger, internal and service actions. It has the following properties.

Property	Type	Description
operationtype	integer	Type of operation. Possible values if `eventsource` of Action object is set to ”event created by a trigger” or ”event created on service status update”: 0 - send message; 1 - global script; 11 - notify all involved. Possible values if `eventsource` of Action object is set to ”internal event”: 0 - send message; 11 - notify all involved. Property behavior: - *required*
opcommand	object	Global script to execute. The global script must have the `scriptid` property defined. Property behavior: - *required* if `operationtype` is set to ”global script”

Property	Type	Description
opcommand_grp	array	Host groups to run global scripts on. The host groups must have the `groupid` property defined. Property behavior: - *required* if `eventsource` of Action object is set to ”event created by a trigger”, and `operationtype` is set to ”global script”, and `opcommand_hst` is not set
opcommand_hst	array	Host to run global scripts on. The hosts must have the `hostid` property defined. Property behavior: - *required* if `eventsource` of Action object is set to ”event created by a trigger”, and `operationtype` is set to ”global script”, and `opcommand_grp` is not set
opmessage	object	Object containing the data about the message sent by the recovery operation. The operation message object is described in detail above. Property behavior: - *required* if `operationtype` is set to ”send message”
opmessage_grp	array	User groups to send messages to. The user groups must have the `usrgrpid` property defined. Property behavior: - *required* if `operationtype` is set to ”send message” and `opmessage_usr` is not set
opmessage_usr	array	Users to send messages to. The users must have the `userid` property defined. Property behavior: - *required* if `operationtype` is set to ”send message” and `opmessage_grp` is not set

Action update operation

The action update operation object defines an operation that will be performed when a problem is updated (commented upon, acknowledged, severity changed, or manually closed). Update operations are possible **only** for trigger and service actions. It has the following properties.

Property	Type	Description
operationtype	integer	Type of operation. Possible values: 0 - send message; 1 - global script; 12 - notify all involved. Property behavior: - *required*
opcommand	object	Global script to execute. The global script must have the `scriptid` property defined. Property behavior: - *required* if `operationtype` is set to ”global script”

Property	Type	Description
opcommand_grp	array	Host groups to run global scripts on. The host groups must have the `groupid` property defined. Property behavior: - *required* if `eventsource` of Action object is set to ”event created by a trigger”, and `operationtype` is set to ”global script”, and `opcommand_hst` is not set
opcommand_hst	array	Host to run global scripts on. The hosts must have the `hostid` property defined. Property behavior: - *required* if `eventsource` of Action object is set to ”event created by a trigger”, and `operationtype` is set to ”global script”, and `opcommand_grp` is not set
opmessage	object	Object containing the data about the message sent by the update operation. The operation message object is described in detail above.
opmessage_grp	array	User groups to send messages to. The user groups must have the `usrgrpid` property defined. Property behavior: - *required* if `operationtype` is set to ”send message” and `opmessage_usr` is not set
opmessage_usr	array	Users to send messages to. The users must have the `userid` property defined. Property behavior: - *required* if `operationtype` is set to ”send message” and `opmessage_grp` is not set

Action filter

The action filter object defines a set of conditions that must be met to perform the configured action operations. It has the following properties.

Property	Type	Description
conditions	array	Set of filter conditions to use for filtering results. Property behavior: - *required*
evaltype	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression. Property behavior: - *required*

Property	Type	Description
eval_formula	string	Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its `formulaid`. The value of `eval_formula` is equal to the value of `formula` for filters with a custom expression. Property behavior: - *read-only*
formula	string	User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its `formulaid`. The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. Property behavior: - *required* if `evaltype` is set to "custom expression"

Action filter condition

The action filter condition object defines a specific condition that must be checked before running the action operations.

Property	Type	Description
conditiontype	integer	Type of condition. Possible values if `eventsource` of Action object is set to ”event created by a trigger”: 0 - host group; 1 - host; 2 - trigger; 3 - event name; 4 - trigger severity; 6 - time period; 13 - host template; 16 - problem is suppressed; 25 - event tag; 26 - event tag value. Possible values if `eventsource` of Action object is set to ”event created by a discovery rule”: 7 - host IP; 8 - discovered service type; 9 - discovered service port; 10 - discovery status; 11 - uptime or downtime duration; 12 - received value; 18 - discovery rule; 19 - discovery check; 20 - proxy; 21 - discovery object. Possible values if `eventsource` of Action object is set to ”event created by active agent autoregistration”: 20 - proxy; 22 - host name; 24 - host metadata. Possible values if `eventsource` of Action object is set to ”internal event”: 0 - host group; 1 - host; 13 - host template; 23 - event type; 25 - event tag; 26 - event tag value. Possible values if `eventsource` of Action object is set to ”event created on service status update”: 25 - event tag; 26 - event tag value; 27 - service; 28 - service name. Property behavior: - *required*
value	string	Value to compare with. Property behavior: - *required*

Property	Type	Description
value2	string	Secondary value to compare with. Property behavior: - *required* if `eventsource` of Action object is set to ”event created by a trigger”, `conditiontype` is set to any possible value for trigger actions, and the type of condition (see below) is ”26” - *required* if `eventsource` of Action object is set to ”internal event”, `conditiontype` is set to any possible value for internal actions, and the type of condition (see below) is ”26” - *required* if `eventsource` of Action object is set to ”event created on service status update”, `conditiontype` is set to any possible value for service actions, and the type of condition (see below) is ”26”
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
operator	integer	Condition operator. Possible values: 0 - *(default)* equals; 1 - does not equal; 2 - contains; 3 - does not contain; 4 - in; 5 - is greater than or equals; 6 - is less than or equals; 7 - not in; 8 - matches; 9 - does not match; 10 - Yes; 11 - No.

Note:
To better understand how to use filters with various types of expressions, see examples on the action.get and action.create method pages.

The following operators and values are supported for each condition type.

Condition	Condition name	Supported operators	Expected value
0	Host group	equals, does not equal	Host group ID.
1	Host	equals, does not equal	Host ID.
2	Trigger	equals, does not equal	Trigger ID.
3	Event name	contains, does not contain	Event name.
4	Trigger severity	equals, does not equal, is greater than or equals, is less than or equals	Trigger severity. Refer to the trigger `severity` property for a list of supported trigger severities.
5	Trigger value	equals	Trigger value. Refer to the trigger `value` property for a list of supported trigger values.
6	Time period	in, not in	Time when the event was triggered as a time period.
7	Host IP	equals, does not equal	One or several IP ranges to check, separated by commas. Refer to the network discovery configuration section for more information on supported formats of IP ranges.

Condition	Condition name	Supported operators	Expected value
8	Discovered service type	equals, does not equal	Type of discovered service. The type of service matches the type of the discovery check used to detect the service. Refer to the discovery check `type` property for a list of supported types.
9	Discovered service port	equals, does not equal	One or several port ranges, separated by commas.
10	Discovery status	equals	Status of a discovered object. Possible values: 0 - host or service up; 1 - host or service down; 2 - host or service discovered; 3 - host or service lost.
11	Uptime or downtime duration	is greater than or equals, is less than or equals	Time indicating how long has the discovered object been in the current status in seconds.
12	Received values	equals, does not equal, is greater than or equals, is less than or equals, contains, does not contain	Value returned when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.
13	Host template	equals, does not equal	Linked template ID.
16	Problem is suppressed	Yes, No	No value required: using the ”Yes” operator means that problem must be suppressed, ”No” - not suppressed.
18	Discovery rule	equals, does not equal	ID of the discovery rule.
19	Discovery check	equals, does not equal	ID of the discovery check.
20	Proxy	equals, does not equal	ID of the proxy.
21	Discovery object	equals	Type of object that triggered the discovery event. Possible values: 1 - discovered host; 2 - discovered service.
22	Host name	contains, does not contain, matches, does not match	Host name. Using a regular expression is supported for operators *matches* and *does not match* in autoregistration conditions.
23	Event type	equals	Specific internal event. Possible values: 0 - item in ”not supported” state; 1 - item in ”normal” state; 2 - LLD rule in ”not supported” state; 3 - LLD rule in ”normal” state; 4 - trigger in ”unknown” state; 5 - trigger in ”normal” state.
24	Host metadata	contains, does not contain, matches, does not match	Metadata of the auto-registered host. Using a regular expression is supported for operators *matches* and *does not match*.
25	Tag	equals, does not equal, contains, does not contain	Event tag.

Condition	Condition name	Supported operators	Expected value
26	Tag value	equals, does not equal, contains, does not contain	Event tag value.
27	Service	equals, does not equal	Service ID.
28	Service name	equals, does not equal	Service name.

action.create

Description

`object action.create(object/array actions)`

This method allows to create new actions.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Actions to create.

Additionally to the standard action properties, the method accepts the following parameters.

Parameter	Type	Description
filter	object	Action filter object for the action.
operations	array	Action operations to create for the action.
recovery_operations	array	Action recovery operations to create for the action.
update_operations	array	Action update operations to create for the action.

Return values

`(object)` Returns an object containing the IDs of the created actions under the `actionids` property. The order of the returned IDs matches the order of the passed actions.

Examples

Create a trigger action

Create a trigger action that will begin once a trigger (with the word ”memory” in its name) from host ”10084” goes into a PROBLEM state. The action will have 4 configured operations. The first and immediate operation will send a message to all users in user group ”7” via media type ”1”. If the event is not resolved in 30 minutes, the second operation will run script ”5” (script with scope ”Action operation”) on all hosts in group ”2”. If the event is resolved, a recovery operation will notify all users who received any messages regarding the problem. If the event is updated, an acknowledge/update operation will notify (with a custom subject and message) all users who received any messages regarding the problem.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "esc_period": "30m",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 1,
```

```
                "operator": 0,
                "value": "10084"
            },
            {
                "conditiontype": 3,
                "operator": 2,
                "value": "memory"
            }
        ]
    },
    "operations": [
        {
            "operationtype": 0,
            "esc_step_from": 1,
            "esc_step_to": 1,
            "opmessage_grp": [
                {
                    "usrgrpid": "7"
                }
            ],
            "opmessage": {
                "default_msg": 1,
                "mediatypeid": "1"
            }
        },
        {
            "operationtype": 1,
            "esc_step_from": 2,
            "esc_step_to": 2,
            "opconditions": [
                {
                    "conditiontype": 14,
                    "operator": 0,
                    "value": "0"
                }
            ],
            "opcommand_grp": [
                {
                    "groupid": "2"
                }
            ],
            "opcommand": {
                "scriptid": "5"
            }
        }
    ],
    "recovery_operations": [
        {
            "operationtype": "11",
            "opmessage": {
                "default_msg": 1
            }
        }
    ],
    "update_operations": [
        {
            "operationtype": "12",
            "opmessage": {
                "default_msg": 0,
                "message": "Custom update operation message body",
                "subject": "Custom update operation message subject"
            }
```

```
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "17"
        ]
    },
    "id": 1
}
```

Create a discovery action

Create a discovery action that will link discovered hosts to template ”10001”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Discovery action",
        "eventsource": 1,
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "conditiontype": 21,
                    "operator": 0,
                    "value": "1"
                },
                {
                    "conditiontype": 10,
                    "operator": 0,
                    "value": "2"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 6,
                "optemplate": [
                    {
                        "templateid": "10001"
                    }
                ]
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
```

```
        ]
    },
    "id": 1
}
```

Using a custom expression filter

Create a trigger action that uses a custom expression - ”A and (B or C)” - for evaluating action conditions. Once a trigger with a severity higher or equal to ”Warning” from host ”10084” or host ”10106” goes into a PROBLEM state, the action will send a message to all users in user group ”7” via media type ”1”. The formula IDs ”A”, ”B” and ”C” have been chosen arbitrarily.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Trigger action",
        "eventsource": 0,
        "esc_period": "15m",
        "filter": {
            "evaltype": 3,
            "formula": "A and (B or C)",
            "conditions": [
                {
                    "conditiontype": 4,
                    "operator": 5,
                    "value": "2",
                    "formulaid": "A"
                },
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10084",
                    "formulaid": "B"
                },
                {
                    "conditiontype": 1,
                    "operator": 0,
                    "value": "10106",
                    "formulaid": "C"
                }
            ]
        },
        "operations": [
            {
                "operationtype": 0,
                "esc_step_from": 1,
                "esc_step_to": 1,
                "opmessage_grp": [
                    {
                        "usrgrpid": "7"
                    }
                ],
                "opmessage": {
                    "default_msg": 1,
                    "mediatypeid": "1"
                }
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
    "id": 1
}
```

Create agent autoregistration rule

Create an autoregistration action that adds a host to host group ”2” when the host name contains ”SRV” or metadata contains ”AlmaLinux”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.create",
    "params": {
        "name": "Register Linux servers",
        "eventsource": "2",
        "filter": {
            "evaltype": "2",
            "conditions": [
                {
                    "conditiontype": "22",
                    "operator": "2",
                    "value": "SRV"
                },
                {
                    "conditiontype": "24",
                    "operator": "2",
                    "value": "AlmaLinux"
                }
            ]
        },
        "operations": [
            {
                "operationtype": "4",
                "opgroup": [
                    {
                        "groupid": "2"
                    }
                ]
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            19
        ]
    },
    "id": 1
}
```

See also

- Action filter
- Action operation
- Script

Source

CAction::create() in *ui/include/classes/api/services/CAction.php*.

action.delete

Description

`object action.delete(array actionIds)`

This method allows to delete actions.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the actions to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted actions under the `actionids` property.

Examples

Delete multiple actions

Delete two actions.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.delete",
    "params": [
        "17",
        "18"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "17",
            "18"
        ]
    },
    "id": 1
}
```

Source

CAction::delete() in *ui/include/classes/api/services/CAction.php*.

action.get

Description

`integer/array action.get(object parameters)`

The method allows to retrieve actions according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
actionids	string/array	Return only actions with the given IDs.
groupids	string/array	Return only actions that use the given host groups in action conditions.
hostids	string/array	Return only actions that use the given hosts in action conditions.
triggerids	string/array	Return only actions that use the given triggers in action conditions.
mediatypeids	string/array	Return only actions that use the given media types to send messages.
usrgrpids	string/array	Return only actions that are configured to send messages to the given user groups.
userids	string/array	Return only actions that are configured to send messages to the given users.
scriptids	string/array	Return only actions that are configured to run the given scripts.
selectFilter	query	Return a filter property with the action condition filter.
selectOperations	query	Return an operations property with action operations.
selectRecoveryOperations	query	Return a recovery_operations property with action recovery operations.
selectUpdateOperations	query	Return an update_operations property with action update operations.
sortfield	string/array	Sort the result by the given properties. Possible values: `actionid`, `name`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in the reference commentary.
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve trigger actions

Retrieve all configured trigger actions together with action conditions and operations.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectRecoveryOperations": "extend",
        "selectUpdateOperations": "extend",
        "selectFilter": "extend",
```

```
        "filter": {
            "eventsource": 0
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "actionid": "3",
            "name": "Report problems to Zabbix administrators",
            "eventsource": "0",
            "status": "1",
            "esc_period": "1h",
            "pause_suppressed": "1",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [],
                "eval_formula": ""
            },
            "operations": [
                {
                    "operationid": "3",
                    "actionid": "3",
                    "operationtype": "0",
                    "esc_period": "0",
                    "esc_step_from": "1",
                    "esc_step_to": "1",
                    "evaltype": "0",
                    "opconditions": [],
                    "opmessage": [
                        {
                            "default_msg": "1",
                            "subject": "",
                            "message": "",
                            "mediatypeid" => "0"
                        }
                    ],
                    "opmessage_grp": [
                        {
                            "usrgrpid": "7"
                        }
                    ]
                }
            ],
            "recovery_operations": [
                {
                    "operationid": "7",
                    "actionid": "3",
                    "operationtype": "11",
                    "evaltype": "0",
                    "opconditions": [],
                    "opmessage": {
                        "default_msg": "0",
                        "subject": "{TRIGGER.STATUS}: {TRIGGER.NAME}",
                        "message": "Trigger: {TRIGGER.NAME}\r\nTrigger status: {TRIGGER.STATUS}\r\nTrigger
                        "mediatypeid": "0"
                    }
```

```
                }
            ],
            "update_operations": [
                {
                    "operationid": "31",
                    "operationtype": "12",
                    "evaltype": "0",
                    "opmessage": {
                        "default_msg": "1",
                        "subject": "",
                        "message": "",
                        "mediatypeid": "0"
                    }
                },
                {
                    "operationid": "32",
                    "operationtype": "0",
                    "evaltype": "0",
                    "opmessage": {
                        "default_msg": "0",
                        "subject": "Updated: {TRIGGER.NAME}",
                        "message": "{USER.FULLNAME} updated problem at {EVENT.UPDATE.DATE} {EVENT.UPDATE.T
                        "mediatypeid": "1"
                    },
                    "opmessage_grp": [
                        {
                            "usrgrpid": "7"
                        }
                    ],
                    "opmessage_usr": []
                },
                {
                    "operationid": "33",
                    "operationtype": "1",
                    "evaltype": "0",
                    "opcommand": {
                        "scriptid": "3"
                    },
                    "opcommand_hst": [
                        {
                            "hostid": "10084"
                        }
                    ],
                    "opcommand_grp": []
                }
            ]
        }
    ],
    "id": 1
}
```


Retrieve discovery actions

Retrieve all configured discovery actions together with action conditions and operations. The filter uses the ”and” evaluation type, so the `formula` property is empty and `eval_formula` is generated automatically.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "action.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
```

```
        "selectFilter": "extend",
        "filter": {
            "eventsource": 1
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "actionid": "2",
            "name": "Auto discovery. Linux servers.",
            "eventsource": "1",
            "status": "1",
            "esc_period": "0s",
            "pause_suppressed": "1",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [
                    {
                        "conditiontype": "10",
                        "operator": "0",
                        "value": "0",
                        "value2": "",
                        "formulaid": "B"
                    },
                    {
                        "conditiontype": "8",
                        "operator": "0",
                        "value": "9",
                        "value2": "",
                        "formulaid": "C"
                    },
                    {
                        "conditiontype": "12",
                        "operator": "2",
                        "value": "Linux",
                        "value2": "",
                        "formulaid": "A"
                    }
                ],
                "eval_formula": "A and B and C"
            },
            "operations": [
                {
                    "operationid": "1",
                    "actionid": "2",
                    "operationtype": "6",
                    "esc_period": "0s",
                    "esc_step_from": "1",
                    "esc_step_to": "1",
                    "evaltype": "0",
                    "opconditions": [],
                    "optemplate": [
                        {
                            "templateid": "10001"
                        }
                    ]
```

```
                },
                {
                    "operationid": "2",
                    "actionid": "2",
                    "operationtype": "4",
                    "esc_period": "0s",
                    "esc_step_from": "1",
                    "esc_step_to": "1",
                    "evaltype": "0",
                    "opconditions": [],
                    "opgroup": [
                        {
                            "groupid": "2"
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Action filter
- Action operation

Source

CAction::get() in *ui/include/classes/api/services/CAction.php*.

action.update

Description

`object action.update(object/array actions)`

This method allows to update existing actions.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Action properties to be updated.

The `actionid` property must be defined for each action, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard action properties, the method accepts the following parameters.

Parameter	Type	Description
filter	object	Action filter object to replace the current filter.
operations	array	Action operations to replace existing operations.
recovery_operations	array	Action recovery operations to replace existing recovery operations. Parameter behavior: - *supported* if `eventsource` of Action object is set to ”event created by a trigger”, ”internal event”, or ”event created on service status update”

Parameter	Type	Description
update_operations	array	Action update operations to replace existing update operations. Parameter behavior: - *supported* if `eventsource` of Action object is set to ”event created by a trigger” or ”event created on service status update”

Return values

`(object)` Returns an object containing the IDs of the updated actions under the `actionids` property.

Examples

Disable action

Disable an action, that is, set its status to ”1”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "action.update",
    "params": {
        "actionid": "2",
        "status": "1"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "2"
        ]
    },
    "id": 1
}
```

See also

- Action filter
- Action operation

Source

CAction::update() in *ui/include/classes/api/services/CAction.php*.

Alert

This class is designed to work with alerts.

Object references:

- Alert

Available methods:

- alert.get - retrieve alerts

> Alert object

The following objects are directly related to the `alert` API.

Alert

> **Note:**
> Alerts are created by Zabbix server and cannot be modified via the API.

The alert object contains information about whether certain action operations have been executed successfully. It has the following properties.

Property	Type	Description
alertid	string	ID of the alert.
actionid	string	ID of the action that generated the alert.
alerttype	integer	Alert type. Possible values: 0 - message; 1 - remote command.
clock	timestamp	Time when the alert was generated.
error	string	Error text if there are problems sending a message or running a command.
esc_step	integer	Action escalation step during which the alert was generated.
eventid	string	ID of the event that triggered the action.
mediatypeid	string	ID of the media type that was used to send the message.
message	text	Message text. Property behavior: - *supported* if `alerttype` is set to ”message”
retries	integer	Number of times Zabbix tried to send the message.
sendto	string	Address, user name or other identifier of the recipient. Property behavior: - *supported* if `alerttype` is set to ”message”
status	integer	Status indicating whether the action operation has been executed successfully. Possible values if `alerttype` is set to ”message”: 0 - message not sent; 1 - message sent; 2 - failed after a number of retries; 3 - new alert is not yet processed by alert manager. Possible values if `alerttype` is set to ”remote command”: 0 - command not run; 1 - command run; 2 - tried to run the command on Zabbix agent, but it was unavailable.
subject	string	Message subject. Property behavior: - *supported* if `alerttype` is set to ”message”
userid	string	ID of the user that the message was sent to.
p_eventid	string	ID of problem event, which generated the alert.
acknowledgeid	string	ID of acknowledgment, which generated the alert.

alert.get

Description

`integer/array alert.get(object parameters)`

The method allows to retrieve alerts according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
alertids	string/array	Return only alerts with the given IDs.
actionids	string/array	Return only alerts generated by the given actions.
eventids	string/array	Return only alerts generated by the given events.
groupids	string/array	Return only alerts generated by objects from the given host groups.
hostids	string/array	Return only alerts generated by objects from the given hosts.
mediatypeids	string/array	Return only message alerts that used the given media types.
objectids	string/array	Return only alerts generated by the given objects
userids	string/array	Return only message alerts that were sent to the given users.
eventobject	integer	Return only alerts generated by events related to objects of the given type. See event "object" for a list of supported object types. Default: 0 - trigger.
eventsource	integer	Return only alerts generated by events of the given type. See event "source" for a list of supported event types. Default: 0 - trigger events.
time_from	timestamp	Return only alerts that have been generated after the given time.
time_till	timestamp	Return only alerts that have been generated before the given time.
selectHosts	query	Return a hosts property with data of hosts that triggered the action operation.
selectMediatypes	query	Return a mediatypes property with an array of the media types that were used for the message alert.
selectUsers	query	Return a users property with an array of the users that the message was addressed to.
sortfield	string/array	Sort the result by the given properties. Possible values: `alertid`, `clock`, `eventid`, `mediatypeid`, `sendto`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve alerts by action ID

Retrieve all alerts generated by action ”3”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "alert.get",
    "params": {
        "output": "extend",
        "actionids": "3"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "alertid": "1",
            "actionid": "3",
            "eventid": "21243",
            "userid": "1",
            "clock": "1362128008",
            "mediatypeid": "1",
            "sendto": "support@company.com",
            "subject": "PROBLEM: Zabbix agent on Linux server is unreachable for 5 minutes: ",
            "message": "Trigger: Zabbix agent on Linux server is unreachable for 5 minutes: \nTrigger stat
            "status": "0",
            "retries": "3",
            "error": "",
            "esc_step": "1",
            "alerttype": "0",
            "p_eventid": "0",
            "acknowledgeid": "0"
        }
    ],
    "id": 1
}
```

See also

- Host
- Media type
- User

Source

CAlert::get() in *ui/include/classes/api/services/CAlert.php*.

API info

This class is designed to retrieve meta information about the API.

Available methods:

- apiinfo.version - retrieving the version of the Zabbix API

apiinfo.version

Description

`string apiinfo.version(array)`

This method allows to retrieve the version of the Zabbix API.

> **Attention:**
> This method is only available to unauthenticated users and must be called without the `auth` parameter in the JSON-RPC request.

Parameters

`(array)` The method accepts an empty array.

Return values

`(string)` Returns the version of the Zabbix API.

> **Note:**
> Starting from Zabbix 2.0.4 the version of the API matches the version of Zabbix.

Examples

Retrieving the version of the API

Retrieve the version of the Zabbix API.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "apiinfo.version",
    "params": [],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "6.4.0",
    "id": 1
}
```

Source

CAPIInfo::version() in *ui/include/classes/api/services/CAPIInfo.php*.

Audit log

This class is designed to work with audit log.

Object references:

- Audit log object

Available methods:

- auditlog.get - retrieve audit log records

> Audit log object

The following objects are directly related to the `auditlog` API.

Audit log

The audit log object contains information about user actions. It has the following properties.

Property	Type	Description
auditid	string	ID of audit log entry. Generated using CUID algorithm.
userid	string	Audit log entry author userid.
username	string	Audit log entry author username.
clock	timestamp	Audit log entry creation timestamp.
ip	string	Audit log entry author IP address.
action	integer	Audit log entry action. Possible values: 0 - Add; 1 - Update; 2 - Delete; 4 - Logout; 7 - Execute; 8 - Login; 9 - Failed login; 10 - History clear; 11 - Config refresh.
resourcetype	integer	Audit log entry resource type. Possible values: 0 - User; 3 - Media type; 4 - Host; 5 - Action; 6 - Graph; 11 - User group; 13 - Trigger; 14 - Host group; 15 - Item; 16 - Image; 17 - Value map; 18 - Service; 19 - Map; 22 - Web scenario; 23 - Discovery rule; 25 - Script; 26 - Proxy; 27 - Maintenance; 28 - Regular expression; 29 - Macro; 30 - Template; 31 - Trigger prototype; 32 - Icon mapping; 33 - Dashboard; 34 - Event correlation; 35 - Graph prototype; 36 - Item prototype; 37 - Host prototype; 38 - Autoregistration; 39 - Module; 40 - Settings; 41 - Housekeeping; 42 - Authentication; 43 - Template dashboard; 44 - User role; 45 - API token; 46 - Scheduled report; 47 - High availability node; 48 - SLA; 49 - User directory; 50 - Template group; 51 - Connector.

Property	Type	Description
resourceid	string	Audit log entry resource identifier.
resourcename	string	Audit log entry resource human readable name.
recordsetid	string	Audit log entry recordset ID. The audit log records created during the same operation will have the same recordset ID. Generated using CUID algorithm.
details	text	Audit log entry details. The details are stored as a JSON object, where each property name is a path to the property or nested object in which the change occurred, and where each value contains the data (in array format) about the change in this property or nested object. Possible value formats: ["add"] - Nested object has been added; ["add", "<value>"] - The property of the added object equals <value>; ["update"] - Nested object has been updated; ["update", "<new value>", "<old value>"] - The property of the updated object was changed from <old value> to <new value>; ["delete"] - Nested object has been deleted.

auditlog.get

Description

`integer/array auditlog.get(object parameters)`

The method allows to retrieve audit log records according to the given parameters.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
auditids	string/array	Return only audit log with the given IDs.
userids	string/array	Return only audit log that were created by the given users.
time_from	timestamp	Returns only audit log entries that have been created after or at the given time.
time_till	timestamp	Returns only audit log entries that have been created before or at the given time.
sortfield	string/array	Sort the result by the given properties. Possible values: `auditid`, `userid`, `clock`.
filter	object	Return only results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against.
search	object	Case insensitive sub-string search in content of fields: `username`, `ip`, `resourcename`, `details`.
countOutput	boolean	These parameters being common for all `get` methods are described in the reference commentary.
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	

Parameter	Type	Description
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve audit log

Retrieve two latest audit log records.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "auditlog.get",
    "params": {
        "output": "extend",
        "sortfield": "clock",
        "sortorder": "DESC",
        "limit": 2
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "auditid": "cksstgfam0001yhdcc41y20q2",
            "userid": "1",
            "username": "Admin",
            "clock": "1629975715",
            "ip": "127.0.0.1",
            "action": "1",
            "resourcetype": "0",
            "resourceid": "0",
            "resourcename": "Jim",
            "recordsetid": "cksstgfal0000yhdcso67ondl",
            "details": "{\"user.name\":[\"update\",\"Jim\",\"\"],\"user.medias[37]\":[\"add\"],\"user.medi
        },
        {
            "auditid": "ckssofl0p0001yhdcqxclsg8r",
            "userid": "1",
            "username": "Admin",
            "clock": "1629967278",
            "ip": "127.0.0.1",
            "action": "0",
            "resourcetype": "0",
            "resourceid": "20",
            "resourcename": "John",
            "recordsetid": "ckssofl0p0000yhdcpxyo1jgo",
            "details": "{\"user.username\":[\"add\",\"John\"], \"user.userid:\":[\"add\",\"20\"],\"user.us
        }
    ],
    "id": 1
}
```

See also

- Audit log object

Source

CAuditLog::get() in *ui/include/classes/api/services/CAuditLog.php*.

Authentication

This class is designed to work with authentication settings.

Object references:

- Authentication

Available methods:

- authentication.get - retrieve authentication
- authentication.update - update authentication

> Authentication object

The following objects are directly related to the `authentication` API.

Authentication

The authentication object has the following properties.

Property	Type	Description
authentication_type	integer	Default authentication. Possible values: 0 - *(default)* Internal; 1 - LDAP.
http_auth_enabled	integer	HTTP authentication. Possible values: 0 - *(default)* Disabled; 1 - Enabled.
http_login_form	integer	Default login form. Possible values: 0 - *(default)* Zabbix login form; 1 - HTTP login form.
http_strip_domains	string	Domain name to remove.
http_case_sensitive	integer	HTTP case sensitive login. Possible values: 0 - Off; 1 - *(default)* On.
ldap_auth_enabled	integer	LDAP authentication. Possible values: 0 - *(default)* Disabled; 1 - Enabled.
ldap_case_sensitive	integer	LDAP case sensitive login. Possible values: 0 - Off; 1 - *(default)* On.

Property	Type	Description
ldap_userdirectoryid	string	Default user directory for LDAP authentication. Used for user groups with `gui_access` set to LDAP or System default. Property behavior: - *required* if `ldap_auth_enabled` is set to ”Enabled”
saml_auth_enabled	integer	SAML authentication. Possible values: 0 - *(default)* Disabled; 1 - Enabled.
saml_case_sensitive	integer	SAML case sensitive login. Possible values: 0 - Off; 1 - *(default)* On.
passwd_min_length	integer	Password minimal length requirement. Valid values range from 1 to 70. Default: 8.
passwd_check_rules	integer	Password checking rules. Possible bitmap values: 0 - Check password length; 1 - Check if password uses uppercase and lowercase Latin letters; 2 - Check if password uses digits; 4 - Check if password uses special characters; 8 - *(default)* Check if password is not in the list of commonly used passwords and does not contain derivations of word ”Zabbix” or user’s name, last name, or username. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 15 for checking all rules).
ldap_jit_status	integer	Status of LDAP provisioning. Possible values: 0 - Disabled for configured LDAP IdPs; 1 - Enabled for configured LDAP IdPs.
saml_jit_status	integer	Status of SAML provisioning. Possible values: 0 - Disabled for configured SAML IdPs; 1 - Enabled for configured SAML IdPs.
jit_provision_interval	string	Time interval between JIT provision requests for logged-in user. Accepts seconds and time unit with suffix with month and year support (3600s,60m,1h,1d,1M,1y). Minimum value: 1h. Default: 1h. Available only for LDAP provisioning.
disabled_usrgrpid	integer	User group ID to assign the deprovisioned user to. The user group must be disabled and cannot be enabled or deleted when configured. Property behavior: - *required* if `ldap_jit_status` is set to ”Enabled for configured LDAP IdPs”, or `saml_jit_status` is set to ”Enabled for configured SAML IdPs”

authentication.get

Description

`object authentication.get(object parameters)`

The method allows to retrieve authentication object according to the given parameters.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports only one parameter.

Parameter	Type	Description
output	query	This parameter being common for all `get` methods described in the reference commentary.

Return values

`(object)` Returns authentication object.

Examples

Request:

```
{
    "jsonrpc": "2.0",
    "method": "authentication.get",
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "authentication_type": "0",
        "http_auth_enabled": "0",
        "http_login_form": "0",
        "http_strip_domains": "",
        "http_case_sensitive": "1",
        "ldap_auth_enabled": "0",
        "ldap_case_sensitive": "1",
        "ldap_userdirectoryid": "0",
        "saml_auth_enabled": "0",
        "saml_case_sensitive": "0",
        "passwd_min_length": "8",
        "passwd_check_rules": "15",
        "jit_provision_interval": "1h",
        "saml_jit_status": "0",
        "ldap_jit_status": "0",
        "disabled_usrgrpid": "9"
    },
    "id": 1
}
```

Source

CAuthentication::get() in *ui/include/classes/api/services/CAuthentication.php*.

authentication.update

Description

`object authentication.update(object authentication)`

This method allows to update existing authentication settings.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Authentication properties to be updated.

Return values

`(array)` Returns an array with the names of updated parameters.

Examples

Request:

```
{
    "jsonrpc": "2.0",
    "method": "authentication.update",
    "params": {
        "http_auth_enabled": 1,
        "http_case_sensitive": 0,
        "http_login_form": 1
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        "http_auth_enabled",
        "http_case_sensitive",
        "http_login_form"
    ],
    "id": 1
}
```

Source

CAuthentication::update() in *ui/include/classes/api/services/CAuthentication.php*.

Autoregistration

This class is designed to work with autoregistration.

Object references:

- Autoregistration

Available methods:

- autoregistration.get - retrieve autoregistration
- autoregistration.update - update autoregistration

> Autoregistration object

The following objects are directly related to the `autoregistration` API.

Autoregistration

The autoregistration object has the following properties.

Property	Type	Description
tls_accept	integer	Type of allowed incoming connections for autoregistration. Possible values: 1 - allow insecure connections; 2 - allow TLS with PSK; 3 - allow both insecure and TLS with PSK connections.
tls_psk_identity	string	PSK identity; must be paired with only one PSK (across autoregistration, hosts, and proxies). Do not include sensitive information in the PSK identity, as it is sent unencrypted over the network to inform the receiver which PSK to use. Property behavior: - *write-only*
tls_psk	string	Pre-shared key (PSK); must be at least 32 hex digits. Property behavior: - *write-only*

autoregistration.get

Description

`object autoregistration.get(object parameters)`

The method allows to retrieve autoregistration object according to the given parameters.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports only one parameter.

Parameter	Type	Description
output	query	This parameter being common for all `get` methods described in the reference commentary.

Return values

`(object)` Returns autoregistration object.

Examples

Request:

```
{
   "jsonrpc": "2.0",
    "method": "autoregistration.get",
    "params": {
        "output": "extend"
    },
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "tls_accept": "3"
    },
    "id": 1
}
```

Source

CAutoregistration::get() in *ui/include/classes/api/services/CAutoregistration.php*.

autoregistration.update

Description

`object autoregistration.update(object autoregistration)`

This method allows to update existing autoregistration.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Autoregistration properties to be updated.

Return values

`(boolean )` Returns boolean true as result on successful update.

Examples

Request:

```
{
    "jsonrpc": "2.0",
    "method": "autoregistration.update",
    "params": {
        "tls_accept": "3",
        "tls_psk_identity": "PSK 001",
        "tls_psk": "11111595725ac58dd977beef14b97461a7c1045b9a1c923453302c5473193478"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CAutoregistration::update() in *ui/include/classes/api/services/CAutoregistration.php*.

Configuration

This class is designed to export and import Zabbix configuration data.

Available methods:

- configuration.export - exporting the configuration
- configuration.import - importing the configuration
- configuration.importcompare - comparing import file with current system elements

configuration.export

Description

`string configuration.export(object parameters)`

This method allows to export configuration data as a serialized string.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the objects to be exported and the format to use.

Parameter	Type	Description
format	string	Format in which the data must be exported. Possible values: `yaml` - YAML; `xml` - XML; `json` - JSON; `raw` - unprocessed PHP array. Parameter behavior: - *required*
prettyprint	boolean	Make the output more human readable by adding indentation. Possible values: `true` - add indentation; `false` - *(default)* do not add indentation.
options	object	Objects to be exported. The `options` object has the following parameters: `host_groups` - `(array)` IDs of host groups to export; `hosts` - `(array)` IDs of hosts to export; `images` - `(array)` IDs of images to export; `maps` - `(array)` IDs of maps to export; `mediaTypes` - `(array)` IDs of media types to export; `template_groups` - `(array)` IDs of template groups to export; `templates` - `(array)` IDs of templates to export. Parameter behavior: - *required*

Return values

`(string)` Returns a serialized string containing the requested configuration data.

Examples

Exporting a template

Export the configuration of template ”10571” as an XML string.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.export",
    "params": {
        "options": {
            "templates": [
                "10571"
            ]
        },
        "format": "xml"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>6.4</version><template_
    "id": 1
}
```

Source

CConfiguration::export() in *ui/include/classes/api/services/CConfiguration.php*.

configuration.import

Description

`boolean configuration.import(object parameters)`

This method allows to import configuration data from a serialized string.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the data to import and rules how the data should be handled.

Parameter	Type	Description
format	string	Format of the serialized string. Possible values: `yaml` - YAML; `xml` - XML; `json` - JSON. Parameter behavior: - *required*
source	string	Serialized string containing the configuration data. Parameter behavior: - *required*
rules	object	Rules on how new and existing objects should be imported. The `rules` parameter is described in detail in the table below. Parameter behavior: - *required*

Note:
If no rules are given, the configuration will not be updated.

The `rules` object supports the following parameters.

Parameter	Type	Description
discoveryRules	object	Rules on how to import LLD rules. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new LLD rules will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing LLD rules will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, LLD rules not present in the imported data will be deleted from the database; default: `false`.
graphs	object	Rules on how to import graphs. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new graphs will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing graphs will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, graphs not present in the imported data will be deleted from the database; default: `false`.
host_groups	object	Rules on how to import host groups. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new host groups will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing host groups will be updated; default: `false`.
template_groups	object	Rules on how to import template groups. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new template groups will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing template groups will be updated; default: `false`.
hosts	object	Rules on how to import hosts. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new hosts will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing hosts will be updated; default: `false`.
httptests	object	Rules on how to import web scenarios. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new web scenarios will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing web scenarios will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, web scenarios not present in the imported data will be deleted from the database; default: `false`.
images	object	Rules on how to import images. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new images will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing images will be updated; default: `false`.

Parameter	Type	Description
items	object	Rules on how to import items. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new items will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing items will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, items not present in the imported data will be deleted from the database; default: `false`.
maps	object	Rules on how to import maps. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new maps will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing maps will be updated; default: `false`.
mediaTypes	object	Rules on how to import media types. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new media types will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing media types will be updated; default: `false`.
templateLinkage	object	Rules on how to import template links. Supported parameters: `createMissing` - `(boolean)` if set to `true`, templates that are not linked to the host or template being imported, but are present in the imported data, will be linked; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, templates that are linked to the host or template being imported, but are not present in the imported data, will be unlinked without removing entities (items, triggers, etc.) inherited from the unlinked templates; default: `false`.
templates	object	Rules on how to import templates. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new templates will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing templates will be updated; default: `false`.
templateDashboards	object	Rules on how to import template dashboards. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new template dashboards will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing template dashboards will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, template dashboards not present in the imported data will be deleted from the database; default: `false`.
triggers	object	Rules on how to import triggers. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new triggers will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing triggers will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, triggers not present in the imported data will be deleted from the database; default: `false`.

Parameter	Type	Description
valueMaps	object	Rules on how to import host or template value maps. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new value maps will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing value maps will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, value maps not present in the imported data will be deleted from the database; default: `false`.

Return values

`(boolean)` Returns `true` if importing has been successful.

Examples

Importing a template

Import the template configuration contained in the XML string. If any items or triggers in the XML string are missing, they will be deleted from the database, and everything else will be left unchanged.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.import",
    "params": {
        "format": "xml",
        "rules": {
            "templates": {
                "createMissing": true,
                "updateExisting": true
            },
            "items": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "triggers": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "valueMaps": {
                "createMissing": true,
                "updateExisting": false
            }
        },
        "source": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>6.4</version><templ
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CConfiguration::import() in *ui/include/classes/api/services/CConfiguration.php*.

configuration.importcompare

Description

`array configuration.importcompare(object parameters)`

This method allows to compare import file with current system elements and shows what will be changed if this import file will be imported.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the possible data to import and rules how the data should be handled.

Parameter	Type	Description
format	string	Format of the serialized string. Possible values: `yaml` - YAML; `xml` - XML; `json` - JSON. Parameter behavior: - *required*
source	string	Serialized string containing the configuration data. Parameter behavior: - *required*
rules	object	Rules on how new and existing objects should be compared. The `rules` parameter is described in detail in the table below. Parameter behavior: - *required*

Note:
If no rules are given, there will be nothing to update and result will be empty.

Note:
Comparison will be done only for host groups and templates. Triggers and graphs will be compared only for imported templates, any other will be considered as ”new”.

The `rules` object supports the following parameters.

Parameter	Type	Description
discoveryRules	object	Rules on how to import LLD rules. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new LLD rules will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing LLD rules will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, LLD rules not present in the imported data will be deleted from the database; default: `false`.

Parameter	Type	Description
graphs	object	Rules on how to import graphs. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new graphs will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing graphs will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, graphs not present in the imported data will be deleted from the database; default: `false`.
host_groups	object	Rules on how to import host groups. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new host groups will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing host groups will be updated; default: `false`.
template_groups	object	Rules on how to import template groups. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new template groups will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing template groups will be updated; default: `false`.
hosts	object	Rules on how to import hosts. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new hosts will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing hosts will be updated; default: `false`. This parameter will make no difference to the output. It is allowed only for consistency with `configuration.import`.
httptests	object	Rules on how to import web scenarios. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new web scenarios will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing web scenarios will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, web scenarios not present in the imported data will be deleted from the database; default: `false`.
images	object	Rules on how to import images. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new images will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing images will be updated; default: `false`. This parameter will make no difference to the output. It is allowed only for consistency with `configuration.import`.
items	object	Rules on how to import items. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new items will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing items will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, items not present in the imported data will be deleted from the database; default: `false`.

Parameter	Type	Description
maps	object	Rules on how to import maps. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new maps will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing maps will be updated; default: `false`. This parameter will make no difference to the output. It is allowed only for consistency with `configuration.import`.
mediaTypes	object	Rules on how to import media types. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new media types will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing media types will be updated; default: `false`. This parameter will make no difference to the output. It is allowed only for consistency with `configuration.import`.
templateLinkage	object	Rules on how to import template links. Supported parameters: `createMissing` - `(boolean)` if set to `true`, templates that are not linked to the host or template being imported, but are present in the imported data, will be linked; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, templates that are linked to the host or template being imported, but are not present in the imported data, will be unlinked without removing entities (items, triggers, etc.) inherited from the unlinked templates; default: `false`.
templates	object	Rules on how to import templates. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new templates will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing templates will be updated; default: `false`.
templateDashboards	object	Rules on how to import template dashboards. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new template dashboards will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing template dashboards will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, template dashboards not present in the imported data will be deleted from the database; default: `false`.
triggers	object	Rules on how to import triggers. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new triggers will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing triggers will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, triggers not present in the imported data will be deleted from the database; default: `false`.

Parameter	Type	Description
valueMaps	object	Rules on how to import host or template value maps. Supported parameters: `createMissing` - `(boolean)` if set to `true`, new value maps will be created; default: `false`; `updateExisting` - `(boolean)` if set to `true`, existing value maps will be updated; default: `false`; `deleteMissing` - `(boolean)` if set to `true`, value maps not present in the imported data will be deleted from the database; default: `false`.

Return values

`(array)` Returns an array with changes in configuration, that will be made.

Examples

Comparing the import of a template

Compare the template contained in the XML string to the current system elements, and show what will be changed if this template will be imported.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "configuration.importcompare",
    "params": {
        "format": "xml",
        "rules": {
            "discoveryRules": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "graphs": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "host_groups": {
                "createMissing": true,
                "updateExisting": true
            },
            "template_groups": {
                "createMissing": true,
                "updateExisting": true
            },
            "httptests": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "items": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "templateLinkage": {
                "createMissing": true,
                "deleteMissing": true
            },
            "templates": {
                "createMissing": true,
                "updateExisting": true
```

```
            },
            "templateDashboards": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "triggers": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            },
            "valueMaps": {
                "createMissing": true,
                "updateExisting": true,
                "deleteMissing": true
            }
        },
        "source": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>6.4</version><templ
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templates": {
            "updated": [
                {
                    "before": {
                        "uuid": "5aef0444a82a4d8cb7a95dc4c0c85330",
                        "template": "New template",
                        "name": "New template",
                        "groups": [
                            {
                                "name": "Templates"
                            }
                        ]
                    },
                    "after": {
                        "uuid": "5aef0444a82a4d8cb7a95dc4c0c85330",
                        "template": "New template",
                        "name": "New template",
                        "groups": [
                            {
                                "name": "Templates"
                            }
                        ]
                    },
                    "items": {
                        "added": [
                            {
                                "after": {
                                    "uuid": "648006da5971424ead0c47ddbbf1ea2e",
                                    "name": "CPU utilization",
                                    "key": "system.cpu.util",
                                    "value_type": "FLOAT",
                                    "units": "%"
                                },
                                "triggers": {
                                    "added": [
                                        {
```

```
                                        "after": {
                                            "uuid": "736225012c534ec480c2a66a91322ce0",
                                            "expression": "avg(/New template/system.cpu.util,3m)>70",
                                            "name": "CPU utilization too high on 'New host' for 3 minu
                                            "priority": "WARNING"
                                        }
                                    }
                                ]
                            }
                        }
                    ],
                    "removed": [
                        {
                            "before": {
                                "uuid": "6805d4c39a624a8bab2cc8ab63df1ab3",
                                "name": "CPU load",
                                "key": "system.cpu.load",
                                "value_type": "FLOAT"
                            },
                            "triggers": {
                                "removed": [
                                    {
                                        "before": {
                                            "uuid": "ab4c2526c2bc42e48a633082255ebcb3",
                                            "expression": "avg(/New template/system.cpu.load,3m)>2",
                                            "name": "CPU load too high on 'New host' for 3 minutes",
                                            "priority": "WARNING"
                                        }
                                    }
                                ]
                            }
                        }
                    ],
                    "updated": [
                        {
                            "before": {
                                "uuid": "7f1e6f1e48aa4a128e5b6a958a5d11c3",
                                "name": "Zabbix agent ping",
                                "key": "agent.ping"
                            },
                            "after": {
                                "uuid": "7f1e6f1e48aa4a128e5b6a958a5d11c3",
                                "name": "Zabbix agent ping",
                                "key": "agent.ping",
                                "delay": "3m"
                            }
                        }
                    ]
                }
            }
        ]
    }
},
"id": 1
}
```


Source

CConfiguration::importcompare() in *ui/include/classes/api/services/CConfiguration.php*.

Connector

This class is designed to work with connector objects.

Object references:

- Connector
- Connector tag filter

Available methods:

- connector.create - creating new connectors
- connector.delete - deleting connectors
- connector.get - retrieving connectors
- connector.update - updating connectors

> Connector object

The following objects are directly related to the `connector` API.

Connector

The connector object has the following properties.

Property	Type	Description
connectorid	string	ID of the connector. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the connector. Property behavior: - *required* for create operations
url	string	Endpoint URL, that is, URL of the receiver. User macros are supported. Property behavior: - *required* for create operations
protocol	integer	Communication protocol. Possible values: 0 - *(default)* Zabbix Streaming Protocol v1.0.
data_type	integer	Data type. Possible values: 0 - *(default)* Item values; 1 - Events.
max_records	integer	Maximum number of events or items that can be sent within one message. Possible values: 0-2147483647 (max value of 32-bit signed integer). Default: 0 - Unlimited.
max_senders	integer	Number of sender processes to run for this connector. Possible values: 1-100. Default: 1.
max_attempts	integer	Number of attempts. Possible values: 1-5. Default: 1.

Property	Type	Description
timeout	string	Timeout. Time suffixes are supported (e.g., 30s, 1m). User macros are supported. Possible values: 1s-60s. Default: 5s.
http_proxy	string	HTTP(S) proxy connection string given as *[protocol]://[username[:password]@]proxy.example.com[:port]*. User macros are supported.
authtype	integer	HTTP authentication method. Possible values: 0 - *(default)* None; 1 - Basic; 2 - NTLM; 3 - Kerberos; 4 - Digest; 5 - Bearer.
username	string	User name. User macros are supported. Property behavior: - *supported* if `authtype` is set to ”Basic”, ”NTLM”, ”Kerberos”, or ”Digest”
password	string	Password. User macros are supported. Property behavior: - *supported* if `authtype` is set to ”Basic”, ”NTLM”, ”Kerberos”, or ”Digest”
token	string	Bearer token. User macros are supported. Property behavior: - *required* if `authtype` is set to ”Bearer”
verify_peer	integer	Whether to validate that the host’s certificate is authentic. Possible values: 0 - Do not validate; 1 - *(default)* Validate.
verify_host	integer	Whether to validate that the host name for the connection matches the one in the host’s certificate. Possible values: 0 - Do not validate; 1 - *(default)* Validate.
ssl_cert_file	string	Public SSL Key file path. User macros are supported.
ssl_key_file	string	Private SSL Key file path. User macros are supported.
ssl_key_password	string	Password for SSL Key file. User macros are supported.
description	text	Description of the connector.
status	integer	Whether the connector is enabled. Possible values: 0 - Disabled; 1 - *(default)* Enabled.

Property	Type	Description
tags_evaltype	integer	Tag evaluation method. Possible values: 0 - *(default)* And/Or; 2 - Or.

Tag filter

Tag filter allows to export only matching item values or events. If not set then everything will be exported. The tag filter object has the following properties.

Property	Type	Description
tag	string	Tag name. Property behavior: - *required*
operator	integer	Condition operator. Possible values: 0 - *(default)* Equals; 1 - Does not equal; 2 - Contains; 3 - Does not contain; 12 - Exists; 1 - Does not exist.
value	string	Tag value.

connector.create

Description

`object connector.create(object/array connectors)`

This method allows to create new connector objects.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Connector objects to create.

Additionally to the standard connector properties, the method accepts the following parameters.

Parameter	Type	Description
tags	array	Connector tag filter.

Return values

`(object)` Returns an object containing the IDs of the created connectors under the `connectorids` property. The order of the returned IDs matches the order of the passed connectors.

Examples

Creating a connector

Create a connector to export trigger events with a tag filter. HTTP authentication will be performed using Bearer token.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "connector.create",
    "params": [
        {
            "name": "Export of events",
            "data_type": 1,
            "url": "{$DATA_EXPORT_URL}",
            "authtype": 5,
            "token": "{$DATA_EXPORT_BEARER_TOKEN}",
            "tags": [
                {
                    "tag": "service",
                    "operator": 0,
                    "value": "mysqld"
                }
            ]
        }
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "connectorid": [
            "3"
        ]
    },
    "id": 1
}
```

Source

CConnector::create() in *ui/include/classes/api/services/CConnector.php*.

connector.delete

Description

`object connector.delete(array connectorids)`

This method allows to delete connector entries.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the connectors to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted connectors under the `connectorids` property.

Examples

Deleting multiple connectors

Delete two connector entries.

Request:

```
{
    "jsonrpc": "2.0",
```

```
    "method": "connector.delete",
    "params": [
        3,
        5
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "connectorids": [
            "3",
            "5"
        ]
    },
    "id": 1
}
```

Source

CConnector::delete() in *ui/include/classes/api/services/CConnector.php*.

connector.get

Description

`integer/array connector.get(object parameters)`

The method allows to retrieve connector objects according to the given parameters.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
connectorids	string/array	Return only connectors with the given IDs.
selectTags	query	Return a `tags` property with connector tag filter. Supports `count`.
sortfield	string/array	Sort the result by the given properties. Possible values: `connectorid`, `name`, `data_type`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving all connectors

Retrieve all data about all connectors and their properties.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "connector.get",
    "params": {
        "output": "extend",
        "selectTags": ["tag", "operator", "value"],
        "preservekeys": true
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "connectorid": "1",
            "name": "Export of item values",
            "protocol": "0",
            "data_type": "0",
            "url": "{$DATA_EXPORT_VALUES_URL}",
            "max_records": "0",
            "max_senders": "4",
            "max_attempts": "2",
            "timeout": "10s",
            "http_proxy": "{$DATA_EXPORT_VALUES_PROXY}",
            "authtype": "4",
            "username": "{$DATA_EXPORT_VALUES_USERNAME}",
            "password": "{$DATA_EXPORT_VALUES_PASSWORD}",
            "token": "",
            "verify_peer": "1",
            "verify_host": "1",
            "ssl_cert_file": "{$DATA_EXPORT_VALUES_SSL_CERT_FILE}",
            "ssl_key_file": "{$DATA_EXPORT_VALUES_SSL_KEY_FILE}",
            "ssl_key_password": "",
            "description": "",
            "status": "1",
            "tags_evaltype": "0",
            "tags": [
                {
                    "tag": "component",
                    "operator": "0",
                    "value": "memory"
                }
            ]
        },
        {
            "connectorid": "2",
            "name": "Export of events",
            "protocol": "0",
            "data_type": "1",
```

```
            "url": "{$DATA_EXPORT_EVENTS_URL}",
            "max_records": "0",
            "max_senders": "2",
            "max_attempts": "2",
            "timeout": "5s",
            "http_proxy": "",
            "authtype": "5",
            "username": "",
            "password": "",
            "token": "{$DATA_EXPORT_EVENTS_BEARER_TOKEN}",
            "verify_peer": "1",
            "verify_host": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "description": "",
            "status": "1",
            "tags_evaltype": "0",
            "tags": [
                {
                    "tag": "scope",
                    "operator": "0",
                    "value": "performance"
                }
            ]
        }
    ],
    "id": 1
}
```


Source

CConnector:get() in *ui/include/classes/api/services/CConnector.php*.

connector.update

Description

`object connector.update(object/array connectors)`

This method allows to update existing connectors.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Connector properties to be updated.

The `connectorid` property must be defined for each connector, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard connector properties, the method accepts the following parameters.

Parameter	Type	Description
tags	array	Connector tag filter to replace the current tag filter.

Return values

`(object)` Returns an object containing the IDs of the updated connectors under the `connectorids` property.

Examples

Changing HTTP authentication type

Change HTTP authentication type to Bearer for connector with ID ”3”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "connector.update",
    "params": {
        "connectorid": 3,
        "authtype": 5,
        "token": "{$DATA_EXPORT_BEARER_TOKEN}"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "connectorids": [
            "3"
        ]
    },
    "id": 1
}
```

Updating tag filter

Change tag filter for connector with ID ”5”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "connector.update",
    "params": [
        {
            "connectorid": 5,
            "tags_evaltype": 2,
            "tags": [
                {
                    "tag": "service",
                    "operator": 0,
                    "value": "mysqld"
                },
                {
                    "tag": "error",
                    "operator": 12,
                    "value": ""
                }
            ]
        }
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "connectorids": [
            "5"
        ]
    },
    "id": 1
```

```
}
```

Source

CConnector::update() in *ui/include/classes/api/services/CConnector.php*.

Correlation

This class is designed to work with correlations.

Object references:

- Correlation

Available methods:

- correlation.create - creating new correlations
- correlation.delete - deleting correlations
- correlation.get - retrieving correlations
- correlation.update - updating correlations

> Correlation object

The following objects are directly related to the `correlation` API.

Correlation

The correlation object has the following properties.

Property	Type	Description
correlationid	string	ID of the correlation. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the correlation. Property behavior: - *required* for create operations
description	string	Description of the correlation.
status	integer	Whether the correlation is enabled or disabled. Possible values: 0 - *(default)* enabled; 1 - disabled.

Correlation operation

The correlation operation object defines an operation that will be performed when a correlation is executed. It has the following properties.

Property	Type	Description
type	integer	Type of operation. Possible values: 0 - close old events; 1 - close new event. Property behavior: - *required*

Correlation filter

The correlation filter object defines a set of conditions that must be met to perform the configured correlation operations. It has the following properties.

Property	Type	Description
evaltype	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression. Property behavior: - *required*
conditions	array	Set of filter conditions to use for filtering results. Property behavior: - *required*
eval_formula	string	Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its `formulaid`. The value of `eval_formula` is equal to the value of `formula` for filters with a custom expression. Property behavior: - *read-only*
formula	string	User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its `formulaid`. The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. Property behavior: - *required* if `evaltype` is set to ”custom expression”

Correlation filter condition

The correlation filter condition object defines a specific condition that must be checked before running the correlation operations.

Property	Type	Description
type	integer	Type of condition. Possible values: 0 - old event tag; 1 - new event tag; 2 - new event host group; 3 - event tag pair; 4 - old event tag value; 5 - new event tag value. Property behavior: - *required*
tag	string	Event tag (old or new). Property behavior: - *required* if `type` is set to ”old event tag”, ”new event tag”, ”old event tag value”, or ”new event tag value”
groupid	string	Host group ID. Property behavior: - *required* if `type` is set to ”new event host group”

Property	Type	Description
oldtag	string	Old event tag. Property behavior: - *required* if `type` is set to ”event tag pair”
newtag	string	Old event tag. Property behavior: - *required* if `type` is set to ”event tag pair”
value	string	Event tag (old or new) value. Property behavior: - *required* if `type` is set to ”old event tag value” or ”new event tag value”
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
operator	integer	Condition operator. Property behavior: - *required* if `type` is set to ”new event host group”, ”old event tag value”, or ”new event tag value”

Note:
To better understand how to use filters with various types of expressions, see examples on the correlation.get and correlation.create method pages.

The following operators and values are supported for each condition type.

Condition	Condition name	Supported operators	Expected value
2	Host group	=, <>	Host group ID.
4	Old event tag value	=, <>, like, not like	string
5	New event tag value	=, <>, like, not like	string

correlation.create

Description

`object correlation.create(object/array correlations)`

This method allows to create new correlations.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Correlations to create.

Additionally to the standard correlation properties, the method accepts the following parameters.

Parameter	Type	Description
operations	array	Correlation operations to create for the correlation. Parameter behavior: - *required*

Parameter	Type	Description
filter	object	Correlation filter object for the correlation. Parameter behavior: - *required*

Return values

(object) Returns an object containing the IDs of the created correlations under the correlationids property. The order of the returned IDs matches the order of the passed correlations.

Examples

Create a new event tag correlation

Create a correlation using evaluation method AND/OR with one condition and one operation. By default the correlation will be enabled.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.create",
    "params": {
        "name": "new event tag correlation",
        "filter": {
            "evaltype": 0,
            "conditions": [
                {
                    "type": 1,
                    "tag": "ok"
                }
            ]
        },
        "operations": [
            {
                "type": 0
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
        ]
    },
    "id": 1
}
```

Using a custom expression filter

Create a correlation that will use a custom filter condition. The formula IDs ”A” or ”B” have been chosen arbitrarily. Condition type will be ”Host group” with operator ”<>”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.create",
    "params": {
        "name": "new host group correlation",
```

```
        "description": "a custom description",
        "status": 0,
        "filter": {
            "evaltype": 3,
            "formula": "A or B",
            "conditions": [
                {
                    "type": 2,
                    "operator": 1,
                    "formulaid": "A"
                },
                {
                    "type": 2,
                    "operator": 1,
                    "formulaid": "B"
                }
            ]
        },
        "operations": [
            {
                "type": 1
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "2"
        ]
    },
    "id": 1
}
```

See also

- Correlation filter
- Correlation operation

Source

CCorrelation::create() in *ui/include/classes/api/services/CCorrelation.php*.

correlation.delete

Description

`object correlation.delete(array correlationids)`

This method allows to delete correlations.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the correlations to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted correlations under the `correlationids` property.

Example

Delete multiple correlations

Delete two correlations.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.delete",
    "params": [
        "1",
        "2"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

Source

CCorrelation::delete() in *ui/include/classes/api/services/CCorrelation.php*.

correlation.get

Description

`integer/array correlation.get(object parameters)`

The method allows to retrieve correlations according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
correlationids	string/array	Return only correlations with the given IDs.
selectFilter	query	Return a filter property with the correlation conditions.
selectOperations	query	Return an operations property with the correlation operations.
sortfield	string/array	Sort the result by the given properties. Possible values: `correlationid`, `name`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	

Parameter	Type	Description
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve correlations

Retrieve all configured correlations together with correlation conditions and operations. The filter uses the "and/or" evaluation type, so the `formula` property is empty and `eval_formula` is generated automatically.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.get",
    "params": {
        "output": "extend",
        "selectOperations": "extend",
        "selectFilter": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "correlationid": "1",
            "name": "Correlation 1",
            "description": "",
            "status": "0",
            "filter": {
                "evaltype": "0",
                "formula": "",
                "conditions": [
                    {
                        "type": "3",
                        "oldtag": "error",
                        "newtag": "ok",
                        "formulaid": "A"
                    }
                ],
                "eval_formula": "A"
            },
            "operations": [
                {
                    "type": "0"
                }
            ]
        }
    ],
```

```
    "id": 1
}
```

See also

- Correlation filter
- Correlation operation

Source

CCorrelation::get() in *ui/include/classes/api/services/CCorrelation.php*.

correlation.update

Description

`object correlation.update(object/array correlations)`

This method allows to update existing correlations.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Correlation properties to be updated.

The `correlationid` property must be defined for each correlation, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard correlation properties, the method accepts the following parameters.

Parameter	Type	Description
filter	object	Correlation filter object to replace the current filter.
operations	array	Correlation operations to replace existing operations.

Return values

`(object)` Returns an object containing the IDs of the updated correlations under the `correlationids` property.

Examples

Disable correlation

Request:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.update",
    "params": {
        "correlationid": "1",
        "status": "1"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
        ]
    },
    "id": 1
}
```

Replace conditions, but keep the evaluation method

Request:

```
{
    "jsonrpc": "2.0",
    "method": "correlation.update",
    "params": {
        "correlationid": "1",
        "filter": {
            "conditions": [
                {
                    "type": 3,
                    "oldtag": "error",
                    "newtag": "ok"
                }
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "correlationids": [
            "1"
        ]
    },
    "id": 1
}
```

See also

- Correlation filter
- Correlation operation

Source

CCorrelation::update() in *ui/include/classes/api/services/CCorrelation.php*.

Dashboard

This class is designed to work with dashboards.

Object references:

- Dashboard
- Dashboard page
- Dashboard widget
- Dashboard widget field
- Dashboard user
- Dashboard user group

Available methods:

- dashboard.create - creating new dashboards
- dashboard.delete - deleting dashboards
- dashboard.get - retrieving dashboards
- dashboard.update - updating dashboards

> Dashboard object

The following objects are directly related to the `dashboard` API.

Dashboard

The dashboard object has the following properties.

Property	Type	Description
dashboardid	string	ID of the dashboard. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the dashboard. Property behavior: - *required* for create operations
userid	string	Dashboard owner user ID.
private	integer	Type of dashboard sharing. Possible values: 0 - public dashboard; 1 - *(default)* private dashboard.
display_period	integer	Default page display period (in seconds). Possible values: 10, 30, 60, 120, 600, 1800, 3600. Default: 30.
auto_start	integer	Auto start slideshow. Possible values: 0 - do not auto start slideshow; 1 - *(default)* auto start slideshow.

Dashboard page

The dashboard page object has the following properties.

Property	Type	Description
dashboard_pageid	string	ID of the dashboard page. Property behavior: - *read-only*
name	string	Dashboard page name. Default: empty string.
display_period	integer	Dashboard page display period (in seconds). Possible values: 0, 10, 30, 60, 120, 600, 1800, 3600. Default: 0 (will use the default page display period).
widgets	array	Array of the dashboard widget objects.

Dashboard widget

The dashboard widget object has the following properties.

Property	Type	Description
widgetid	string	ID of the dashboard widget. Property behavior: - *read-only*

Property	Type	Description
type	string	Type of the dashboard widget. Possible values: actionlog - Action log; clock - Clock; *(deprecated)* dataover - Data overview; discovery - Discovery status; favgraphs - Favorite graphs; favmaps - Favorite maps; geomap - Geomap; graph - Graph (classic); graphprototype - Graph prototype; hostavail - Host availability; item - Item value; map - Map; navtree - Map Navigation Tree; plaintext - Plain text; problemhosts - Problem hosts; problems - Problems; problemsbysv - Problems by severity; slareport - SLA report; svggraph - Graph; systeminfo - System information; tophosts - Top hosts; trigover - Trigger overview; url - URL; web - Web monitoring. Property behavior: - *required*
name	string	Custom widget name.
x	integer	A horizontal position from the left side of the dashboard. Valid values range from 0 to 23.
y	integer	A vertical position from the top of the dashboard. Valid values range from 0 to 62.
width	integer	The widget width. Valid values range from 1 to 24.
height	integer	The widget height. Valid values range from 2 to 32.
view_mode	integer	The widget view mode. Possible values: 0 - *(default)* default widget view; 1 - with hidden header;
fields	array	Array of the dashboard widget field objects. Property behavior: - see individual widgets in Dashboard widget fields

Dashboard widget field

The dashboard widget field object has the following properties.

Property	Type	Description
type	integer	Type of the widget field. Possible values: 0 - Integer; 1 - String; 2 - Host group; 3 - Host; 4 - Item; 5 - Item prototype; 6 - Graph; 7 - Graph prototype; 8 - Map; 9 - Service; 10 - SLA; 11 - User; 12 - Action; 13 - Media type. Property behavior: - *required*
name	string	Widget field name. Possible values: see Dashboard widget fields. Property behavior: - *required*
value	mixed	Widget field value depending on the type. Possible values: see Dashboard widget fields. Property behavior: - *required*

Dashboard user group

List of dashboard permissions based on user groups. It has the following properties.

Property	Type	Description
usrgrpid	string	User group ID. Property behavior: - *required*
permission	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write. Property behavior: - *required*

Dashboard user

List of dashboard permissions based on users. It has the following properties.

Property	Type	Description
userid	string	User ID. Property behavior: - *required*
permission	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write. Property behavior: - *required*

Dashboard widget fields

This page contains navigation links for dashboard widget parameters and possible property values for the respective dashboard widget field objects.

To see the parameters and property values for each widget, go to individual widget pages for:

- Action log
- Clock
- Discovery status
- Favorite graphs
- Favorite maps
- Geomap
- Graph
- Graph (classic)
- Graph prototype
- Host availability
- Item value
- Map
- Map navigation tree
- Plain text
- Problem hosts
- Problems
- SLA report
- System information
- Problems by severity
- Top hosts
- Trigger overview
- URL
- Web monitoring

Deprecated widgets:

- Data overview

Attention:
Deprecated widgets will be removed in the upcoming major release.

1 Action log

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Action log* widget in `dashboard.create` and `dashboard.update` methods.

Parameters

The following parameters are supported for the *Action log* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Recipients	11	userids	User ID. Note: To configure multiple users, create a dashboard widget field object for each user.
Actions	12	actionids	Action ID. Note: To configure multiple actions, create a dashboard widget field object for each action.
Media types	13	mediatypeids	Media type ID. Note: To configure multiple media types, create a dashboard widget field object for each media type.
Status	0	statuses	0 - In progress; 1 - Sent/Executed; 2 - Failed. Note: To configure multiple values, create a dashboard widget field object for each value.
Search string	1	message	Any string value.
Sort entries by	0	sort_triggers	3 - Time (ascending); 4 - *(default)* Time (descending); 5 - Type (ascending); 6 - Type (descending); 7 - Status (ascending); 8 - Status (descending); 11 - Recipient (ascending); 12 - Recipient (descending).
Show lines	0	show_lines	Valid values range from 1-100. Default: 25.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Action log* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring an *Action log* widget

Configure an *Action log* widget that displays 10 entries of action operation details, sorted by time (in ascending order). In addition, display details only for those action operations that attempted to send an email to user "1", but were unsuccessful.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "actionlog",
                        "name": "Action log",
```

```
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "show_lines",
                                "value": 10
                            },
                            {
                                "type": 0,
                                "name": "sort_triggers",
                                "value": 3
                            },
                            {
                                "type": 11,
                                "name": "userids",
                                "value": 1
                            },
                            {
                                "type": 13,
                                "name": "mediatypeids",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "statuses",
                                "value": 2
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

2 Clock

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Clock* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Clock* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Clock* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - *(default)* 15 minutes.
Time type	0	time_type	0 - *(default)* Local time; 1 - Server time; 2 - Host time.
Clock type	0	clock_type	0 - *(default)* Analog; 1 - Digital.

The following parameters are supported if *Time type* is set to ”Host time”.

Parameter	type	name	value
Item	4	itemid	Item ID. Parameter behavior: - *required*

The following parameters are supported if *Clock type* is set to ”Digital”.

Parameter	type	name	value
Show	0	show	1 - Date; 2 - *(default)* Time; 3 - Time zone. Note: To configure multiple values, create a dashboard widget field object for each value.
Advanced configuration	0	adv_conf	0 - *(default)* Disabled; 1 - Enabled. Parameter *Advanced configuration* must be set to ”Enabled” to configure *Background color* and all options for *Date*, *Time* and *Time zone*.

Parameter	type	name	value
Background color	1	bg_color	Hexadecimal color code (e.g. `FF0000`). Default: `""` (empty).
Date			
Size	0	date_size	Valid values range from 1-100. Default: 20.
Bold	0	date_bold	0 - *(default)* Disabled; 1 - Enabled.
Color	1	date_color	Hexadecimal color code (e.g. `FF0000`). Default: `""` (empty).
Time			
Size	0	time_size	Valid values range from 1-100. Default: 30.
Bold	0	time_bold	0 - *(default)* Disabled; 1 - Enabled.
Color	1	time_color	Hexadecimal color code (e.g. `FF0000`). Default: `""` (empty).
Seconds	0	time_sec	0 - Disabled; 1 - *(default)* Enabled.
Format	0	time_format	0 - *(default)* 24-hour; 1 - 12-hour.
Time zone			
Size	0	tzone_size	Valid values range from 1-100. Default: 20.
Bold	0	tzone_bold	0 - *(default)* Disabled; 1 - Enabled.
Color	1	tzone_color	Hexadecimal color code (e.g. `FF0000`). Default: `""` (empty).
Time zone	1	tzone_timezone	Valid timezone string (e.g. `Europe/Riga`, `system`, `UTC`, etc.). For the full list of supported time zones please refer to PHP documentation. Default: `local`. Parameter *Time zone* not available if *Time type* is set to ”Host time”.
Format	0	tzone_format	0 - *(default)* Short; 1 - Full. Parameter *Format* not available if *Time type* is set to ”Host time”.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Clock* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Clock* widget

Configure a *Clock* widget that displays local date, time and time zone in a customized digital clock.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
```

```
    "display_period": 30,
    "auto_start": 1,
    "pages": [
        {
            "widgets": [
                {
                    "type": "clock",
                    "name": "Clock",
                    "x": 0,
                    "y": 0,
                    "width": 4,
                    "height": 3,
                    "view_mode": 0,
                    "fields": [
                        {
                            "type": 0,
                            "name": "clock_type",
                            "value": 1
                        },
                        {
                            "type": 0,
                            "name": "show",
                            "value": 1
                        },
                        {
                            "type": 0,
                            "name": "show",
                            "value": 2
                        },
                        {
                            "type": 0,
                            "name": "show",
                            "value": 3
                        },
                        {
                            "type": 0,
                            "name": "adv_conf",
                            "value": 1
                        },
                        {
                            "type": 0,
                            "name": "date_size",
                            "value": 20
                        },
                        {
                            "type": 1,
                            "name": "date_color",
                            "value": "E1E1E1"
                        },
                        {
                            "type": 0,
                            "name": "time_bold",
                            "value": 1
                        },
                        {
                            "type": 0,
                            "name": "tzone_size",
                            "value": 10
                        },
                        {
                            "type": 1,
                            "name": "tzone_color",
```

```
                                "value": "E1E1E1"
                            },
                            {
                                "type": 1,
                                "name": "tzone_timezone",
                                "value": "Europe/Riga"
                            },
                            {
                                "type": 0,
                                "name": "tzone_format",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

3 Data overview

Attention:
This widget is deprecated and will be removed in the upcoming major release. Consider using the *Top hosts* widget instead.

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Data overview* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Data overview* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Data overview* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Host groups	2	groupids	Host group ID. Note: To configure multiple host groups, create a dashboard widget field object for each host group.
Hosts	3	hostids	Host ID. Note: To configure multiple hosts, create a dashboard widget field object for each host. For multiple hosts, the parameter *Host groups* must either be not configured at all or configured with at least one host group that the configured hosts belong to.
Tags (the number in the property name (e.g. tags.tag.0) references tag order in the tag evaluation list)			
Evaluation type	0	evaltype	0 - *(default)* And/Or; 2 - Or.
Tag name	1	tags.tag.0	Any string value. Parameter behavior: - *required* if configuring *Tags*

Parameter	type	name	value
Operator	0	tags.operator.0	0 - Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist. Parameter behavior: - *required* if configuring *Tags*
Tag value	1	tags.value.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Show suppressed problems	0	show_suppressed	0 - *(default)* Disabled; 1 - Enabled.
Hosts location	0	style	0 - *(default)* Left; 1 - Top.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Data overview* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Data overview* widget

Configure a *Data overview* widget that displays data for host "10084" and only for items for which the tag with the name "component" contains value "cpu". In addition, display the data with hosts located on top.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "dataover",
                        "name": "Data overview",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 3,
                                "name": "hostids",
                                "value": 10084
                            },
                            {
                                "type": 1,
                                "name": "tags.tag.0",
                                "value": "component"
```

```
                            },
                            {
                                "type": 0,
                                "name": "tags.operator.0",
                                "value": 0
                            },
                            {
                                "type": 1,
                                "name": "tags.value.0",
                                "value": "cpu"
                            },
                            {
                                "type": 0,
                                "name": "style",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

4 Discovery status

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Discovery status* widget in `dashboard.create` and `dashboard.update` methods.

Parameters

The following parameters are supported for the *Discovery status* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Discovery status* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring *Discovery status* widget

Configure a *Discovery status* widget with the refresh interval set to 15 minutes.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "discovery",
                        "name": "Discovery status",
                        "x": 0,
                        "y": 0,
                        "width": 6,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "rf_rate",
                                "value": 900
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

5 Favorite graphs

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Favorite graphs* widget in `dashboard.create` and `dashboard.update` methods.

Parameters

The following parameters are supported for the *Favorite graphs* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - *(default)* 15 minutes.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Favorite graphs* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Favorite graphs* widget

Configure a *Favorite graphs* widget with the refresh interval set to 10 minutes.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "favgraphs",
                        "name": "Favorite graphs",
```

```
                        "x": 0,
                        "y": 0,
                        "width": 4,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "rf_rate",
                                "value": 600
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

6 Favorite maps

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Favorite maps* widget in `dashboard.create` and `dashboard.update` methods.

Parameters

The following parameters are supported for the *Favorite maps* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - *(default)* 15 minutes.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Favorite maps* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Favorite maps* widget

Configure a *Favorite maps* widget with the refresh interval set to 10 minutes.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "favmaps",
                        "name": "Favorite maps",
                        "x": 0,
                        "y": 0,
                        "width": 4,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "rf_rate",
                                "value": 600
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

7 Geomap

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Geomap* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Geomap* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Geomap* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Host groups	2	groupids	Host group ID. Note: To configure multiple host groups, create a dashboard widget field object for each host group.
Hosts	3	hostids	Host ID. Note: To configure multiple hosts, create a dashboard widget field object for each host. For multiple hosts, the parameter *Host groups* must either be not configured at all or configured with at least one host group that the configured hosts belong to.

Parameter	type	name	value
Tags (the number in the property name (e.g. tags.tag.0) references tag order in the tag evaluation list)			
Evaluation type	0	evaltype	0 - *(default)* And/Or; 2 - Or.
Tag name	1	tags.tag.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Operator	0	tags.operator.0	0 - Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist. Parameter behavior: - *required* if configuring *Tags*
Tag value	1	tags.value.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Initial view	1	default_view	Comma separated *latitude*, *longitude*, *zoom level* (*optional*, valid values range from 0-30). Example: `40.6892494,-74.0466891,10`.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Geomap* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Geomap* widget

Configure a *Geomap* widget that displays hosts from host groups "2" and "22" based on the following tag configuration: tag with the name "component" contains value "node", and tag with the name "location" equals value "New York". In addition, set the map initial view to coordinates "40.6892494" (latitude), "-74.0466891" (longitude) with the zoom level "10".

Request:

```
{
    "jsonrpc": "2.0",
```

```
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "geomap",
                        "name": "Geomap",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids",
                                "value": 22
                            },
                            {
                                "type": 2,
                                "name": "groupids",
                                "value": 2
                            },
                            {
                                "type": 1,
                                "name": "default_view",
                                "value": "40.6892494,-74.0466891,10"
                            },
                            {
                                "type": 0,
                                "name": "evaltype",
                                "value": 2
                            },
                            {
                                "type": 1,
                                "name": "tags.tag.0",
                                "value": "component"
                            },
                            {
                                "type": 0,
                                "name": "tags.operator.0",
                                "value": 0
                            },
                            {
                                "type": 1,
                                "name": "tags.value.0",
                                "value": "node"
                            },
                            {
                                "type": 1,
                                "name": "tags.tag.1",
                                "value": "location"
                            },
                            {
                                "type": 0,
                                "name": "tags.operator.1",
                                "value": 1
                            },
```

```
                            {
                                "type": 1,
                                "name": "tags.value.1",
                                "value": "New York"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

8 Graph

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Graph* widget in `dashboard.create` and `dashboard.update` methods.

> **Attention:**
> Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Graph* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Graph* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.

Data set

The following parameters are supported for configuring a *Data set*.

Note:
The first number in the property name (e.g. ds.hosts.0.0, ds.items.0.0) represents the particular data set, while the second number, if present, represents the configured host or item.

Parameter	type	name	value
Data set type	0	ds.dataset_type.0	0 - Item list; 1 - *(default)* Item pattern.
Parameters if *Data set type* is set to "Item list"			
Items	4	ds.itemids.0.0	Item ID. Note: To configure multiple items, create a dashboard widget field object for each item. Parameter behavior: - *required*
Color	1	ds.color.0.0	Hexadecimal color code (e.g. FF0000). Parameter behavior: - *required*
Parameters if *Data set type* is set to "Item pattern"			
Host pattern	1	ds.hosts.0.0	Host name or pattern (e.g., "Zabbix*"). Parameter behavior: - *required*

Parameter	type	name	value
Item pattern	1	ds.items.0.0	Item name or pattern (e.g., "*: Number of processed *values per second"). Parameter behavior: - *required*
Color	1	ds.color.0	Hexadecimal color code (e.g. FF0000). Default: FF465C.
Draw	0	ds.type.0	0 - *(default)* Line; 1 - Points; 2 - Staircase; 3 - Bar.
Stacked	0	ds.stacked.0	0 - *(default)* Disabled; 1 - Enabled. Parameter *Stacked* not available if *Draw* is set to "Points".
Width	0	ds.width.0	Valid values range from 1-10. Default: 1. Parameter *Width* not available if *Draw* is set to "Points" or "Bar".
Point size	0	ds.pointsize.0	Valid values range from 1-10. Default: 3. Parameter *Point size* not available if *Draw* is set to "Line", "Staircase" or "Bar".
Transparency	0	ds.transparency.0	Valid values range from 1-10. Default: 5.
Fill	0	ds.fill.0	Valid values range from 1-10. Default: 3. Parameter *Fill* not available if *Draw* is set to "Points" or "Bar".
Missing data	0	ds.missingdatafunc.0	0 - *(default)* None; 1 - Connected; 2 - Treat as 0; 3 - Last known. Parameter *Missing data* not available if *Draw* is set to "Points" or "Bar".
Y-axis	0	ds.axisy.0	0 - *(default)* Left; 1 - Right.
Time shift	1	ds.timeshift.0	Valid time string (e.g. 3600, 1h, etc.). You may use time suffixes. Negative values are also allowed. Default: "" (empty).
Aggregation function	0	ds.aggregate_function.0	0 - *(default)* none; 1 - min; 2 - max; 3 - avg; 4 - count; 5 - sum; 6 - first; 7 - last.
Aggregation interval	1	ds.aggregate_interval.0	Valid time string (e.g. 3600, 1h, etc.). You may use time suffixes. Default: 1h.

Parameter	type	name	value
Aggregate	0	ds.aggregate_grouping.0	0 - *(default)* Each item; 1 - Data set. Parameter *Aggregate* not available if *Aggregation function* is set to ”none”.
Approximation	0	ds.approximation.0	1 - min; 2 - *(default)* avg; 4 - max; 7 - all.
Data set label	1	ds.data_set_label.0	Any string value. Default: "" (empty).

Display options

The following parameters are supported for configuring *Display options*.

Parameter	type	name	value
History data selection	0	source	0 - *(default)* Auto; 1 - History; 2 - Trends.
Simple triggers	0	simple_triggers	0 - *(default)* Disabled; 1 - Enabled.
Working time	0	working_time	0 - *(default)* Disabled; 1 - Enabled.
Percentile line (left) (parameter available if *Y-axis* (in *Data set* configuration) is set to ”Left”)			
Status	0	percentile_left	0 - *(default)* Disabled; 1 - Enabled.
Value	0	percentile_left_value	Valid values range from 1-100.

Parameter	type	name	value
Percentile line (right) (parameter available if *Y-axis* (in *Data set* configuration) is set to "Right")			
Status	0	percentile_right	0 - *(default)* Disabled; 1 - Enabled.
Value	0	percentile_right_value	Valid values range from 1-100.

Time period

The following parameters are supported for configuring *Time period*.

Parameter	type	name	value
Set custom time period	0	graph_time	0 - *(default)* Disabled; 1 - Enabled.
From	1	time_from	Valid time string in format `YYYY-MM-DD hh:mm:ss`. Relative time period values (`now`, `now/d`, `now/w-1w`, etc.) are also supported. Default: `now-1h`.
To	1	time_to	Valid time string value in format `YYYY-MM-DD hh:mm:ss`. Relative time period values (`now`, `now/d`, `now/w-1w`, etc.) are also supported. Default: `now`.

Axes

The following parameters are supported for configuring *Axes*.

Parameter	type	name	value
Left Y	0	lefty	0 - Disabled; 1 - *(default)* Enabled. Parameter available if *Y-axis* (in *Data set* configuration) is set to "Left".
Right Y	0	righty	0 - *(default)* Disabled; 1 - Enabled. Parameter available if *Y-axis* (in *Data set* configuration) is set to "Right".

Parameter	type	name	value
Min	1	lefty_min	Any numeric value. Default: "" (empty).
		righty_min	
Max	1	lefty_max	Any numeric value. Default: "" (empty).
		righty_max	
Units (type)	0	lefty_units	0 - *(default)* Auto; 1 - Static.
		righty_units	
Units (value)	1	lefty_static_units	Any string value. Default: "" (empty).
		righty_static_units	
X-Axis	0	xaxis	0 - Disabled; 1 - *(default)* Enabled.

Legend

The following parameters are supported for configuring *Legend*.

Parameter	type	name	value
Show legend	0	legend	0 - Disabled; 1 - *(default)* Enabled.
Display min/max/avg	0	legend_statistic	0 - *(default)* Disabled; 1 - Enabled.
Number of rows	0	legend_lines	Valid values range from 1-10. Default: 1.
Number of columns	0	legend_columns	Valid values range from 1-4. Default: 4.

Problems

The following parameters are supported for configuring *Problems*.

Parameter	type	name	value
Show problems	0	show_problems	0 - *(default)* Disabled; 1 - Enabled.
Selected items only	0	graph_item_problems	0 - Disabled; 1 - *(default)* Enabled.
Problem hosts	1	problemhosts.0	Host name. Note: The number in the property name references the configured host. To configure multiple hosts, create a dashboard widget field object for each host.

Parameter	type	name	value
Severity	0	severities	0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster. Default: empty (all enabled). Note: To configure multiple values, create a dashboard widget field object for each value.
Problem	1	problem_name	Problem event name (case insensitive, full name or part of it).
Tags (the number in the property name (e.g. tags.tag.0) references tag order in the tag evaluation list)			
Evaluation type	0	evaltype	0 - *(default)* And/Or; 2 - Or.
Tag name	1	tags.tag.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Operator	0	tags.operator.0	0 - Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist. Parameter behavior: - *required* if configuring *Tags*
Tag value	1	tags.value.0	Any string value. Parameter behavior: - *required* if configuring *Tags*

Overrides

The following parameters are supported for configuring *Overrides*.

Note:
The first number in the property name (e.g. or.hosts.0.0, or.items.0.0) represents the particular data set, while the second number, if present, represents the configured host or item.

Parameter	type	name	value
Host pattern	1	or.hosts.0.0	Host name or pattern (e.g. `Zabbix*`). Parameter behavior: - *required*
Item pattern	1	or.items.0.0	Item name or pattern (e.g. `*: Number of processed *values per second`). Parameter behavior: - *required*
Base color	1	or.color.0	Hexadecimal color code (e.g. FF0000).
Width	0	or.width.0	Valid values range from 1-10.
Draw	0	or.type.0	0 - Line; 1 - Points; 2 - Staircase; 3 - Bar.
Transparency	0	or.transparency.0	Valid values range from 1-10.
Fill	0	or.fill.0	Valid values range from 1-10.
Point size	0	or.pointsize.0	Valid values range from 1-10.
Missing data	0	or.missingdatafunc.0	0 - None; 1 - Connected; 2 - Treat as 0; 3 - Last known.
Y-axis	0	or.axisy.0	0 - Left; 1 - Right.
Time shift	1	or.timeshift.0	Valid time string (e.g. 3600, 1h, etc.). You may use time suffixes. Negative values are allowed.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Graph* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Graph* widget

Configure a *Graph* widget in the following way:

- 2 data sets for a total of 9 items on 1 host.
- The first data set is of type "Item list" and consists of 3 items that are represented by lines with a different color, but the same width, transparency, and fill.
- The second data set is of type "Item pattern", consists of 6 items, has a configured aggregation, and is represented by a line with a custom color, width, transparency, and fill.
- The second data set also has a custom data set label.
- Data in the graph are displayed for a time period of the last 3 hours.
- Problems in the graph are displayed only for the configured items.
- Graph has two Y axes of which the right Y axis displays values only for the second data set.
- Graph legend displays configured items in 4 rows, as well as minimum, maximum and average values of the data sets.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
```

```
            "widgets": [
                {
                    "type": "svggraph",
                    "name": "Graph",
                    "x": 0,
                    "y": 0,
                    "width": 12,
                    "height": 5,
                    "view_mode": 0,
                    "fields": [
                        {
                            "type": 0,
                            "name": "ds.dataset_type.0",
                            "value": 0
                        },
                        {
                            "type": 4,
                            "name": "ds.itemids.0.1",
                            "value": 23264
                        },
                        {
                            "type": 1,
                            "name": "ds.color.0.1",
                            "value": "FF0000"
                        },
                        {
                            "type": 4,
                            "name": "ds.itemids.0.2",
                            "value": 23269
                        },
                        {
                            "type": 1,
                            "name": "ds.color.0.2",
                            "value": "BF00FF"
                        },
                        {
                            "type": 4,
                            "name": "ds.itemids.0.3",
                            "value": 23257
                        },
                        {
                            "type": 1,
                            "name": "ds.color.0.3",
                            "value": "0040FF"
                        },
                        {
                            "type": 0,
                            "name": "ds.width.0",
                            "value": 3
                        },
                        {
                            "type": 0,
                            "name": "ds.transparency.0",
                            "value": 3
                        },
                        {
                            "type": 0,
                            "name": "ds.fill.0",
                            "value": 1
                        },
                        {
                            "type": 1,
```

```
                "name": "ds.hosts.1.0",
                "value": "Zabbix server"
            },
            {
                "type": 1,
                "name": "ds.items.1.0",
                "value": "*: Number of processed *values per second"
            },
            {
                "type": 1,
                "name": "ds.color.1",
                "value": "000000"
            },
            {
                "type": 0,
                "name": "ds.transparency.1",
                "value": 0
            },
            {
                "type": 0,
                "name": "ds.fill.1",
                "value": 0
            },
            {
                "type": 0,
                "name": "ds.axisy.1",
                "value": 1
            },
            {
                "type": 0,
                "name": "ds.aggregate_function.1",
                "value": 3
            },
            {
                "type": 1,
                "name": "ds.aggregate_interval.1",
                "value": "1m"
            },
            {
                "type": 0,
                "name": "ds.aggregate_grouping.1",
                "value": 1
            },
            {
                "type": 1,
                "name": "ds.data_set_label.1",
                "value": "Number of processed values per second"
            },
            {
                "type": 0,
                "name": "graph_time",
                "value": 1
            },
            {
                "type": 1,
                "name": "time_from",
                "value": "now-3h"
            },
            {
                "type": 0,
                "name": "legend_statistic",
                "value": 1
```

```
                            },
                            {
                                "type": 0,
                                "name": "legend_lines",
                                "value": 4
                            },
                            {
                                "type": 0,
                                "name": "show_problems",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

9 Graph (classic)

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Graph (classic)* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Graph (classic)* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Graph (classic)* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Source	0	source_type	0 - *(default)* Graph; 1 - Simple graph.
Graph	6	graphid	Graph ID. Parameter behavior: - *required* if *Source* is set to ”Graph”
Item	4	itemid	Item ID. Parameter behavior: - *required* if *Source* is set to ”Simple graph”
Show legend	0	show_legend	0 - Disabled; 1 - *(default)* Enabled.
Enable host selection	0	dynamic	0 - *(default)* Disabled; 1 - Enabled.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Graph (classic)* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Graph (classic)* widget

Configure a *Graph (classic)* widget that displays a simple graph for the item ”42269”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "graph",
                        "name": "Graph (classic)",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "source_type",
                                "value": 1
                            },
                            {
                                "type": 4,
                                "name": "itemid",
```

```
                                "value": 42269
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

10 Graph prototype

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Graph prototype* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Graph prototype* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Graph prototype* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Source	0	source_type	2 - *(default)* Graph prototype; 3 - Simple graph prototype.
Graph prototype	7	graphid	Graph prototype ID. Parameter behavior: - *required* if *Source* is set to ”Graph prototype”
Item prototype	5	itemid	Item prototype ID. Parameter behavior: - *required* if *Source* is set to ”Simple graph prototype”
Show legend	0	show_legend	0 - Disabled; 1 - *(default)* Enabled.
Enable host selection	0	dynamic	0 - *(default)* Disabled; 1 - Enabled.
Columns	0	columns	Valid values range from 1-24. Default: 2.
Rows	0	rows	Valid values range from 1-16. Default: 1.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Graph prototype* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Graph prototype* widget

Configure a *Graph prototype* widget that displays a grid of 3 graphs (3 columns, 1 row) created from an item prototype (ID: ”42316”) by low-level discovery.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "graphprototype",
                        "name": "Graph prototype",
                        "x": 0,
                        "y": 0,
                        "width": 16,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "source_type",
                                "value": 3
```

```
                            },
                            {
                                "type": 5,
                                "name": "itemid",
                                "value": 42316
                            },
                            {
                                "type": 0,
                                "name": "columns",
                                "value": 3
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

11 Host availability

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Host availability* widget in `dashboard.create` and `dashboard.update` methods.

Parameters

The following parameters are supported for the *Host availability* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - *(default)* 15 minutes.
Host groups	2	groupids	Host group ID. Note: To configure multiple host groups, create a dashboard widget field object for each host group.
Interface type	0	interface_type	0 - None; 1 - Zabbix agent; 2 - SNMP; 3 - IPMI; 4 - JMX. Default: 1, 2, 3, 4 (all enabled). Note: To configure multiple values, create a dashboard widget field object for each value.
Layout	0	layout	0 - *(default)* Horizontal; 1 - Vertical.
Show hosts in maintenance	0	maintenance	0 - *(default)* Disabled; 1 - Enabled.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Host availability* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Host availability* widget

Configure a *Host availability* widget that displays availability information (in a vertical layout) for hosts in host group "4" with "Zabbix agent" and "SNMP" interfaces configured.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "hostavail",
                        "name": "Host availability",
                        "x": 0,
                        "y": 0,
                        "width": 6,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids",
                                "value": 4
                            },
                            {
```

```
                                "type": 0,
                                "name": "interface_type",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "interface_type",
                                "value": 2
                            },
                            {
                                "type": 0,
                                "name": "layout",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

12 Item value

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Item value* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Item value* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Item value* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Item	4	itemid	Item ID. Parameter behavior: - *required*
Show	0	show	1 - Description; 2 - Value; 3 - Time; 4 - Change indicator. Default: 1, 2, 3, 4 (all enabled). Note: To configure multiple values, create a dashboard widget field object for each value.
Enable host selection	0	dynamic	0 - *(default)* Disabled; 1 - Enabled.
Advanced configuration	0	adv_conf	0 - *(default)* Disabled; 1 - Enabled.

Advanced configuration

The following parameters are supported if *Advanced configuration* is set to "Enabled".

Note:
The number in the *Thresholds* property name (e.g. thresholds.color.0) references the threshold place in a list, sorted in ascending order. However, if thresholds are configured in a different order, the values will be sorted in ascending order after updating widget configuration in Zabbix frontend (e.g. `"threshold.threshold.0":"5"` → `"threshold.threshold.0":"1"`; `"threshold.threshold.1":"1"` → `"threshold.threshold.1": "5"`).

Parameter	type	name	value
Background color	1	bg_color	Hexadecimal color code (e.g. FF0000). Default: "" (empty).
Thresholds			
Color	1	thresholds.color.0	Hexadecimal color code (e.g. FF0000).
Threshold	1	thresholds.threshold.0	Any string value.

Description

The following parameters are supported if *Advanced configuration* is set to "Enabled", and *Show* is set to "Description".

Parameter	type	name	value
Description	1	description	Any string value, including macros. Supported macros: {HOST.*}, {ITEM.*}, {INVENTORY.*}, User macros. Default: {ITEM.NAME}.
Horizontal position	0	desc_h_pos	0 - Left; 1 - *(default)* Center; 2 - Right. Two or more elements (Description, Value, Time) cannot share the same *Horizontal position* and *Vertical position*.
Vertical position	0	desc_v_pos	0 - Top; 1 - Middle; 2 - *(default)* Bottom. Two or more elements (Description, Value, Time) cannot share the same *Horizontal position* and *Vertical position*.
Size	0	desc_size	Valid values range from 1-100. Default: 15.
Bold	0	desc_bold	0 - *(default)* Disabled; 1 - Enabled.
Color	1	desc_color	Hexadecimal color code (e.g. FF0000). Default: "" (empty).

Value

The following parameters are supported if *Advanced configuration* is set to ”Enabled”, and *Show* is set to ”Value”.

Parameter	type	name	value
Decimal places			
Decimal places	0	decimal_places	Valid values range from 1-10. Default: 2.
Size	0	decimal_size	Valid values range from 1-100. Default: 35.
Position			
Horizontal position	0	value_h_pos	0 - Left; 1 - *(default)* Center; 2 - Right. Two or more elements (Description, Value, Time) cannot share the same *Horizontal position* and *Vertical position*.
Vertical position	0	value_v_pos	0 - Top; 1 - *(default)* Middle; 2 - Bottom. Two or more elements (Description, Value, Time) cannot share the same *Horizontal position* and *Vertical position*.
Size	0	value_size	Valid values range from 1-100. Default: 45.
Bold	0	value_bold	0 - Disabled; 1 - *(default)* Enabled.
Color	1	value_color	Hexadecimal color code (e.g. FF0000). Default: "" (empty).
Units			

Parameter	type	name	value
Units (checkbox)	0	units_show	0 - Disabled; 1 - *(default)* Enabled.
Units (value)	1	units	Any string value.
Position	0	units_pos	0 - Before value; 1 - Above value; 2 - *(default)* After value; 3 - Below value.
Size	0	units_size	Valid values range from 1-100. Default: 35.
Bold	0	units_bold	0 - Disabled; 1 - *(default)* Enabled.
Color	1	units_color	Hexadecimal color code (e.g. FF0000). Default: "" (empty).

Time

The following parameters are supported if *Advanced configuration* is set to ”Enabled”, and *Show* is set to ”Time”.

Parameter	type	name	value
Horizontal position	0	time_h_pos	0 - Left; 1 - *(default)* Center; 2 - Right. Two or more elements (Description, Value, Time) cannot share the same *Horizontal position* and *Vertical position*.
Vertical position	0	time_v_pos	0 - *(default)* Top; 1 - Middle; 2 - Bottom. Two or more elements (Description, Value, Time) cannot share the same *Horizontal position* and *Vertical position*.
Size	0	time_size	Valid values range from 1-100. Default: 15.
Bold	0	time_bold	0 - *(default)* Disabled; 1 - Enabled.
Color	1	time_color	Hexadecimal color code (e.g. FF0000). Default: "" (empty).

Change indicator

The following parameters are supported if *Advanced configuration* is set to ”Enabled”, and *Show* is set to ”Change indicator”.

Parameter	type	name	value
Change indicator ↑ color	1	up_color	Hexadecimal color code (e.g. FF0000). Default: "" (empty).
Change indicator ↓ color	1	down_color	Hexadecimal color code (e.g. FF0000). Default: "" (empty).
Change indicator ↕ color	1	updown_color	Hexadecimal color code (e.g. FF0000). Default: "" (empty).

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Item value* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring an *Item value* widget

Configure an *Item value* widget that displays the item value for the item "42266" (Zabbix agent availability). In addition, visually fine-tune the widget with multiple advanced options, including a dynamic background color that changes based on the availability status of Zabbix agent.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "item",
                        "name": "Item value",
                        "x": 0,
                        "y": 0,
                        "width": 4,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 4,
                                "name": "itemid",
                                "value": 42266
                            },
                            {
                                "type": 0,
                                "name": "show",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "show",
                                "value": 2
                            },
                            {
                                "type": 0,
                                "name": "show",
                                "value": 3
                            },
                            {
                                "type": 0,
                                "name": "adv_conf",
                                "value": 1
                            },
                            {
                                "type": 1,
                                "name": "description",
                                "value": "Agent status"
                            },
                            {
                                "type": 0,
                                "name": "desc_h_pos",
                                "value": 0
                            },
```

```
                {
                    "type": 0,
                    "name": "desc_v_pos",
                    "value": 0
                },
                {
                    "type": 0,
                    "name": "desc_bold",
                    "value": 1
                },
                {
                    "type": 1,
                    "name": "desc_color",
                    "value": "F06291"
                },
                {
                    "type": 0,
                    "name": "value_h_pos",
                    "value": 0
                },
                {
                    "type": 0,
                    "name": "value_size",
                    "value": 25
                },
                {
                    "type": 1,
                    "name": "value_color",
                    "value": "FFFF00"
                },
                {
                    "type": 0,
                    "name": "units_show",
                    "value": 0
                },
                {
                    "type": 0,
                    "name": "time_h_pos",
                    "value": 2
                },
                {
                    "type": 0,
                    "name": "time_v_pos",
                    "value": 2
                },
                {
                    "type": 0,
                    "name": "time_size",
                    "value": 10
                },
                {
                    "type": 0,
                    "name": "time_bold",
                    "value": 1
                },
                {
                    "type": 1,
                    "name": "time_color",
                    "value": "9FA8DA"
                },
                {
                    "type": 1,
```

```
                                "name": "thresholds.color.0",
                                "value": "E1E1E1"
                            },
                            {
                                "type": 1,
                                "name": "thresholds.threshold.0",
                                "value": "0"
                            },
                            {
                                "type": 1,
                                "name": "thresholds.color.1",
                                "value": "D1C4E9"
                            },
                            {
                                "type": 1,
                                "name": "thresholds.threshold.1",
                                "value": "1"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```


Response:


```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```


See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

13 Map

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Map* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Map* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Map* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - *(default)* 15 minutes.
Source type	0	source_type	1 - *(default)* Map; 2 - Map navigation tree.
Map	8	sysmapid	Map ID. Parameter behavior: - *required* if *Source type* is set to ”Map”
Linked widget reference	1	filter_widget_reference	Valid *Map navigation tree* widget parameter *Reference* value. Parameter behavior: - *required* if *Source type* is set to ”Map navigation tree”

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Map* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Map* widget

Configure a *Map* widget that displays the map ”1”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "map",
                        "name": "Map",
                        "x": 0,
                        "y": 0,
                        "width": 18,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 8,
                                "name": "sysmapid",
                                "value": 1
                            }
```

```
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

Configuring a linked *Map* widget

Configure a *Map* widget that is linked to a *Map navigation tree* widget.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "map",
                        "name": "Map",
                        "x": 0,
                        "y": 5,
                        "width": 18,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "source_type",
                                "value": 2
                            },
                            {
                                "type": 1,
```

```
                    "name": "filter_widget_reference",
                    "value": "ABCDE"
                }
            ]
        },
        {
            "type": "navtree",
            "name": "Map navigation tree",
            "x": 0,
            "y": 0,
            "width": 6,
            "height": 5,
            "view_mode": 0,
            "fields": [
                {
                    "type": 1,
                    "name": "navtree.name.1",
                    "value": "Element A"
                },
                {
                    "type": 1,
                    "name": "navtree.name.2",
                    "value": "Element B"
                },
                {
                    "type": 1,
                    "name": "navtree.name.3",
                    "value": "Element C"
                },
                {
                    "type": 1,
                    "name": "navtree.name.4",
                    "value": "Element A1"
                },
                {
                    "type": 1,
                    "name": "navtree.name.5",
                    "value": "Element A2"
                },
                {
                    "type": 1,
                    "name": "navtree.name.6",
                    "value": "Element B1"
                },
                {
                    "type": 1,
                    "name": "navtree.name.7",
                    "value": "Element B2"
                },
                {
                    "type": 0,
                    "name": "navtree.parent.4",
                    "value": 1
                },
                {
                    "type": 0,
                    "name": "navtree.parent.5",
                    "value": 1
                },
                {
                    "type": 0,
                    "name": "navtree.parent.6",
```

```
                        "value": 2
                    },
                    {
                        "type": 0,
                        "name": "navtree.parent.7",
                        "value": 2
                    },
                    {
                        "type": 0,
                        "name": "navtree.order.1",
                        "value": 1
                    },
                    {
                        "type": 0,
                        "name": "navtree.order.2",
                        "value": 2
                    },
                    {
                        "type": 0,
                        "name": "navtree.order.3",
                        "value": 3
                    },
                    {
                        "type": 0,
                        "name": "navtree.order.4",
                        "value": 1
                    },
                    {
                        "type": 0,
                        "name": "navtree.order.5",
                        "value": 2
                    },
                    {
                        "type": 0,
                        "name": "navtree.order.6",
                        "value": 1
                    },
                    {
                        "type": 0,
                        "name": "navtree.order.7",
                        "value": 2
                    },
                    {
                        "type": 8,
                        "name": "navtree.sysmapid.6",
                        "value": 1
                    },
                    {
                        "type": 1,
                        "name": "reference",
                        "value": "ABCDE"
                    }
                ]
            }
        ]
    }
],
"userGroups": [
    {
        "usrgrpid": 7,
        "permission": 2
    }
```

```
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`
- Map navigation tree

14 Map navigation tree

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Map navigation tree* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Map navigation tree* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Map navigation tree* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - *(default)* 15 minutes.
Show unavailable maps	1	show_unavailable	0 - *(default)* Disabled; 1 - Enabled.

Parameter	type	name	value
Reference	1	reference	Any string value consisting of 5 characters (e.g. `ABCDE`, `JBPNL`, etc.). Parameter *Reference* value is used in the *Map* widget (*Linked widget reference*) for linking with the *Map navigation tree* widget. Parameter behavior: - *required*

The following parameters are supported for configuring map navigation tree elements.

Parameter	type	name	value
Name	1	navtree.name.1	Any string value. Note: The number in the property name sets the element number.
Linked map	8	navtree.sysmapid.1	Map ID. Note: The number in the property name references the element to which the map is linked.
Parameters for creating element hierarchy	0	navtree.parent.1	Parent element number. Note: The number in the property name references the child element. The property value references the parent element.
	0	navtree.order.1	Element position in the map navigation tree. Note: The number in the property name references the element number. The property value references the element position in the map navigation tree. Parent element position is determined within the whole map navigation tree. Child element position is determined within the parent element.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Map navigation tree* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Map navigation tree* widget

Configure a *Map navigation tree* widget that displays the following map navigation tree:

- Element A
  - Element A1
  - Element A2
- Element B
  - Element B1 (contains linked map "1" that can be displayed in a linked *Map widget*)
  - Element B2
- Element C

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "navtree",
                        "name": "Map navigation tree",
```

```
                "x": 0,
                "y": 0,
                "width": 6,
                "height": 5,
                "view_mode": 0,
                "fields": [
                    {
                        "type": 1,
                        "name": "navtree.name.1",
                        "value": "Element A"
                    },
                    {
                        "type": 1,
                        "name": "navtree.name.2",
                        "value": "Element B"
                    },
                    {
                        "type": 1,
                        "name": "navtree.name.3",
                        "value": "Element C"
                    },
                    {
                        "type": 1,
                        "name": "navtree.name.4",
                        "value": "Element A1"
                    },
                    {
                        "type": 1,
                        "name": "navtree.name.5",
                        "value": "Element A2"
                    },
                    {
                        "type": 1,
                        "name": "navtree.name.6",
                        "value": "Element B1"
                    },
                    {
                        "type": 1,
                        "name": "navtree.name.7",
                        "value": "Element B2"
                    },
                    {
                        "type": 0,
                        "name": "navtree.parent.4",
                        "value": 1
                    },
                    {
                        "type": 0,
                        "name": "navtree.parent.5",
                        "value": 1
                    },
                    {
                        "type": 0,
                        "name": "navtree.parent.6",
                        "value": 2
                    },
                    {
                        "type": 0,
                        "name": "navtree.parent.7",
                        "value": 2
                    },
                    {
```

```
                            "type": 0,
                            "name": "navtree.order.1",
                            "value": 1
                        },
                        {
                            "type": 0,
                            "name": "navtree.order.2",
                            "value": 2
                        },
                        {
                            "type": 0,
                            "name": "navtree.order.3",
                            "value": 3
                        },
                        {
                            "type": 0,
                            "name": "navtree.order.4",
                            "value": 1
                        },
                        {
                            "type": 0,
                            "name": "navtree.order.5",
                            "value": 2
                        },
                        {
                            "type": 0,
                            "name": "navtree.order.6",
                            "value": 1
                        },
                        {
                            "type": 0,
                            "name": "navtree.order.7",
                            "value": 2
                        },
                        {
                            "type": 8,
                            "name": "navtree.sysmapid.6",
                            "value": 1
                        },
                        {
                            "type": 1,
                            "name": "reference",
                            "value": "ABCDE"
                        }
                    ]
                }
            ]
        }
    ],
    "userGroups": [
        {
            "usrgrpid": 7,
            "permission": 2
        }
    ],
    "users": [
        {
            "userid": 1,
            "permission": 3
        }
    ]
},
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`
- Map

15 Plain text

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Plain text* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Plain text* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Plain text* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Items	4	itemids	Item ID. Note: To configure multiple items, create a dashboard widget field object for each item. Parameter behavior: - *required*
Items location	0	style	0 - *(default)* Left; 1 - Top.
Show lines	0	show_lines	Valid values range from 1-100. Default: 25.
Show text as HTML	0	show_as_html	0 - *(default)* Disabled; 1 - Enabled.

Parameter	type	name	value
Enable host selection	0	dynamic	0 - *(default)* Disabled; 1 - Enabled.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Plain text* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Plain text* widget

Configure a *Plain text* widget that displays latest data for items "42269" and "42253". In addition, configure the item names to be located at the top of the data columns, and only 15 lines of data to be displayed.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "plaintext",
                        "name": "Plain text",
                        "x": 0,
                        "y": 0,
                        "width": 6,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 4,
                                "name": "itemids",
                                "value": 42269
                            },
                            {
                                "type": 4,
                                "name": "itemids",
                                "value": 42253
                            },
                            {
                                "type": 0,
                                "name": "style",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "show_lines",
                                "value": 15
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
```

```
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

16 Problem hosts

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Problem hosts* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Problem hosts* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Problem hosts* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Host groups	2	groupids	Host group ID. Note: To configure multiple host groups, create a dashboard widget field object for each host group.
Exclude host groups	2	exclude_groupids	Host group ID. Note: To exclude multiple host groups, create a dashboard widget field object for each host group.

Parameter	type	name	value
Hosts	3	hostids	Host ID. Note: To configure multiple hosts, create a dashboard widget field object for each host. For multiple hosts, the parameter *Host groups* must either be not configured at all or configured with at least one host group that the configured hosts belong to.
Problem	1	problem	Problem event name (case insensitive, full name or part of it).
Severity	0	severities	0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster. Default: empty (all enabled). Note: To configure multiple values, create a dashboard widget field object for each value.
Tags (the number in the property name (e.g. tags.tag.0) references tag order in the tag evaluation list)			
Evaluation type	0	evaltype	0 - *(default)* And/Or; 2 - Or.
Tag name	1	tags.tag.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Operator	0	tags.operator.0	0 - Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist. Parameter behavior: - *required* if configuring *Tags*
Tag value	1	tags.value.0	Any string value. Parameter behavior: - *required* if configuring *Tags*

Parameter	type	name	value
Show suppressed problems	0	show_suppressed	0 - *(default)* Disabled; 1 - Enabled.
Hide groups without problems	0	hide_empty_groups	0 - *(default)* Disabled; 1 - Enabled.
Problem display	0	ext_ack	0 - *(default)* All; 1 - Unacknowledged only; 2 - Separated.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Problem hosts* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Problem hosts* widget

Configure a *Problem hosts* widget that displays hosts from host groups "2" and "4" that have problems with a name that includes the string "CPU" and that have the following severities: "Warning", "Average", "High", "Disaster".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "problemhosts",
                        "name": "Problem hosts",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids",
                                "value": 2
                            },
                            {
                                "type": 2,
                                "name": "groupids",
                                "value": 4
                            },
                            {
                                "type": 1,
                                "name": "problem",
                                "value": "cpu"
                            },
                            {
                                "type": 0,
```

```
                                "name": "severities",
                                "value": 2
                            },
                            {
                                "type": 0,
                                "name": "severities",
                                "value": 3
                            },
                            {
                                "type": 0,
                                "name": "severities",
                                "value": 4
                            },
                            {
                                "type": 0,
                                "name": "severities",
                                "value": 5
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

17 Problems

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Problems* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Problems* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Problems* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Show	0	show	1 - *(default)* Recent problems; 2 - History; 3 - Problems.
Host groups	2	groupids	Host group ID. Note: To configure multiple host groups, create a dashboard widget field object for each host group.
Exclude host groups	2	exclude_groupids	Host group ID. Note: To exclude multiple host groups, create a dashboard widget field object for each host group.
Hosts	3	hostids	Host ID. Note: To configure multiple hosts, create a dashboard widget field object for each host. For multiple hosts, the parameter *Host groups* must either be not configured at all or configured with at least one host group that the configured hosts belong to.
Problem	1	problem	Problem event name (case insensitive, full name or part of it).
Severity	0	severities	0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster. Default: empty (all enabled). Note: To configure multiple values, create a dashboard widget field object for each value.

Parameter	type	name	value
Tags (the number in the property name (e.g. tags.tag.0) references tag order in the tag evaluation list)			
Evaluation type	0	evaltype	0 - *(default)* And/Or; 2 - Or.
Tag name	1	tags.tag.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Operator	0	tags.operator.0	0 - Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist. Parameter behavior: - *required* if configuring *Tags*
Tag value	1	tags.value.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Show tags	0	show_tags	0 - *(default)* None; 1 - 1; 2 - 2; 3 - 3.
Tag name (format)	0	tag_name_format	0 - *(default)* Full; 1 - Shortened; 2 - None. Parameter *Tag name* (format) not available if *Show tags* is set to "None".
Tag display priority	1	tag_priority	Comma-separated list of tags. Parameter *Tag display priority* not available if *Show tags* is set to "None".

Parameter	type	name	value
Show operational data	0	show_opdata	0 - *(default)* None; 1 - Separately; 2 - With problem name.
Show suppressed problems	0	show_suppressed	0 - *(default)* Disabled; 1 - Enabled.
Show unacknowledged only	0	unacknowledged	0 - *(default)* Disabled; 1 - Enabled.
Sort entries by	0	sort_triggers	1 - Severity (descending); 2 - Host (ascending); 3 - Time (ascending); 4 - *(default)* Time (descending); 13 - Severity (ascending); 14 - Host (descending); 15 - Problem (ascending); 16 - Problem (descending). For all values, except ”Time (descending)” and ”Time (ascending)”, the parameter *Show timeline* must be set to ”Disabled”.
Show timeline	0	show_timeline	0 - Disabled; 1 - *(default)* Enabled. Parameter *Show timeline* available if *Sort entries by* is set to ”Time (descending)” or ”Time (ascending)”.
Show lines	0	show_lines	Valid values range from 1-100. Default: 25.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Problems* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Problems* widget

Configure a *Problems* widget that displays problems for host group ”4” that satisfy the following conditions:

- Problems that have a tag with the name ”scope” that contains values ”performance” or ”availability”, or ”capacity”.
- Problems that have the following severities: ”Warning”, ”Average”, ”High”, ”Disaster”.

In addition, configure the widget to show tags and operational data.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
```

```
                "type": "problems",
                "name": "Problems",
                "x": 0,
                "y": 0,
                "width": 12,
                "height": 5,
                "view_mode": 0,
                "fields": [
                    {
                        "type": 2,
                        "name": "groupids",
                        "value": 4
                    },
                    {
                        "type": 1,
                        "name": "tags.tag.0",
                        "value": "scope"
                    },
                    {
                        "type": 0,
                        "name": "tags.operator.0",
                        "value": 0
                    },
                    {
                        "type": 1,
                        "name": "tags.value.0",
                        "value": "performance"
                    },
                    {
                        "type": 1,
                        "name": "tags.tag.1",
                        "value": "scope"
                    },
                    {
                        "type": 0,
                        "name": "tags.operator.1",
                        "value": 0
                    },
                    {
                        "type": 1,
                        "name": "tags.value.1",
                        "value": "availability"
                    },
                    {
                        "type": 1,
                        "name": "tags.tag.2",
                        "value": "scope"
                    },
                    {
                        "type": 0,
                        "name": "tags.operator.2",
                        "value": 0
                    },
                    {
                        "type": 1,
                        "name": "tags.value.2",
                        "value": "capacity"
                    },
                    {
                        "type": 0,
                        "name": "severities",
                        "value": 2
```

```
                            },
                            {
                                "type": 0,
                                "name": "severities",
                                "value": 3
                            },
                            {
                                "type": 0,
                                "name": "severities",
                                "value": 4
                            },
                            {
                                "type": 0,
                                "name": "severities",
                                "value": 5
                            },
                            {
                                "type": 0,
                                "name": "show_tags",
                                "value": 1
                            },
                            {
                                "type": 0,
                                "name": "show_opdata",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- dashboard.create
- dashboard.update

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Problems by severity* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Problems by severity* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Problems by severity* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Host groups	2	groupids	Host group ID. Note: To configure multiple host groups, create a dashboard widget field object for each host group.
Exclude host groups	2	exclude_groupids	Host group ID. Note: To exclude multiple host groups, create a dashboard widget field object for each host group.
Hosts	3	hostids	Host ID. Note: To configure multiple hosts, create a dashboard widget field object for each host. For multiple hosts, the parameter *Host groups* must either be not configured at all or configured with at least one host group that the configured hosts belong to.
Problem	1	problem	Problem event name (case insensitive, full name or part of it).
Severity	0	severities	0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster. Default: empty (all enabled). Note: To configure multiple values, create a dashboard widget field object for each value.

Parameter	type	name	value
Tags (the number in the property name (e.g. tags.tag.0) references tag order in the tag evaluation list)			
Evaluation type	0	evaltype	0 - *(default)* And/Or; 2 - Or.
Tag name	1	tags.tag.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Operator	0	tags.operator.0	0 - Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist. Parameter behavior: - *required* if configuring *Tags*
Tag value	1	tags.value.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Show	0	show_type	0 - *(default)* Host groups; 1 - Totals.
Layout	0	layout	0 - *(default)* Horizontal; 1 - Vertical. Parameter *Layout* not available if *Show* is set to ”Host groups”.
Show operational data	0	show_opdata	0 - *(default)* None; 1 - Separately; 2 - With problem name.
Show suppressed problems	0	show_suppressed	0 - *(default)* Disabled; 1 - Enabled.

Parameter	type	name	value
Hide groups without problems	0	hide_empty_groups	0 - *(default)* Disabled; 1 - Enabled. Parameter *Hide groups without problems* not available if *Show* is set to ”Totals”.
Problem display	0	ext_ack	0 - *(default)* All; 1 - Unacknowledged only; 2 - Separated.
Show timeline	0	show_timeline	0 - Disabled; 1 - *(default)* Enabled.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Problems by severity* widget. For more information on configuring a dashboard, see dashboard.create.

Configuring a *Problems by severity* widget

Configure a *Problems by severity* widget that displays problem totals for all host groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "problemsbysv",
                        "name": "Problems by severity",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "show_type",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
```

```
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

19 SLA report

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *SLA report* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *SLA report* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *SLA report* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - *(default)* No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
SLA	10	slaid	SLA ID. Parameter behavior: - *required*
Service	9	serviceid	Service ID.
Show periods	0	show_periods	Valid values range from 1-100. Default: 20.
From	1	date_from	Valid date string in format `YYYY-MM-DD`. Relative dates with modifiers `d`, `w`, `M`, `y` (e.g. `now`, `now/d`, `now/w-1w`, etc.) are supported.

Parameter	type	name	value
To	1	date_to	Valid date string in format `YYYY-MM-DD`. Relative dates with modifiers `d`, `w`, `M`, `y` (e.g. `now`, `now/d`, `now/w-1w`, etc.) are supported.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *SLA report* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring an *SLA report* widget

Configure an *SLA report* widget that displays the SLA report for SLA "4" service "2" for the period of last 30 days.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "slareport",
                        "name": "SLA report",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 10,
                                "name": "slaid",
                                "value": 4
                            },
                            {
                                "type": 9,
                                "name": "serviceid",
                                "value": 2
                            },
                            {
                                "type": 1,
                                "name": "date_from",
                                "value": "now-30d"
                            },
                            {
                                "type": 1,
                                "name": "date_to",
                                "value": "now"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
```

```
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

20 System information

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *System Information* widget in `dashboard.create` and `dashboard.update` methods.

Parameters

The following parameters are supported for the *System Information* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - *(default)* 15 minutes.
Show	0	info_type	0 - *(default)* System stats; 1 - High availability nodes.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *System information* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *System information* widget

Configure a *System information* widget that displays system stats with a refresh interval of 10 minutes.

Request:

```
{
    "jsonrpc": "2.0",
```

```
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "systeminfo",
                        "name": "System information",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 0,
                                "name": "rf_rate",
                                "value": 600
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- dashboard.create
- dashboard.update

21 Top hosts

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Top hosts* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Top hosts* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Top hosts* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Host groups	2	groupids	Host group ID. Note: To configure multiple host groups, create a dashboard widget field object for each host group.
Hosts	3	hostids	Host ID. Note: To configure multiple hosts, create a dashboard widget field object for each host. For multiple hosts, the parameter *Host groups* must either be not configured at all or configured with at least one host group that the configured hosts belong to.
Host Tags (the number in the property name (e.g. tags.tag.0) references tag order in the tag evaluation list)			
Evaluation type	0	evaltype	0 - *(default)* And/Or; 2 - Or.

Parameter	type	name	value
Tag name	1	tags.tag.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Operator	0	tags.operator.0	0 - Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist. Parameter behavior: - *required* if configuring *Tags*
Tag value	1	tags.value.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Columns (see below)			
Order	0	order	2 - *(default)* Top N; 3 - Bottom N.
Order column	0	column	Column numeric value from the configured columns.
Host count	0	count	Valid values range from 1-100. Default: 10.

Columns

Columns have common parameters and additional parameters depending on the configuration of the parameter *Data*.

Note:
For all parameters related to columns the number in the property name (e.g. columns.name.0) references a column for which the parameter is configured.

The following parameters are supported for all columns.

Parameter	type	name	value
Name	1	columns.name.0	Any string value.
Data	0	columns.data.0	1 - Item value; 2 - Host name; 3 - Text. Parameter behavior: - *required*
Base color	1	columns.base_color.0	Hexadecimal color code (e.g. FF0000). Parameter behavior: - *required*

Item value

The following parameters are supported if *Data* is set to "Item value".

Note:
The first number in the *Thresholds* property name (e.g. columnsthresholds.color.0.0) references the column for which thresholds are configured, while the second number references threshold place in a list, sorted in ascending order. However, if thresholds are configured in a different order, the values will be sorted in ascending order after updating widget configuration in Zabbix frontend (e.g. `"threshold.threshold.0":"5"` → `"threshold.threshold.0":"1"`; `"threshold.threshold.1":"1"` → `"threshold.threshold.1": "5"`).

Parameter	type	name	value
Item	1	columns.item.0	Valid item name.
Time shift	1	columns.timeshift.0	Valid numeric or time string value (e.g. 3600 or 1h). You may use time suffixes. Negative values are allowed. Parameter behavior: - *required*
Aggregation function	0	columns.aggregate_function.0	0 - *(default)* none; 1 - min; 2 - max; 3 - avg; 4 - count; 5 - sum; 6 - first; 7 - last.
Aggregation interval	1	columns.aggregate_interval.0	Valid time string (e.g. 3600, 1h, etc.). You may use time suffixes. Parameter *Aggregation interval* not available if *Aggregation function* is set to *none*. Default: 1h.
Display	0	columns.display.0	1 - *(default)* As is; 2 - Bar; 3 - Indicators.
Min	1	columns.min.0	Any numeric value. Parameter *Min* not available if *Display* is set to "As is".
Max	1	columns.max.0	Any numeric value. Parameter *Max* not available if *Display* is set to "As is".
Decimal places	0	columns.decimal_places.0	Valid values range from 0-10. Default: 2.
History data	0	columns.history.0	1 - *(default)* Auto; 2 - History; 3 - Trends.
Thresholds			
Color	1	columnsthresholds.color.0.0	Hexadecimal color code (e.g. FF0000). Default: "" (empty).
Threshold	1	columnsthresholds.threshold.0.0	Any string value.

Text

The following parameters are supported if *Data* is set to "Text".

Parameter	type	name	value
Text	1	columns.text.0	Any string value, including macros. Supported macros: {HOST.*}, {INVENTORY.*}. Parameter behavior: - *required* if *Data* is set to "Text"

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Top hosts* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Top hosts* widget

Configure a *Top hosts* widget that displays top hosts by CPU utilization in host group "4". In addition, configure the following custom columns: "Host name", "CPU utilization in %", "1m avg", "5m avg", "15m avg", "Processes".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "tophosts",
                        "name": "Top hosts",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids",
                                "value": 4
                            },
                            {
                                "type": 1,
                                "name": "columns.name.0",
                                "value": ""
                            },
                            {
                                "type": 0,
                                "name": "columns.data.0",
                                "value": 2
                            },
                            {
                                "type": 1,
                                "name": "columns.base_color.0",
                                "value": "FFFFFF"
                            },
                            {
                                "type": 1,
                                "name": "columns.timeshift.0",
                                "value": ""
                            },
                            {
                                "type": 1,
                                "name": "columns.name.1",
                                "value": "CPU utilization in %"
                            },
                            {
                                "type": 0,
                                "name": "columns.data.1",
                                "value": 1
```

```
                },
                {
                    "type": 1,
                    "name": "columns.base_color.1",
                    "value": "4CAF50"
                },
                {
                    "type": 1,
                    "name": "columns.item.1",
                    "value": "CPU utilization"
                },
                {
                    "type": 1,
                    "name": "columns.timeshift.1",
                    "value": ""
                },
                {
                    "type": 0,
                    "name": "columns.display.1",
                    "value": 3
                },
                {
                    "type": 1,
                    "name": "columns.min.1",
                    "value": "0"
                },
                {
                    "type": 1,
                    "name": "columns.max.1",
                    "value": "100"
                },
                {
                    "type": 1,
                    "name": "columnsthresholds.color.1.0",
                    "value": "FFFF00"
                },
                {
                    "type": 1,
                    "name": "columnsthresholds.threshold.1.0",
                    "value": "50"
                },
                {
                    "type": 1,
                    "name": "columnsthresholds.color.1.1",
                    "value": "FF8000"
                },
                {
                    "type": 1,
                    "name": "columnsthresholds.threshold.1.1",
                    "value": "80"
                },
                {
                    "type": 1,
                    "name": "columnsthresholds.color.1.2",
                    "value": "FF4000"
                },
                {
                    "type": 1,
                    "name": "columnsthresholds.threshold.1.2",
                    "value": "90"
                },
                {
```

```
            "type": 1,
            "name": "columns.name.2",
            "value": "1m avg"
        },
        {
            "type": 0,
            "name": "columns.data.2",
            "value": 1
        },
        {
            "type": 1,
            "name": "columns.base_color.2",
            "value": "FFFFFF"
        },
        {
            "type": 1,
            "name": "columns.item.2",
            "value": "Load average (1m avg)"
        },
        {
            "type": 1,
            "name": "columns.timeshift.2",
            "value": ""
        },
        {
            "type": 1,
            "name": "columns.name.3",
            "value": "5m avg"
        },
        {
            "type": 0,
            "name": "columns.data.3",
            "value": 1
        },
        {
            "type": 1,
            "name": "columns.base_color.3",
            "value": "FFFFFF"
        },
        {
            "type": 1,
            "name": "columns.item.3",
            "value": "Load average (5m avg)"
        },
        {
            "type": 1,
            "name": "columns.timeshift.3",
            "value": ""
        },
        {
            "type": 1,
            "name": "columns.name.4",
            "value": "15m avg"
        },
        {
            "type": 0,
            "name": "columns.data.4",
            "value": 1
        },
        {
            "type": 1,
            "name": "columns.base_color.4",
```

```
                            "value": "FFFFFF"
                        },
                        {
                            "type": 1,
                            "name": "columns.item.4",
                            "value": "Load average (15m avg)"
                        },
                        {
                            "type": 1,
                            "name": "columns.timeshift.4",
                            "value": ""
                        },
                        {
                            "type": 1,
                            "name": "columns.name.5",
                            "value": "Processes"
                        },
                        {
                            "type": 0,
                            "name": "columns.data.5",
                            "value": 1
                        },
                        {
                            "type": 1,
                            "name": "columns.base_color.5",
                            "value": "FFFFFF"
                        },
                        {
                            "type": 1,
                            "name": "columns.item.5",
                            "value": "Number of processes"
                        },
                        {
                            "type": 1,
                            "name": "columns.timeshift.5",
                            "value": ""
                        },
                        {
                            "type": 0,
                            "name": "columns.decimal_places.5",
                            "value": 0
                        },
                        {
                            "type": 0,
                            "name": "column",
                            "value": 1
                        }
                    ]
                }
            ]
        }
    ],
    "userGroups": [
        {
            "usrgrpid": 7,
            "permission": 2
        }
    ],
    "users": [
        {
            "userid": 1,
            "permission": 3
```

```
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

22 Trigger overview

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Trigger overview* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Trigger overview* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Trigger Overview* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Show	0	show	1 - *(default)* Recent problems; 2 - Any; 3 - Problems.
Host groups	2	groupids	Host group ID. Note: To configure multiple host groups, create a dashboard widget field object for each host group.
Hosts	3	hostids	Host ID. Note: To configure multiple hosts, create a dashboard widget field object for each host. For multiple hosts, the parameter *Host groups* must either be not configured at all or configured with at least one host group that the configured hosts belong to.

Parameter	type	name	value
Tags (the number in the property name (e.g. tags.tag.0) references tag order in the tag evaluation list)			
Evaluation type	0	evaltype	0 - *(default)* And/Or; 2 - Or.
Tag name	1	tags.tag.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Operator	0	tags.operator.0	0 - Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist. Parameter behavior: - *required* if configuring *Tags*
Tag value	1	tags.value.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Show suppressed problems	0	show_suppressed	0 - *(default)* Disabled; 1 - Enabled.
Hosts location	0	style	0 - *(default)* Left; 1 - Top.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Trigger overview* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Trigger overview* widget

Configure a *Trigger overview* widget that displays trigger states for all host groups that have triggers with a tag that has the name

”scope” and contains value ”availability”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "trigover",
                        "name": "Trigger overview",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 1,
                                "name": "tags.tag.0",
                                "value": "scope"
                            },
                            {
                                "type": 0,
                                "name": "tags.operator.0",
                                "value": 0
                            },
                            {
                                "type": 1,
                                "name": "tags.value.0",
                                "value": "availability"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
```

```
                "3"
            ]
        },
        "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

23 URL

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *URL* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *URL* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *URL* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - *(default)* No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
URL	1	url	Valid URL string. Parameter behavior: - *required*
Enable host selection	0	dynamic	0 - *(default)* Disabled; 1 - Enabled.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *URL* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *URL* widget

Configure a *URL* widget that displays the home page of Zabbix manual.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
```

```
                "widgets": [
                    {
                        "type": "url",
                        "name": "URL",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 1,
                                "name": "url",
                                "value": "https://www.zabbix.com/documentation/6.4/en"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

24 Web monitoring

Description

These parameters and the possible property values for the respective dashboard widget field objects allow to configure the *Web monitoring* widget in `dashboard.create` and `dashboard.update` methods.

Attention:
Widget `fields` properties are not validated during the creation or update of a dashboard. This allows users to modify built-in widgets and create custom widgets, but also introduces the risk of creating or updating widgets incorrectly. To ensure the successful creation or update of the *Web monitoring* widget, please refer to the parameter behavior outlined in the tables below.

Parameters

The following parameters are supported for the *Web monitoring* widget.

Parameter	type	name	value
Refresh interval	0	rf_rate	0 - No refresh; 10 - 10 seconds; 30 - 30 seconds; 60 - *(default)* 1 minute; 120 - 2 minutes; 600 - 10 minutes; 900 - 15 minutes.
Host groups	2	groupids	Host group ID. Note: To configure multiple host groups, create a dashboard widget field object for each host group.
Exclude host groups	2	exclude_groupids	Host group ID. Note: To exclude multiple host groups, create a dashboard widget field object for each host group.
Hosts	3	hostids	Host ID. Note: To configure multiple hosts, create a dashboard widget field object for each host. For multiple hosts, the parameter *Host groups* must either be not configured at all or configured with at least one host group that the configured hosts belong to.
Tags (the number in the property name (e.g. tags.tag.0) references tag order in the tag evaluation list)			
Evaluation type	0	evaltype	0 - *(default)* And/Or; 2 - Or.

Parameter	type	name	value
Tag name	1	tags.tag.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Operator	0	tags.operator.0	0 - Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist. Parameter behavior: - *required* if configuring *Tags*
Tag value	1	tags.value.0	Any string value. Parameter behavior: - *required* if configuring *Tags*
Show hosts in maintenance	0	maintenance	0 - Disabled; 1 - *(default)* Enabled.

Examples

The following examples aim to only describe the configuration of the dashboard widget field objects for the *Web monitoring* widget. For more information on configuring a dashboard, see `dashboard.create`.

Configuring a *Web monitoring* widget

Configure a *Web monitoring* widget that displays a status summary of the active web monitoring scenarios for host group "4".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "web",
                        "name": "Web monitoring",
                        "x": 0,
                        "y": 0,
                        "width": 6,
                        "height": 3,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 2,
                                "name": "groupids",
                                "value": 4
                            }
                        ]
                    }
                ]
            }
```

```
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

dashboard.create

Description

`object dashboard.create(object/array dashboards)`

This method allows to create new dashboards.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Dashboards to create.

Additionally to the standard dashboard properties, the method accepts the following parameters.

Parameter	Type	Description
pages	array	Dashboard pages to be created for the dashboard. Dashboard pages will be ordered in the same order as specified. Parameter behavior: *- required*
users	array	Dashboard user shares to be created on the dashboard.
userGroups	array	Dashboard user group shares to be created on the dashboard.

Return values

(object) Returns an object containing the IDs of the created dashboards under the dashboardids property. The order of the returned IDs matches the order of the passed dashboards.

Examples

Creating a dashboard

Create a dashboard named "My dashboard" with one Problems widget with tags and using two types of sharing (user group and user) on a single dashboard page.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.create",
    "params": {
        "name": "My dashboard",
        "display_period": 30,
        "auto_start": 1,
        "pages": [
            {
                "widgets": [
                    {
                        "type": "problems",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 1,
                                "name": "tags.tag.0",
                                "value": "service"
                            },
                            {
                                "type": 0,
                                "name": "tags.operator.0",
                                "value": 1
                            },
                            {
                                "type": 1,
                                "name": "tags.value.0",
                                "value": "zabbix_server"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": "7",
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": "4",
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

See also

- Dashboard page
- Dashboard widget
- Dashboard widget field
- Dashboard user
- Dashboard user group

Source

CDashboard::create() in *ui/include/classes/api/services/CDashboard.php*.

dashboard.delete

Description

`object dashboard.delete(array dashboardids)`

This method allows to delete dashboards.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the dashboards to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted dashboards under the `dashboardids` property.

Examples

Deleting multiple dashboards

Delete two dashboards.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.delete",
    "params": [
        "2",
        "3"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2",
            "3"
```

```
        ]
    },
    "id": 1
}
```

Source

CDashboard::delete() in *ui/include/classes/api/services/CDashboard.php*.

dashboard.get

Description

`integer/array dashboard.get(object parameters)`

The method allows to retrieve dashboards according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dashboardids	string/array	Return only dashboards with the given IDs.
selectPages	query	Return a pages property with dashboard pages, correctly ordered.
selectUsers	query	Return a users property with users that the dashboard is shared with.
selectUserGroups	query	Return a userGroups property with user groups that the dashboard is shared with.
sortfield	string/array	Sort the result by the given properties. Possible values: `dashboardid`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving a dashboard by ID

Retrieve all data about dashboards "1" and "2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.get",
    "params": {
        "output": "extend",
        "selectPages": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "dashboardids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dashboardid": "1",
            "name": "Dashboard",
            "userid": "1",
            "private": "0",
            "display_period": "30",
            "auto_start": "1",
            "users": [],
            "userGroups": [],
            "pages": [
                {
                    "dashboard_pageid": "1",
                    "name": "",
                    "display_period": "0",
                    "widgets": [
                        {
                            "widgetid": "9",
                            "type": "systeminfo",
                            "name": "",
                            "x": "12",
                            "y": "8",
                            "width": "12",
                            "height": "5",
                            "view_mode": "0",
                            "fields": []
                        },
                        {
                            "widgetid": "8",
                            "type": "problemsbysv",
                            "name": "",
                            "x": "12",
                            "y": "4",
                            "width": "12",
                            "height": "4",
                            "view_mode": "0",
                            "fields": []
                        },
                        {
                            "widgetid": "7",
                            "type": "problemhosts",
                            "name": "",
                            "x": "12",
```

```
                "y": "0",
                "width": "12",
                "height": "4",
                "view_mode": "0",
                "fields": []
            },
            {
                "widgetid": "6",
                "type": "discovery",
                "name": "",
                "x": "6",
                "y": "9",
                "width": "6",
                "height": "4",
                "view_mode": "0",
                "fields": []
            },
            {
                "widgetid": "5",
                "type": "web",
                "name": "",
                "x": "0",
                "y": "9",
                "width": "6",
                "height": "4",
                "view_mode": "0",
                "fields": []
            },
            {
                "widgetid": "4",
                "type": "problems",
                "name": "",
                "x": "0",
                "y": "3",
                "width": "12",
                "height": "6",
                "view_mode": "0",
                "fields": []
            },
            {
                "widgetid": "3",
                "type": "favmaps",
                "name": "",
                "x": "8",
                "y": "0",
                "width": "4",
                "height": "3",
                "view_mode": "0",
                "fields": []
            },
            {
                "widgetid": "1",
                "type": "favgraphs",
                "name": "",
                "x": "0",
                "y": "0",
                "width": "4",
                "height": "3",
                "view_mode": "0",
                "fields": []
            }
        ]
```

```
            },
            {
                "dashboard_pageid": "2",
                "name": "",
                "display_period": "0",
                "widgets": []
            },
            {
                "dashboard_pageid": "3",
                "name": "Custom page name",
                "display_period": "60",
                "widgets": []
            }
        ]
    },
    {
        "dashboardid": "2",
        "name": "My dashboard",
        "userid": "1",
        "private": "1",
        "display_period": "60",
        "auto_start": "1",
        "users": [
            {
                "userid": "4",
                "permission": "3"
            }
        ],
        "userGroups": [
            {
                "usrgrpid": "7",
                "permission": "2"
            }
        ],
        "pages": [
            {
                "dashboard_pageid": "4",
                "name": "",
                "display_period": "0",
                "widgets": [
                    {
                        "widgetid": "10",
                        "type": "problems",
                        "name": "",
                        "x": "0",
                        "y": "0",
                        "width": "12",
                        "height": "5",
                        "view_mode": "0",
                        "fields": [
                            {
                                "type": "2",
                                "name": "groupids",
                                "value": "4"
                            }
                        ]
                    }
                ]
            }
        ]
    }
],
```

```
    "id": 1
}
```

See also

- Dashboard page
- Dashboard widget
- Dashboard widget field
- Dashboard user
- Dashboard user group

Source

CDashboard::get() in *ui/include/classes/api/services/CDashboard.php*.

dashboard.update

Description

`object dashboard.update(object/array dashboards)`

This method allows to update existing dashboards.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Dashboard properties to be updated.

The `dashboardid` property must be defined for each dashboard, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard dashboard properties, the method accepts the following parameters.

Parameter	Type	Description
pages	array	Dashboard pages to replace the existing dashboard pages. Dashboard pages are updated by the `dashboard_pageid` property. New dashboard pages will be created for objects without `dashboard_pageid` property and the existing dashboard pages will be deleted if not reused. Dashboard pages will be ordered in the same order as specified. Only the specified properties of the dashboard pages will be updated.
users	array	Dashboard user shares to replace the existing elements.
userGroups	array	Dashboard user group shares to replace the existing elements.

Return values

`(object)` Returns an object containing the IDs of the updated dashboards under the `dashboardids` property.

Examples

Renaming a dashboard

Rename a dashboard to ”SQL server status”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "name": "SQL server status"
    },
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

Updating dashboard pages

Rename the first dashboard page, replace widgets on the second dashboard page and add a new page as the third one. Delete all other dashboard pages.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "pages": [
            {
                "dashboard_pageid": 1,
                "name": "Renamed Page"
            },
            {
                "dashboard_pageid": 2,
                "widgets": [
                    {
                        "type": "clock",
                        "x": 0,
                        "y": 0,
                        "width": 4,
                        "height": 3
                    }
                ]
            },
            {
                "display_period": 60
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

Change dashboard owner

Available only for admins and super admins.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "userid": "1"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

See also

- Dashboard page
- Dashboard widget
- Dashboard widget field
- Dashboard user
- Dashboard user group

Source

CDashboard::update() in *ui/include/classes/api/services/CDashboard.php*.

Discovered host

This class is designed to work with discovered hosts.

Object references:

- Discovered host

Available methods:

- dhost.get - retrieve discovered hosts

> Discovered host object

The following objects are directly related to the `dhost` API.

Discovered host

Note:
Discovered host are created by the Zabbix server and cannot be modified via the API.

The discovered host object contains information about a host discovered by a network discovery rule. It has the following properties.

Property	Type	Description
dhostid	string	ID of the discovered host.
druleid	string	ID of the discovery rule that detected the host.
lastdown	timestamp	Time when the discovered host last went down.
lastup	timestamp	Time when the discovered host last went up.

Property	Type	Description
status	integer	Whether the discovered host is up or down. A host is up if it has at least one active discovered service. Possible values: 0 - host up; 1 - host down.

dhost.get

Description

`integer/array dhost.get(object parameters)`

The method allows to retrieve discovered hosts according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dhostids	string/array	Return only discovered hosts with the given IDs.
druleids	string/array	Return only discovered hosts that have been created by the given discovery rules.
dserviceids	string/array	Return only discovered hosts that are running the given services.
selectDRules	query	Return a drules property with an array of the discovery rules that detected the host.
selectDServices	query	Return a dservices property with the discovered services running on the host. Supports `count`.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectDServices` - results will be sorted by `dserviceid`.
sortfield	string/array	Sort the result by the given properties. Possible values: `dhostid`, `druleid`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;

- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve discovered hosts by discovery rule

Retrieve all hosts and the discovered services they are running that have been detected by discovery rule "4".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dhost.get",
    "params": {
        "output": "extend",
        "selectDServices": "extend",
        "druleids": "4"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dservices": [
                {
                    "dserviceid": "1",
                    "dhostid": "1",
                    "type": "4",
                    "key_": "",
                    "value": "",
                    "port": "80",
                    "status": "0",
                    "lastup": "1337697227",
                    "lastdown": "0",
                    "dcheckid": "5",
                    "ip": "192.168.1.1",
                    "dns": "station.company.lan"
                }
            ],
            "dhostid": "1",
            "druleid": "4",
            "status": "0",
            "lastup": "1337697227",
            "lastdown": "0"
        },
        {
            "dservices": [
                {
                    "dserviceid": "2",
                    "dhostid": "2",
                    "type": "4",
                    "key_": "",
                    "value": "",
                    "port": "80",
                    "status": "0",
                    "lastup": "1337697234",
                    "lastdown": "0",
                    "dcheckid": "5",
                    "ip": "192.168.1.4",
                    "dns": "john.company.lan"
                }
            ],
```

```
            "dhostid": "2",
            "druleid": "4",
            "status": "0",
            "lastup": "1337697234",
            "lastdown": "0"
        },
        {
            "dservices": [
                {
                    "dserviceid": "3",
                    "dhostid": "3",
                    "type": "4",
                    "key_": "",
                    "value": "",
                    "port": "80",
                    "status": "0",
                    "lastup": "1337697234",
                    "lastdown": "0",
                    "dcheckid": "5",
                    "ip": "192.168.1.26",
                    "dns": "printer.company.lan"
                }
            ],
            "dhostid": "3",
            "druleid": "4",
            "status": "0",
            "lastup": "1337697234",
            "lastdown": "0"
        },
        {
            "dservices": [
                {
                    "dserviceid": "4",
                    "dhostid": "4",
                    "type": "4",
                    "key_": "",
                    "value": "",
                    "port": "80",
                    "status": "0",
                    "lastup": "1337697234",
                    "lastdown": "0",
                    "dcheckid": "5",
                    "ip": "192.168.1.7",
                    "dns": "mail.company.lan"
                }
            ],
            "dhostid": "4",
            "druleid": "4",
            "status": "0",
            "lastup": "1337697234",
            "lastdown": "0"
        }
    ],
    "id": 1
}
```


See also

- Discovered service
- Discovery rule

Source

CDHost::get() in *ui/include/classes/api/services/CDHost.php*.

Discovered service

This class is designed to work with discovered services.

Object references:

- Discovered service

Available methods:

- dservice.get - retrieve discovered services

> Discovered service object

The following objects are directly related to the `dservice` API.

Discovered service

Note:
Discovered services are created by the Zabbix server and cannot be modified via the API.

The discovered service object contains information about a service discovered by a network discovery rule on a host. It has the following properties.

Property	Type	Description
dserviceid	string	ID of the discovered service.
dcheckid	string	ID of the discovery check used to detect the service.
dhostid	string	ID of the discovered host running the service.
dns	string	DNS of the host running the service.
ip	string	IP address of the host running the service.
lastdown	timestamp	Time when the discovered service last went down.
lastup	timestamp	Time when the discovered service last went up.
port	integer	Service port number.
status	integer	Status of the service. Possible values: 0 - service up; 1 - service down.
value	string	Value returned by the service when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.

dservice.get

Description

`integer/array dservice.get(object parameters)`

The method allows to retrieve discovered services according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dserviceids	string/array	Return only discovered services with the given IDs.

Parameter	Type	Description
dhostids	string/array	Return only discovered services that belong to the given discovered hosts.
dcheckids	string/array	Return only discovered services that have been detected by the given discovery checks.
druleids	string/array	Return only discovered services that have been detected by the given discovery rules.
selectDRules	query	Return a drules property with an array of the discovery rules that detected the service.
selectDHosts	query	Return a dhosts property with an array the discovered hosts that the service belongs to.
selectHosts	query	Return a hosts property with the hosts with the same IP address and proxy as the service. Supports `count`.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectHosts` - result will be sorted by `hostid`.
sortfield	string/array	Sort the result by the given properties. Possible values: `dserviceid`, `dhostid`, `ip`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve services discovered on a host

Retrieve all discovered services detected on discovered host "11".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dservice.get",
    "params": {
        "output": "extend",
        "dhostids": "11"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
```

```
        {
            "dserviceid": "12",
            "dhostid": "11",
            "value": "",
            "port": "80",
            "status": "1",
            "lastup": "0",
            "lastdown": "1348650607",
            "dcheckid": "5",
            "ip": "192.168.1.134",
            "dns": "john.local"
        },
        {
            "dserviceid": "13",
            "dhostid": "11",
            "value": "",
            "port": "21",
            "status": "1",
            "lastup": "0",
            "lastdown": "1348650610",
            "dcheckid": "6",
            "ip": "192.168.1.134",
            "dns": "john.local"
        }
    ],
    "id": 1
}
```


See also

- Discovered host
- Discovery check
- Host

Source

CDService::get() in *ui/include/classes/api/services/CDService.php*.

Discovery check

This class is designed to work with discovery checks.

Object references:

- Discovery check

Available methods:

- dcheck.get - retrieve discovery checks

> Discovery check object

The following objects are directly related to the `dcheck` API.

Discovery check

The discovery check object defines a specific check performed by a network discovery rule. It has the following properties.

Property	Type	Description
dcheckid	string	ID of the discovery check.
druleid	string	ID of the discovery rule that the check belongs to.

Property	Type	Description
key_	string	Item key (if `type` is set to ”Zabbix agent”) or SNMP OID (if `type` is set to ”SNMPv1 agent”, ”SNMPv2 agent”, or ”SNMPv3 agent”). Property behavior: - *required* if `type` is set to ”Zabbix agent”, ”SNMPv1 agent”, ”SNMPv2 agent”, or ”SNMPv3 agent”
ports	string	One or several port ranges to check, separated by commas. Default: 0. Property behavior: - *supported* if `type` is set to ”SSH” (0), ”LDAP” (1), ”SMTP” (2), ”FTP” (3), ”HTTP” (4), ”POP” (5), ”NNTP” (6), ”IMAP” (7), ”TCP” (8), ”Zabbix agent” (9), ”SNMPv1 agent” (10), ”SNMPv2 agent” (11), ”SNMPv3 agent” (13), ”HTTPS” (14), or ”Telnet” (15)
snmp_community	string	SNMP community. Property behavior: - *required* if `type` is set to ”SNMPv1 agent” or ”SNMPv2 agent”
snmpv3_authpassphrase	string	Authentication passphrase. Property behavior: - *supported* if `type` is set to ”SNMPv3 agent” and `snmpv3_securitylevel` is set to ”authNoPriv” or ”authPriv”
snmpv3_authprotocol	integer	Authentication protocol. Possible values: 0 - *(default)* MD5; 1 - SHA1; 2 - SHA224; 3 - SHA256; 4 - SHA384; 5 - SHA512. Property behavior: - *supported* if `type` is set to ”SNMPv3 agent” and `snmpv3_securitylevel` is set to ”authNoPriv” or ”authPriv”
snmpv3_contextname	string	SNMPv3 context name. Property behavior: - *supported* if `type` is set to ”SNMPv3 agent”
snmpv3_privpassphrase	string	Privacy passphrase. Property behavior: - *supported* if `type` is set to ”SNMPv3 agent” and `snmpv3_securitylevel` is set to ”authPriv”
snmpv3_privprotocol	integer	Privacy protocol. Possible values: 0 - *(default)* DES; 1 - AES128; 2 - AES192; 3 - AES256; 4 - AES192C; 5 - AES256C. Property behavior: - *supported* if `type` is set to ”SNMPv3 agent” and `snmpv3_securitylevel` is set to ”authPriv”

Property	Type	Description
snmpv3_securitylevel	string	Security level. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv. Property behavior: - *supported* if `type` is set to ”SNMPv3 agent”
snmpv3_securityname	string	Security name. Property behavior: - *supported* if `type` is set to ”SNMPv3 agent”
type	integer	Type of check. Possible values: 0 - SSH; 1 - LDAP; 2 - SMTP; 3 - FTP; 4 - HTTP; 5 - POP; 6 - NNTP; 7 - IMAP; 8 - TCP; 9 - Zabbix agent; 10 - SNMPv1 agent; 11 - SNMPv2 agent; 12 - ICMP ping; 13 - SNMPv3 agent; 14 - HTTPS; 15 - Telnet. Property behavior: - *required*
uniq	integer	Whether to use this check as a device uniqueness criteria. Only a single unique check can be configured for a discovery rule. Possible values: 0 - *(default)* do not use this check as a uniqueness criteria; 1 - use this check as a uniqueness criteria. Property behavior: - *supported* if `type` is set to ”Zabbix agent”, ”SNMPv1 agent”, ”SNMPv2 agent”, or ”SNMPv3 agent”
host_source	integer	Source for host name. Possible values: 1 - *(default)* DNS; 2 - IP; 3 - discovery value of this check.
name_source	integer	Source for visible name. Possible values: 0 - *(default)* not specified; 1 - DNS; 2 - IP; 3 - discovery value of this check.

dcheck.get

Description

`integer/array dcheck.get(object parameters)`

The method allows to retrieve discovery checks according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dcheckids	string/array	Return only discovery checks with the given IDs.
druleids	string/array	Return only discovery checks that belong to the given discovery rules.
dserviceids	string/array	Return only discovery checks that have detected the given discovered services.
sortfield	string/array	Sort the result by the given properties. Possible values: `dcheckid`, `druleid`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve discovery checks for a discovery rule

Retrieve all discovery checks used by discovery rule "6".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dcheck.get",
    "params": {
        "output": "extend",
        "dcheckids": "6"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
```

```
        {
            "dcheckid": "6",
            "druleid": "4",
            "type": "3",
            "key_": "",
            "snmp_community": "",
            "ports": "21",
            "snmpv3_securityname": "",
            "snmpv3_securitylevel": "0",
            "snmpv3_authpassphrase": "",
            "snmpv3_privpassphrase": "",
            "uniq": "0",
            "snmpv3_authprotocol": "0",
            "snmpv3_privprotocol": "0",
            "host_source": "1",
            "name_source": "0"
        }
    ],
    "id": 1
}
```


Source

CDCheck::get() in *ui/include/classes/api/services/CDCheck.php*.

Discovery rule

This class is designed to work with network discovery rules.

> **Note:**
> This API is meant to work with network discovery rules. For the low-level discovery rules see the LLD rule API.

Object references:

- Discovery rule

Available methods:

- drule.create - create new discovery rules
- drule.delete - delete discovery rules
- drule.get - retrieve discovery rules
- drule.update - update discovery rules

> Discovery rule object

The following objects are directly related to the `drule` API.

Discovery rule

The discovery rule object defines a network discovery rule. It has the following properties.

Property	Type	Description
druleid	string	ID of the discovery rule. Property behavior: - *read-only* - *required* for update operations

Property	Type	Description
iprange	string	One or several IP ranges to check, separated by commas. Refer to the network discovery configuration section for more information on supported formats of IP ranges. Property behavior: - *required* for create operations
name	string	Name of the discovery rule. Property behavior: - *required* for create operations
delay	string	Execution interval of the discovery rule. Accepts seconds, time unit with suffix, or a user macro. Default: 1h.
proxy_hostid	string	ID of the proxy used for discovery.
status	integer	Whether the discovery rule is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.

drule.create

Description

`object drule.create(object/array discoveryRules)`

This method allows to create new discovery rules.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Discovery rules to create.

Additionally to the standard discovery rule properties, the method accepts the following parameters.

Parameter	Type	Description
dchecks	array	Discovery checks to create for the discovery rule. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the created discovery rules under the `druleids` property. The order of the returned IDs matches the order of the passed discovery rules.

Examples

Create a discovery rule

Create a discovery rule to find machines running the Zabbix agent in the local network. The rule must use a single Zabbix agent check on port 10050.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "drule.create",
```

```
    "params": {
        "name": "Zabbix agent discovery",
        "iprange": "192.168.1.1-255",
        "dchecks": [
            {
                "type": "9",
                "key_": "system.uname",
                "ports": "10050",
                "uniq": "0"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "6"
        ]
    },
    "id": 1
}
```

See also

- Discovery check

Source

CDRule::create() in *ui/include/classes/api/services/CDRule.php*.

drule.delete

Description

`object drule.delete(array discoveryRuleIds)`

This method allows to delete discovery rules.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the discovery rules to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted discovery rules under the `druleids` property.

Examples

Delete multiple discovery rules

Delete two discovery rules.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "drule.delete",
    "params": [
        "4",
        "6"
```

```
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "4",
            "6"
        ]
    },
    "id": 1
}
```

Source

CDRule::delete() in *ui/include/classes/api/services/CDRule.php*.

drule.get

Description

`integer/array drule.get(object parameters)`

The method allows to retrieve discovery rules according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dhostids	string/array	Return only discovery rules that created the given discovered hosts.
druleids	string/array	Return only discovery rules with the given IDs.
dserviceids	string/array	Return only discovery rules that created the given discovered services.
selectDChecks	query	Return a dchecks property with the discovery checks used by the discovery rule. Supports `count`.
selectDHosts	query	Return a dhosts property with the discovered hosts created by the discovery rule. Supports `count`.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectDChecks` - results will be sorted by `dcheckid`; `selectDHosts` - results will be sorted by `dhostsid`.
sortfield	string/array	Sort the result by the given properties. Possible values: `druleid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	

Parameter	Type	Description
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve all discovery rules

Retrieve all configured discovery rules and the discovery checks they use.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "drule.get",
    "params": {
        "output": "extend",
        "selectDChecks": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "druleid": "2",
            "proxy_hostid": "0",
            "name": "Local network",
            "iprange": "192.168.3.1-255",
            "delay": "5s",
            "status": "0",
            "dchecks": [
                {
                    "dcheckid": "7",
                    "druleid": "2",
                    "type": "3",
                    "key_": "",
                    "snmp_community": "",
                    "ports": "21",
                    "snmpv3_securityname": "",
                    "snmpv3_securitylevel": "0",
                    "snmpv3_authpassphrase": "",
                    "snmpv3_privpassphrase": "",
                    "uniq": "0",
                    "snmpv3_authprotocol": "0",
                    "snmpv3_privprotocol": "0",
                    "host_source": "1",
                    "name_source": "0"
                },
                {
                    "dcheckid": "8",
```

```
                    "druleid": "2",
                    "type": "4",
                    "key_": "",
                    "snmp_community": "",
                    "ports": "80",
                    "snmpv3_securityname": "",
                    "snmpv3_securitylevel": "0",
                    "snmpv3_authpassphrase": "",
                    "snmpv3_privpassphrase": "",
                    "uniq": "0",
                    "snmpv3_authprotocol": "0",
                    "snmpv3_privprotocol": "0",
                    "host_source": "1",
                    "name_source": "0"
                }
            ]
        },
        {
            "druleid": "6",
            "proxy_hostid": "0",
            "name": "Zabbix agent discovery",
            "iprange": "192.168.1.1-255",
            "delay": "1h",
            "status": "0",
            "dchecks": [
                {
                    "dcheckid": "10",
                    "druleid": "6",
                    "type": "9",
                    "key_": "system.uname",
                    "snmp_community": "",
                    "ports": "10050",
                    "snmpv3_securityname": "",
                    "snmpv3_securitylevel": "0",
                    "snmpv3_authpassphrase": "",
                    "snmpv3_privpassphrase": "",
                    "uniq": "0",
                    "snmpv3_authprotocol": "0",
                    "snmpv3_privprotocol": "0",
                    "host_source": "2",
                    "name_source": "3"
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Discovered host
- Discovery check

Source

CDRule::get() in *ui/include/classes/api/services/CDRule.php*.

drule.update

Description

`object drule.update(object/array discoveryRules)`

This method allows to update existing discovery rules.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Discovery rule properties to be updated.

The `druleid` property must be defined for each discovery rule, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard discovery rule properties, the method accepts the following parameters.

Parameter	Type	Description
dchecks	array	Discovery checks to replace existing checks.

Return values

`(object)` Returns an object containing the IDs of the updated discovery rules under the `druleids` property.

Examples

Change the IP range of a discovery rule

Change the IP range of a discovery rule to "192.168.2.1-255".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "drule.update",
    "params": {
        "druleid": "6",
        "iprange": "192.168.2.1-255"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "druleids": [
            "6"
        ]
    },
    "id": 1
}
```

See also

- Discovery check

Source

CDRule::update() in *ui/include/classes/api/services/CDRule.php*.

Event

This class is designed to work with events.

Object references:

- Event

Available methods:

- event.get - retrieving events
- event.acknowledge - acknowledging events

> Event object

The following objects are directly related to the `event` API.

Event

Note:
Events are created by the Zabbix server and cannot be modified via the API.

The event object has the following properties.

Property	Type	Description
eventid	string	ID of the event.
source	integer	Type of the event. Possible values: 0 - event created by a trigger; 1 - event created by a discovery rule; 2 - event created by active agent autoregistration; 3 - internal event; 4 - event created on service status update.
object	integer	Type of object that is related to the event. Possible values if `source` is set to ”event created by a trigger”: 0 - trigger. Possible values if `source` is set to ”event created by a discovery rule”: 1 - discovered host; 2 - discovered service. Possible values if `source` is set to ”event created by active agent autoregistration”: 3 - auto-registered host. Possible values if `source` is set to ”internal event”: 0 - trigger; 4 - item; 5 - LLD rule. Possible values if `source` is set to ”event created on service status update”: 6 - service.
objectid	string	ID of the related object.
acknowledged	integer	Whether the event has been acknowledged.
clock	timestamp	Time when the event was created.
ns	integer	Nanoseconds when the event was created.
name	string	Resolved event name.

Property	Type	Description
value	integer	State of the related object. Possible values if `source` is set to ”event created by a trigger” or ”event created on service status update”: 0 - OK; 1 - problem. Possible values if `source` is set to ”event created by a discovery rule”: 0 - host or service up; 1 - host or service down; 2 - host or service discovered; 3 - host or service lost. Possible values if `source` is set to ”internal event”: 0 - ”normal” state; 1 - ”unknown” or ”not supported” state. Property behavior: - *supported* if `source` is set to ”event created by a trigger”, ”event created by a discovery rule”, ”internal event”, or ”event created on service status update”
severity	integer	Event current severity. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
r_eventid	string	Recovery event ID.
c_eventid	string	ID of the event that was used to override (close) current event under global correlation rule. See `correlationid` to identify exact correlation rule. This parameter is only defined when the event is closed by global correlation rule.
cause_eventid	string	Cause event ID.
correlationid	string	ID of the correlation rule that generated closing of the problem. This parameter is only defined when the event is closed by global correlation rule.
userid	string	User ID if the event was manually closed.
suppressed	integer	Whether the event is suppressed. Possible values: 0 - event is in normal state; 1 - event is suppressed.
opdata	string	Operational data with expanded macros.
urls	array	Active media type URLs.

Event tag

The event tag object has the following properties.

Property	Type	Description
tag	string	Event tag name.
value	string	Event tag value.

Media type URL

The media type URL object has the following properties.

Property	Type	Description
name	string	Media type defined URL name.
url	string	Media type defined URL value.

Results will contain entries only for active media types with enabled event menu entry. Macro used in properties will be expanded, but if one of the properties contains an unexpanded macro, both properties will be excluded from results. For supported macros, see *Supported macros*.

event.acknowledge

Description

`object event.acknowledge(object/array parameters)`

This method allows to update events. The following update actions can be performed:

- Close event. If event is already resolved, this action will be skipped.
- Acknowledge event. If event is already acknowledged, this action will be skipped.
- Unacknowledge event. If event is not acknowledged, this action will be skipped.
- Add message.
- Change event severity. If event already has same severity, this action will be skipped.
- Suppress event. If event is already suppressed, this action will be skipped.
- Unsuppress event. If event is not suppressed, this action will be skipped.

Attention:
Only trigger events can be updated.
Only problem events can be updated.
Read/Write rights for trigger are required to close the event or to change event's severity.
To close an event, manual close should be allowed in the trigger.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Parameters containing the IDs of the events and update operations that should be performed.

Parameter	Type	Description
eventids	string/object	IDs of the events to acknowledge. Parameter behavior: - *required*
action	integer	Event update action(s). Possible bitmap values: 1 - close problem; 2 - acknowledge event; 4 - add message; 8 - change severity; 16 - unacknowledge event; 32 - suppress event; 64 - unsuppress event; 128 - change event rank to cause; 256 - change event rank to symptom. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 34 for acknowledge and suppress event). Parameter behavior: - *required*

Parameter	Type	Description
cause_eventid	string	Cause event ID. Parameter behavior: - *required* if `action` contains the ”change event rank to symptom” bit
message	string	Text of the message. Parameter behavior: - *required* if `action` contains the ”add message” bit
severity	integer	New severity for events. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster. Parameter behavior: - *required* if `action` contains the ”change severity” bit
suppress_until	integer	Unix timestamp until which event must be suppressed. If set to ”0”, the suppression will be indefinite. Parameter behavior: - *required* if `action` contains the ”suppress event” bit

Return values

`(object)` Returns an object containing the IDs of the updated events under the `eventids` property.

Examples

Acknowledging an event

Acknowledge a single event and leave a message.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": "20427",
        "action": 6,
        "message": "Problem resolved."
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            "20427"
        ]
    },
    "id": 1
}
```

Changing event’s severity

Change severity for multiple events and leave a message.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "event.acknowledge",
    "params": {
        "eventids": ["20427", "20428"],
        "action": 12,
        "message": "Maintenance required to fix it.",
        "severity": 4
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "eventids": [
            "20427",
            "20428"
        ]
    },
    "id": 1
}
```

Source

CEvent::acknowledge() in *ui/include/classes/api/services/CEvent.php*.

event.get

Description

`integer/array event.get(object parameters)`

The method allows to retrieve events according to the given parameters.

Attention:
This method may return events of a deleted entity if these events have not been removed by the housekeeper yet.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
eventids	string/array	Return only events with the given IDs.
groupids	string/array	Return only events created by objects that belong to the given host groups.
hostids	string/array	Return only events created by objects that belong to the given hosts.
objectids	string/array	Return only events created by the given objects.
source	integer	Return only events with the given type. Refer to the event object page for a list of supported event types. Default: 0 - trigger events.

Parameter	Type	Description
object	integer	Return only events created by objects of the given type. Refer to the event object page for a list of supported object types. Default: 0 - trigger.
acknowledged	boolean	If set to `true` return only acknowledged events.
suppressed	boolean	`true` - return only suppressed events; `false` - return events in the normal state.
symptom	boolean	`true` - return only symptom events; `false` - return only cause events.
severities	integer/array	Return only events with given event severities. Applies only if object is trigger.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	array of objects	Return only events with given tags. Exact match by tag and case-insensitive search by value and operator. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all events. Possible operator types: 0 - (default) Like; 1 - Equal; 2 - Not like; 3 - Not equal 4 - Exists; 5 - Not exists.
eventid_from	string	Return only events with IDs greater or equal to the given ID.
eventid_till	string	Return only events with IDs less or equal to the given ID.
time_from	timestamp	Return only events that have been created after or at the given time.
time_till	timestamp	Return only events that have been created before or at the given time.
problem_time_from	timestamp	Returns only events that were in the problem state starting with `problem_time_from`. Applies only if the source is trigger event and object is trigger. Mandatory if `problem_time_till` is specified.
problem_time_till	timestamp	Returns only events that were in the problem state until `problem_time_till`. Applies only if the source is trigger event and object is trigger. Mandatory if `problem_time_from` is specified.
value	integer/array	Return only events with the given values.
selectHosts	query	Return a hosts property with hosts containing the object that created the event. Supported only for events generated by triggers, items or LLD rules.
selectRelatedObject	query	Return a `relatedObject` property with the object that created the event. The type of object returned depends on the event type.
select_alerts	query	Return an alerts property with alerts generated by the event. Alerts are sorted in reverse chronological order.

Parameter	Type	Description
select_acknowledges	query	Return an `acknowledges` property with event updates. Event updates are sorted in reverse chronological order. The event update object has the following properties: `acknowledgeid` - `(string)` acknowledgment's ID; `userid` - `(string)` ID of the user that updated the event; `eventid` - `(string)` ID of the updated event; `clock` - `(timestamp)` time when the event was updated; `message` - `(string)` text of the message; `action` - `(integer)` update action that was performed see event.acknowledge; `old_severity` - `(integer)` event severity before this update action; `new_severity` - `(integer)` event severity after this update action; `suppress_until` - `(timestamp)` time till event will be suppressed; `taskid` - `(string)` ID of task if current event is undergoing a rank change; `username` - `(string)` username of the user that updated the event; `name` - `(string)` name of the user that updated the event; `surname` - `(string)` surname of the user that updated the event. Supports `count`.
selectTags	query	Return a tags property with event tags.
selectSuppressionData	query	Return a `suppression_data` property with the list of active maintenances and manual suppressions: `maintenanceid` - `(string)` ID of the maintenance; `userid` - `(string)` ID of user who suppressed the event; `suppress_until` - `(integer)` time until the event is suppressed.
sortfield	string/array	Sort the result by the given properties. Possible values: `eventid`, `objectid`, `clock`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving trigger events

Retrieve the latest events from trigger "13926."

Request:

```
{
    "jsonrpc": "2.0",
    "method": "event.get",
    "params": {
        "output": "extend",
```

```
        "select_acknowledges": "extend",
        "selectTags": "extend",
        "selectSuppressionData": "extend",
        "objectids": "13926",
        "sortfield": ["clock", "eventid"],
        "sortorder": "DESC"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "9695",
            "source": "0",
            "object": "0",
            "objectid": "13926",
            "clock": "1347970410",
            "value": "1",
            "acknowledged": "1",
            "ns": "413316245",
            "name": "MySQL is down",
            "severity": "5",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "cause_eventid": "0",
            "opdata": "",
            "acknowledges": [
                {
                    "acknowledgeid": "1",
                    "userid": "1",
                    "eventid": "9695",
                    "clock": "1350640590",
                    "message": "Problem resolved.\n\r----[BULK ACKNOWLEDGE]----",
                    "action": "6",
                    "old_severity": "0",
                    "new_severity": "0",
                    "suppress_until": "1472511600",
                    "taskid": "0",
                    "username": "Admin",
                    "name": "Zabbix",
                    "surname": "Administrator"
                }
            ],
            "suppression_data": [
                {
                    "maintenanceid": "15",
                    "suppress_until": "1472511600",
                    "userid": "0"
                }
            ],
            "suppressed": "1",
            "tags": [
                {
                    "tag": "service",
                    "value": "mysqld"
                },
                {
```

```
                    "tag": "error",
                    "value": ""
                }
            ]
        },
        {
            "eventid": "9671",
            "source": "0",
            "object": "0",
            "objectid": "13926",
            "clock": "1347970347",
            "value": "0",
            "acknowledged": "0",
            "ns": "0",
            "name": "Unavailable by ICMP ping",
            "severity": "4",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "cause_eventid": "0",
            "opdata": "",
            "acknowledges": [],
            "suppression_data": [],
            "suppressed": "0",
            "tags": []
        }
    ],
    "id": 1
}
```


Retrieving events by time period

Retrieve all events that have been created between October 9 and 10, 2012, in reverse chronological order.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "event.get",
    "params": {
        "output": "extend",
        "time_from": "1349797228",
        "time_till": "1350661228",
        "sortfield": ["clock", "eventid"],
        "sortorder": "desc"
    },
    "id": 1
}
```


Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "20616",
            "source": "0",
            "object": "0",
            "objectid": "14282",
            "clock": "1350477814",
            "value": "1",
            "acknowledged": "0",
            "ns": "0",
            "name": "Less than 25% free in the history cache",
```

```
            "severity": "3",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "cause_eventid": "0",
            "opdata": "",
            "suppressed": "0"
        },
        {
            "eventid": "20617",
            "source": "0",
            "object": "0",
            "objectid": "14283",
            "clock": "1350477814",
            "value": "0",
            "acknowledged": "0",
            "ns": "0",
            "name": "Zabbix trapper processes more than 75% busy",
            "severity": "3",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "cause_eventid": "0",
            "opdata": "",
            "suppressed": "0"
        },
        {
            "eventid": "20618",
            "source": "0",
            "object": "0",
            "objectid": "14284",
            "clock": "1350477815",
            "value": "1",
            "acknowledged": "0",
            "ns": "0",
            "name": "High ICMP ping loss",
            "severity": "3",
            "r_eventid": "0",
            "c_eventid": "0",
            "correlationid": "0",
            "userid": "0",
            "cause_eventid": "0",
            "opdata": "",
            "suppressed": "0"
        }
    ],
    "id": 1
}
```


See also

- Alert
- Item
- Host
- LLD rule
- Trigger

Source

CEvent::get() in *ui/include/classes/api/services/CEvent.php*.

Graph

This class is designed to work with graphs.

Object references:

- Graph

Available methods:

- graph.create - creating new graphs
- graph.delete - deleting graphs
- graph.get - retrieving graphs
- graph.update - updating graphs

> Graph object

The following objects are directly related to the `graph` API.

Graph

The graph object has the following properties.

Property	Type	Description
graphid	string	ID of the graph. Property behavior: - *read-only* - *required* for update operations
height	integer	Height of the graph in pixels. Property behavior: - *required* for create operations
name	string	Name of the graph. Property behavior: - *required* for create operations
width	integer	Width of the graph in pixels. Property behavior: - *required* for create operations
flags	integer	Origin of the graph. Possible values: 0 - *(default)* a plain graph; 4 - a discovered graph. Property behavior: - *read-only*
graphtype	integer	Graph's layout type. Possible values: 0 - *(default)* normal; 1 - stacked; 2 - pie; 3 - exploded.
percent_left	float	Left percentile. Default: 0.
percent_right	float	Right percentile. Default: 0.

Property	Type	Description
show_3d	integer	Whether to show pie and exploded graphs in 3D. Possible values: 0 - *(default)* show in 2D; 1 - show in 3D.
show_legend	integer	Whether to show the legend on the graph. Possible values: 0 - hide; 1 - *(default)* show.
show_work_period	integer	Whether to show the working time on the graph. Possible values: 0 - hide; 1 - *(default)* show.
show_triggers	integer	Whether to show the trigger line on the graph. Possible values: 0 - hide; 1 - *(default)* show.
templateid	string	ID of the parent template graph. Property behavior: - *read-only*
yaxismax	float	The fixed maximum value for the Y axis. Default: 100.
yaxismin	float	The fixed minimum value for the Y axis. Default: 0.
ymax_itemid	string	ID of the item that is used as the maximum value for the Y axis. If a user has no access to the specified item, the graph is rendered as if `ymax_type` is set to ”calculated”.
ymax_type	integer	Maximum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.
ymin_itemid	string	ID of the item that is used as the minimum value for the Y axis. If a user has no access to the specified item, the graph is rendered as if `ymin_type` is set to ”calculated”.
ymin_type	integer	Minimum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.
uuid	string	Universal unique identifier, used for linking imported graphs to already existing ones. Auto-generated, if not given. Property behavior: - *supported* if the graph belongs to a template

graph.create

Description

`object graph.create(object/array graphs)`

This method allows to create new graphs.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Graphs to create.

Additionally to the standard graph properties, the method accepts the following parameters.

Parameter	Type	Description
gitems	array	Graph items to be created for the graph. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the created graphs under the `graphids` property. The order of the returned IDs matches the order of the passed graphs.

Examples

Creating a graph

Create a graph with two items.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graph.create",
    "params": {
        "name": "MySQL bandwidth",
        "width": 900,
        "height": 200,
        "gitems": [
            {
                "itemid": "22828",
                "color": "00AA00"
            },
            {
                "itemid": "22829",
                "color": "3333FF"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652"
        ]
    },
    "id": 1
}
```

See also

- Graph item

Source

CGraph::create() in *ui/include/classes/api/services/CGraph.php*.

graph.delete

Description

`object graph.delete(array graphIds)`

This method allows to delete graphs.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the graphs to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted graphs under the `graphids` property.

Examples

Deleting multiple graphs

Delete two graphs.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graph.delete",
    "params": [
        "652",
        "653"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652",
            "653"
        ]
    },
    "id": 1
}
```

Source

CGraph::delete() in *ui/include/classes/api/services/CGraph.php*.

graph.get

Description

`integer/array graph.get(object parameters)`

The method allows to retrieve graphs according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
graphids	string/array	Return only graphs with the given IDs.
groupids	string/array	Return only graphs that belong to hosts or templates in the given host groups or template groups.
templateids	string/array	Return only graph that belong to the given templates.
hostids	string/array	Return only graphs that belong to the given hosts.
itemids	string/array	Return only graphs that contain the given items.
templated	boolean	If set to `true` return only graphs that belong to templates.
inherited	boolean	If set to `true` return only graphs inherited from a template.
expandName	flag	Expand macros in the graph name.
selectHostGroups	query	Return a hostgroups property with the host groups that the graph belongs to.
selectTemplateGroups	query	Return a templategroups property with the template groups that the graph belongs to.
selectTemplates	query	Return a templates property with the templates that the graph belongs to.
selectHosts	query	Return a hosts property with the hosts that the graph belongs to.
selectItems	query	Return an items property with the items used in the graph.
selectGraphDiscovery	query	Return a `graphDiscovery` property with the graph discovery object. The graph discovery objects links the graph to a graph prototype from which it was created. It has the following properties: `graphid` - `(string)` ID of the graph; `parent_graphid` - `(string)` ID of the graph prototype from which the graph has been created.
selectGraphItems	query	Return a gitems property with the items used in the graph.
selectDiscoveryRule	query	Return a discoveryRule property with the low-level discovery rule that created the graph.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: `host` - technical name of the host that the graph belongs to; `hostid` - ID of the host that the graph belongs to.
sortfield	string/array	Sort the result by the given properties. Possible values: `graphid`, `name`, `graphtype`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Parameter	Type	Description
selectGroups (deprecated)	query	This parameter is deprecated, please use `selectHostGroups` or `selectTemplateGroups` instead. Return a `groups` property with the host groups and template groups that the graph belongs to.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving graphs from hosts

Retrieve all graphs from host "10107" and sort them by name.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graph.get",
    "params": {
        "output": "extend",
        "hostids": 10107,
        "sortfield": "name"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "graphid": "612",
            "name": "CPU jumps",
            "width": "900",
            "height": "200",
            "yaxismin": "0",
            "yaxismax": "100",
            "templateid": "439",
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "613",
            "name": "CPU load",
            "width": "900",
            "height": "200",
            "yaxismin": "0",
            "yaxismax": "100",
            "templateid": "433",
```

```
            "show_work_period": "1",
            "show_triggers": "1",
            "graphtype": "0",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "1",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "614",
            "name": "CPU utilization",
            "width": "900",
            "height": "200",
            "yaxismin": "0",
            "yaxismax": "100",
            "templateid": "387",
            "show_work_period": "1",
            "show_triggers": "0",
            "graphtype": "1",
            "show_legend": "1",
            "show_3d": "0",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "1",
            "ymax_type": "1",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "0"
        },
        {
            "graphid": "645",
            "name": "Disk space usage /",
            "width": "600",
            "height": "340",
            "yaxismin": "0",
            "yaxismax": "0",
            "templateid": "0",
            "show_work_period": "0",
            "show_triggers": "0",
            "graphtype": "2",
            "show_legend": "1",
            "show_3d": "1",
            "percent_left": "0",
            "percent_right": "0",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "flags": "4"
        }
    ],
    "id": 1
}
```


See also

- Discovery rule
- Graph item

- Item
- Host
- Host group
- Template
- Template group

Source

CGraph::get() in *ui/include/classes/api/services/CGraph.php*.

graph.update

Description

`object graph.update(object/array graphs)`

This method allows to update existing graphs.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Graph properties to be updated.

The `graphid` property must be defined for each graph, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard graph properties the method accepts the following parameters.

Parameter	Type	Description
gitems	array	Graph items to replace existing graph items. If a graph item has the `gitemid` property defined it will be updated, otherwise a new graph item will be created.

Return values

`(object)` Returns an object containing the IDs of the updated graphs under the `graphids` property.

Examples

Setting the maximum for the Y scale

Set the maximum of the Y scale to a fixed value of 100.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graph.update",
    "params": {
        "graphid": "439",
        "ymax_type": 1,
        "yaxismax": 100
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "439"
        ]
    },
```

```
    "id": 1
}
```

Source

CGraph::update() in *ui/include/classes/api/services/CGraph.php*.

Graph item

This class is designed to work with graph items.

Object references:

- Graph item

Available methods:

- graphitem.get - retrieving graph items

> Graph item object

The following objects are directly related to the `graphitem` API.

Graph item

> **Note:**
> Graph items can only be modified via the `graph` API.

The graph item object has the following properties.

Property	Type	Description
gitemid	string	ID of the graph item. Property behavior: - *read-only*
color	string	Graph item's draw color as a hexadecimal color code. Property behavior: - *required* for create operations
itemid	string	ID of the item. Property behavior: - *required* for create operations
calc_fnc	integer	Value of the item that will be displayed. Possible values: 1 - minimum value; 2 - *(default)* average value; 4 - maximum value; 7 - all values; 9 - last value, used only by pie and exploded graphs.
drawtype	integer	Draw style of the graph item. Possible values: 0 - *(default)* line; 1 - filled region; 2 - bold line; 3 - dot; 4 - dashed line; 5 - gradient line.
graphid	string	ID of the graph that the graph item belongs to.

Property	Type	Description
sortorder	integer	Position of the item in the graph. Default: starts with "0" and increases by one with each entry.
type	integer	Type of graph item. Possible values: 0 - *(default)* simple; 2 - graph sum, used only by pie and exploded graphs.
yaxisside	integer	Side of the graph where the graph item's Y scale will be drawn. Possible values: 0 - *(default)* left side; 1 - right side.

graphitem.get

Description

`integer/array graphitem.get(object parameters)`

The method allows to retrieve graph items according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
graphids	string/array	Return only graph items that belong to the given graphs.
itemids	string/array	Return only graph items with the given item IDs.
type	integer	Return only graph items with the given type. Refer to the graph item object page for a list of supported graph item types.
selectGraphs	query	Return a graphs property with an array of graphs that the item belongs to.
sortfield	string/array	Sort the result by the given properties. Possible values: `gitemid`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
limit	integer	
output	query	
preservekeys	boolean	
sortorder	string/array	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving graph items from a graph

Retrieve all graph items used in a graph with additional information about the item and the host.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graphitem.get",
    "params": {
        "output": "extend",
        "graphids": "387"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "gitemid": "1242",
            "graphid": "387",
            "itemid": "22665",
            "drawtype": "1",
            "sortorder": "1",
            "color": "FF5555",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0"
        },
        {
            "gitemid": "1243",
            "graphid": "387",
            "itemid": "22668",
            "drawtype": "1",
            "sortorder": "2",
            "color": "55FF55",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0"
        },
        {
            "gitemid": "1244",
            "graphid": "387",
            "itemid": "22671",
            "drawtype": "1",
            "sortorder": "3",
            "color": "009999",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0"
        }
    ],
    "id": 1
}
```

See also

- Graph

Source

CGraphItem::get() in *ui/include/classes/api/services/CGraphItem.php*.

Graph prototype

This class is designed to work with graph prototypes.

Object references:

- Graph prototype

Available methods:

- graphprototype.create - creating new graph prototypes
- graphprototype.delete - deleting graph prototypes
- graphprototype.get - retrieving graph prototypes
- graphprototype.update - updating graph prototypes

> Graph prototype object

The following objects are directly related to the `graphprototype` API.

Graph prototype

The graph prototype object has the following properties.

Property	Type	Description
graphid	string	ID of the graph prototype. Property behavior: - *read-only* - *required* for update operations
height	integer	Height of the graph prototype in pixels. Property behavior: - *required* for create operations
name	string	Name of the graph prototype. Property behavior: - *required* for create operations
width	integer	Width of the graph prototype in pixels. Property behavior: - *required* for create operations
graphtype	integer	Graph prototypes's layout type. Possible values: 0 - *(default)* normal; 1 - stacked; 2 - pie; 3 - exploded.
percent_left	float	Left percentile. Default: 0.
percent_right	float	Right percentile. Default: 0.
show_3d	integer	Whether to show discovered pie and exploded graphs in 3D. Possible values: 0 - *(default)* show in 2D; 1 - show in 3D.
show_legend	integer	Whether to show the legend on the discovered graph. Possible values: 0 - hide; 1 - *(default)* show.

Property	Type	Description
show_work_period	integer	Whether to show the working time on the discovered graph. Possible values: 0 - hide; 1 - *(default)* show.
templateid	string	ID of the parent template graph prototype. Property behavior: - *read-only*
yaxismax	float	The fixed maximum value for the Y axis.
yaxismin	float	The fixed minimum value for the Y axis.
ymax_itemid	string	ID of the item that is used as the maximum value for the Y axis. If a user has no access to the specified item, the graph is rendered as if `ymax_type` is set to ”calculated”.
ymax_type	integer	Maximum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.
ymin_itemid	string	ID of the item that is used as the minimum value for the Y axis. If a user has no access to the specified item, the graph is rendered as if `ymin_type` is set to ”calculated”.
ymin_type	integer	Minimum value calculation method for the Y axis. Possible values: 0 - *(default)* calculated; 1 - fixed; 2 - item.
discover	integer	Graph prototype discovery status. Possible values: 0 - *(default)* new graphs will be discovered; 1 - new graphs will not be discovered and existing graphs will be marked as lost.
uuid	string	Universal unique identifier, used for linking imported graph prototypes to already existing ones. Auto-generated, if not given. Property behavior: - *supported* if the graph prototype belongs to a template

graphprototype.create

Description

`object graphprototype.create(object/array graphPrototypes)`

This method allows to create new graph prototypes.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Graph prototypes to create.

Additionally to the standard graph prototype properties, the method accepts the following parameters.

Parameter	Type	Description
gitems	array	Graph items to be created for the graph prototypes. Graph items can reference both items and item prototypes, but at least one item prototype must be present. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the created graph prototypes under the `graphids` property. The order of the returned IDs matches the order of the passed graph prototypes.

Examples

Creating a graph prototype

Create a graph prototype with two items.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.create",
    "params": {
        "name": "Disk space usage {#FSNAME}",
        "width": 900,
        "height": 200,
        "gitems": [
            {
                "itemid": "22828",
                "color": "00AA00"
            },
            {
                "itemid": "22829",
                "color": "3333FF"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652"
        ]
    },
    "id": 1
}
```

See also

- Graph item

Source

CGraphPrototype::create() in *ui/include/classes/api/services/CGraphPrototype.php*.

graphprototype.delete

Description

`object graphprototype.delete(array graphPrototypeIds)`

This method allows to delete graph prototypes.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the graph prototypes to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted graph prototypes under the `graphids` property.

Examples

Deleting multiple graph prototypes

Delete two graph prototypes.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.delete",
    "params": [
        "652",
        "653"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "652",
            "653"
        ]
    },
    "id": 1
}
```

Source

CGraphPrototype::delete() in *ui/include/classes/api/services/CGraphPrototype.php*.

graphprototype.get

Description

`integer/array graphprototype.get(object parameters)`

The method allows to retrieve graph prototypes according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
discoveryids	string/array	Return only graph prototypes that belong to the given discovery rules.

Parameter	Type	Description
graphids	string/array	Return only graph prototypes with the given IDs.
groupids	string/array	Return only graph prototypes that belong to hosts or templates in the given host groups or template groups.
hostids	string/array	Return only graph prototypes that belong to the given hosts.
inherited	boolean	If set to `true` return only graph prototypes inherited from a template.
itemids	string/array	Return only graph prototypes that contain the given item prototypes.
templated	boolean	If set to `true` return only graph prototypes that belong to templates.
templateids	string/array	Return only graph prototypes that belong to the given templates.
selectDiscoveryRule	query	Return a discoveryRule property with the LLD rule that the graph prototype belongs to.
selectGraphItems	query	Return a gitems property with the graph items used in the graph prototype.
selectHostGroups	query	Return a hostgroups property with the host groups that the graph prototype belongs to.
selectHosts	query	Return a hosts property with the hosts that the graph prototype belongs to.
selectItems	query	Return an `items` property with the items and item prototypes used in the graph prototype.
selectTemplateGroups	query	Return a templategroups property with the template groups that the graph prototype belongs to.
selectTemplates	query	Return a templates property with the templates that the graph prototype belongs to.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: `host` - technical name of the host that the graph prototype belongs to; `hostid` - ID of the host that the graph prototype belongs to.
sortfield	string/array	Sort the result by the given properties. Possible values: `graphid`, `name`, `graphtype`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectGroups (deprecated)	query	This parameter is deprecated, please use `selectHostGroups` or `selectTemplateGroups` instead. Return a `groups` property with the host groups and template groups that the graph prototype belongs to.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving graph prototypes from an LLD rule

Retrieve all graph prototypes from an LLD rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "graphid": "1017",
            "parent_itemid": "27426",
            "name": "Disk space usage {#FSNAME}",
            "width": "600",
            "height": "340",
            "yaxismin": "0.0000",
            "yaxismax": "0.0000",
            "templateid": "442",
            "show_work_period": "0",
            "show_triggers": "0",
            "graphtype": "2",
            "show_legend": "1",
            "show_3d": "1",
            "percent_left": "0.0000",
            "percent_right": "0.0000",
            "ymin_type": "0",
            "ymax_type": "0",
            "ymin_itemid": "0",
            "ymax_itemid": "0",
            "discover": "0"
        }
    ],
    "id": 1
}
```

See also

- Discovery rule
- Graph item
- Item
- Host
- Host group
- Template
- Template group

Source

CGraphPrototype::get() in *ui/include/classes/api/services/CGraphPrototype.php*.

graphprototype.update

Description

`object graphprototype.update(object/array graphPrototypes)`

This method allows to update existing graph prototypes.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Graph prototype properties to be updated.

The `graphid` property must be defined for each graph prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard graph prototype properties, the method accepts the following parameters.

Parameter	Type	Description
gitems	array	Graph items to replace existing graph items. If a graph item has the `gitemid` property defined it will be updated, otherwise a new graph item will be created.

Return values

`(object)` Returns an object containing the IDs of the updated graph prototypes under the `graphids` property.

Examples

Changing the size of a graph prototype

Change the size of a graph prototype to 1100 to 400 pixels.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.update",
    "params": {
        "graphid": "439",
        "width": 1100,
        "height": 400
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "graphids": [
            "439"
        ]
    },
    "id": 1
}
```

Source

CGraphPrototype::update() in *ui/include/classes/api/services/CGraphPrototype.php*.

High availability node

This class is designed to work with server nodes that are part of a High availability cluster, or a standalone server instance.

Object references:

- High availability node

Available methods:

- hanode.get - retrieving nodes

> High availability node object

The following object is related to operating a High availability cluster of Zabbix servers.

High availability node

Note:
Nodes are created by the Zabbix server and cannot be modified via the API.

The High availability node object has the following properties.

Property	Type	Description
ha_nodeid	string	ID of the node.
name	string	Name assigned to the node, using the HANodeName configuration entry of zabbix_server.conf. Empty for a server running in standalone mode.
address	string	IP or DNS name where the node connects from.
port	integer	Port on which the node is running.
lastaccess	integer	Heartbeat time, that is, time of last update from the node. UTC timestamp.
status	integer	State of the node. Possible values: 0 - standby; 1 - stopped manually; 2 - unavailable; 3 - active.

hanode.get

Description

`integer/array hanode.get(object parameters)`

The method allows to retrieve a list of High availability cluster nodes according to the given parameters.

Note:
This method is only available to *Super admin* user types. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
ha_nodeids	string/array	Return only nodes with the given node IDs.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Allows filtering by the node properties: `name`, `address`, `status`.
sortfield	string/array	Sort the result by the given properties. Possible values: `name`, `lastaccess`, `status`.
countOutput	flag	These parameters being common for all `get` methods are described in detail in the reference commentary.
limit	integer	
output	query	

Parameter	Type	Description
preservekeys	boolean	
sortorder	string/array	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Get a list of nodes ordered by status

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "preservekeys": true,
        "sortfield": "status",
        "sortorder": "DESC"
    },
    "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "ckuo7i1nw000h0sajj3l3hh8u": {
      "ha_nodeid": "ckuo7i1nw000h0sajj3l3hh8u",
      "name": "node-active",
      "address": "192.168.1.13",
      "port": "10051",
      "lastaccess": "1635335704",
      "status": "3"
    },
    "ckuo7i1nw000e0sajwfttc1mp": {
      "ha_nodeid": "ckuo7i1nw000e0sajwfttc1mp",
      "name": "node6",
      "address": "192.168.1.10",
      "port": "10053",
      "lastaccess": "1635332902",
      "status": "2"
    },
    "ckuo7i1nv000c0sajz85xcrtt": {
      "ha_nodeid": "ckuo7i1nv000c0sajz85xcrtt",
      "name": "node4",
      "address": "192.168.1.8",
      "port": "10052",
      "lastaccess": "1635334214",
      "status": "1"
    },
    "ckuo7i1nv000a0saj1fcdkeu4": {
      "ha_nodeid": "ckuo7i1nv000a0saj1fcdkeu4",
      "name": "node2",
      "address": "192.168.1.6",
      "port": "10051",
      "lastaccess": "1635335705",
      "status": "0"
    }
```

```
  },
  "id": 1
}
```

Get a list of specific nodes by their IDs

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "ha_nodeids": ["ckuo7i1nw000e0sajwfttc1mp", "ckuo7i1nv000c0sajz85xcrtt"]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "ha_nodeid": "ckuo7i1nv000c0sajz85xcrtt",
            "name": "node4",
            "address": "192.168.1.8",
            "port": "10052",
            "lastaccess": "1635334214",
            "status": "1"
        },
        {
            "ha_nodeid": "ckuo7i1nw000e0sajwfttc1mp",
            "name": "node6",
            "address": "192.168.1.10",
            "port": "10053",
            "lastaccess": "1635332902",
            "status": "2"
        }
    ],
    "id": 1
}
```

Get a list of stopped nodes

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "output": ["ha_nodeid", "address", "port"],
        "filter": {
            "status": 1
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "ha_nodeid": "ckuo7i1nw000g0sajjsjre7e3",
            "address": "192.168.1.12",
            "port": "10051"
```

```
        },
        {
            "ha_nodeid": "ckuo7i1nv000c0sajz85xcrtt",
            "address": "192.168.1.8",
            "port": "10052"
        },
        {
            "ha_nodeid": "ckuo7i1nv000d0sajd95y1b6x",
            "address": "192.168.1.9",
            "port": "10053"
        }
    ],
    "id": 1
}
```

Get a count of standby nodes

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "countOutput": true,
        "filter": {
            "status": 0
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "3",
    "id": 1
}
```

Check status of nodes at specific IP addresses

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hanode.get",
    "params": {
        "output": ["name", "status"],
        "filter": {
            "address": ["192.168.1.7", "192.168.1.13"]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "name": "node3",
            "status": "0"
        },
        {
            "name": "node-active",
            "status": "3"
```

```
        }
    ],
    "id": 1
}
```

Source

CHaNode::get() in *ui/include/classes/api/services/CHaNode.php*.

History

This class is designed to work with history data.

Object references:

- History

Available methods:

- history.get - retrieving history data.

> History object

The following objects are directly related to the `history` API.

Note:
History objects differ depending on the item's type of information. They are created by the Zabbix server and cannot be modified via the API.

Float history

The float history object has the following properties.

Property	Type	Description
clock	timestamp	Time when that value was received.
itemid	string	ID of the related item.
ns	integer	Nanoseconds when the value was received.
value	float	Received value.

Integer history

The integer history object has the following properties.

Property	Type	Description
clock	timestamp	Time when that value was received.
itemid	string	ID of the related item.
ns	integer	Nanoseconds when the value was received.
value	integer	Received value.

String history

The string history object has the following properties.

Property	Type	Description
clock	timestamp	Time when that value was received.
itemid	string	ID of the related item.
ns	integer	Nanoseconds when the value was received.
value	string	Received value.

Text history

The text history object has the following properties.

Property	Type	Description
id	string	ID of the history entry.
clock	timestamp	Time when that value was received.
itemid	string	ID of the related item.
ns	integer	Nanoseconds when the value was received.
value	text	Received value.

Log history

The log history object has the following properties.

Property	Type	Description
id	string	ID of the history entry.
clock	timestamp	Time when that value was received.
itemid	string	ID of the related item.
logeventid	integer	Windows event log entry ID.
ns	integer	Nanoseconds when the value was received.
severity	integer	Windows event log entry level.
source	string	Windows event log entry source.
timestamp	timestamp	Windows event log entry time.
value	text	Received value.

history.clear

Description

`object history.clear(array itemids)`

This method allows to clear item history.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of items to clear.

Return values

`(object)` Returns an object containing the IDs of the cleared items under the `itemids` property.

Examples

Clear history

Request:

```
{
    "jsonrpc": "2.0",
    "method": "history.clear",
    "params": [
        "10325",
        "13205"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "10325",
            "13205"
        ]
    },
    "id": 1
}
```

Source

CHistory::clear() in *ui/include/classes/api/services/CHistory.php*.

history.get

Description

`integer/array history.get(object parameters)`

The method allows to retrieve history data according to the given parameters.

Attention:
This method may return historical data of a deleted entity if this data has not been removed by the housekeeper yet.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
history	integer	History object types to return. Possible values: 0 - numeric float; 1 - character; 2 - log; 3 - *(default)* numeric unsigned; 4 - text.
hostids	string/array	Return only history from the given hosts.
itemids	string/array	Return only history from the given items.
time_from	timestamp	Return only values that have been received after or at the given time.
time_till	timestamp	Return only values that have been received before or at the given time.
sortfield	string/array	Sort the result by the given properties. Possible values: `itemid`, `clock`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	

Parameter	Type	Description
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving item history data

Return 10 latest values received from a numeric(float) item.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "history.get",
    "params": {
        "output": "extend",
        "history": 0,
        "itemids": "23296",
        "sortfield": "clock",
        "sortorder": "DESC",
        "limit": 10
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23296",
            "clock": "1351090996",
            "value": "0.085",
            "ns": "563157632"
        },
        {
            "itemid": "23296",
            "clock": "1351090936",
            "value": "0.16",
            "ns": "549216402"
        },
        {
            "itemid": "23296",
            "clock": "1351090876",
            "value": "0.18",
            "ns": "537418114"
        },
        {
            "itemid": "23296",
            "clock": "1351090816",
            "value": "0.21",
            "ns": "522659528"
        },
        {
            "itemid": "23296",
            "clock": "1351090756",
            "value": "0.215",
            "ns": "507809457"
```

```
        },
        {
            "itemid": "23296",
            "clock": "1351090696",
            "value": "0.255",
            "ns": "495509699"
        },
        {
            "itemid": "23296",
            "clock": "1351090636",
            "value": "0.36",
            "ns": "477708209"
        },
        {
            "itemid": "23296",
            "clock": "1351090576",
            "value": "0.375",
            "ns": "463251343"
        },
        {
            "itemid": "23296",
            "clock": "1351090516",
            "value": "0.315",
            "ns": "447947017"
        },
        {
            "itemid": "23296",
            "clock": "1351090456",
            "value": "0.275",
            "ns": "435307141"
        }
    ],
    "id": 1
}
```


Source

CHistory::get() in *ui/include/classes/api/services/CHistory.php*.

Host

This class is designed to work with hosts.

Object references:

- Host
- Host inventory

Available methods:

- host.create - creating new hosts
- host.delete - deleting hosts
- host.get - retrieving hosts
- host.massadd - adding related objects to hosts
- host.massremove - removing related objects from hosts
- host.massupdate - replacing or removing related objects from hosts
- host.update - updating hosts

> Host object

The following objects are directly related to the host API.

Host

The host object has the following properties.

Property	Type	Description
hostid	string	ID of the host. Property behavior: - *read-only* - *required* for update operations
host	string	Technical name of the host. Property behavior: - *required* for create operations
description	text	Description of the host.
flags	integer	Origin of the host. Possible values: 0 - a plain host; 4 - a discovered host. Property behavior: - *read-only*
inventory_mode	integer	Host inventory population mode. Possible values: -1 - *(default)* disabled; 0 - manual; 1 - automatic.
ipmi_authtype	integer	IPMI authentication algorithm. Possible values: -1 - *(default)* default; 0 - none; 1 - MD2; 2 - MD5 4 - straight; 5 - OEM; 6 - RMCP+.
ipmi_password	string	IPMI password.
ipmi_privilege	integer	IPMI privilege level. Possible values: 1 - callback; 2 - *(default)* user; 3 - operator; 4 - admin; 5 - OEM.
ipmi_username	string	IPMI username.
maintenance_from	timestamp	Starting time of the effective maintenance. Property behavior: - *read-only*
maintenance_status	integer	Effective maintenance status. Possible values: 0 - *(default)* no maintenance; 1 - maintenance in effect. Property behavior: - *read-only*

Property	Type	Description
maintenance_type	integer	Effective maintenance type. Possible values: 0 - *(default)* maintenance with data collection; 1 - maintenance without data collection. Property behavior: - *read-only*
maintenanceid	string	ID of the maintenance that is currently in effect on the host. Property behavior: - *read-only*
name	string	Visible name of the host. Default: `host` property value.
proxy_hostid	string	ID of the proxy that is used to monitor the host.
status	integer	Status and function of the host. Possible values: 0 - *(default)* monitored host; 1 - unmonitored host.
tls_connect	integer	Connections to host. Possible values: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate.
tls_accept	integer	Connections from host. Possible bitmap values: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 6 for PSK and certificate).
tls_issuer	string	Certificate issuer.
tls_subject	string	Certificate subject.
tls_psk_identity	string	PSK identity; must be paired with only one PSK (across autoregistration, hosts, and proxies). Do not include sensitive information in the PSK identity, as it is sent unencrypted over the network to inform the receiver which PSK to use. Property behavior: - *write-only* - *required* if `tls_connect` is set to "PSK", or `tls_accept` contains the "PSK" bit
tls_psk	string	Pre-shared key (PSK); must be at least 32 hex digits. Property behavior: - *write-only* - *required* if `tls_connect` is set to "PSK", or `tls_accept` contains the "PSK" bit

Property	Type	Description
active_available	integer	Host active interface availability status. Possible values: 0 - interface status is unknown; 1 - interface is available; 2 - interface is not available. Property behavior: - *read-only*

Host inventory

The host inventory object has the following properties.

Note:
Each property has it's own unique ID number, which is used to associate host inventory fields with items.

ID	Property	Type	Description	Maximum length
4	alias	string	Alias.	128 characters
11	asset_tag	string	Asset tag.	64 characters
28	chassis	string	Chassis.	64 characters
23	contact	string	Contact person.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
32	contract_number	string	Contract number.	64 characters
47	date_hw_decomm	string	HW decommissioning date.	64 characters
46	date_hw_expiry	string	HW maintenance expiry date.	64 characters
45	date_hw_install	string	HW installation date.	64 characters
44	date_hw_purchase	string	HW purchase date.	64 characters
34	deployment_status	string	Deployment status.	64 characters
14	hardware	string	Hardware.	255 characters
15	hardware_full	string	Detailed hardware.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
39	host_netmask	string	Host subnet mask.	39 characters
38	host_networks	string	Host networks.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
40	host_router	string	Host router.	39 characters
30	hw_arch	string	HW architecture.	32 characters
33	installer_name	string	Installer name.	64 characters
24	location	string	Location.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
25	location_lat	string	Location latitude.	16 characters
26	location_lon	string	Location longitude.	16 characters
12	macaddress_a	string	MAC address A.	64 characters
13	macaddress_b	string	MAC address B.	64 characters
29	model	string	Model.	64 characters
3	name	string	Name.	128 characters
27	notes	string	Notes.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
41	oob_ip	string	OOB IP address.	39 characters
42	oob_netmask	string	OOB host subnet mask.	39 characters
43	oob_router	string	OOB router.	39 characters
5	os	string	OS name.	128 characters
6	os_full	string	Detailed OS name.	255 characters
7	os_short	string	Short OS name.	128 characters
61	poc_1_cell	string	Primary POC mobile number.	64 characters

ID	Property	Type	Description	Maximum length
58	poc_1_email	string	Primary email.	128 characters
57	poc_1_name	string	Primary POC name.	128 characters
63	poc_1_notes	string	Primary POC notes.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
59	poc_1_phone_a	string	Primary POC phone A.	64 characters
60	poc_1_phone_b	string	Primary POC phone B.	64 characters
62	poc_1_screen	string	Primary POC screen name.	64 characters
68	poc_2_cell	string	Secondary POC mobile number.	64 characters
65	poc_2_email	string	Secondary POC email.	128 characters
64	poc_2_name	string	Secondary POC name.	128 characters
70	poc_2_notes	string	Secondary POC notes.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
66	poc_2_phone_a	string	Secondary POC phone A.	64 characters
67	poc_2_phone_b	string	Secondary POC phone B.	64 characters
69	poc_2_screen	string	Secondary POC screen name.	64 characters
8	serialno_a	string	Serial number A.	64 characters
9	serialno_b	string	Serial number B.	64 characters
48	site_address_a	string	Site address A.	128 characters
49	site_address_b	string	Site address B.	128 characters
50	site_address_c	string	Site address C.	128 characters
51	site_city	string	Site city.	128 characters
53	site_country	string	Site country.	64 characters
56	site_notes	string	Site notes.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
55	site_rack	string	Site rack location.	128 characters
52	site_state	string	Site state.	64 characters
54	site_zip	string	Site ZIP/postal code.	64 characters
16	software	string	Software.	255 characters
18	software_app_a	string	Software application A.	64 characters
19	software_app_b	string	Software application B.	64 characters
20	software_app_c	string	Software application C.	64 characters
21	software_app_d	string	Software application D.	64 characters
22	software_app_e	string	Software application E.	64 characters
17	software_full	string	Software details.	Depends on the database used: - 65535 characters for SQL databases - 2048 characters for Oracle databases
10	tag	string	Tag.	64 characters
1	type	string	Type.	64 characters
2	type_full	string	Type details.	64 characters
35	url_a	string	URL A.	255 characters
36	url_b	string	URL B.	255 characters
37	url_c	string	URL C.	255 characters
31	vendor	string	Vendor.	64 characters

Host tag

The host tag object has the following properties.

Property	Type	Description
tag	string	Host tag name. Property behavior: - *required*
value	string	Host tag value.

Property	Type	Description
automatic	integer	Type of host tag. Possible values: 0 - *(default)* manual (tag created by user); 1 - automatic (tag created by low-level discovery)

host.create

Description

`object host.create(object/array hosts)`

This method allows to create new hosts.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Hosts to create.

Additionally to the standard host properties, the method accepts the following parameters.

Parameter	Type	Description
groups	object/array	Host groups to add the host to. The host groups must have the `groupid` property defined. Parameter behavior: *- required*
interfaces	object/array	Interfaces to be created for the host.
tags	object/array	Host tags.
templates	object/array	Templates to be linked to the host. The templates must have the `templateid` property defined.
macros	object/array	User macros to be created for the host.
inventory	object	Host inventory properties.

Return values

`(object)` Returns an object containing the IDs of the created hosts under the `hostids` property. The order of the returned IDs matches the order of the passed hosts.

Examples

Creating a host

Create a host called "Linux server" with an IP interface and tags, add it to a group, link a template to it and set the MAC addresses in the host inventory.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "Linux server",
        "interfaces": [
            {
                "type": 1,
                "main": 1,
                "useip": 1,
```

```
                "ip": "192.168.3.1",
                "dns": "",
                "port": "10050"
            }
        ],
        "groups": [
            {
                "groupid": "50"
            }
        ],
        "tags": [
            {
                "tag": "Host name",
                "value": "Linux server"
            }
        ],
        "templates": [
            {
                "templateid": "20045"
            }
        ],
        "macros": [
            {
                "macro": "{$USER_ID}",
                "value": "123321"
            },
            {
                "macro": "{$USER_LOCATION}",
                "value": "0:0:0",
                "description": "latitude, longitude and altitude coordinates"
            }
        ],
        "inventory_mode": 0,
        "inventory": {
            "macaddress_a": "01234",
            "macaddress_b": "56768"
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "107819"
        ]
    },
    "id": 1
}
```

Creating a host with SNMP interface

Create a host called "SNMP host" with an SNMPv3 interface with details.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "SNMP host",
        "interfaces": [
```

```
            {
                "type": 2,
                "main": 1,
                "useip": 1,
                "ip": "127.0.0.1",
                "dns": "",
                "port": "161",
                "details": {
                    "version": 3,
                    "bulk": 0,
                    "securityname": "mysecurityname",
                    "contextname": "",
                    "securitylevel": 1
                }
            }
        ],
        "groups": [
            {
                "groupid": "4"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10658"
        ]
    },
    "id": 1
}
```

Creating a host with PSK encryption

Create a host called "PSK host" with PSK encryption configured. Note that the host has to be pre-configured to use PSK.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.create",
    "params": {
        "host": "PSK host",
        "interfaces": [
            {
                "type": 1,
                "ip": "192.168.3.1",
                "dns": "",
                "port": "10050",
                "useip": 1,
                "main": 1
            }
        ],
        "groups": [
            {
                "groupid": "2"
            }
        ],
        "tls_accept": 2,
        "tls_connect": 2,
```

```
        "tls_psk_identity": "PSK 001",
        "tls_psk": "1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10590"
        ]
    },
    "id": 1
}
```

See also

- Host group
- Template
- User macro
- Host interface
- Host inventory
- Host tag

Source

CHost::create() in *ui/include/classes/api/services/CHost.php*.

host.delete

Description

`object host.delete(array hosts)`

This method allows to delete hosts.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of hosts to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted hosts under the `hostids` property.

Examples

Deleting multiple hosts

Delete two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.delete",
    "params": [
        "13",
        "32"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "13",
            "32"
        ]
    },
    "id": 1
}
```

Source

CHost::delete() in *ui/include/classes/api/services/CHost.php*.

host.get

Description

`integer/array host.get(object parameters)`

The method allows to retrieve hosts according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
groupids	string/array	Return only hosts that belong to the given groups.
dserviceids	string/array	Return only hosts that are related to the given discovered services.
graphids	string/array	Return only hosts that have the given graphs.
hostids	string/array	Return only hosts with the given host IDs.
httptestids	string/array	Return only hosts that have the given web checks.
interfaceids	string/array	Return only hosts that use the given interfaces.
itemids	string/array	Return only hosts that have the given items.
maintenanceids	string/array	Return only hosts that are affected by the given maintenances.
monitored_hosts	flag	Return only monitored hosts.
proxy_hosts	flag	Return only proxies.
proxyids	string/array	Return only hosts that are monitored by the given proxies.
templated_hosts	flag	Return both hosts and templates.
templateids	string/array	Return only hosts that are linked to the given templates.
triggerids	string/array	Return only hosts that have the given triggers.
with_items	flag	Return only hosts that have items. Overrides the `with_monitored_items` and `with_simple_graph_items` parameters.
with_item_prototypes	flag	Return only hosts that have item prototypes. Overrides the `with_simple_graph_item_prototypes` parameter.
with_simple_graph_item_prototypes	flag	Return only hosts that have item prototypes, which are enabled for creation and have numeric type of information.
with_graphs	flag	Return only hosts that have graphs.
with_graph_prototypes	flag	Return only hosts that have graph prototypes.
with_httptests	flag	Return only hosts that have web checks. Overrides the `with_monitored_httptests` parameter.
with_monitored_httptests	flag	Return only hosts that have enabled web checks.

Parameter	Type	Description
with_monitored_items	flag	Return only hosts that have enabled items. Overrides the `with_simple_graph_items` parameter.
with_monitored_triggers	flag	Return only hosts that have enabled triggers. All of the items used in the trigger must also be enabled.
with_simple_graph_items	flag	Return only hosts that have items with numeric type of information.
with_triggers	flag	Return only hosts that have triggers. Overrides the `with_monitored_triggers` parameter.
withProblemsSuppressed	boolean	Return hosts that have suppressed problems. Possible values: `null` - *(default)* all hosts; `true` - only hosts with suppressed problems; `false` - only hosts with unsuppressed problems.
evaltype	integer	Rules for tag searching. Possible values: 0 - *(default)* And/Or; 2 - Or.
severities	integer/array	Return hosts that have only problems with given severities. Applies only if problem object is trigger.
tags	object/array	Return only hosts with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all hosts. Possible operator values: 0 - *(default)* Contains; 1 - Equals; 2 - Not like; 3 - Not equal; 4 - Exists; 5 - Not exists.
inheritedTags	boolean	Return hosts that have given `tags` also in all of their linked templates. Default: Possible values: `true` - linked templates must also have given `tags`; `false` - *(default)* linked template tags are ignored.
selectDiscoveries	query	Return a discoveries property with host low-level discovery rules. Supports `count`.
selectDiscoveryRule	query	Return a discoveryRule property with the low-level discovery rule that created the host (from host prototype in VMware monitoring).
selectGraphs	query	Return a graphs property with host graphs. Supports `count`.

Parameter	Type	Description
selectHostDiscovery	query	Return a `hostDiscovery` property with host discovery object data. The host discovery object links a discovered host to a host prototype or a host prototypes to an LLD rule and has the following properties: `host` - `(string)` host of the host prototype; `hostid` - `(string)` ID of the discovered host or host prototype; `parent_hostid` - `(string)` ID of the host prototype from which the host has been created; `parent_itemid` - `(string)` ID of the LLD rule that created the discovered host; `lastcheck` - `(timestamp)` time when the host was last discovered; `ts_delete` - `(timestamp)` time when a host that is no longer discovered will be deleted.
selectHostGroups	query	Return a hostgroups property with host groups data that the host belongs to.
selectHttpTests	query	Return an httpTests property with host web scenarios. Supports `count`.
selectInterfaces	query	Return an interfaces property with host interfaces. Supports `count`.
selectInventory	query	Return an inventory property with host inventory data.
selectItems	query	Return an items property with host items. Supports `count`.
selectMacros	query	Return a macros property with host macros.
selectParentTemplates	query	Return a `parentTemplates` property with templates that the host is linked to. In addition to Template object fields, it contains `link_type` - `(integer)` the way that the template is linked to host. Possible values: 0 - *(default)* manually linked; 1 - automatically linked by LLD. Supports `count`.
selectDashboards	query	Return a dashboards property. Supports `count`.
selectTags	query	Return a tags property with host tags.
selectInheritedTags	query	Return an inheritedTags property with tags that are on all templates which are linked to host.
selectTriggers	query	Return a triggers property with host triggers. Supports `count`.
selectValueMaps	query	Return a valuemaps property with host value maps.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Allows filtering by interface properties. Doesn't work for `text` fields.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectParentTemplates` - results will be sorted by `host`; `selectInterfaces`; `selectItems` - sorted by `name`; `selectDiscoveries` - sorted by `name`; `selectTriggers` - sorted by `description`; `selectGraphs` - sorted by `name`; `selectDashboards` - sorted by `name`.

Parameter	Type	Description
search	object	Return results that match the given pattern (case-insensitive). Accepts an array, where the keys are property names, and the values are strings to search for. If no additional options are given, this will perform a `LIKE "%…%"` search. Allows searching by interface properties. Works only for `string` and `text` fields.
searchInventory	object	Return only hosts that have inventory data matching the given wildcard search. This parameter is affected by the same additional parameters as `search`.
sortfield	string/array	Sort the result by the given properties. Possible values: `hostid`, `host`, `name`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectGroups (deprecated)	query	This parameter is deprecated, please use `selectHostGroups` instead. Return a groups property with host groups data that the host belongs to.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving data by name

Retrieve all data about two hosts named "Zabbix server" and "Linux server".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "filter": {
            "host": [
                "Zabbix server",
                "Linux server"
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
```

```
    "result": [
        {
            "hostid": "10160",
            "proxy_hostid": "0",
            "host": "Zabbix server",
            "status": "0",
            "lastaccess": "0",
            "ipmi_authtype": "-1",
            "ipmi_privilege": "2",
            "ipmi_username": "",
            "ipmi_password": "",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "name": "Zabbix server",
            "flags": "0",
            "description": "The Zabbix monitoring server.",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "inventory_mode": "1",
            "active_available": "1"
        },
        {
            "hostid": "10167",
            "proxy_hostid": "0",
            "host": "Linux server",
            "status": "0",
            "ipmi_authtype": "-1",
            "ipmi_privilege": "2",
            "ipmi_username": "",
            "ipmi_password": "",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "name": "Linux server",
            "flags": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "inventory_mode": "1",
            "active_available": "1"
        }
    ],
    "id": 1
}
```


Retrieving host groups

Retrieve host groups that the host ”Zabbix server” is a member of.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectHostGroups": "extend",
```

```
        "filter": {
            "host": [
                "Zabbix server"
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10085",
            "hostgroups": [
                {
                    "groupid": "2",
                    "name": "Linux servers",
                    "flags": "0",
                    "uuid": "dc579cd7a1a34222933f24f52a68bcd8"
                },
                {
                    "groupid": "4",
                    "name": "Zabbix servers",
                    "flags": "0",
                    "uuid": "6f6799aa69e844b4b3918f779f2abf08"
                }
            ]
        }
    ],
    "id": 1
}
```

Retrieving linked templates

Retrieve the IDs and names of templates linked to host ”10084”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectParentTemplates": [
            "templateid",
            "name"
        ],
        "hostids": "10084"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "parentTemplates": [
                {
                    "name": "Linux",
                    "templateid": "10001"
```

```
                },
                {
                    "name": "Zabbix Server",
                    "templateid": "10047"
                }
            ]
        }
    ],
    "id": 1
}
```

Retrieving hosts by template

Retrieve hosts that have the ”10001” (*Linux by Zabbix agent*) template linked to them.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "name"],
        "templateids": "10001"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "templateid": "10001",
            "hosts": [
                {
                    "hostid": "10084",
                    "name": "Zabbix server"
                },
                {
                    "hostid": "10603",
                    "name": "Host 1"
                },
                {
                    "hostid": "10604",
                    "name": "Host 2"
                }
            ]
        }
    ],
    "id": 1
}
```

Searching by host inventory data

Retrieve hosts that contain ”Linux” in the host inventory ”OS” field.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "host"
        ],
        "selectInventory": [
```

```
            "os"
        ],
        "searchInventory": {
            "os": "Linux"
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "host": "Zabbix server",
            "inventory": {
                "os": "Linux Ubuntu"
            }
        },
        {
            "hostid": "10107",
            "host": "Linux server",
            "inventory": {
                "os": "Linux Mint"
            }
        }
    ],
    "id": 1
}
```

Searching by host tags

Retrieve hosts that have tag ”Host name” equal to ”Linux server”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "Host name",
                "value": "Linux server",
                "operator": 1
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10085",
            "tags": [
                {
                    "tag": "Host name",
```

```
                    "value": "Linux server"
                },
                {
                    "tag": "OS",
                    "value": "RHEL 7"
                }
            ]
        }
    ],
    "id": 1
}
```

Retrieve hosts that have these tags not only on host level but also in their linked parent templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "tags": [
            {
                "tag": "A",
                "value": "1",
                "operator": 1
            }
        ],
        "inheritedTags": true
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10623",
            "name": "PC room 1"
        },
        {
            "hostid": "10601",
            "name": "Office"
        }
    ],
    "id": 1
}
```

Searching host with tags and template tags

Retrieve a host with tags and all tags that are linked to parent templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "hostids": 10502,
        "selectTags": ["tag", "value"],
        "selectInheritedTags": ["tag", "value"]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10502",
            "name": "Desktop",
            "tags": [
                {
                    "tag": "A",
                    "value": "1"
                }
            ],
            "inheritedTags": [
                {
                    "tag": "B",
                    "value": "2"
                }
            ]
        }
    ],
    "id": 1
}
```

Searching hosts by problem severity

Retrieve hosts that have ”Disaster” problems.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "severities": 5
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10160",
            "name": "Zabbix server"
        }
    ],
    "id": 1
}
```

Retrieve hosts that have ”Average” and ”High” problems.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["name"],
        "severities": [3, 4]
    },
    "id": 1
```

```
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "20170",
            "name": "Database"
        },
        {
            "hostid": "20183",
            "name": "workstation"
        }
    ],
    "id": 1
}
```

See also

- Host group
- Template
- User macro
- Host interface

Source

CHost::get() in *ui/include/classes/api/services/CHost.php*.

host.massadd

Description

`object host.massadd(object parameters)`

This method allows to simultaneously add multiple related objects to all the given hosts.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the hosts to update and the objects to add to all the hosts.

The method accepts the following parameters.

Parameter	Type	Description
hosts	object/array	Hosts to be updated. The hosts must have the `hostid` property defined. Parameter behavior: - *required*
groups	object/array	Host groups to add to the given hosts. The host groups must have the `groupid` property defined.
interfaces	object/array	Host interfaces to be created for the given hosts.
macros	object/array	User macros to be created for the given hosts.
templates	object/array	Templates to link to the given hosts. The templates must have the `templateid` property defined.

Return values

(object) Returns an object containing the IDs of the updated hosts under the hostids property.

Examples

Adding macros

Add two new macros to two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.massadd",
    "params": {
        "hosts": [
            {
                "hostid": "10160"
            },
            {
                "hostid": "10167"
            }
        ],
        "macros": [
            {
                "macro": "{$TEST1}",
                "value": "MACROTEST1"
            },
            {
                "macro": "{$TEST2}",
                "value": "MACROTEST2",
                "description": "Test description"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10160",
            "10167"
        ]
    },
    "id": 1
}
```

See also

- host.update
- Host group
- Template
- User macro
- Host interface

Source

CHost::massAdd() in *ui/include/classes/api/services/CHost.php*.

host.massremove

Description

`object host.massremove(object parameters)`

This method allows to remove related objects from multiple hosts.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the hosts to update and the objects that should be removed.

Parameter	Type	Description
hostids	string/array	IDs of the hosts to be updated. Parameter behavior: - *required*
groupids	string/array	Host groups to remove the given hosts from.
interfaces	object/array	Host interfaces to remove from the given hosts. The host interface object must have the `ip`, `dns` and `port` properties defined.
macros	string/array	User macros to delete from the given hosts.
templateids	string/array	Templates to unlink from the given hosts.
templateids_clear	string/array	Templates to unlink and clear from the given hosts.

Return values

`(object)` Returns an object containing the IDs of the updated hosts under the `hostids` property.

Examples

Unlinking templates

Unlink a template from two hosts and delete all of the templated entities.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.massremove",
    "params": {
        "hostids": ["69665", "69666"],
        "templateids_clear": "325"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "69665",
            "69666"
        ]
    },
    "id": 1
}
```

See also

- host.update
- User macro
- Host interface

Source

CHost::massRemove() in *ui/include/classes/api/services/CHost.php*.

host.massupdate

Description

`object host.massupdate(object parameters)`

This method allows to simultaneously replace or remove related objects and update properties on multiple hosts.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the hosts to update and the properties that should be updated.

Additionally to the standard host properties, the method accepts the following parameters.

Parameter	Type	Description
hosts	object/array	Hosts to be updated. The hosts must have the `hostid` property defined. Parameter behavior: - *required*
groups	object/array	Host groups to replace the current host groups the hosts belong to. The host groups must have the `groupid` property defined.
interfaces	object/array	Host interfaces to replace the current host interfaces on the given hosts.
inventory	object	Host inventory properties. Host inventory mode cannot be updated using the `inventory` parameter, use `inventory_mode` instead.
macros	object/array	User macros to replace the current user macros on the given hosts.
templates	object/array	Templates to replace the currently linked templates on the given hosts. The templates must have the `templateid` property defined.
templates_clear	object/array	Templates to unlink and clear from the given hosts. The templates must have the `templateid` property defined.

Return values

`(object)` Returns an object containing the IDs of the updated hosts under the `hostids` property.

Examples

Enabling multiple hosts

Enable monitoring of two hosts, that is, set their status to "0".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.massupdate",
    "params": {
        "hosts": [
            {
                "hostid": "69665"
            },
            {
                "hostid": "69666"
            }
        ],
```

```
        "status": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "69665",
            "69666"
        ]
    },
    "id": 1
}
```

See also

- host.update
- host.massadd
- host.massremove
- Host group
- Template
- User macro
- Host interface

Source

CHost::massUpdate() in *ui/include/classes/api/services/CHost.php*.

host.update

Description

`object host.update(object/array hosts)`

This method allows to update existing hosts.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Host properties to be updated.

The `hostid` property must be defined for each host, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.

Note, however, that updating the host technical name will also update the host's visible name (if not given or empty) by the host's technical name value.

Additionally to the standard host properties, the method accepts the following parameters.

Parameter	Type	Description
groups	object/array	Host groups to replace the current host groups the host belongs to. The host groups must have the `groupid` property defined. All host groups that are not listed in the request will be unlinked.
interfaces	object/array	Host interfaces to replace the current host interfaces. All interfaces that are not listed in the request will be removed.
tags	object/array	Host tags to replace the current host tags. All tags that are not listed in the request will be removed.

Parameter	Type	Description
inventory	object	Host inventory properties.
macros	object/array	User macros to replace the current user macros. All macros that are not listed in the request will be removed.
templates	object/array	Templates to replace the currently linked templates. All templates that are not listed in the request will be only unlinked. The templates must have the `templateid` property defined.
templates_clear	object/array	Templates to unlink and clear from the host. The templates must have the `templateid` property defined.

Note:
As opposed to the Zabbix frontend, when `name` (visible host name) is the same as `host` (technical host name), updating `host` via API will not automatically update `name`. Both properties need to be updated explicitly.

Return values

`(object)` Returns an object containing the IDs of the updated hosts under the `hostids` property.

Examples

Enabling a host

Enable host monitoring, that is, set its status to "0".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "status": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

Unlinking templates

Unlink and clear two templates from host.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "templates_clear": [
            {
                "templateid": "10124"
            },
```

```
            {
                "templateid": "10125"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

Updating host macros

Replace all host macros with two new ones.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "macros": [
            {
                "macro": "{$PASS}",
                "value": "password"
            },
            {
                "macro": "{$DISC}",
                "value": "sda",
                "description": "Updated description"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10126"
        ]
    },
    "id": 1
}
```

Updating host inventory

Change inventory mode and add location

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
```

```
        "hostid": "10387",
        "inventory_mode": 0,
        "inventory": {
            "location": "Latvia, Riga"
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10387"
        ]
    },
    "id": 1
}
```

Updating host tags

Replace all host tags with a new one.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10387",
        "tags": {
            "tag": "OS",
            "value": "RHEL 7"
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10387"
        ]
    },
    "id": 1
}
```

Updating discovered host macros

Convert discovery rule created ”automatic” macro to ”manual” and change its value to ”new-value”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10387",
        "macros": {
            "hostmacroid": "5541",
            "value": "new-value",
            "automatic": "0"
        }
```

```
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10387"
        ]
    },
    "id": 1
}
```

Updating host encryption

Update the host "10590" to use PSK encryption only for connections from host to Zabbix server, and change the PSK identity and PSK key. Note that the host has to be pre-configured to use PSK.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10590",
        "tls_connect": 1,
        "tls_accept": 2,
        "tls_psk_identity": "PSK 002",
        "tls_psk": "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10590"
        ]
    },
    "id": 1
}
```

See also

- host.massadd
- host.massupdate
- host.massremove
- Host group
- Template
- User macro
- Host interface
- Host inventory
- Host tag

Source

CHost::update() in *ui/include/classes/api/services/CHost.php*.

Host group

This class is designed to work with host groups.

Object references:

- Host group

Available methods:

- hostgroup.create - creating new host groups
- hostgroup.delete - deleting host groups
- hostgroup.get - retrieving host groups
- hostgroup.massadd - adding related objects to host groups
- hostgroup.massremove - removing related objects from host groups
- hostgroup.massupdate - replacing or removing related objects from host groups
- hostgroup.propagate - propagating permissions and tag filters to host groups' subgroups
- hostgroup.update - updating host groups

> Host group object

The following objects are directly related to the `hostgroup` API.

Host group

The host group object has the following properties.

Property	Type	Description
groupid	string	ID of the host group. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the host group. Property behavior: - *required* for create operations
flags	integer	Origin of the host group. Possible values: 0 - a plain host group; 4 - a discovered host group. Property behavior: - *read-only*
uuid	string	Universal unique identifier, used for linking imported host groups to already existing ones. Auto-generated, if not given.

hostgroup.create

Description

`object hostgroup.create(object/array hostGroups)`

This method allows to create new host groups.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Host groups to create.

The method accepts host groups with the standard host group properties.

Return values

(object) Returns an object containing the IDs of the created host groups under the groupids property. The order of the returned IDs matches the order of the passed host groups.

Examples

Creating a host group

Create a host group called "Linux servers".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.create",
    "params": {
        "name": "Linux servers"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107819"
        ]
    },
    "id": 1
}
```

Source

CHostGroup::create() in *ui/include/classes/api/services/CHostGroup.php*.

hostgroup.delete

Description

`object hostgroup.delete(array hostGroupIds)`

This method allows to delete host groups.

A host group cannot be deleted if:

- it contains hosts that belong to this group only;
- it is marked as internal;
- it is used by a host prototype;
- it is used in a global script;
- it is used in a correlation condition.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(array) IDs of the host groups to delete.

Return values

(object) Returns an object containing the IDs of the deleted host groups under the groupids property.

Examples

Deleting multiple host groups

Delete two host groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.delete",
    "params": [
        "107824",
        "107825"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107824",
            "107825"
        ]
    },
    "id": 1
}
```

Source

CHostGroup::delete() in *ui/include/classes/api/services/CHostGroup.php*.

hostgroup.get

Description

`integer/array hostgroup.get(object parameters)`

The method allows to retrieve host groups according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
graphids	string/array	Return only host groups that contain hosts with the given graphs.
groupids	string/array	Return only host groups with the given host group IDs.
hostids	string/array	Return only host groups that contain the given hosts.
maintenanceids	string/array	Return only host groups that are affected by the given maintenances.
triggerids	string/array	Return only host groups that contain hosts with the given triggers.
with_graphs	flag	Return only host groups that contain hosts with graphs.
with_graph_prototypes	flag	Return only host groups that contain hosts with graph prototypes.
with_hosts	flag	Return only host groups that contain hosts.
with_httptests	flag	Return only host groups that contain hosts with web checks. Overrides the `with_monitored_httptests` parameter.
with_items	flag	Return only host groups that contain hosts with items. Overrides the `with_monitored_items` and `with_simple_graph_items` parameters.
with_item_prototypes	flag	Return only host groups that contain hosts with item prototypes. Overrides the `with_simple_graph_item_prototypes` parameter.

Parameter	Type	Description
with_simple_graph_item_prototypes	flag	Return only host groups that contain hosts with item prototypes, which are enabled for creation and have numeric type of information.
with_monitored_httptests	flag	Return only host groups that contain hosts with enabled web checks.
with_monitored_hosts	flag	Return only host groups that contain monitored hosts.
with_monitored_items	flag	Return only host groups that contain hosts with enabled items. Overrides the `with_simple_graph_items` parameter.
with_monitored_triggers	flag	Return only host groups that contain hosts with enabled triggers. All of the items used in the trigger must also be enabled.
with_simple_graph_items	flag	Return only host groups that contain hosts with numeric items.
with_triggers	flag	Return only host groups that contain hosts with triggers. Overrides the `with_monitored_triggers` parameter.
selectDiscoveryRule	query	Return a discoveryRule property with the LLD rule that created the host group.
selectGroupDiscovery	query	Return a `groupDiscovery` property with the host group discovery object. The host group discovery object links a discovered host group to a host group prototype and has the following properties: `groupid` - `(string)` ID of the discovered host group; `lastcheck` - `(timestamp)` time when the host group was last discovered; `name` - `(string)` name of the host group prototype; `parent_group_prototypeid` - `(string)` ID of the host group prototype from which the host group has been created; `ts_delete` - `(timestamp)` time when a host group that is no longer discovered will be deleted.
selectHosts	query	Return a hosts property with the hosts that belong to the host group. Supports `count`.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectHosts` - results will be sorted by `host`.
sortfield	string/array	Sort the result by the given properties. Possible values: `groupid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
monitored_hosts (deprecated)	flag	This parameter is deprecated, please use `with_monitored_hosts` instead. Return only host groups that contain monitored hosts.
real_hosts (deprecated)	flag	This parameter is deprecated, please use `with_hosts` instead. Return only host groups that contain hosts.

Return values

`(integer/array)` Returns either:

- an array of objects;

- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving data by name

Retrieve all data about two host groups named ”Zabbix servers” and ”Linux servers”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.get",
    "params": {
        "output": "extend",
        "filter": {
            "name": [
                "Zabbix servers",
                "Linux servers"
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "groupid": "2",
            "name": "Linux servers",
            "internal": "0"
        },
        {
            "groupid": "4",
            "name": "Zabbix servers",
            "internal": "0"
        }
    ],
    "id": 1
}
```

See also

- Host

Source

CHostGroup::get() in *ui/include/classes/api/services/CHostGroup.php*.

hostgroup.massadd

Description

`object hostgroup.massadd(object parameters)`

This method allows to simultaneously add multiple related objects to all the given host groups.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the host groups to update and the objects to add to all the host groups.

The method accepts the following parameters.

Parameter	Type	Description
groups	object/array	Host groups to be updated. The host groups must have the `groupid` property defined. Parameter behavior: - *required*
hosts	object/array	Hosts to add to all host groups. The hosts must have the `hostid` property defined.

Return values

`(object)` Returns an object containing the IDs of the updated host groups under the `groupids` property.

Examples

Adding hosts to host groups

Add two hosts to host groups with IDs 5 and 6.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massadd",
    "params": {
        "groups": [
            {
                "groupid": "5"
            },
            {
                "groupid": "6"
            }
        ],
        "hosts": [
            {
                "hostid": "30050"
            },
            {
                "hostid": "30001"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "5",
            "6"
        ]
    },
    "id": 1
}
```

See also

- Host

Source

CHostGroup::massAdd() in *ui/include/classes/api/services/CHostGroup.php*.

hostgroup.massremove

Description

`object hostgroup.massremove(object parameters)`

This method allows to remove related objects from multiple host groups.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the host groups to update and the objects that should be removed.

Parameter	Type	Description
groupids	string/array	IDs of the host groups to be updated. Parameter behavior: *- required*
hostids	string/array	Hosts to remove from all host groups.

Return values

`(object)` Returns an object containing the IDs of the updated host groups under the `groupids` property.

Examples

Removing hosts from host groups

Remove two hosts from the given host groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massremove",
    "params": {
        "groupids": [
            "5",
            "6"
        ],
        "hostids": [
            "30050",
            "30001"
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "5",
            "6"
        ]
    },
    "id": 1
}
```

Source

CHostGroup::massRemove() in *ui/include/classes/api/services/CHostGroup.php*.

hostgroup.massupdate

Description

`object hostgroup.massupdate(object parameters)`

This method allows to replace hosts and templates with the specified ones in multiple host groups.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the host groups to update and the objects that should be updated.

Parameter	Type	Description
groups	object/array	Host groups to be updated. The host groups must have the `groupid` property defined. Parameter behavior: - *required*
hosts	object/array	Hosts to replace the current hosts on the given host groups. All other hosts, except the ones mentioned, will be excluded from host groups. Discovered hosts will not be affected. The hosts must have the `hostid` property defined. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the updated host groups under the `groupids` property.

Examples

Replacing hosts in a host group

Replace all hosts in a host group to ones mentioned host.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.massupdate",
    "params": {
        "groups": [
            {
                "groupid": "6"
            }
        ],
        "hosts": [
            {
                "hostid": "30050"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "6",
        ]
    },
    "id": 1
}
```


See also

- hostgroup.update
- hostgroup.massadd
- Host

Source

CHostGroup::massUpdate() in *ui/include/classes/api/services/CHostGroup.php*.

hostgroup.propagate

Description

`object hostgroup.propagate(object parameters)`

This method allows to apply permissions and tag filters to all subgroups of a host group.

> **Note:**
> This method is only available to *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
groups	object/array	Host groups to propagate. The host groups must have the `groupid` property defined. Parameter behavior: - *required*
permissions	boolean	Set to ”true” to propagate permissions. Parameter behavior: - *required* if `tag_filters` is not set
tag_filters	boolean	Set to ”true” to propagate tag filters. Parameter behavior: - *required* if `permissions` is not set

Return values

`(object)` Returns an object containing the IDs of the propagated host groups under the `groupids` property.

Examples

Propagating host group permissions and tag filters to its subgroups.

Propagate host group permissions and tag filters to its subgroups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.propagate",
    "params": {
        "groups": [
            {
                "groupid": "6"
            }
        ],
        "permissions": true,
        "tag_filters": true
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "6",
        ]
    },
    "id": 1
}
```

See also

- hostgroup.update
- hostgroup.massadd
- Host

Source

CHostGroup::propagate() in *ui/include/classes/api/services/CHostGroup.php*.

hostgroup.update

Description

`object hostgroup.update(object/array hostGroups)`

This method allows to update existing hosts groups.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Host group properties to be updated.

The `groupid` property must be defined for each host group, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.

Return values

`(object)` Returns an object containing the IDs of the updated host groups under the `groupids` property.

Examples

Renaming a host group

Rename a host group to ”Linux hosts.”

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostgroup.update",
    "params": {
        "groupid": "7",
        "name": "Linux hosts"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "7"
        ]
    },
    "id": 1
}
```

Source

CHostGroup::update() in *ui/include/classes/api/services/CHostGroup.php*.

Host interface

This class is designed to work with host interfaces.

Object references:

- Host interface

Available methods:

- hostinterface.create - creating new host interfaces
- hostinterface.delete - deleting host interfaces
- hostinterface.get - retrieving host interfaces
- hostinterface.massadd - adding host interfaces to hosts
- hostinterface.massremove - removing host interfaces from hosts
- hostinterface.replacehostinterfaces - replacing host interfaces on a host
- hostinterface.update - updating host interfaces

> Host interface object

The following objects are directly related to the `hostinterface` API.

Host interface

The host interface object has the following properties.

> **Attention:**
> Note that both `ip` and `dns` properties are *required* for create operations. If you do not want to use DNS, set it to an empty string.

Property	Type	Description
interfaceid	string	ID of the interface. Property behavior: - *read-only* - *required* for update operations

Property	Type	Description
available	integer	Availability of host interface. Possible values: 0 - *(default)* unknown; 1 - available; 2 - unavailable. Property behavior: - *read-only*
hostid	string	ID of the host that the interface belongs to. Property behavior: - *constant* - *required* for create operations
type	integer	Interface type. Possible values: 1 - Agent; 2 - SNMP; 3 - IPMI; 4 - JMX. Property behavior: - *required* for create operations
ip	string	IP address used by the interface. Can be empty if the connection is made via DNS. Property behavior: - *required* for create operations
dns	string	DNS name used by the interface. Can be empty if the connection is made via IP. Property behavior: - *required* for create operations
port	string	Port number used by the interface. Can contain user macros. Property behavior: - *required* for create operations
useip	integer	Whether the connection should be made via IP. Possible values: 0 - connect using host DNS name; 1 - connect using host IP address. Property behavior: - *required* for create operations
main	integer	Whether the interface is used as default on the host. Only one interface of some type can be set as default on a host. Possible values: 0 - not default; 1 - default. Property behavior: - *required* for create operations
details	array	Additional details object for interface. Property behavior: - *required* if `type` is set to ”SNMP”

Property	Type	Description
disable_until	timestamp	The next polling time of an unavailable host interface. Property behavior: - *read-only*
error	string	Error text if host interface is unavailable. Property behavior: - *read-only*
errors_from	timestamp	Time when host interface became unavailable. Property behavior: - *read-only*

Details

The details object has the following properties.

Property	Type	Description
version	integer	SNMP interface version. Possible values: 1 - SNMPv1; 2 - SNMPv2c; 3 - SNMPv3. Property behavior: - *required*
bulk	integer	Whether to use bulk SNMP requests. Possible values: 0 - don't use bulk requests; 1 - (default) - use bulk requests.
community	string	SNMP community. Used only by SNMPv1 and SNMPv2 interfaces. Property behavior: - *required* if `version` is set to "SNMPv1" or "SNMPv2c"
max_repetitions	integer	Max repetition value for native SNMP bulk requests (GetBulkRequest-PDUs). Used only for `discovery[]` and `walk[]` items in SNMPv2 and v3. Default: 10.
securityname	string	SNMPv3 security name. Used only by SNMPv3 interfaces.
securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 interfaces. Possible values: 0 - (default) - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
authpassphrase	string	SNMPv3 authentication passphrase. Used only by SNMPv3 interfaces.
privpassphrase	string	SNMPv3 privacy passphrase. Used only by SNMPv3 interfaces.
authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 interfaces. Possible values: 0 - (default) - MD5; 1 - SHA1; 2 - SHA224; 3 - SHA256; 4 - SHA384; 5 - SHA512.

Property	Type	Description
privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 interfaces. Possible values: 0 - (default) - DES; 1 - AES128; 2 - AES192; 3 - AES256; 4 - AES192C; 5 - AES256C.
contextname	string	SNMPv3 context name. Used only by SNMPv3 interfaces.

hostinterface.create

Description

`object hostinterface.create(object/array hostInterfaces)`

This method allows to create new host interfaces.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Host interfaces to create.

The method accepts host interfaces with the standard host interface properties.

Return values

`(object)` Returns an object containing the IDs of the created host interfaces under the `interfaceids` property. The order of the returned IDs matches the order of the passed host interfaces.

Examples

Create a new interface

Create a secondary IP agent interface on host "30052."

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.create",
    "params": {
        "hostid": "30052",
        "main": "0",
        "type": "1",
        "useip": "1",
        "ip": "127.0.0.1",
        "dns": "",
        "port": "10050"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30062"
        ]
    },
```

```
    "id": 1
}
```

Create an interface with SNMP details

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.create",
    "params": {
        "hostid": "10456",
        "main": "0",
        "type": "2",
        "useip": "1",
        "ip": "127.0.0.1",
        "dns": "",
        "port": "1601",
        "details": {
            "version": "2",
            "bulk": "1",
            "community": "{$SNMP_COMMUNITY}"
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30063"
        ]
    },
    "id": 1
}
```

See also

- hostinterface.massadd
- host.massadd

Source

CHostInterface::create() in *ui/include/classes/api/services/CHostInterface.php*.

hostinterface.delete

Description

`object hostinterface.delete(array hostInterfaceIds)`

This method allows to delete host interfaces.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the host interfaces to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted host interfaces under the `interfaceids` property.

Examples

Delete a host interface

Delete the host interface with ID 30062.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.delete",
    "params": [
        "30062"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30062"
        ]
    },
    "id": 1
}
```

See also

- hostinterface.massremove
- host.massremove

Source

CHostInterface::delete() in *ui/include/classes/api/services/CHostInterface.php*.

hostinterface.get

Description

`integer/array hostinterface.get(object parameters)`

The method allows to retrieve host interfaces according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
hostids	string/array	Return only host interfaces used by the given hosts.
interfaceids	string/array	Return only host interfaces with the given IDs.
itemids	string/array	Return only host interfaces used by the given items.
triggerids	string/array	Return only host interfaces used by items in the given triggers.
selectItems	query	Return an items property with the items that use the interface. Supports `count`.
selectHosts	query	Return a hosts property with an array of hosts that use the interface.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectItems`.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties. Possible values: `interfaceid`, `dns`, `ip`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve host interfaces

Retrieve all data about the interfaces used by host "30057."

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.get",
    "params": {
        "output": "extend",
        "hostids": "30057"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "interfaceid": "50039",
            "hostid": "30057",
            "main": "1",
            "type": "1",
            "useip": "0",
            "ip": "",
            "dns": "localhost",
            "port": "10050",
            "available": "0",
            "error": "",
            "errors_from": "0",
            "disable_until": "0",
            "details": []
        },
        {
            "interfaceid": "55082",
            "hostid": "30057",
```

```
            "main": "1",
            "type": "2",
            "useip": "1",
            "ip": "127.0.0.1",
            "dns": "",
            "port": "161",
            "available": "0",
            "error": "",
            "errors_from": "0",
            "disable_until": "0",
            "details": {
                "version": "2",
                "bulk": "0",
                "community": "{$SNMP_COMMUNITY}",
                "max_repetitions": "10"
            }
        }
    ],
    "id": 1
}
```


See also

- Host
- Item

Source

CHostInterface::get() in *ui/include/classes/api/services/CHostInterface.php*.

hostinterface.massadd

Description

`object hostinterface.massadd(object parameters)`

This method allows to simultaneously add host interfaces to multiple hosts.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the host interfaces to be created on the given hosts.

The method accepts the following parameters.

Parameter	Type	Description
interfaces	object/array	Host interfaces to create on the given hosts. Parameter behavior: - *required*
hosts	object/array	Hosts to be updated. The hosts must have the `hostid` property defined. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the created host interfaces under the `interfaceids` property.

Examples

Creating interfaces

Create an interface on two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.massadd",
    "params": {
        "hosts": [
            {
                "hostid": "30050"
            },
            {
                "hostid": "30052"
            }
        ],
        "interfaces": {
            "dns": "",
            "ip": "127.0.0.1",
            "main": 0,
            "port": "10050",
            "type": 1,
            "useip": 1
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30069",
            "30070"
        ]
    },
    "id": 1
}
```

See also

- hostinterface.create
- host.massadd
- Host

Source

CHostInterface::massAdd() in *ui/include/classes/api/services/CHostInterface.php*.

hostinterface.massremove

Description

`object hostinterface.massremove(object parameters)`

This method allows to remove host interfaces from the given hosts.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the hosts to be updated and the interfaces to be removed.

Parameter	Type	Description
interfaces	object/array	Host interfaces to remove from the given hosts. The host interface object must have the `ip`, `dns` and `port` properties defined. Parameter behavior: - *required*
hostids	string/array	IDs of the hosts to be updated. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the deleted host interfaces under the `interfaceids` property.

Examples

Removing interfaces

Remove the ”127.0.0.1” SNMP interface from two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.massremove",
    "params": {
        "hostids": [
            "30050",
            "30052"
        ],
        "interfaces": {
            "dns": "",
            "ip": "127.0.0.1",
            "port": "161"
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30069",
            "30070"
        ]
    },
    "id": 1
}
```

See also

- hostinterface.delete
- host.massremove

Source

CHostInterface::massRemove() in *ui/include/classes/api/services/CHostInterface.php*.

hostinterface.replacehostinterfaces

Description

`object hostinterface.replacehostinterfaces(object parameters)`

This method allows to replace all host interfaces on a given host.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the ID of the host to be updated and the new host interfaces.

Parameter	Type	Description
interfaces	object/array	Host interfaces to replace the current host interfaces with. Parameter behavior: - *required*
hostid	string	ID of the host to be updated. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the created host interfaces under the `interfaceids` property.

Examples

Replacing host interfaces

Replace all host interfaces with a single agent interface.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.replacehostinterfaces",
    "params": {
        "hostid": "30052",
        "interfaces": {
            "dns": "",
            "ip": "127.0.0.1",
            "main": 1,
            "port": "10050",
            "type": 1,
            "useip": 1
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30081"
        ]
    },
    "id": 1
}
```

See also

- host.update
- host.massupdate

Source

CHostInterface::replaceHostInterfaces() in *ui/include/classes/api/services/CHostInterface.php*.

hostinterface.update

Description

`object hostinterface.update(object/array hostInterfaces)`

This method allows to update existing host interfaces.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Host interface properties to be updated.

The `interfaceid` property must be defined for each host interface, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.

Return values

`(object)` Returns an object containing the IDs of the updated host interfaces under the `interfaceids` property.

Examples

Changing a host interface port

Change the port of a host interface.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostinterface.update",
    "params": {
        "interfaceid": "30048",
        "port": "30050"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "interfaceids": [
            "30048"
        ]
    },
    "id": 1
}
```

Source

CHostInterface::update() in *ui/include/classes/api/services/CHostInterface.php*.

Host prototype

This class is designed to work with host prototypes.

Object references:

- Host prototype
- Host prototype inventory

- Group link
- Group prototype

Available methods:

- hostprototype.create - creating new host prototypes
- hostprototype.delete - deleting host prototypes
- hostprototype.get - retrieving host prototypes
- hostprototype.update - updating host prototypes

> Host prototype object

The following objects are directly related to the `hostprototype` API.

Host prototype

The host prototype object has the following properties.

Property	Type	Description
hostid	string	ID of the host prototype. Property behavior: - *read-only* - *required* for update operations
host	string	Technical name of the host prototype. Property behavior: - *required* for create operations - *read-only* for inherited objects
name	string	Visible name of the host prototype. Default: `host` property value. Property behavior: - *read-only* for inherited objects
status	integer	Status of the host prototype. Possible values: 0 - *(default)* monitored host; 1 - unmonitored host.
inventory_mode	integer	Host inventory population mode. Possible values: -1 - *(default)* disabled; 0 - manual; 1 - automatic.
templateid	string	ID of the parent template host prototype. Property behavior: - *read-only*
discover	integer	Host prototype discovery status. Possible values: 0 - *(default)* new hosts will be discovered; 1 - new hosts will not be discovered and existing hosts will be marked as lost.

Property	Type	Description
custom_interfaces	integer	Source of custom interfaces for hosts created by the host prototype. Possible values: 0 - *(default)* inherit interfaces from parent host; 1 - use host prototypes custom interfaces. Property behavior: - *read-only* for inherited objects
uuid	string	Universal unique identifier, used for linking imported host prototypes to already existing ones. Auto-generated, if not given. Property behavior: - *supported* if the host prototype belongs to a template

Group link

The group link object links a host prototype with a host group. It has the following properties.

Property	Type	Description
groupid	string	ID of the host group. Property behavior: - *required*

Group prototype

The group prototype object defines a group that will be created for a discovered host. It has the following properties.

Property	Type	Description
name	string	Name of the group prototype. Property behavior: - *required*

Host prototype tag

The host prototype tag object has the following properties.

Property	Type	Description
tag	string	Host prototype tag name. Property behavior: - *required*
value	string	Host prototype tag value.

Custom interface

Custom interfaces are supported if `custom_interfaces` of Host prototype object is set to "use host prototypes custom interfaces". The custom interface object has the following properties.

Property	Type	Description
type	integer	Interface type. Possible values: 1 - Agent; 2 - SNMP; 3 - IPMI; 4 - JMX. Property behavior: - *required*
useip	integer	Whether the connection should be made via IP. Possible values: 0 - connect using host DNS name; 1 - connect using host IP address. Property behavior: - *required*
ip	string	IP address used by the interface. Can contain macros. Property behavior: - *required* if `useip` is set to ”connect using host IP address”
dns	string	DNS name used by the interface. Can contain macros. Property behavior: - *required* if `useip` is set to ”connect using host DNS name”
port	string	Port number used by the interface. Can contain user and LLD macros. Property behavior: - *required*
main	integer	Whether the interface is used as default on the host. Only one interface of some type can be set as default on a host. Possible values: 0 - not default; 1 - default. Property behavior: - *required*
details	array	Additional object for interface. Property behavior: - *required* if `type` is set to ”SNMP”

Custom interface details

The details object has the following properties.

Property	Type	Description
version	integer	SNMP interface version. Possible values: 1 - SNMPv1; 2 - SNMPv2c; 3 - SNMPv3. Property behavior: - *required*
bulk	integer	Whether to use bulk SNMP requests. Possible values: 0 - don't use bulk requests; 1 - *(default)* - use bulk requests.
community	string	SNMP community. Property behavior: - *required* if `version` is set to "SNMPv1" or "SNMPv2c"
max_repetitions	integer	Max repetition value for native SNMP bulk requests (GetBulkRequest-PDUs). Used only for `discovery[]` and `walk[]` items in SNMPv2 and v3. Default: 10.
securityname	string	SNMPv3 security name. Property behavior: - *supported* if `version` is set to "SNMPv3"
securitylevel	integer	SNMPv3 security level. Possible values: 0 - *(default)* - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv. Property behavior: - *supported* if `version` is set to "SNMPv3"
authpassphrase	string	SNMPv3 authentication passphrase. Property behavior: - *supported* if `version` is set to "SNMPv3" and `securitylevel` is set to "authNoPriv" or "authPriv"
privpassphrase	string	SNMPv3 privacy passphrase. Property behavior: - *supported* if `version` is set to "SNMPv3" and `securitylevel` is set to "authPriv"
authprotocol	integer	SNMPv3 authentication protocol. Possible values: 0 - *(default)* - MD5; 1 - SHA1; 2 - SHA224; 3 - SHA256; 4 - SHA384; 5 - SHA512. Property behavior: - *supported* if `version` is set to "SNMPv3" and `securitylevel` is set to "authNoPriv" or "authPriv"

Property	Type	Description
privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 interfaces. Possible values: 0 - *(default)* - DES; 1 - AES128; 2 - AES192; 3 - AES256; 4 - AES192C; 5 - AES256C. Property behavior: - *supported* if `version` is set to "SNMPv3" and `securitylevel` is set to "authPriv"
contextname	string	SNMPv3 context name. Property behavior: - *supported* if `version` is set to "SNMPv3"

hostprototype.create

Description

`object hostprototype.create(object/array hostPrototypes)`

This method allows to create new host prototypes.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Host prototypes to create.

Additionally to the standard host prototype properties, the method accepts the following parameters.

Parameter	Type	Description
groupLinks	array	Group links to be created for the host prototype. Parameter behavior: - *required*
ruleid	string	ID of the LLD rule that the host prototype belongs to. Parameter behavior: - *required*
groupPrototypes	array	Group prototypes to be created for the host prototype.
macros	object/array	User macros to be created for the host prototype.
tags	object/array	Host prototype tags.
interfaces	object/array	Host prototype custom interfaces.
templates	object/array	Templates to be linked to the host prototype. The templates must have the `templateid` property defined.

Return values

`(object)` Returns an object containing the IDs of the created host prototypes under the `hostids` property. The order of the returned IDs matches the order of the passed host prototypes.

Examples

Creating a host prototype

Create a host prototype "{#VM.NAME}" on LLD rule "23542" with a group prototype "{#HV.NAME}", tag pair "Datacenter": "{#DATACENTER.NAME}" and custom SNMPv2 interface 127.0.0.1:161 with community {$SNMP_COMMUNITY}. Link it to host group "2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.create",
    "params": {
        "host": "{#VM.NAME}",
        "ruleid": "23542",
        "custom_interfaces": "1",
        "groupLinks": [
            {
                "groupid": "2"
            }
        ],
        "groupPrototypes": [
            {
                "name": "{#HV.NAME}"
            }
        ],
        "tags": [
            {
                "tag": "Datacenter",
                "value": "{#DATACENTER.NAME}"
            }
        ],
        "interfaces": [
            {
                "main": "1",
                "type": "2",
                "useip": "1",
                "ip": "127.0.0.1",
                "dns": "",
                "port": "161",
                "details": {
                    "version": "2",
                    "bulk": "1",
                    "community": "{$SNMP_COMMUNITY}"
                }
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10103"
        ]
    },
    "id": 1
}
```

See also

- Group link
- Group prototype
- Host prototype tag
- Custom interface

- User macro

Source

CHostPrototype::create() in *ui/include/classes/api/services/CHostPrototype.php*.

hostprototype.delete

Description

`object hostprototype.delete(array hostPrototypeIds)`

This method allows to delete host prototypes.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the host prototypes to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted host prototypes under the `hostids` property.

Examples

Deleting multiple host prototypes

Delete two host prototypes.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.delete",
    "params": [
        "10103",
        "10105"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10103",
            "10105"
        ]
    },
    "id": 1
}
```

Source

CHostPrototype::delete() in *ui/include/classes/api/services/CHostPrototype.php*.

hostprototype.get

Description

`integer/array hostprototype.get(object parameters)`

The method allows to retrieve host prototypes according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
hostids	string/array	Return only host prototypes with the given IDs.
discoveryids	string/array	Return only host prototype that belong to the given LLD rules.
inherited	boolean	If set to `true` return only items inherited from a template.
selectDiscoveryRule	query	Return a discoveryRule property with the LLD rule that the host prototype belongs to.
selectInterfaces	query	Return an interfaces property with host prototype custom interfaces.
selectGroupLinks	query	Return a groupLinks property with the group links of the host prototype.
selectGroupPrototypes	query	Return a groupPrototypes property with the group prototypes of the host prototype.
selectMacros	query	Return a macros property with host prototype macros.
selectParentHost	query	Return a parentHost property with the host that the host prototype belongs to.
selectTags	query	Return a tags property with host prototype tags.
selectTemplates	query	Return a templates property with the templates linked to the host prototype. Supports `count`.
sortfield	string/array	Sort the result by the given properties. Possible values: `hostid`, `host`, `name`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving host prototypes from an LLD rule

Retrieve all host prototypes, their group links, group prototypes and tags from an LLD rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.get",
    "params": {
        "output": "extend",
```

```
        "selectInterfaces": "extend",
        "selectGroupLinks": "extend",
        "selectGroupPrototypes": "extend",
        "selectTags": "extend",
        "discoveryids": "23554"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10092",
            "host": "{#HV.UUID}",
            "name": "{#HV.UUID}",
            "status": "0",
            "templateid": "0",
            "discover": "0",
            "custom_interfaces": "1",
            "inventory_mode": "-1",
            "groupLinks": [
                {
                    "group_prototypeid": "4",
                    "hostid": "10092",
                    "groupid": "7",
                    "templateid": "0"
                }
            ],
            "groupPrototypes": [
                {
                    "group_prototypeid": "7",
                    "hostid": "10092",
                    "name": "{#CLUSTER.NAME}",
                    "templateid": "0"
                }
            ],
            "tags": [
                {
                    "tag": "Datacenter",
                    "value": "{#DATACENTER.NAME}"
                },
                {
                    "tag": "Instance type",
                    "value": "{#INSTANCE_TYPE}"
                }
            ],
            "interfaces": [
                {
                    "main": "1",
                    "type": "2",
                    "useip": "1",
                    "ip": "127.0.0.1",
                    "dns": "",
                    "port": "161",
                    "details": {
                        "version": "2",
                        "bulk": "1",
                        "community": "{$SNMP_COMMUNITY}",
                        "max_repetitions": "10"
                    }
```

```
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Group link
- Group prototype
- User macro

Source

CHostPrototype::get() in *ui/include/classes/api/services/CHostPrototype.php*.

hostprototype.update

Description

`object hostprototype.update(object/array hostPrototypes)`

This method allows to update existing host prototypes.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Host prototype properties to be updated.

The `hostid` property must be defined for each host prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard host prototype properties, the method accepts the following parameters.

Parameter	Type	Description
groupLinks	array	Group links to replace the current group links on the host prototype. Parameter behavior: - *read-only* for inherited objects
groupPrototypes	array	Group prototypes to replace the existing group prototypes on the host prototype. Parameter behavior: - *read-only* for inherited objects
macros	object/array	User macros to replace the current user macros. All macros that are not listed in the request will be removed.
tags	object/array	Host prototype tags to replace the current tags. All tags that are not listed in the request will be removed. Parameter behavior: - *read-only* for inherited objects

Parameter	Type	Description
interfaces	object/array	Host prototype custom interfaces to replace the current interfaces. Custom interface object should contain all its parameters. All interfaces that are not listed in the request will be removed. Parameter behavior: - *supported* if `custom_interfaces` of Host prototype object is set to "use host prototypes custom interfaces" - *read-only* for inherited objects
templates	object/array	Templates to replace the currently linked templates. The templates must have the `templateid` property defined.

Return values

`(object)` Returns an object containing the IDs of the updated host prototypes under the `hostids` property.

Examples

Disabling a host prototype

Disable a host prototype, that is, set its status to "1".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.update",
    "params": {
        "hostid": "10092",
        "status": 1
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10092"
        ]
    },
    "id": 1
}
```

Updating host prototype tags

Replace host prototype tags with new ones.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.update",
    "params": {
        "hostid": "10092",
        "tags": [
            {
                "tag": "Datacenter",
                "value": "{#DATACENTER.NAME}"
            },
            {
                "tag": "Instance type",
                "value": "{#INSTANCE_TYPE}"
```

```
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10092"
        ]
    },
    "id": 1
}
```

Updating host prototype custom interfaces

Replace inherited interfaces with host prototype custom interfaces.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "hostprototype.update",
    "params": {
        "hostid": "10092",
        "custom_interfaces": "1",
        "interfaces": [
            {
                "main": "1",
                "type": "2",
                "useip": "1",
                "ip": "127.0.0.1",
                "dns": "",
                "port": "161",
                "details": {
                    "version": "2",
                    "bulk": "1",
                    "community": "{$SNMP_COMMUNITY}"
                }
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10092"
        ]
    },
    "id": 1
}
```

See also

- Group link
- Group prototype
- Host prototype tag
- Custom interface

- User macro

Source

CHostPrototype::update() in *ui/include/classes/api/services/CHostPrototype.php*.

Housekeeping

This class is designed to work with housekeeping.

Object references:

- Housekeeping

Available methods:

- housekeeping.get - retrieve housekeeping
- housekeeping.update - update housekeeping

> Housekeeping object

The following objects are directly related to the `housekeeping` API.

Housekeeping

The settings object has the following properties.

Property	Type	Description
hk_events_mode	integer	Enable internal housekeeping for events and alerts. Possible values: 0 - Disable; 1 - *(default)* Enable.
hk_events_trigger	string	Trigger data storage period. Accepts seconds and time unit with suffix. Default: 365d.
hk_events_service	string	Service data storage period. Accepts seconds and time unit with suffix. Default: 1d.
hk_events_internal	string	Internal data storage period. Accepts seconds and time unit with suffix. Default: 1d.
hk_events_discovery	string	Network discovery data storage period. Accepts seconds and time unit with suffix. Default: 1d.
hk_events_autoreg	string	Autoregistration data storage period. Accepts seconds and time unit with suffix. Default: 1d.
hk_services_mode	integer	Enable internal housekeeping for services. Possible values: 0 - Disable; 1 - *(default)* Enable.
hk_services	string	Services data storage period. Accepts seconds and time unit with suffix. Default: 365d.

Property	Type	Description
hk_audit_mode	integer	Enable internal housekeeping for audit. Possible values: 0 - Disable; 1 - *(default)* Enable.
hk_audit	string	Audit data storage period. Accepts seconds and time unit with suffix. Default: 365d.
hk_sessions_mode	integer	Enable internal housekeeping for sessions. Possible values: 0 - Disable; 1 - *(default)* Enable.
hk_sessions	string	Sessions data storage period. Accepts seconds and time unit with suffix. Default: 365d.
hk_history_mode	integer	Enable internal housekeeping for history. Possible values: 0 - Disable; 1 - *(default)* Enable.
hk_history_global	integer	Override item history period. Possible values: 0 - Do not override; 1 - *(default)* Override.
hk_history	string	History data storage period. Accepts seconds and time unit with suffix. Default: 90d.
hk_trends_mode	integer	Enable internal housekeeping for trends. Possible values: 0 - Disable; 1 - *(default)* Enable.
hk_trends_global	integer	Override item trend period. Possible values: 0 - Do not override; 1 - *(default)* Override.
hk_trends	string	Trends data storage period. Accepts seconds and time unit with suffix. Default: 365d.
db_extension	string	Configuration flag DB extension. If this flag is set to ”timescaledb” then the server changes its behavior for housekeeping and item deletion. Property behavior: - *read-only*
compression_availability	integer	Whether data compression is supported by the database (or its extension). Possible values: 0 - Unavailable; 1 - Available. Property behavior: - *read-only*
compression_status	integer	Enable TimescaleDB compression for history and trends. Possible values: 0 - *(default)* Off; 1 - On.

Property	Type	Description
compress_older	string	Compress history and trends records older than specified period. Accepts seconds and time unit with suffix. Default: 7d.

housekeeping.get

Description

`object housekeeping.get(object parameters)`

The method allows to retrieve housekeeping object according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports only one parameter.

Parameter	Type	Description
output	query	This parameter being common for all `get` methods described in the reference commentary.

Return values

`(object)` Returns housekeeping object.

Examples

Request:

```
{
    "jsonrpc": "2.0",
    "method": "housekeeping.get",
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hk_events_mode": "1",
        "hk_events_trigger": "365d",
        "hk_events_service": "1d",
        "hk_events_internal": "1d",
        "hk_events_discovery": "1d",
        "hk_events_autoreg": "1d",
        "hk_services_mode": "1",
        "hk_services": "365d",
        "hk_audit_mode": "1",
        "hk_audit": "365d",
        "hk_sessions_mode": "1",
        "hk_sessions": "365d",
        "hk_history_mode": "1",
        "hk_history_global": "0",
```

```
        "hk_history": "90d",
        "hk_trends_mode": "1",
        "hk_trends_global": "0",
        "hk_trends": "365d",
        "db_extension": "",
        "compression_status": "0",
        "compress_older": "7d"
    },
    "id": 1
}
```

Source

CHousekeeping ::get() in *ui/include/classes/api/services/CHousekeeping.php*.

housekeeping.update

Description

`object housekeeping.update(object housekeeping)`

This method allows to update existing housekeeping settings.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Housekeeping properties to be updated.

Return values

`(array)` Returns an array with the names of updated parameters.

Examples

Request:

```
{
    "jsonrpc": "2.0",
    "method": "housekeeping.update",
    "params": {
        "hk_events_mode": "1",
        "hk_events_trigger": "200d",
        "hk_events_internal": "2d",
        "hk_events_discovery": "2d"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        "hk_events_mode",
        "hk_events_trigger",
        "hk_events_internal",
        "hk_events_discovery"
    ],
    "id": 1
}
```

Source

CHousekeeping::update() in *ui/include/classes/api/services/CHousekeeping.php*.

Icon map

This class is designed to work with icon maps.

Object references:

- Icon map
- Icon mapping

Available methods:

- iconmap.create - create new icon maps
- iconmap.delete - delete icon maps
- iconmap.get - retrieve icon maps
- iconmap.update - update icon maps

> Icon map object

The following objects are directly related to the `iconmap` API.

Icon map

The icon map object has the following properties.

Property	Type	Description
iconmapid	string	ID of the icon map. Property behavior: - *read-only* - *required* for update operations
default_iconid	string	ID of the default icon. Property behavior: - *required* for create operations
name	string	Name of the icon map. Property behavior: - *required* for create operations

Icon mapping

The icon mapping object defines a specific icon to be used for hosts with a certain inventory field value. It has the following properties.

Property	Type	Description
iconid	string	ID of the icon used by the icon mapping. Property behavior: - *required*
expression	string	Expression to match the inventory field against. Property behavior: - *required*
inventory_link	integer	ID of the host inventory field. Refer to the host inventory object for a list of supported inventory fields. Property behavior: - *required*

Property	Type	Description
sortorder	integer	Position of the icon mapping in the icon map. Property behavior: - *read-only*

iconmap.create

Description

`object iconmap.create(object/array iconMaps)`

This method allows to create new icon maps.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Icon maps to create.

Additionally to the standard icon map properties, the method accepts the following parameters.

Parameter	Type	Description
mappings	array	Icon mappings to be created for the icon map. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the created icon maps under the `iconmapids` property. The order of the returned IDs matches the order of the passed icon maps.

Examples

Create an icon map

Create an icon map to display hosts of different types.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.create",
    "params": {
        "name": "Type icons",
        "default_iconid": "2",
        "mappings": [
            {
                "inventory_link": 1,
                "expression": "server",
                "iconid": "3"
            },
            {
                "inventory_link": 1,
                "expression": "switch",
                "iconid": "4"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "2"
        ]
    },
    "id": 1
}
```

See also

- Icon mapping

Source

CIconMap::create() in *ui/include/classes/api/services/CIconMap.php*.

iconmap.delete

Description

`object iconmap.delete(array iconMapIds)`

This method allows to delete icon maps.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the icon maps to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted icon maps under the `iconmapids` property.

Examples

Delete multiple icon maps

Delete two icon maps.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.delete",
    "params": [
        "2",
        "5"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "2",
            "5"
        ]
    },
    "id": 1
}
```

Source

CIconMap::delete() in *ui/include/classes/api/services/CIconMap.php*.

iconmap.get

Description

`integer/array iconmap.get(object parameters)`

The method allows to retrieve icon maps according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
iconmapids	string/array	Return only icon maps with the given IDs.
sysmapids	string/array	Return only icon maps that are used in the given maps.
selectMappings	query	Return a mappings property with the icon mappings used.
sortfield	string/array	Sort the result by the given properties. Possible values: `iconmapid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve an icon map

Retrieve all data about icon map "3".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.get",
    "params": {
        "iconmapids": "3",
        "output": "extend",
        "selectMappings": "extend"
    },
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mappings": [
                {
                    "iconmappingid": "3",
                    "iconmapid": "3",
                    "iconid": "6",
                    "inventory_link": "1",
                    "expression": "server",
                    "sortorder": "0"
                },
                {
                    "iconmappingid": "4",
                    "iconmapid": "3",
                    "iconid": "10",
                    "inventory_link": "1",
                    "expression": "switch",
                    "sortorder": "1"
                }
            ],
            "iconmapid": "3",
            "name": "Host type icons",
            "default_iconid": "2"
        }
    ],
    "id": 1
}
```

See also

- Icon mapping

Source

CIconMap::get() in *ui/include/classes/api/services/CIconMap.php*.

iconmap.update

Description

`object iconmap.update(object/array iconMaps)`

This method allows to update existing icon maps.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Icon map properties to be updated.

The `iconmapid` property must be defined for each icon map, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard icon map properties, the method accepts the following parameters.

Parameter	Type	Description
mappings	array	Icon mappings to replace the existing icon mappings.

Return values

`(object)` Returns an object containing the IDs of the updated icon maps under the `iconmapids` property.

Examples

Rename icon map

Rename an icon map to ”OS icons”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "iconmap.update",
    "params": {
        "iconmapid": "1",
        "name": "OS icons"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "iconmapids": [
            "1"
        ]
    },
    "id": 1
}
```

See also

- Icon mapping

Source

CIconMap::update() in *ui/include/classes/api/services/CIconMap.php*.

Image

This class is designed to work with images.

Object references:

- Image

Available methods:

- image.create - create new images
- image.delete - delete images
- image.get - retrieve images
- image.update - update images

> Image object

The following objects are directly related to the `image` API.

Image

The image object has the following properties.

Property	Type	Description
imageid	string	ID of the image. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the image. Property behavior: - *required* for create operations
imagetype	integer	Type of image. Possible values: 1 - *(default)* icon; 2 - background image. Property behavior: - *constant* - *required* for create operations
image	string	Base64 encoded image. The maximum size of the encoded image is 1 MB. Maximum size can be adjusted by changing ZBX_MAX_IMAGE_SIZE constant value. Supported image formats: PNG, JPEG, GIF. Property behavior: - *required* for create operations

image.create

Description

`object image.create(object/array images)`

This method allows to create new images.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Images to create.

The method accepts images with the standard image properties.

Return values

`(object)` Returns an object containing the IDs of the created images under the `imageids` property. The order of the returned IDs matches the order of the passed images.

Examples

Create an image

Create a cloud icon.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "image.create",
    "params": {
        "imagetype": 1,
        "name": "Cloud_(24)",
        "image": "iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAAACzbK7QAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAACmAAAApgB
    },
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": [
            "188"
        ]
    },
    "id": 1
}
```

Source

CImage::create() in *ui/include/classes/api/services/CImage.php*.

image.delete

Description

`object image.delete(array imageIds)`

This method allows to delete images.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the images to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted images under the `imageids` property.

Examples

Delete multiple images

Delete two images.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "image.delete",
    "params": [
        "188",
        "192"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "imageids": [
            "188",
            "192"
        ]
    },
    "id": 1
}
```

Source

CImage::delete() in *ui/include/classes/api/services/CImage.php*.

image.get

Description

`integer/array image.get(object parameters)`

The method allows to retrieve images according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
imageids	string/array	Return only images with the given IDs.
sysmapids	string/array	Return images that are used on the given maps.
select_image	flag	Return an `image` property with the Base64 encoded image.
sortfield	string/array	Sort the result by the given properties. Possible values: `imageid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve an image

Retrieve all data for image with ID "2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "image.get",
    "params": {
        "output": "extend",
        "select_image": true,
        "imageids": "2"
    },
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "imageid": "2",
            "imagetype": "1",
            "name": "Cloud_(24)",
            "image": "iVBORw0KGgoAAAANSUhEUgAAABgAAAANCAYAAACzbK7QAAAABHNCSVQICAgIfAhkiAAAAAlwSFlzAAACmAAA
        }
    ],
    "id": 1
}
```

Source

CImage::get() in *ui/include/classes/api/services/CImage.php*.

image.update

Description

`object image.update(object/array images)`

This method allows to update existing images.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Image properties to be updated.

The `imageid` property must be defined for each image, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

The method accepts images with the standard image properties.

Return values

`(object)` Returns an object containing the IDs of the updated images under the `imageids` property.

Examples

Rename image

Rename image to ”Cloud icon”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "image.update",
    "params": {
        "imageid": "2",
        "name": "Cloud icon"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
```

```
        "imageids": [
            "2"
        ]
    },
    "id": 1
}
```

Source

CImage::update() in *ui/include/classes/api/services/CImage.php*.

Item

This class is designed to work with items.

Object references:

- Item

Available methods:

- item.create - creating new items
- item.delete - deleting items
- item.get - retrieving items
- item.update - updating items

> Item object

The following objects are directly related to the `item` API.

Item

Note:
Web items cannot be directly created, updated or deleted via the Zabbix API.

The item object has the following properties.

Property	Type	Description
itemid	string	ID of the item. Property behavior: - *read-only* - *required* for update operations
delay	string	Update interval of the item. Accepts seconds or a time unit with suffix (30s,1m,2h,1d). Optionally one or more custom intervals can be specified either as flexible intervals or scheduling. Multiple intervals are separated by a semicolon. User macros may be used. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Flexible intervals may be written as two macros separated by a forward slash (e.g. `{$FLEX_INTERVAL}/{$FLEX_PERIOD}`). Property behavior: - *required* if `type` is set to "Zabbix agent" (0), "Simple check" (3), "Zabbix internal" (5), "External check" (10), "Database monitor" (11), "IPMI agent" (12), "SSH agent" (13), "TELNET agent" (14), "Calculated" (15), "JMX agent" (16), "HTTP agent" (19), "SNMP agent" (20), "Script" (21), or if `type` is set to "Zabbix agent (active)" (7) and `key_` does not contain "mqtt.get"

Property	Type	Description
hostid	string	ID of the host or template that the item belongs to. Property behavior: - *constant* - *required* for create operations
interfaceid	string	ID of the item's host interface. Property behavior: - *required* if item belongs to host and `type` is set to "Zabbix agent", "IPMI agent", "JMX agent", "SNMP trap", or "SNMP agent" - *supported* if item belongs to host and `type` is set to "Simple check", "External check", "SSH agent", "TELNET agent", or "HTTP agent" - *read-only* for discovered objects
key_	string	Item key. Property behavior: - *required* for create operations - *read-only* for inherited objects or discovered objects
name	string	Name of the item. Property behavior: - *required* for create operations - *read-only* for inherited objects or discovered objects
type	integer	Type of the item. Possible values: 0 - Zabbix agent; 2 - Zabbix trapper; 3 - Simple check; 5 - Zabbix internal; 7 - Zabbix agent (active); 9 - Web item; 10 - External check; 11 - Database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 15 - Calculated; 16 - JMX agent; 17 - SNMP trap; 18 - Dependent item; 19 - HTTP agent; 20 - SNMP agent; 21 - Script. Property behavior: - *required* for create operations - *read-only* for inherited objects or discovered objects
url	string	URL string. Supports user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}. Property behavior: - *required* if `type` is set to "HTTP agent" - *read-only* for inherited objects or discovered objects

Property	Type	Description
value_type	integer	Type of information of the item. Possible values: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text. Property behavior: - *required* for create operations - *read-only* for inherited objects or discovered objects
allow_traps	integer	Allow to populate value similarly to the trapper item. 0 - *(default)* Do not allow to accept incoming data; 1 - Allow to accept incoming data. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for discovered objects
authtype	integer	Authentication method. Possible values if `type` is set to ”SSH agent”: 0 - *(default)* password; 1 - public key. Possible values if `type` is set to ”HTTP agent”: 0 - *(default)* none; 1 - basic; 2 - NTLM; 3 - Kerberos. Property behavior: - *supported* if `type` is set to ”SSH agent” or ”HTTP agent” - *read-only* for inherited objects (if `type` is set to ”HTTP agent”) or discovered objects
description	string	Description of the item. Property behavior: - *read-only* for discovered objects
error	string	Error text if there are problems updating the item value. Property behavior: - *read-only*
flags	integer	Origin of the item. Possible values: 0 - a plain item; 4 - a discovered item. Property behavior: - *read-only*
follow_redirects	integer	Follow response redirects while polling data. Possible values: 0 - Do not follow redirects; 1 - *(default)* Follow redirects. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects

Property	Type	Description
headers	object	Object with HTTP(S) request headers, where header name is used as key and header value as value. Example: { "User-Agent": "Zabbix" } Property behavior: - *supported* if `type` is set to "HTTP agent" - *read-only* for inherited objects or discovered objects
history	string	A time unit of how long the history data should be stored. Also accepts user macro. Default: 90d. Property behavior: - *read-only* for discovered objects
http_proxy	string	HTTP(S) proxy connection string. Property behavior: - *supported* if `type` is set to "HTTP agent" - *read-only* for inherited objects or discovered objects
inventory_link	integer	ID of the host inventory field that is populated by the item. Refer to the host inventory page for a list of supported host inventory fields and their IDs. Default: 0. Property behavior: - *supported* if `value_type` is set to "numeric float", "character", "numeric unsigned", or "text" - *read-only* for discovered objects
ipmi_sensor	string	IPMI sensor. Property behavior: - *required* if `type` is set to "IPMI agent" and `key_` is not set to "ipmi.get" - *supported* if `type` is set to "IPMI agent" - *read-only* for inherited objects or discovered objects
jmx_endpoint	string	JMX agent custom connection string. Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi Property behavior: - *supported* if `type` is set to "JMX agent" - *read-only* for discovered objects
lastclock	timestamp	Time when the item value was last updated. By default, only values that fall within the last 24 hours are displayed. You can extend this time period by changing the value of *Max history display period* parameter in the *Administration → General* menu section. Property behavior: - *read-only*

Property	Type	Description
lastns	integer	Nanoseconds when the item value was last updated. By default, only values that fall within the last 24 hours are displayed. You can extend this time period by changing the value of *Max history display period* parameter in the *Administration → General* menu section. Property behavior: - *read-only*
lastvalue	string	Last value of the item. By default, only values that fall within the last 24 hours are displayed. You can extend this time period by changing the value of *Max history display period* parameter in the *Administration → General* menu section. Property behavior: - *read-only*
logtimefmt	string	Format of the time in log entries. Property behavior: - *supported* if `value_type` is set to ”log” - *read-only* for inherited objects or discovered objects
master_itemid	integer	Master item ID. Recursion up to 3 dependent items and maximum count of dependent items equal to 29999 are allowed. Property behavior: - *required* if `type` is set to ”Dependent item” - *read-only* for inherited objects or discovered objects
output_format	integer	Should the response be converted to JSON. 0 - *(default)* Store raw; 1 - Convert to JSON. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects
params	string	Additional parameters depending on the type of the item: - executed script for SSH agent and TELNET agent items; - SQL query for database monitor items; - formula for calculated items; - the script for script item. Property behavior: - *required* if `type` is set to ”Database monitor”, ”SSH agent”, ”TELNET agent”, ”Calculated”, or ”Script” - *read-only* for inherited objects (if `type` is set to ”Script”) or discovered objects
parameters	array	Additional parameters if `type` is set to ”Script”. Array of objects with `name` and `value` properties, where `name` must be unique. Property behavior: - *supported* if `type` is set to ”Script” - *read-only* for inherited objects or discovered objects

Property	Type	Description
password	string	Password for authentication. Property behavior: - *required* if `type` is set to ”JMX agent” and `username` is set - *supported* if `type` is set to ”Simple check”, ”SSH agent”, ”TELNET agent”, ”Database monitor”, or ”HTTP agent” - *read-only* for inherited objects (if `type` is set to ”HTTP agent”) or discovered objects
post_type	integer	Type of post data body stored in `posts` property. Possible values: 0 - *(default)* Raw data; 2 - JSON data; 3 - XML data. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects
posts	string	HTTP(S) request body data. Property behavior: - *required* if `type` is set to ”HTTP agent” and `post_type` is set to ”JSON data” or ”XML data” - *supported* if `type` is set to ”HTTP agent” and `post_type` is set to ”Raw data” - *read-only* for inherited objects or discovered objects
prevvalue	string	Previous value of the item. By default, only values that fall within the last 24 hours are displayed. You can extend this time period by changing the value of *Max history display period* parameter in the *Administration → General* menu section. Property behavior: - *read-only*
privatekey	string	Name of the private key file. Property behavior: - *required* if `type` is set to ”SSH agent” and `authtype` is set to ”public key” - *read-only* for discovered objects
publickey	string	Name of the public key file. Property behavior: - *required* if `type` is set to ”SSH agent” and `authtype` is set to ”public key” - *read-only* for discovered objects
query_fields	array	Query parameters. Array of objects with `key`:`value` pairs, where `value` can be an empty string. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects

Property	Type	Description
request_method	integer	Type of request method. Possible values: 0 - *(default)* GET; 1 - POST; 2 - PUT; 3 - HEAD. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects
retrieve_mode	integer	What part of response should be stored. Possible values if `request_method` is set to ”GET”, ”POST”, or ”PUT”: 0 - *(default)* Body; 1 - Headers; 2 - Both body and headers will be stored. Possible values if `request_method` is set to ”HEAD”: 1 - Headers. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects
snmp_oid	string	SNMP OID. Property behavior: - *required* if `type` is set to ”SNMP agent” - *read-only* for inherited objects or discovered objects
ssl_cert_file	string	Public SSL Key file path. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects
ssl_key_file	string	Private SSL Key file path. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects
ssl_key_password	string	Password for SSL Key file. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects
state	integer	State of the item. Possible values: 0 - *(default)* normal; 1 - not supported. Property behavior: - *read-only*
status	integer	Status of the item. Possible values: 0 - *(default)* enabled item; 1 - disabled item.

Property	Type	Description
status_codes	string	Ranges of required HTTP status codes, separated by commas. Also supports user macros as part of comma separated list. Example: 200,200-{$M},{$M},200-400 Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects
templateid	string	ID of the parent template item. *Hint*: Use the `hostid` property to specify the template that the item belongs to. Property behavior: - *read-only*
timeout	string	Item data polling request timeout. Supports user macros. Default: 3s. Maximum value: 60s. Property behavior: - *supported* if `type` is set to ”HTTP agent” or ”Script” - *read-only* for inherited objects or discovered objects
trapper_hosts	string	Allowed hosts. Property behavior: - *readonly* for discovered objects - *supported* if `type` is set to ”Zabbix trapper”, or if `type` is set to ”HTTP agent” and `allow_traps` is set to ”Allow to accept incoming data”
trends	string	A time unit of how long the trends data should be stored. Also accepts user macro. Default: 365d. Property behavior: - *supported* if `value_type` is set to ”numeric float” or ”numeric unsigned” - *read-only* for discovered objects
units	string	Value units. Property behavior: - *supported* if `value_type` is set to ”numeric float” or ”numeric unsigned” - *read-only* for inherited objects or discovered objects
username	string	Username for authentication. Property behavior: - *required* if `type` is set to ”SSH agent”, ”TELNET agent”, or if `type` is set to ”JMX agent” and `password` is set - *supported* if `type` is set to ”Simple check”, ”Database monitor”, or ”HTTP agent” - *read-only* for inherited objects (if `type` is set to ”HTTP agent”) or discovered objects
uuid	string	Universal unique identifier, used for linking imported item to already existing ones. Auto-generated, if not given. Property behavior: - *supported* if the item belongs to a template

Property	Type	Description
valuemapid	string	ID of the associated value map. Property behavior: - *supported* if `value_type` is set to ”numeric float”, ”character”, or ”numeric unsigned” - *read-only* for inherited objects or discovered objects
verify_host	integer	Whether to validate that the host name for the connection matches the one in the host’s certificate. Possible values: 0 - *(default)* Do not validate; 1 - Validate. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects
verify_peer	integer	Whether to validate that the host’s certificate is authentic. Possible values: 0 - *(default)* Do not validate; 1 - Validate. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects or discovered objects

Item tag

The item tag object has the following properties.

Property	Type	Description
tag	string	Item tag name. Property behavior: - *required*
value	string	Item tag value.

Item preprocessing

The item preprocessing object has the following properties.

Property	Type	Description
type	integer	The preprocessing option type. Possible values: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second; 11 - XML XPath; 12 - JSONPath; 13 - In range; 14 - Matches regular expression; 15 - Does not match regular expression; 16 - Check for error in JSON; 17 - Check for error in XML; 18 - Check for error using regular expression; 19 - Discard unchanged; 20 - Discard unchanged with heartbeat; 21 - JavaScript; 22 - Prometheus pattern; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace; 26 - Check unsupported; 27 - XML to JSON; 28 - SNMP walk value; 29 - SNMP walk to JSON. Property behavior: - *required*
params	string	Additional parameters used by preprocessing option. Multiple parameters are separated by the newline (\n) character. Property behavior: - *required* if `type` is set to ”Custom multiplier” (1), ”Right trim” (2), ”Left trim” (3), ”Trim” (4), ”Regular expression” (5), ”XML XPath” (11), ”JSONPath” (12), ”In range” (13), ”Matches regular expression” (14), ”Does not match regular expression” (15), ”Check for error in JSON” (16), ”Check for error in XML” (17), ”Check for error using regular expression” (18), ”Discard unchanged with heartbeat” (20), ”JavaScript” (21), ”Prometheus pattern” (22), ”Prometheus to JSON” (23), ”CSV to JSON” (24), ”Replace” (25), ”SNMP walk value” (28), or ”SNMP walk to JSON” (29)

Property	Type	Description
error_handler	integer	Action type used in case of preprocessing step failure. Possible values: 0 - Error message is set by Zabbix server; 1 - Discard value; 2 - Set custom value; 3 - Set custom error message. Possible values if `type` is set to ”Check unsupported”: 1 - Discard value; 2 - Set custom value; 3 - Set custom error message. Property behavior: - *required* if `type` is set to ”Custom multiplier” (1), ”Regular expression” (5), ”Boolean to decimal” (6), ”Octal to decimal” (7), ”Hexadecimal to decimal” (8), ”Simple change” (9), ”Change per second” (10), ”XML XPath” (11), ”JSONPath” (12), ”In range” (13), ”Matches regular expression” (14), ”Does not match regular expression” (15), ”Check for error in JSON” (16), ”Check for error in XML” (17), ”Check for error using regular expression” (18), ”Prometheus pattern” (22), ”Prometheus to JSON” (23), ”CSV to JSON” (24), ”Check unsupported” (26), ”XML to JSON” (27), ”SNMP walk value” (28), or ”SNMP walk to JSON” (29)
error_handler_params	string	Error handler parameters. Property behavior: - *required* if `error_handler` is set to ”Set custom value” or ”Set custom error message”

The following parameters and error handlers are supported for each preprocessing type.

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
1	Custom multiplier	number[1, 6]			0, 1, 2, 3
2	Right trim	list of characters[2]			
3	Left trim	list of characters[2]			
4	Trim	list of characters[2]			
5	Regular expression	pattern[3]	output[2]		0, 1, 2, 3
6	Boolean to decimal				0, 1, 2, 3
7	Octal to decimal				0, 1, 2, 3
8	Hexadecimal to decimal				0, 1, 2, 3

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
9	Simple change				0, 1, 2, 3
10	Change per second				0, 1, 2, 3
11	XML XPath	path[4]			0, 1, 2, 3
12	JSONPath	path[4]			0, 1, 2, 3
13	In range	min[1, 6]	max[1, 6]		0, 1, 2, 3
14	Matches regular expression	pattern[3]			0, 1, 2, 3
15	Does not match regular expression	pattern[3]			0, 1, 2, 3
16	Check for error in JSON	path[4]			0, 1, 2, 3
17	Check for error in XML	path[4]			0, 1, 2, 3
18	Check for error using regular expression	pattern[3]	output[2]		0, 1, 2, 3
19	Discard unchanged				
20	Discard unchanged with heartbeat	seconds[5, 6]			
21	JavaScript	script[2]			
22	Prometheus pattern	pattern[6, 7]	`value`, `label`, `function`	output[8, 9]	0, 1, 2, 3
23	Prometheus to JSON	pattern[6, 7]			0, 1, 2, 3

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
24	CSV to JSON	character[2]	character[2]	0,1	0, 1, 2, 3
25	Replace	search string[2]	replacement[2]		
26	Check unsupported				1, 2, 3
27	XML to JSON				0, 1, 2, 3
28	SNMP walk value	OID[2]	Format: 0 - Unchanged 1 - UTF-8 from Hex-STRING 2 - MAC from Hex-STRING 3 - Integer from BITS		0, 1, 2, 3
29	SNMP walk to JSON[10]	Field name[2]	OID prefix[2]	Format: 0 - Unchanged 1 - UTF-8 from Hex-STRING 2 - MAC from Hex-STRING 3 - Integer from BITS	0, 1, 2, 3

[1] integer or floating-point number
[2] string
[3] regular expression
[4] JSONPath or XML XPath
[5] positive integer (with support of time suffixes, e.g. 30s, 1m, 2h, 1d)
[6] user macro
[7] Prometheus pattern following the syntax: `<metric name>{<label name>="<label value>", ...} == <value>`. Each Prometheus pattern component (metric, label name, label value and metric value) can be user macro.
[8] Prometheus output following the syntax: `<label name>` (can be a user macro) if `label` is selected as the second parameter.
[9] One of the aggregation functions: `sum`, `min`, `max`, `avg`, `count` if `function` is selected as the second parameter.
[10] Supports multiple "Field name,OID prefix,Format records" records delimited by a new line character.

item.create

Description

`object item.create(object/array items)`

This method allows to create new items.

> **Note:**
> Web items cannot be created via the Zabbix API.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Items to create.

Additionally to the standard item properties, the method accepts the following parameters.

Parameter	Type	Description
preprocessing	array	Item preprocessing options.
tags	array	Item tags.

Return values

(object) Returns an object containing the IDs of the created items under the itemids property. The order of the returned IDs matches the order of the passed items.

Examples

Creating an item

Create a numeric Zabbix agent item with 2 item tags to monitor free disk space on host with ID "30074".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Free disk space on /home/joe/",
        "key_": "vfs.fs.size[/home/joe/,free]",
        "hostid": "30074",
        "type": 0,
        "value_type": 3,
        "interfaceid": "30084",
        "tags": [
            {
                "tag": "Disk usage"
            },
            {
                "tag": "Equipment",
                "value": "Workstation"
            }
        ],
        "delay": "30s"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24758"
        ]
    },
    "id": 1
}
```

Creating a host inventory item

Create a Zabbix agent item to populate the host's "OS" inventory field.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "uname",
        "key_": "system.uname",
        "hostid": "30021",
        "type": 0,
```

```
        "interfaceid": "30007",
        "value_type": 1,
        "delay": "10s",
        "inventory_link": 5
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "24759"
        ]
    },
    "id": 1
}
```

Creating an item with preprocessing

Create an item using custom multiplier.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Device uptime",
        "key_": "sysUpTime",
        "hostid": "11312",
        "type": 4,
        "snmp_oid": "SNMPv2-MIB::sysUpTime.0",
        "value_type": 1,
        "delay": "60s",
        "units": "uptime",
        "interfaceid": "1156",
        "preprocessing": [
            {
                "type": 1,
                "params": "0.01",
                "error_handler": 1,
                "error_handler_params": ""
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44210"
        ]
    },
    "id": 1
}
```

Creating dependent item

Create a dependent item for the master item with ID 24759. Only dependencies on the same host are allowed, therefore master and the dependent item should have the same hostid.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
      "hostid": "30074",
      "name": "Dependent test item",
      "key_": "dependent.item",
      "type": 18,
      "master_itemid": "24759",
      "value_type": 2
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

Create HTTP agent item

Create POST request method item with JSON response preprocessing.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "url":"http://127.0.0.1/http.php",
        "query_fields": [
            {
                "mode":"json"
            },
            {
                "min": "10"
            },
            {
                "max": "100"
            }
        ],
        "interfaceid": "1",
        "type": 19,
        "hostid": "10254",
        "delay": "5s",
        "key_": "json",
        "name": "HTTP agent example JSON",
        "value_type": 0,
        "output_format": 1,
        "preprocessing": [
            {
                "type": 12,
                "params": "$.random",
                "error_handler": 0,
                "error_handler_params": ""
            }
        ]
```

```
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

Create script item

Create a simple data collection using a script item.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Script example",
        "key_": "custom.script.item",
        "hostid": "12345",
        "type": 21,
        "value_type": 4,
        "params": "var request = new HttpRequest();\nreturn request.post(\"https://postman-echo.com/post\"
        "parameters": [
            {
                "name": "host",
                "value": "{HOST.CONN}"
            }
        ],
        "timeout": "6s",
        "delay": "30s"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

Source

CItem::create() in *ui/include/classes/api/services/CItem.php*.

item.delete

Description

`object item.delete(array itemIds)`

This method allows to delete items.

Note:
Web items cannot be deleted via the Zabbix API.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the items to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted items under the `itemids` property.

Examples

Deleting multiple items

Delete two items.
Dependent items and item prototypes are removed automatically if master item is deleted.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.delete",
    "params": [
        "22982",
        "22986"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22982",
            "22986"
        ]
    },
    "id": 1
}
```

Source

CItem::delete() in *ui/include/classes/api/services/CItem.php*.

item.get

Description

`integer/array item.get(object parameters)`

The method allows to retrieve items according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
itemids	string/array	Return only items with the given IDs.
groupids	string/array	Return only items that belong to the hosts from the given groups.
templateids	string/array	Return only items that belong to the given templates.
hostids	string/array	Return only items that belong to the given hosts.
proxyids	string/array	Return only items that are monitored by the given proxies.
interfaceids	string/array	Return only items that use the given host interfaces.
graphids	string/array	Return only items that are used in the given graphs.
triggerids	string/array	Return only items that are used in the given triggers.
webitems	flag	Include web items in the result.
inherited	boolean	If set to `true` return only items inherited from a template.
templated	boolean	If set to `true` return only items that belong to templates.
monitored	boolean	If set to `true` return only enabled items that belong to monitored hosts.
group	string	Return only items that belong to a group with the given name.
host	string	Return only items that belong to a host with the given name.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	array of objects	Return only items with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all items. Possible operator types: 0 - (default) Like; 1 - Equal; 2 - Not like; 3 - Not equal 4 - Exists; 5 - Not exists.
with_triggers	boolean	If set to `true` return only items that are used in triggers.
selectHosts	query	Return a `hosts` property with an array of hosts that the item belongs to.
selectInterfaces	query	Return an `interfaces` property with an array of host interfaces used by the item.
selectTriggers	query	Return a `triggers` property with the triggers that the item is used in. Supports `count`.
selectGraphs	query	Return a `graphs` property with the graphs that contain the item. Supports `count`.
selectDiscoveryRule	query	Return a `discoveryRule` property with the LLD rule that created the item.
selectItemDiscovery	query	Return an `itemDiscovery` property with the item discovery object. The item discovery object links the item to an item prototype from which it was created. It has the following properties: `itemdiscoveryid` - `(string)` ID of the item discovery; `itemid` - `(string)` ID of the discovered item; `parent_itemid` - `(string)` ID of the item prototype from which the item has been created; `key_` - `(string)` key of the item prototype; `lastcheck` - `(timestamp)` time when the item was last discovered; `ts_delete` - `(timestamp)` time when an item that is no longer discovered will be deleted.

Parameter	Type	Description
selectPreprocessing	query	Return a `preprocessing` property with item preprocessing options. It has the following properties: `type` - `(string)` The preprocessing option type: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second; 11 - XML XPath; 12 - JSONPath; 13 - In range; 14 - Matches regular expression; 15 - Does not match regular expression; 16 - Check for error in JSON; 17 - Check for error in XML; 18 - Check for error using regular expression; 19 - Discard unchanged; 20 - Discard unchanged with heartbeat; 21 - JavaScript; 22 - Prometheus pattern; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace; 26 - Check for not supported value; 27 - XML to JSON; 28 - SNMP walk value; 29 - SNMP walk to JSON. `params` - `(string)` Additional parameters used by preprocessing option. Multiple parameters are separated by the newline (\n) character. `error_handler` - `(string)` Action type used in case of preprocessing step failure: 0 - Error message is set by Zabbix server; 1 - Discard value; 2 - Set custom value; 3 - Set custom error message. `error_handler_params` - `(string)` Error handler parameters.
selectTags	query	Return the item tags in `tags` property.
selectValueMap	query	Return a `valuemap` property with item value map.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: `host` - technical name of the host that the item belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectGraphs` - results will be sorted by `name`; `selectTriggers` - results will be sorted by `description`.
sortfield	string/array	Sort the result by the given properties. Possible values: `itemid`, `name`, `key_`, `delay`, `history`, `trends`, `type`, `status`.

Parameter	Type	Description
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Finding items by key

Retrieve all items used in triggers for specific host ID that have word ”system.cpu” in the item key and sort results by name.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "hostids": "10084",
        "with_triggers": true,
        "search": {
            "key_": "system.cpu"
        },
        "sortfield": "name"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "42269",
            "type": "18",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "CPU utilization",
            "key_": "system.cpu.util",
            "delay": "0",
            "history": "7d",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "%",
            "logtimefmt": "",
            "templateid": "42267",
            "valuemapid": "0",
```

```
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "CPU utilization in %.",
            "inventory_link": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "42264",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "state": "0",
            "error": "",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        },
        {
            "itemid": "42259",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Load average (15m avg)",
            "key_": "system.cpu.load[all,avg15]",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "",
            "logtimefmt": "",
            "templateid": "42219",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
```

```
        "password": "",
        "publickey": "",
        "privatekey": "",
        "flags": "0",
        "interfaceid": "1",
        "description": "",
        "inventory_link": "0",
        "evaltype": "0",
        "jmx_endpoint": "",
        "master_itemid": "0",
        "timeout": "3s",
        "url": "",
        "query_fields": [],
        "posts": "",
        "status_codes": "200",
        "follow_redirects": "1",
        "post_type": "0",
        "http_proxy": "",
        "headers": [],
        "retrieve_mode": "0",
        "request_method": "0",
        "output_format": "0",
        "ssl_cert_file": "",
        "ssl_key_file": "",
        "ssl_key_password": "",
        "verify_peer": "0",
        "verify_host": "0",
        "allow_traps": "0",
        "uuid": "",
        "state": "0",
        "error": "",
        "parameters": [],
        "lastclock": "0",
        "lastns": "0",
        "lastvalue": "0",
        "prevvalue": "0"
    },
    {
        "itemid": "42249",
        "type": "0",
        "snmp_oid": "",
        "hostid": "10084",
        "name": "Load average (1m avg)",
        "key_": "system.cpu.load[all,avg1]",
        "delay": "1m",
        "history": "7d",
        "trends": "365d",
        "status": "0",
        "value_type": "0",
        "trapper_hosts": "",
        "units": "",
        "logtimefmt": "",
        "templateid": "42209",
        "valuemapid": "0",
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "flags": "0",
```

```
            "interfaceid": "1",
            "description": "",
            "inventory_link": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "state": "0",
            "error": "",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        },
        {
            "itemid": "42257",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Load average (5m avg)",
            "key_": "system.cpu.load[all,avg5]",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "",
            "logtimefmt": "",
            "templateid": "42217",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "1",
            "description": "",
            "inventory_link": "0",
            "evaltype": "0",
```

```
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "state": "0",
            "error": "",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        },
        {
            "itemid": "42260",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10084",
            "name": "Number of CPUs",
            "key_": "system.cpu.num",
            "delay": "1m",
            "history": "7d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "logtimefmt": "",
            "templateid": "42220",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "1",
            "description": "",
            "inventory_link": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
```

```
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "state": "0",
            "error": "",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
    "id": 1
}
```


Finding dependent items by key

Retrieve all dependent items from host with ID ”10116” that have the word ”apache” in the key.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "hostids": "10116",
        "search": {
            "key_": "apache"
        },
        "filter": {
            "type": 18
        }
    },
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "25550",
            "type": "18",
            "snmp_oid": "",
            "hostid": "10116",
            "name": "Days",
            "key_": "apache.status.uptime.days",
            "delay": "0",
            "history": "90d",
```

```
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "25545",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "state": "0",
            "error": "",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        },
        {
            "itemid": "25555",
            "type": "18",
            "snmp_oid": "",
            "hostid": "10116",
            "name": "Hours",
            "key_": "apache.status.uptime.hours",
            "delay": "0",
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
```

```
            "units": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "25545",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "state": "0",
            "error": "",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "0",
            "prevvalue": "0"
        }
    ],
    "id": 1
}
```


Find HTTP agent item

Find HTTP agent item with post body type XML for specific host ID.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "hostids": "10255",
        "filter": {
            "type": 19,
            "post_type": 3
        }
```

```
    },
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28252",
            "type": "19",
            "snmp_oid": "",
            "hostid": "10255",
            "name": "template item",
            "key_": "ti",
            "delay": "30s",
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "0",
            "interfaceid": "0",
            "description": "",
            "inventory_link": "0",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "localhost",
            "query_fields": [
                {
                    "mode": "xml"
                }
            ],
            "posts": "<body>\r\n<![CDATA[{$MACRO}<foo></bar>]]>\r\n</body>",
            "status_codes": "200",
            "follow_redirects": "0",
            "post_type": "3",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "1",
            "request_method": "3",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
```

```
            "state": "0",
            "error": "",
            "parameters": [],
            "lastclock": "0",
            "lastns": "0",
            "lastvalue": "",
            "prevvalue": ""
        }
    ],
    "id": 1
}
```

Retrieving items with preprocessing rules

Retrieve all items and their preprocessing rules for specific host ID.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": ["itemid", "name", "key_"],
        "selectPreprocessing": "extend",
        "hostids": "10254"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemid": "23865",
        "name": "HTTP agent example JSON",
        "key_": "json",
        "preprocessing": [
            {
                "type": "12",
                "params": "$.random",
                "error_handler": "1",
                "error_handler_params": ""
            }
        ]
    },
    "id": 1
}
```

See also

- Discovery rule
- Graph
- Host
- Host interface
- Trigger

Source

CItem::get() in *ui/include/classes/api/services/CItem.php*.

item.update

Description

`object item.update(object/array items)`

This method allows to update existing items.

> **Note:**
> Web items cannot be updated via the Zabbix API.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Item properties to be updated.

The `itemid` property must be defined for each item, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard item properties, the method accepts the following parameters.

Parameter	Type	Description
preprocessing	array	Item preprocessing options to replace the current preprocessing options. Parameter behavior: - *read-only* for inherited objects or discovered objects
tags	array	Item tags. Parameter behavior: - *read-only* for discovered objects

Return values

`(object)` Returns an object containing the IDs of the updated items under the `itemids` property.

Examples

Enabling an item

Enable an item, that is, set its status to "0".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "10092",
        "status": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "10092"
        ]
    },
    "id": 1
}
```

Update dependent item

Update Dependent item name and Master item ID. Only dependencies on same host are allowed, therefore Master and Dependent item should have same hostid.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "name": "Dependent item updated name",
        "master_itemid": "25562",
        "itemid": "189019"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "189019"
        ]
    },
    "id": 1
}
```

Update HTTP agent item

Enable item value trapping.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "23856",
        "allow_traps": 1
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23856"
        ]
    },
    "id": 1
}
```

Updating an item with preprocessing

Update an item with item preprocessing rule ”In range”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "23856",
        "preprocessing": [
            {
                "type": 13,
                "params": "\n100",
                "error_handler": 1,
```

```
                "error_handler_params": ""
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23856"
        ]
    },
    "id": 1
}
```

Updating a script item

Update a script item with a different script and remove unnecessary parameters that were used by previous script.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "23865",
        "parameters": [],
        "script": "Zabbix.log(3, 'Log test');\nreturn 1;"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

Source

CItem::update() in *ui/include/classes/api/services/CItem.php*.

Item prototype

This class is designed to work with item prototypes.

Object references:

- Item prototype

Available methods:

- itemprototype.create - creating new item prototypes
- itemprototype.delete - deleting item prototypes
- itemprototype.get - retrieving item prototypes
- itemprototype.update - updating item prototypes

> Item prototype object

The following objects are directly related to the `itemprototype` API.

Item prototype

The item prototype object has the following properties.

Property	Type	Description
itemid	string	ID of the item prototype. Property behavior: - *read-only* - *required* for update operations
delay	string	Update interval of the item prototype. Accepts seconds or a time unit with suffix (30s,1m,2h,1d). Optionally one or more custom intervals can be specified either as flexible intervals or scheduling. Multiple intervals are separated by a semicolon. User macros and LLD macros may be used. A single macro has to fill the whole field. Multiple macros in a field or macros mixed with text are not supported. Flexible intervals may be written as two macros separated by a forward slash (e.g. `{$FLEX_INTERVAL}/{$FLEX_PERIOD}`). Property behavior: - *required* if `type` is set to ”Zabbix agent” (0), ”Simple check” (3), ”Zabbix internal” (5), ”External check” (10), ”Database monitor” (11), ”IPMI agent” (12), ”SSH agent” (13), ”TELNET agent” (14), ”Calculated” (15), ”JMX agent” (16), ”HTTP agent” (19), ”SNMP agent” (20), ”Script” (21), or if `type` is set to ”Zabbix agent (active)” (7) and `key_` does not contain ”mqtt.get”
hostid	string	ID of the host that the item prototype belongs to. Property behavior: - *constant* - *required* for create operations
interfaceid	string	ID of the item prototype’s host interface. Property behavior: - *required* if item prototype belongs to host and `type` is set to ”Zabbix agent”, ”IPMI agent”, ”JMX agent”, ”SNMP trap”, or ”SNMP agent” - *supported* if item prototype belongs to host and `type` is set to ”Simple check”, ”External check”, ”SSH agent”, ”TELNET agent”, or ”HTTP agent”
key_	string	Item prototype key. Property behavior: - *required* for create operations - *read-only* for inherited objects
name	string	Name of the item prototype. Property behavior: - *required* for create operations - *read-only* for inherited objects

Property	Type	Description
type	integer	Type of the item prototype. Possible values: 0 - Zabbix agent; 2 - Zabbix trapper; 3 - Simple check; 5 - Zabbix internal; 7 - Zabbix agent (active); 10 - External check; 11 - Database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 15 - Calculated; 16 - JMX agent; 17 - SNMP trap; 18 - Dependent item; 19 - HTTP agent; 20 - SNMP agent; 21 - Script. Property behavior: - *required* for create operations - *read-only* for inherited objects
url	string	URL string. Supports LLD macros, user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}. Property behavior: - *required* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
value_type	integer	Type of information of the item prototype. Possible values: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text. Property behavior: - *required* for create operations - *read-only* for inherited objects
allow_traps	integer	Allow to populate value similarly to the trapper item. 0 - *(default)* Do not allow to accept incoming data; 1 - Allow to accept incoming data. Property behavior: - *supported* if `type` is set to ”HTTP agent”

Property	Type	Description
authtype	integer	Authentication method. Possible values if `type` is set to ”SSH agent”: 0 - *(default)* password; 1 - public key. Possible values if `type` is set to ”HTTP agent”: 0 - *(default)* none; 1 - basic; 2 - NTLM; 3 - Kerberos. Property behavior: - *supported* if `type` is set to ”SSH agent” or ”HTTP agent” - *read-only* for inherited objects (if `type` is set to ”HTTP agent”)
description	string	Description of the item prototype.
follow_redirects	integer	Follow response redirects while polling data. Possible values: 0 - Do not follow redirects; 1 - *(default)* Follow redirects. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
headers	object	Object with HTTP(S) request headers, where header name is used as key and header value as value. Example: { ”User-Agent”: ”Zabbix” } Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
history	string	A time unit of how long the history data should be stored. Also accepts user macro and LLD macro. Default: 90d.
http_proxy	string	HTTP(S) proxy connection string. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
ipmi_sensor	string	IPMI sensor. Property behavior: - *required* if `type` is set to ”IPMI agent” and `key_` is not set to ”ipmi.get” - *supported* if `type` is set to ”IPMI agent” - *read-only* for inherited objects
jmx_endpoint	string	JMX agent custom connection string. Default: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi Property behavior: - *supported* if `type` is set to ”JMX agent”
logtimefmt	string	Format of the time in log entries. Property behavior: - *supported* if `value_type` is set to ”log” - *read-only* for inherited objects

Property	Type	Description
master_itemid	integer	Master item ID. Recursion up to 3 dependent items and item prototypes and maximum count of dependent items and item prototypes equal to 29999 are allowed. Property behavior: - *required* if `type` is set to ”Dependent item” - *read-only* for inherited objects
output_format	integer	Should the response be converted to JSON. Possible values: 0 - *(default)* Store raw; 1 - Convert to JSON. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
params	string	Additional parameters depending on the type of the item prototype: - executed script for SSH agent and TELNET agent item prototypes; - SQL query for database monitor item prototypes; - formula for calculated item prototypes. Property behavior: - *required* if `type` is set to ”Database monitor”, ”SSH agent”, ”TELNET agent”, ”Calculated”, or ”Script” - *read-only* for inherited objects (if `type` is set to ”Script”)
parameters	array	Additional parameters if `type` is set to ”Script”. Array of objects with `name` and `value` properties, where `name` must be unique. Property behavior: - *supported* if `type` is set to ”Script” - *read-only* for inherited objects
password	string	Password for authentication. Property behavior: - *required* if `type` is set to ”JMX agent” and `username` is set - *supported* if `type` is set to ”Simple check”, ”SSH agent”, ”TELNET agent”, ”Database monitor”, or ”HTTP agent” - *read-only* for inherited objects (if `type` is set to ”HTTP agent”)
post_type	integer	Type of post data body stored in `posts` property. Possible values: 0 - *(default)* Raw data. 2 - JSON data. 3 - XML data. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
posts	string	HTTP(S) request body data. Property behavior: - *required* if `type` is set to ”HTTP agent” and `post_type` is set to ”JSON data” or ”XML data” - *supported* if `type` is set to ”HTTP agent” and `post_type` is set to ”Raw data” - *read-only* for inherited objects
privatekey	string	Name of the private key file. Property behavior: - *required* if `type` is set to ”SSH agent” and `authtype` is set to ”public key”

Property	Type	Description
publickey	string	Name of the public key file. Property behavior: - *required* if `type` is set to ”SSH agent” and `authtype` is set to ”public key”
query_fields	array	Query parameters. Array of objects with `key:value` pairs, where `value` can be empty string. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
request_method	integer	Type of request method. Possible values: 0 - *(default)* GET; 1 - POST; 2 - PUT; 3 - HEAD. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
retrieve_mode	integer	What part of response should be stored. Possible values if `request_method` is set to ”GET”, ”POST”, or ”PUT”: 0 - *(default)* Body; 1 - Headers; 2 - Both body and headers will be stored. Possible values if `request_method` is set to ”HEAD”: 1 - Headers. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
snmp_oid	string	SNMP OID. Property behavior: - *required* if `type` is set to ”SNMP agent” - *read-only* for inherited objects
ssl_cert_file	string	Public SSL Key file path. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
ssl_key_file	string	Private SSL Key file path. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
ssl_key_password	string	Password for SSL Key file. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
status	integer	Status of the item prototype. Possible values: 0 - *(default)* enabled item prototype; 1 - disabled item prototype; 3 - unsupported item prototype.

Property	Type	Description
status_codes	string	Ranges of required HTTP status codes, separated by commas. Also supports user macros or LLD macros as part of comma separated list. Example: 200,200-{$M},{$M},200-400 Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
templateid	string	ID of the parent template item prototype. Property behavior: - *read-only*
timeout	string	Item data polling request timeout. Supports user macros and LLD macros. Default: 3s. Maximum value: 60s. Property behavior: - *supported* if `type` is set to ”HTTP agent” or ”Script” - *read-only* for inherited objects
trapper_hosts	string	Allowed hosts. Property behavior: - *supported* if `type` is set to ”Zabbix trapper”, or if `type` is set to ”HTTP agent” and `allow_traps` is set to ”Allow to accept incoming data”
trends	string	A time unit of how long the trends data should be stored. Also accepts user macro and LLD macro. Default: 365d. Property behavior: - *supported* if `value_type` is set to ”numeric float” or ”numeric unsigned”
units	string	Value units. Property behavior: - *supported* if `value_type` is set to ”numeric float” or ”numeric unsigned” - *read-only* for inherited objects
username	string	Username for authentication. Property behavior: - *required* if `type` is set to ”SSH agent” or ”TELNET agent”, or if `type` is set to ”JMX agent” and `password` is set - *supported* if `type` is set to ”Simple check”, ”Database monitor”, or ”HTTP agent” - *read-only* for inherited objects (if `type` is set to ”HTTP agent”)
uuid	string	Universal unique identifier, used for linking imported item prototypes to already existing ones. Auto-generated, if not given. Property behavior: - *supported* if the item prototype belongs to a template
valuemapid	string	ID of the associated value map. Property behavior: - *supported* if `value_type` is set to ”numeric float”, ”character”, or ”numeric unsigned” - *read-only* for inherited objects

Property	Type	Description
verify_host	integer	Whether to validate that the host name for the connection matches the one in the host's certificate. Possible values: 0 - *(default)* Do not validate; 1 - Validate. Property behavior: - *supported* if `type` is set to "HTTP agent" - *read-only* for inherited objects
verify_peer	integer	Whether to validate that the host's certificate is authentic. Possible values: 0 - *(default)* Do not validate; 1 - Validate. Property behavior: - *supported* if `type` is set to "HTTP agent" - *read-only* for inherited objects
discover	integer	Item prototype discovery status. Possible values: 0 - *(default)* new items will be discovered; 1 - new items will not be discovered and existing items will be marked as lost.

Item prototype tag

The item prototype tag object has the following properties.

Property	Type	Description
tag	string	Item prototype tag name. Property behavior: - *required*
value	string	Item prototype tag value.

Item prototype preprocessing

The item prototype preprocessing object has the following properties.

Property	Type	Description
type	integer	The preprocessing option type. Possible values: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second; 11 - XML XPath; 12 - JSONPath; 13 - In range; 14 - Matches regular expression; 15 - Does not match regular expression; 16 - Check for error in JSON; 17 - Check for error in XML; 18 - Check for error using regular expression; 19 - Discard unchanged; 20 - Discard unchanged with heartbeat; 21 - JavaScript; 22 - Prometheus pattern; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace; 26 - Check unsupported; 27 - XML to JSON; 28 - SNMP walk value; 29 - SNMP walk to JSON. Property behavior: - *required*
params	string	Additional parameters used by preprocessing option. Multiple parameters are separated by the newline (\n) character. Property behavior: - *required* if `type` is set to ”Custom multiplier” (1), ”Right trim” (2), ”Left trim” (3), ”Trim” (4), ”Regular expression” (5), ”XML XPath” (11), ”JSONPath” (12), ”In range” (13), ”Matches regular expression” (14), ”Does not match regular expression” (15), ”Check for error in JSON” (16), ”Check for error in XML” (17), ”Check for error using regular expression” (18), ”Discard unchanged with heartbeat” (20), ”JavaScript” (21), ”Prometheus pattern” (22), ”Prometheus to JSON” (23), ”CSV to JSON” (24), ”Replace” (25), ”SNMP walk value” (28), or ”SNMP walk to JSON” (29)

Property	Type	Description
error_handler	integer	Action type used in case of preprocessing step failure. Possible values: 0 - Error message is set by Zabbix server; 1 - Discard value; 2 - Set custom value; 3 - Set custom error message. Possible values if `type` is set to ”Check unsupported”: 1 - Discard value; 2 - Set custom value; 3 - Set custom error message. Property behavior: - *required* if `type` is set to ”Custom multiplier” (1), ”Regular expression” (5), ”Boolean to decimal” (6), ”Octal to decimal” (7), ”Hexadecimal to decimal” (8), ”Simple change” (9), ”Change per second” (10), ”XML XPath” (11), ”JSONPath” (12), ”In range” (13), ”Matches regular expression” (14), ”Does not match regular expression” (15), ”Check for error in JSON” (16), ”Check for error in XML” (17), ”Check for error using regular expression” (18), ”Prometheus pattern” (22), ”Prometheus to JSON” (23), ”CSV to JSON” (24), ”Check unsupported” (26), ”XML to JSON” (27), ”SNMP walk value” (28), or ”SNMP walk to JSON” (29)
error_handler_params	string	Error handler parameters. Property behavior: - *required* if `error_handler` is set to ”Set custom value” or ”Set custom error message”

The following parameters and error handlers are supported for each preprocessing type.

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
1	Custom multiplier	number[1, 6]			0, 1, 2, 3
2	Right trim	list of characters[2]			
3	Left trim	list of characters[2]			
4	Trim	list of characters[2]			
5	Regular expression	pattern[3]	output[2]		0, 1, 2, 3
6	Boolean to decimal				0, 1, 2, 3
7	Octal to decimal				0, 1, 2, 3
8	Hexadecimal to decimal				0, 1, 2, 3

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
9	Simple change				0, 1, 2, 3
10	Change per second				0, 1, 2, 3
11	XML XPath	path[4]			0, 1, 2, 3
12	JSONPath	path[4]			0, 1, 2, 3
13	In range	min[1, 6]	max[1, 6]		0, 1, 2, 3
14	Matches regular expression	pattern[3]			0, 1, 2, 3
15	Does not match regular expression	pattern[3]			0, 1, 2, 3
16	Check for error in JSON	path[4]			0, 1, 2, 3
17	Check for error in XML	path[4]			0, 1, 2, 3
18	Check for error using regular expression	pattern[3]	output[2]		0, 1, 2, 3
19	Discard unchanged				
20	Discard unchanged with heartbeat	seconds[5, 6]			
21	JavaScript	script[2]			
22	Prometheus pattern	pattern[6, 7]	`value`, `label`, `function`	output[8, 9]	0, 1, 2, 3
23	Prometheus to JSON	pattern[6, 7]			0, 1, 2, 3

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
24	CSV to JSON	character[2]	character[2]	0,1	0, 1, 2, 3
25	Replace	search string[2]	replacement[2]		
26	Check unsupported				1, 2, 3
27	XML to JSON				0, 1, 2, 3
28	SNMP walk value	OID[2]	Format: 0 - Unchanged 1 - UTF-8 from Hex-STRING 2 - MAC from Hex-STRING 3 - Integer from BITS		0, 1, 2, 3
29	SNMP walk to JSON[10]	Field name[2]	OID prefix[2]	Format: 0 - Unchanged 1 - UTF-8 from Hex-STRING 2 - MAC from Hex-STRING 3 - Integer from BITS	0, 1, 2, 3

[1] integer or floating-point number
[2] string
[3] regular expression
[4] JSONPath or XML XPath
[5] positive integer (with support of time suffixes, e.g. 30s, 1m, 2h, 1d)
[6] user macro, LLD macro
[7] Prometheus pattern following the syntax: `<metric name>{<label name>="<label value>", ...} == <value>`. Each Prometheus pattern component (metric, label name, label value and metric value) can be user macro or LLD macro.
[8] Prometheus output following the syntax: `<label name>` (can be a user macro or an LLD macro) if `label` is selected as the second parameter.
[9] One of the aggregation functions: `sum`, `min`, `max`, `avg`, `count` if `function` is selected as the second parameter.
[10] Supports multiple "Field name,OID prefix,Format records" records delimited by a new line character.

itemprototype.create

Description

`object itemprototype.create(object/array itemPrototypes)`

This method allows to create new item prototypes.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Item prototype to create.

Additionally to the standard item prototype properties, the method accepts the following parameters.

Parameter	Type	Description
ruleid	string	ID of the LLD rule that the item belongs to. Parameter behavior: - *required*
preprocessing	array	Item prototype preprocessing options.
tags	array	Item prototype tags.

Return values

`(object)` Returns an object containing the IDs of the created item prototypes under the `itemids` property. The order of the returned IDs matches the order of the passed item prototypes.

Examples

Creating an item prototype

Create an item prototype to monitor free disk space on a discovered file system. Discovered items should be numeric Zabbix agent items updated every 30 seconds.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Free disk space on {#FSNAME}",
        "key_": "vfs.fs.size[{#FSNAME},free]",
        "hostid": "10197",
        "ruleid": "27665",
        "type": 0,
        "value_type": 3,
        "interfaceid": "112",
        "delay": "30s"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27666"
        ]
    },
    "id": 1
}
```

Creating an item prototype with preprocessing

Create an item using change per second and a custom multiplier as a second step.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Incoming network traffic on {#IFNAME}",
        "key_": "net.if.in[{#IFNAME}]",
        "hostid": "10001",
        "ruleid": "27665",
        "type": 0,
        "value_type": 3,
        "delay": "60s",
```

```
        "units": "bps",
        "interfaceid": "1155",
        "preprocessing": [
            {
                "type": 10,
                "params": "",
                "error_handler": 0,
                "error_handler_params": ""
            },
            {
                "type": 1,
                "params": "8",
                "error_handler": 2,
                "error_handler_params": "10"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

Creating dependent item prototype

Create Dependent item prototype for Master item prototype with ID 44211. Only dependencies on same host (template/discovery rule) are allowed, therefore Master and Dependent item should have same hostid and ruleid.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
      "hostid": "10001",
      "ruleid": "27665",
      "name": "Dependent test item prototype",
      "key_": "dependent.prototype",
      "type": 18,
      "master_itemid": "44211",
      "value_type": 3
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44212"
        ]
    },
    "id": 1
}
```

Create HTTP agent item prototype

Create item prototype with URL using user macro, query fields and custom headers.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "type": "19",
        "hostid": "10254",
        "ruleid": "28256",
        "interfaceid": "2",
        "name": "api item prototype example",
        "key_": "api_http_item",
        "value_type": 3,
        "url": "{$URL_PROTOTYPE}",
        "query_fields": [
            {
                "min": "10"
            },
            {
                "max": "100"
            }
        ],
        "headers": {
            "X-Source": "api"
        },
        "delay": "35"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28305"
        ]
    },
    "id": 1
}
```

Create script item prototype

Create a simple data collection using a script item prototype.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.create",
    "params": {
        "name": "Script example",
        "key_": "custom.script.itemprototype",
        "hostid": "12345",
        "type": 21,
        "value_type": 4,
        "params": "var request = new HttpRequest();\nreturn request.post(\"https://postman-echo.com/post\"
        "parameters": [
            {
                "name": "host",
                "value": "{HOST.CONN}"
            }
```

```
        ],
        "timeout": "6s",
        "delay": "30s"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

Source

CItemPrototype::create() in *ui/include/classes/api/services/CItemPrototype.php*.

itemprototype.delete

Description

`object itemprototype.delete(array itemPrototypeIds)`

This method allows to delete item prototypes.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the item prototypes to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted item prototypes under the `prototypeids` property.

Examples

Deleting multiple item prototypes

Delete two item prototypes.
Dependent item prototypes are removed automatically if master item or item prototype is deleted.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.delete",
    "params": [
        "27352",
        "27356"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "prototypeids": [
            "27352",
```

```
            "27356"
        ]
    },
    "id": 1
}
```

Source

CItemPrototype::delete() in *ui/include/classes/api/services/CItemPrototype.php*.

itemprototype.get

Description

`integer/array itemprototype.get(object parameters)`

The method allows to retrieve item prototypes according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
discoveryids	string/array	Return only item prototypes that belong to the given LLD rules.
graphids	string/array	Return only item prototypes that are used in the given graph prototypes.
hostids	string/array	Return only item prototypes that belong to the given hosts.
inherited	boolean	If set to `true` return only item prototypes inherited from a template.
itemids	string/array	Return only item prototypes with the given IDs.
monitored	boolean	If set to `true` return only enabled item prototypes that belong to monitored hosts.
templated	boolean	If set to `true` return only item prototypes that belong to templates.
templateids	string/array	Return only item prototypes that belong to the given templates.
triggerids	string/array	Return only item prototypes that are used in the given trigger prototypes.
selectDiscoveryRule	query	Return a discoveryRule property with the low-level discovery rule that the item prototype belongs to.
selectGraphs	query	Return a graphs property with graph prototypes that the item prototype is used in. Supports `count`.
selectHosts	query	Return a hosts property with an array of hosts that the item prototype belongs to.
selectTags	query	Return the item prototype tags in tags property.
selectTriggers	query	Return a triggers property with trigger prototypes that the item prototype is used in. Supports `count`.

Parameter	Type	Description
selectPreprocessing	query	Return a preprocessing property with item preprocessing options. It has the following properties: `type` - (`string`) The preprocessing option type: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression; 6 - Boolean to decimal; 7 - Octal to decimal; 8 - Hexadecimal to decimal; 9 - Simple change; 10 - Change per second; 11 - XML XPath; 12 - JSONPath; 13 - In range; 14 - Matches regular expression; 15 - Does not match regular expression; 16 - Check for error in JSON; 17 - Check for error in XML; 18 - Check for error using regular expression; 19 - Discard unchanged; 20 - Discard unchanged with heartbeat; 21 - JavaScript; 22 - Prometheus pattern; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace; 26 - Check for not supported value; 27 - XML to JSON; 28 - SNMP walk value; 29 - SNMP walk to JSON. `params` - (`string`) Additional parameters used by preprocessing option. Multiple parameters are separated by the newline (\n) character. `error_handler` - (`string`) Action type used in case of preprocessing step failure: 0 - Error message is set by Zabbix server; 1 - Discard value; 2 - Set custom value; 3 - Set custom error message. `error_handler_params` - (`string`) Error handler parameters.
selectValueMap	query	Return a valuemap property with item prototype value map.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: `host` - technical name of the host that the item prototype belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectGraphs` - results will be sorted by `name`; `selectTriggers` - results will be sorted by `description`.
sortfield	string/array	Sort the result by the given properties. Possible values: `itemid`, `name`, `key_`, `delay`, `type`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.

Parameter	Type	Description
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving item prototypes from an LLD rule

Retrieve all item prototypes for specific LLD rule ID.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23077",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10079",
            "name": "Incoming network traffic on en0",
            "key_": "net.if.in[en0]",
            "delay": "1m",
            "history": "1w",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "bps",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
```

```
            "interfaceid": "0",
            "description": "",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0",
            "uuid": "",
            "parameters": []
        },
        {
            "itemid": "10010",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10001",
            "name": "Processor load (1 min average per core)",
            "key_": "system.cpu.load[percpu,avg1]",
            "delay": "1m",
            "history": "1w",
            "trends": "365d",
            "status": "0",
            "value_type": "0",
            "trapper_hosts": "",
            "units": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "0",
            "description": "The processor load is calculated as system CPU load divided by number of CPU c
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
```

```
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0",
            "uuid": "",
            "parameters": []
        }
    ],
    "id": 1
}
```

Finding dependent item

Find one Dependent item for specific item ID.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "item.get",
    "params": {
        "output": "extend",
        "filter": {
            "type": 18,
            "master_itemid": "25545"
        },
        "limit": "1"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "25547",
            "type": "18",
            "snmp_oid": "",
            "hostid": "10116",
            "name": "Seconds",
            "key_": "apache.status.uptime.seconds",
            "delay": "0",
            "history": "90d",
            "trends": "365d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "logtimefmt": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
```

```
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "0",
            "description": "",
            "evaltype": "0",
            "master_itemid": "25545",
            "jmx_endpoint": "",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0",
            "uuid": "",
            "parameters": []
        }
    ],
    "id": 1
}
```


Find HTTP agent item prototype

Find HTTP agent item prototype with request method HEAD for specific host ID.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.get",
    "params": {
        "hostids": "10254",
        "filter": {
            "type": 19,
            "request_method": 3
        }
    },
    "id": 1
}
```


Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28257",
            "type": "19",
            "snmp_oid": "",
            "hostid": "10254",
            "name": "discovered",
```

```
            "key_": "item[{#INAME}]",
            "delay": "{#IUPDATE}",
            "history": "90d",
            "trends": "30d",
            "status": "0",
            "value_type": "3",
            "trapper_hosts": "",
            "units": "",
            "logtimefmt": "",
            "templateid": "28255",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "flags": "2",
            "interfaceid": "2",
            "description": "",
            "evaltype": "0",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "{#IURL}",
            "query_fields": [],
            "posts": "",
            "status_codes": "",
            "follow_redirects": "0",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "3",
            "output_format": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "discover": "0",
            "uuid": "",
            "parameters": []
        }
    ],
    "id": 1
}
```


See also

- Host
- Graph prototype
- Trigger prototype

Source

CItemPrototype::get() in *ui/include/classes/api/services/CItemPrototype.php*.

itemprototype.update

Description

`object itemprototype.update(object/array itemPrototypes)`

This method allows to update existing item prototypes.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Item prototype properties to be updated.

The `itemid` property must be defined for each item prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard item prototype properties, the method accepts the following parameters.

Parameter	Type	Description
preprocessing	array	Item prototype preprocessing options to replace the current preprocessing options. Parameter behavior: - *read-only* for inherited objects
tags	array	Item prototype tags.

Return values

`(object)` Returns an object containing the IDs of the updated item prototypes under the `itemids` property.

Examples

Changing the interface of an item prototype

Change the host interface that will be used by discovered items.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "27428",
        "interfaceid": "132"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27428"
        ]
    },
    "id": 1
}
```

Update dependent item prototype

Update Dependent item prototype with new Master item prototype ID. Only dependencies on same host (template/discovery rule) are allowed, therefore Master and Dependent item should have same hostid and ruleid.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
```

```
    "params": {
        "master_itemid": "25570",
        "itemid": "189030"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "189030"
        ]
    },
    "id": 1
}
```

Update HTTP agent item prototype

Change query fields and remove all custom headers.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid":"28305",
        "query_fields": [
            {
                "random": "qwertyuiopasdfghjklzxcvbnm"
            }
        ],
        "headers": []
    }
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28305"
        ]
    },
    "id": 1
}
```

Updating item preprocessing options

Update an item prototype with item preprocessing rule “Custom multiplier”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "44211",
        "preprocessing": [
            {
                "type": 1,
                "params": "4",
                "error_handler": 2,
```

```
                "error_handler_params": "5"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

Updating a script item prototype

Update a script item prototype with a different script and remove unnecessary parameters that were used by previous script.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "itemprototype.update",
    "params": {
        "itemid": "23865",
        "parameters": [],
        "script": "Zabbix.log(3, 'Log test');\nreturn 1;"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

Source

CItemPrototype::update() in *ui/include/classes/api/services/CItemPrototype.php*.

LLD rule

This class is designed to work with low level discovery rules.

Object references:

- LLD rule

Available methods:

- discoveryrule.copy - copying LLD rules
- discoveryrule.create - creating new LLD rules
- discoveryrule.delete - deleting LLD rules
- discoveryrule.get - retrieving LLD rules

- discoveryrule.update - updating LLD rules

> LLD rule object

The following objects are directly related to the `discoveryrule` API.

LLD rule

The low-level discovery rule object has the following properties.

Property	Type	Description
itemid	string	ID of the LLD rule. Property behavior: - *read-only* - *required* for update operations
delay	string	Update interval of the LLD rule. Accepts seconds or time unit with suffix and with or without one or more custom intervals that consist of either flexible intervals and scheduling intervals as serialized strings. Also accepts user macros. Flexible intervals could be written as two macros separated by a forward slash. Intervals are separated by a semicolon. Property behavior: - *required* if `type` is set to ”Zabbix agent” (0), ”Simple check” (3), ”Zabbix internal” (5), ”External check” (10), ”Database monitor” (11), ”IPMI agent” (12), ”SSH agent” (13), ”TELNET agent” (14), ”JMX agent” (16), ”HTTP agent” (19), ”SNMP agent” (20), ”Script” (21), or if `type` is set to ”Zabbix agent (active)” (7) and `key_` does not contain ”mqtt.get”
hostid	string	ID of the host that the LLD rule belongs to. Property behavior: - *constant* - *required* for create operations
interfaceid	string	ID of the LLD rule’s host interface. Property behavior: - *required* if LLD rule belongs to host and `type` is set to ”Zabbix agent”, ”IPMI agent”, ”JMX agent”, or ”SNMP agent” - *supported* if LLD rule belongs to host and `type` is set to ”Simple check”, ”External check”, ”SSH agent”, ”TELNET agent”, or ”HTTP agent”
key_	string	LLD rule key. Property behavior: - *required* for create operations - *read-only* for inherited objects
name	string	Name of the LLD rule. Property behavior: - *required* for create operations - *read-only* for inherited objects

Property	Type	Description
type	integer	Type of the LLD rule. Possible values: 0 - Zabbix agent; 2 - Zabbix trapper; 3 - Simple check; 5 - Zabbix internal; 7 - Zabbix agent (active); 10 - External check; 11 - Database monitor; 12 - IPMI agent; 13 - SSH agent; 14 - TELNET agent; 16 - JMX agent; 18 - Dependent item; 19 - HTTP agent; 20 - SNMP agent; 21 - Script. Property behavior: - *required* for create operations - *read-only* for inherited objects
url	string	URL string. Supports user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}. Property behavior: - *required* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
allow_traps	integer	Allow to populate value similarly to the trapper item. Possible values: 0 - *(default)* Do not allow to accept incoming data; 1 - Allow to accept incoming data. Property behavior: - *supported* if `type` is set to ”HTTP agent”
authtype	integer	Authentication method. Possible values if `type` is set to ”SSH agent”: 0 - *(default)* password; 1 - public key. Possible values if `type` is set to ”HTTP agent”: 0 - *(default)* none; 1 - basic; 2 - NTLM. Property behavior: - *supported* if `type` is set to ”SSH agent” or ”HTTP agent” - *read-only* for inherited objects (if `type` is set to ”HTTP agent”)
description	string	Description of the LLD rule.
error	string	Error text if there are problems updating the LLD rule value. Property behavior: - *read-only*

Property	Type	Description
follow_redirects	integer	Follow response redirects while polling data. Possible values: 0 - Do not follow redirects; 1 - *(default)* Follow redirects. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
headers	object	Object with HTTP(S) request headers, where header name is used as key and header value as value. Example: { ”User-Agent”: ”Zabbix” } Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
http_proxy	string	HTTP(S) proxy connection string. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
ipmi_sensor	string	IPMI sensor. Property behavior: - *required* if `type` is set to ”IPMI agent” and `key_` is not set to ”ipmi.get” - *supported* if `type` is set to ”IPMI agent” - *read-only* for inherited objects
jmx_endpoint	string	JMX agent custom connection string. Default: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi Property behavior: - *supported* if `type` is set to ”JMX agent”
lifetime	string	Time period after which items that are no longer discovered will be deleted. Accepts seconds, time unit with suffix, or a user macro. Default: 30d.
master_itemid	integer	Master item ID. Recursion up to 3 dependent items and maximum count of dependent items equal to 999 are allowed. Discovery rule cannot be master item for another discovery rule. Property behavior: - *required* if `type` is set to ”Dependent item” - *read-only* for inherited objects
output_format	integer	Should the response be converted to JSON. Possible values: 0 - *(default)* Store raw; 1 - Convert to JSON. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects

Property	Type	Description
params	string	Additional parameters depending on the type of the LLD rule: - executed script for SSH and Telnet LLD rules; - SQL query for database monitor LLD rules; - formula for calculated LLD rules. Property behavior: - *required* if `type` is set to ”Database monitor”, ”SSH agent”, ”TELNET agent”, or ”Script” - *read-only* for inherited objects (if `type` is set to ”Script”)
parameters	array	Additional parameters if `type` is set to ”Script”. Array of objects with `name` and `value` properties, where `name` must be unique. Property behavior: - *supported* if `type` is set to ”Script” - *read-only* for inherited objects
password	string	Password for authentication. Property behavior: - *required* if `type` is set to ”JMX agent” and `username` is set - *supported* if `type` is set to ”Simple check”, ”Database monitor”, ”SSH agent”, ”TELNET agent”, or ”HTTP agent” - *read-only* for inherited objects (if `type` is set to ”HTTP agent”)
post_type	integer	Type of post data body stored in `posts` property. Possible values: 0 - *(default)* Raw data; 2 - JSON data; 3 - XML data. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
posts	string	HTTP(S) request body data. Property behavior: - *required* if `type` is set to ”HTTP agent” and `post_type` is set to ”JSON data” or ”XML data” - *supported* if `type` is set to ”HTTP agent” and `post_type` is set to ”Raw data” - *read-only* for inherited objects
privatekey	string	Name of the private key file. Property behavior: - *required* if `type` is set to ”SSH agent” and `authtype` is set to ”public key”
publickey	string	Name of the public key file. Property behavior: - *required* if `type` is set to ”SSH agent” and `authtype` is set to ”public key”
query_fields	array	Query parameters. Array of objects with `key:value` pairs, where `value` can be an empty string.

Property	Type	Description
request_method	integer	Type of request method. Possible values: 0 - *(default)* GET; 1 - POST; 2 - PUT; 3 - HEAD. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
retrieve_mode	integer	What part of response should be stored. Possible values if `request_method` is set to ”GET”, ”POST”, or ”PUT”: 0 - *(default)* Body; 1 - Headers; 2 - Both body and headers will be stored. Possible values if `request_method` is set to ”HEAD”: 1 - Headers. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
snmp_oid	string	SNMP OID. Property behavior: - *required* if `type` is set to ”SNMP agent” - *read-only* for inherited objects
ssl_cert_file	string	Public SSL Key file path. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
ssl_key_file	string	Private SSL Key file path. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
ssl_key_password	string	Password for SSL Key file. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
state	integer	State of the LLD rule. Possible values: 0 - *(default)* normal; 1 - not supported. Property behavior: - *read-only*
status	integer	Status of the LLD rule. Possible values: 0 - *(default)* enabled LLD rule; 1 - disabled LLD rule.

Property	Type	Description
status_codes	string	Ranges of required HTTP status codes, separated by commas. Also supports user macros as part of comma separated list. Example: 200,200-{$M},{$M},200-400 Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
templateid	string	ID of the parent template LLD rule. Property behavior: - *read-only*
timeout	string	Item data polling request timeout. Supports user macros. Default: 3s. Maximum value: 60s. Property behavior: - *supported* if `type` is set to ”HTTP agent” or ”Script” - *read-only* for inherited objects
trapper_hosts	string	Allowed hosts. Property behavior: - *supported* if `type` is set to ”Zabbix trapper”, or if `type` is set to ”HTTP agent” and `allow_traps` is set to ”Allow to accept incoming data”
username	string	Username for authentication. Property behavior: - *required* if `type` is set to ”SSH agent”, ”TELNET agent”, or if `type` is set to ”JMX agent” and `password` is set - *supported* if `type` is set to ”Simple check”, ”Database monitor”, or ”HTTP agent” - *read-only* for inherited objects (if `type` is set to ”HTTP agent”)
uuid	string	Universal unique identifier, used for linking imported LLD rules to already existing ones. Auto-generated, if not given. Property behavior: - *supported* if the LLD rule belongs to a template
verify_host	integer	Whether to validate that the host name for the connection matches the one in the host’s certificate. Possible values: 0 - *(default)* Do not validate; 1 - Validate. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects
verify_peer	integer	Whether to validate that the host’s certificate is authentic. Possible values: 0 - *(default)* Do not validate; 1 - Validate. Property behavior: - *supported* if `type` is set to ”HTTP agent” - *read-only* for inherited objects

LLD rule filter

The LLD rule filter object defines a set of conditions that can be used to filter discovered objects. It has the following properties:

Property	Type	Description
conditions	array	Set of filter conditions to use for filtering results. Property behavior: - *required*
evaltype	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression. Property behavior: - *required*
eval_formula	string	Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its `formulaid`. The value of `eval_formula` is equal to the value of `formula` for filters with a custom expression. Property behavior: - *read-only*
formula	string	User-defined expression to be used for evaluating conditions of filters with a custom expression. The expression must contain IDs that reference specific filter conditions by its `formulaid`. The IDs used in the expression must exactly match the ones defined in the filter conditions: no condition can remain unused or omitted. Property behavior: - *required* if `evaltype` is set to "custom expression"

LLD rule filter condition

The LLD rule filter condition object defines a separate check to perform on the value of an LLD macro. It has the following properties:

Property	Type	Description
macro	string	LLD macro to perform the check on. Property behavior: - *required*
value	string	Value to compare with. Property behavior: - *required*
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
operator	integer	Condition operator. Possible values: 8 - *(default)* matches regular expression; 9 - does not match regular expression; 12 - exists; 13 - does not exist.

Note:
To better understand how to use filters with various types of expressions, see examples on the discoveryrule.get and discoveryrule.create method pages.

LLD macro path

The LLD macro path has the following properties:

Property	Type	Description
lld_macro	string	LLD macro. Property behavior: - *required*
path	string	Selector for value which will be assigned to corresponding macro. Property behavior: - *required*

LLD rule preprocessing

The LLD rule preprocessing object has the following properties.

Property	Type	Description
type	integer	The preprocessing option type. Possible values: 5 - Regular expression; 11 - XML XPath; 12 - JSONPath; 15 - Does not match regular expression; 16 - Check for error in JSON; 17 - Check for error in XML; 20 - Discard unchanged with heartbeat; 21 - JavaScript; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace; 27 - XML to JSON; 28 - SNMP walk value; 29 - SNMP walk to JSON. Property behavior: - *required*
params	string	Additional parameters used by preprocessing option. Multiple parameters are separated by the newline (\n) character. Property behavior: - *required* if `type` is set to ”Regular expression” (5), ”XML XPath” (11), ”JSONPath” (12), ”Does not match regular expression” (15), ”Check for error in JSON” (16), ”Check for error in XML” (17), ”Discard unchanged with heartbeat” (20), ”JavaScript” (21), ”Prometheus to JSON” (23), ”CSV to JSON” (24), ”Replace” (25), ”SNMP walk value” (28), or ”SNMP walk to JSON” (29)

Property	Type	Description
error_handler	integer	Action type used in case of preprocessing step failure. Possible values: 0 - Error message is set by Zabbix server; 1 - Discard value; 2 - Set custom value; 3 - Set custom error message. Property behavior: - *required* if `type` is set to ”Regular expression” (5), ”XML XPath” (11), ”JSONPath” (12), ”Does not match regular expression” (15), ”Check for error in JSON” (16), ”Check for error in XML” (17), ”Prometheus to JSON” (23), ”CSV to JSON” (24), ”XML to JSON” (27), ”SNMP walk value” (28), or ”SNMP walk to JSON” (29)
error_handler_params	string	Error handler parameters. Property behavior: - *required* if `error_handler` is set to ”Set custom value” or ”Set custom error message”

The following parameters and error handlers are supported for each preprocessing type.

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
5	Regular expression	pattern[1]	output[2]		0, 1, 2, 3
11	XML XPath	path[3]			0, 1, 2, 3
12	JSONPath	path[3]			0, 1, 2, 3
15	Does not match regular expression	pattern[1]			0, 1, 2, 3
16	Check for error in JSON	path[3]			0, 1, 2, 3
17	Check for error in XML	path[3]			0, 1, 2, 3
20	Discard unchanged with heartbeat	seconds[4, 5]			
21	JavaScript	script[2]			
23	Prometheus to JSON	pattern[5, 6]			0, 1, 2, 3

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
24	CSV to JSON	character[2]	character[2]	0,1	0, 1, 2, 3
25	Replace	search string[2]	replacement[2]		
27	XML to JSON				0, 1, 2, 3
28	SNMP walk value	OID[2]	Format: 0 - Unchanged 1 - UTF-8 from Hex-STRING 2 - MAC from Hex-STRING 3 - Integer from BITS		0, 1, 2, 3
29	SNMP walk to JSON[7]	Field name[2]	OID prefix[2]	Format: 0 - Unchanged 1 - UTF-8 from Hex-STRING 2 - MAC from Hex-STRING 3 - Integer from BITS	0, 1, 2, 3

[1] regular expression
[2] string
[3] JSONPath or XML XPath
[4] positive integer (with support of time suffixes, e.g. 30s, 1m, 2h, 1d)
[5] user macro
[6] Prometheus pattern following the syntax: `<metric name>{<label name>="<label value>", ...} == <value>`. Each Prometheus pattern component (metric, label name, label value and metric value) can be user macro.
[7] Supports multiple "Field name,OID prefix,Format records" records delimited by a new line character.

LLD rule overrides

The LLD rule overrides object defines a set of rules (filters, conditions and operations) that are used to override properties of different prototype objects. It has the following properties:

Property	Type	Description
name	string	Unique override name. Property behavior: - *required*
step	integer	Unique order number of the override. Property behavior: - *required*
stop	integer	Stop processing next overrides if matches. Possible values: 0 - *(default)* don't stop processing overrides; 1 - stop processing overrides if filter matches.
filter	object	Override filter.
operations	array	Override operations.

LLD rule override filter

The LLD rule override filter object defines a set of conditions that if they match the discovered object the override is applied. It has the following properties:

Property	Type	Description
evaltype	integer	Override filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression. Property behavior: - *required*
conditions	array	Set of override filter conditions to use for matching the discovered objects. Property behavior: - *required*
eval_formula	string	Generated expression that will be used for evaluating override filter conditions. The expression contains IDs that reference specific override filter conditions by its `formulaid`. The value of `eval_formula` is equal to the value of `formula` for filters with a custom expression. Property behavior: - *read-only*
formula	string	User-defined expression to be used for evaluating conditions of override filters with a custom expression. The expression must contain IDs that reference specific override filter conditions by its `formulaid`. The IDs used in the expression must exactly match the ones defined in the override filter conditions: no condition can remain unused or omitted. Property behavior: - *required* if `evaltype` is set to ”custom expression”

LLD rule override filter condition

The LLD rule override filter condition object defines a separate check to perform on the value of an LLD macro. It has the following properties:

Property	Type	Description
macro	string	LLD macro to perform the check on. Property behavior: - *required*
value	string	Value to compare with. Property behavior: - *required*
formulaid	string	Arbitrary unique ID that is used to reference the condition from a custom expression. Can only contain capital-case letters. The ID must be defined by the user when modifying filter conditions, but will be generated anew when requesting them afterward.
operator	integer	Condition operator. Possible values: 8 - *(default)* matches regular expression; 9 - does not match regular expression; 12 - exists; 13 - does not exist.

LLD rule override operation

The LLD rule override operation is combination of conditions and actions to perform on the prototype object. It has the following properties:

Property	Type	Description
operationobject	integer	Type of discovered object to perform the action. Possible values: 0 - Item prototype; 1 - Trigger prototype; 2 - Graph prototype; 3 - Host prototype. Property behavior: - *required*
operator	integer	Override condition operator. Possible values: 0 - *(default)* equals; 1 - does not equal; 2 - contains; 3 - does not contain; 8 - matches; 9 - does not match.
value	string	Pattern to match item, trigger, graph or host prototype name depending on selected object.
opstatus	object	Override operation status object for item, trigger and host prototype objects.
opdiscover	object	Override operation discover status object (all object types).
opperiod	object	Override operation period (update interval) object for item prototype object.
ophistory	object	Override operation history object for item prototype object.
optrends	object	Override operation trends object for item prototype object.
opseverity	object	Override operation severity object for trigger prototype object.
optag	array	Override operation tag object for trigger and host prototype objects.
optemplate	array	Override operation template object for host prototype object.
opinventory	object	Override operation inventory object for host prototype object.

LLD rule override operation status

LLD rule override operation status that is set to discovered object. It has the following properties:

Property	Type	Description
status	integer	Override the status for selected object. Possible values: 0 - Create enabled; 1 - Create disabled. Property behavior: - *required*

LLD rule override operation discover

LLD rule override operation discover status that is set to discovered object. It has the following properties:

Property	Type	Description
discover	integer	Override the discover status for selected object. Possible values: 0 - Yes, continue discovering the objects; 1 - No, new objects will not be discovered and existing ones will be marked as lost. Property behavior: - *required*

LLD rule override operation period

LLD rule override operation period is an update interval value (supports custom intervals) that is set to discovered item. It has the following properties:

Property	Type	Description
delay	string	Override the update interval of the item prototype. Accepts seconds or a time unit with suffix (30s,1m,2h,1d) as well as flexible and scheduling intervals and user macros or LLD macros. Multiple intervals are separated by a semicolon. Property behavior: - *required*

LLD rule override operation history

LLD rule override operation history value that is set to discovered item. It has the following properties:

Property	Type	Description
history	string	Override the history of item prototype which is a time unit of how long the history data should be stored. Also accepts user macro and LLD macro. Property behavior: - *required*

LLD rule override operation trends

LLD rule override operation trends value that is set to discovered item. It has the following properties:

Property	Type	Description
trends	string	Override the trends of item prototype which is a time unit of how long the trends data should be stored. Also accepts user macro and LLD macro. Property behavior: - *required*

LLD rule override operation severity

LLD rule override operation severity value that is set to discovered trigger. It has the following properties:

Property	Type	Description
severity	integer	Override the severity of trigger prototype. Possible values: 0 - *(default)* not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster. Property behavior: - *required*

LLD rule override operation tag

LLD rule override operation tag object contains tag name and value that are set to discovered object. It has the following properties:

Property	Type	Description
tag	string	New tag name. Property behavior: - *required*
value	string	New tag value.

LLD rule override operation template

LLD rule override operation template object that is linked to discovered host. It has the following properties:

Property	Type	Description
templateid	string	Override the template of host prototype linked templates. Property behavior: - *required*

LLD rule override operation inventory

LLD rule override operation inventory mode value that is set to discovered host. It has the following properties:

Property	Type	Description
inventory_mode	integer	Override the host prototype inventory mode. Possible values: -1 - disabled; 0 - *(default)* manual; 1 - automatic. Property behavior: - *required*

discoveryrule.copy

Description

`object discoveryrule.copy(object parameters)`

This method allows to copy LLD rules with all of the prototypes to the given hosts.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the LLD rules to copy and the target hosts.

Parameter	Type	Description
discoveryids	array	IDs of the LLD rules to be copied.
hostids	array	IDs of the hosts to copy the LLD rules to.

Return values

`(boolean)` Returns `true` if the copying was successful.

Examples

Copy an LLD rule to multiple hosts

Copy an LLD rule to two hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.copy",
    "params": {
        "discoveryids": [
            "27426"
        ],
        "hostids": [
            "10196",
            "10197"
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Source

CDiscoveryRule::copy() in *ui/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.create

Description

`object discoveryrule.create(object/array lldRules)`

This method allows to create new LLD rules.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` LLD rules to create.

Additionally to the standard LLD rule properties, the method accepts the following parameters.

Parameter	Type	Description
filter	object	LLD rule filter for the LLD rule.

Parameter	Type	Description
preprocessing	array	LLD rule preprocessing options.
lld_macro_paths	array	LLD rule lld_macro_path options.
overrides	array	LLD rule overrides options.

Return values

(object) Returns an object containing the IDs of the created LLD rules under the itemids property. The order of the returned IDs matches the order of the passed LLD rules.

Examples

Creating an LLD rule

Create a Zabbix agent LLD rule to discover mounted file systems. Discovered items will be updated every 30 seconds.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Mounted filesystem discovery",
        "key_": "vfs.fs.discovery",
        "hostid": "10197",
        "type": 0,
        "interfaceid": "112",
        "delay": "30s"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

Using a filter

Create an LLD rule with a set of conditions to filter the results by. The conditions will be grouped together using the logical "and" operator.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Filtered LLD rule",
        "key_": "lld",
        "hostid": "10116",
        "type": 0,
        "interfaceid": "13",
        "delay": "30s",
        "filter": {
            "evaltype": 1,
            "conditions": [
                {
                    "macro": "{#MACRO1}",
                    "value": "@regex1"
```

```
                },
                {
                    "macro": "{#MACRO2}",
                    "value": "@regex2",
                    "operator": "9"
                },
                {
                    "macro": "{#MACRO3}",
                    "value": "",
                    "operator": "12"
                },
                {
                    "macro": "{#MACRO4}",
                    "value": "",
                    "operator": "13"
                }
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

Creating an LLD rule with macro paths

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "LLD rule with LLD macro paths",
        "key_": "lld",
        "hostid": "10116",
        "type": 0,
        "interfaceid": "13",
        "delay": "30s",
        "lld_macro_paths": [
            {
                "lld_macro": "{#MACRO1}",
                "path": "$.path.1"
            },
            {
                "lld_macro": "{#MACRO2}",
                "path": "$.path.2"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

Using a custom expression filter

Create an LLD rule with a filter that will use a custom expression to evaluate the conditions. The LLD rule must only discover objects the "{#MACRO1}" macro value of which matches both regular expression "regex1" and "regex2", and the value of "{#MACRO2}" matches either "regex3" or "regex4". The formula IDs "A", "B", "C" and "D" have been chosen arbitrarily.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Filtered LLD rule",
        "key_": "lld",
        "hostid": "10116",
        "type": 0,
        "interfaceid": "13",
        "delay": "30s",
        "filter": {
            "evaltype": 3,
            "formula": "(A and B) and (C or D)",
            "conditions": [
                {
                    "macro": "{#MACRO1}",
                    "value": "@regex1",
                    "formulaid": "A"
                },
                {
                    "macro": "{#MACRO1}",
                    "value": "@regex2",
                    "formulaid": "B"
                },
                {
                    "macro": "{#MACRO2}",
                    "value": "@regex3",
                    "formulaid": "C"
                },
                {
                    "macro": "{#MACRO2}",
                    "value": "@regex4",
                    "formulaid": "D"
                }
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
```

```
        ]
    },
    "id": 1
}
```

Using custom query fields and headers

Create LLD rule with custom query fields and headers.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "hostid": "10257",
        "interfaceid": "5",
        "type": 19,
        "name": "API HTTP agent",
        "key_": "api_discovery_rule",
        "value_type": 3,
        "delay": "5s",
        "url": "http://127.0.0.1?discoverer.php",
        "query_fields": [
            {
                "mode": "json"
            },
            {
                "elements": "2"
            }
        ],
        "headers": {
            "X-Type": "api",
            "Authorization": "Bearer mF_A.B5f-2.1JcM"
        },
        "allow_traps": 1,
        "trapper_hosts": "127.0.0.1"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 1
}
```

Creating an LLD rule with preprocessing

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Discovery rule with preprocessing",
        "key_": "lld.with.preprocessing",
        "hostid": "10001",
        "ruleid": "27665",
        "type": 0,
```

```
        "value_type": 3,
        "delay": "60s",
        "interfaceid": "1155",
        "preprocessing": [
            {
                "type": 20,
                "params": "20",
                "error_handler": 0,
                "error_handler_params": ""
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

Creating an LLD rule with overrides

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Discover database host",
        "key_": "lld.with.overrides",
        "hostid": "10001",
        "type": 0,
        "value_type": 3,
        "delay": "60s",
        "interfaceid": "1155",
        "overrides": [
            {
                "name": "Discover MySQL host",
                "step": "1",
                "stop": "1",
                "filter": {
                    "evaltype": "2",
                    "conditions": [
                        {
                            "macro": "{#UNIT.NAME}",
                            "operator": "8",
                            "value": "^mysqld\\.service$"
                        },
                        {
                            "macro": "{#UNIT.NAME}",
                            "operator": "8",
                            "value": "^mariadb\\.service$"
                        }
                    ]
                },
                "operations": [
                    {
                        "operationobject": "3",
```

```
                        "operator": "2",
                        "value": "Database host",
                        "opstatus": {
                            "status": "0"
                        },
                        "optemplate": [
                            {
                                "templateid": "10170"
                            }
                        ],
                        "optag": [
                            {
                                "tag": "Database",
                                "value": "MySQL"
                            }
                        ]
                    }
                ]
            },
            {
                "name": "Discover PostgreSQL host",
                "step": "2",
                "stop": "1",
                "filter": {
                    "evaltype": "0",
                    "conditions": [
                        {
                            "macro": "{#UNIT.NAME}",
                            "operator": "8",
                            "value": "^postgresql\\.service$"
                        }
                    ]
                },
                "operations": [
                    {
                        "operationobject": "3",
                        "operator": "2",
                        "value": "Database host",
                        "opstatus": {
                            "status": "0"
                        },
                        "optemplate": [
                            {
                                "templateid": "10263"
                            }
                        ],
                        "optag": [
                            {
                                "tag": "Database",
                                "value": "PostgreSQL"
                            }
                        ]
                    }
                ]
            }
        ]
    },
    "id": 1
}
```


Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "30980"
        ]
    },
    "id": 1
}
```

Create script LLD rule

Create a simple data collection using a script LLD rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.create",
    "params": {
        "name": "Script example",
        "key_": "custom.script.lldrule",
        "hostid": "12345",
        "type": 21,
        "value_type": 4,
        "params": "var request = new HttpRequest();\nreturn request.post(\"https://postman-echo.com/post\"
        "parameters": [{
            "name": "host",
            "value": "{HOST.CONN}"
        }],
        "timeout": "6s",
        "delay": "30s"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

See also

- LLD rule filter
- LLD macro paths
- LLD rule preprocessing

Source

CDiscoveryRule::create() in *ui/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.delete

Description

`object discoveryrule.delete(array lldRuleIds)`

This method allows to delete LLD rules.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the LLD rules to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted LLD rules under the `ruleids` property.

Examples

Deleting multiple LLD rules

Delete two LLD rules.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.delete",
    "params": [
        "27665",
        "27668"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "ruleids": [
            "27665",
            "27668"
        ]
    },
    "id": 1
}
```

Source

CDiscoveryRule::delete() in *ui/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.get

Description

`integer/array discoveryrule.get(object parameters)`

The method allows to retrieve LLD rules according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
itemids	string/array	Return only LLD rules with the given IDs.
groupids	string/array	Return only LLD rules that belong to the hosts from the given groups.
hostids	string/array	Return only LLD rules that belong to the given hosts.

Parameter	Type	Description
inherited	boolean	If set to `true` return only LLD rules inherited from a template.
interfaceids	string/array	Return only LLD rules use the given host interfaces.
monitored	boolean	If set to `true` return only enabled LLD rules that belong to monitored hosts.
templated	boolean	If set to `true` return only LLD rules that belong to templates.
templateids	string/array	Return only LLD rules that belong to the given templates.
selectFilter	query	Return a filter property with data of the filter used by the LLD rule.
selectGraphs	query	Returns a graphs property with graph prototypes that belong to the LLD rule. Supports `count`.
selectHostPrototypes	query	Return a hostPrototypes property with host prototypes that belong to the LLD rule. Supports `count`.
selectHosts	query	Return a hosts property with an array of hosts that the LLD rule belongs to.
selectItems	query	Return an items property with item prototypes that belong to the LLD rule. Supports `count`.
selectTriggers	query	Return a triggers property with trigger prototypes that belong to the LLD rule. Supports `count`.
selectLLDMacroPaths	query	Return an lld_macro_paths property with a list of LLD macros and paths to values assigned to each corresponding macro.
selectPreprocessing	query	Return a preprocessing property with LLD rule preprocessing options.
selectOverrides	query	Return an lld_rule_overrides property with a list of override filters, conditions and operations that are performed on prototype objects.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: `host` - technical name of the host that the LLD rule belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectItems`, `selectGraphs`, `selectTriggers`.
sortfield	string/array	Sort the result by the given properties. Possible values: `itemid`, `name`, `key_`, `delay`, `type`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;

- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving discovery rules from a host

Retrieve all discovery rules for specific host ID.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": "extend",
        "hostids": "10202"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "27425",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Network interface discovery",
            "key_": "net.if.discovery",
            "delay": "1h",
            "status": "0",
            "trapper_hosts": "",
            "templateid": "22444",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "119",
            "description": "Discovery of network interfaces as defined in global regular expression \"Netw
            "lifetime": "30d",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
```

```
            "uuid": "",
            "state": "0",
            "error": "",
            "parameters": []
        },
        {
            "itemid": "27426",
            "type": "0",
            "snmp_oid": "",
            "hostid": "10202",
            "name": "Mounted filesystem discovery",
            "key_": "vfs.fs.discovery",
            "delay": "1h",
            "status": "0",
            "trapper_hosts": "",
            "templateid": "22450",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "119",
            "description": "Discovery of file systems of different types as defined in global regular expr
            "lifetime": "30d",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "",
            "query_fields": [],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": [],
            "retrieve_mode": "0",
            "request_method": "0",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
            "uuid": "",
            "state": "0",
            "error": "",
            "parameters": []
        }
    ],
    "id": 1
}
```


Retrieving filter conditions

Retrieve the name of the LLD rule "24681" and its filter conditions. The filter uses the "and" evaluation type, so the `formula` property is empty and `eval_formula` is generated automatically.

Request:


```
{
    "jsonrpc": "2.0",
```

```
    "method": "discoveryrule.get",
    "params": {
        "output": ["name"],
        "selectFilter": "extend",
        "itemids": ["24681"]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "24681",
            "name": "Filtered LLD rule",
            "filter": {
                "evaltype": "1",
                "formula": "",
                "conditions": [
                    {
                        "macro": "{#MACRO1}",
                        "value": "@regex1",
                        "operator": "8",
                        "formulaid": "A"
                    },
                    {
                        "macro": "{#MACRO2}",
                        "value": "@regex2",
                        "operator": "9",
                        "formulaid": "B"
                    },
                    {
                        "macro": "{#MACRO3}",
                        "value": "",
                        "operator": "12",
                        "formulaid": "C"
                    },
                    {
                        "macro": "{#MACRO4}",
                        "value": "",
                        "operator": "13",
                        "formulaid": "D"
                    }
                ],
                "eval_formula": "A and B and C and D"
            }
        }
    ],
    "id": 1
}
```

Retrieve LLD rule by URL

Retrieve LLD rule for host by rule URL field value. Only exact match of URL string defined for LLD rule is supported.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "hostids": "10257",
        "filter": {
```

```
            "type": 19,
            "url": "http://127.0.0.1/discoverer.php"
        }
    },
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "28336",
            "type": "19",
            "snmp_oid": "",
            "hostid": "10257",
            "name": "API HTTP agent",
            "key_": "api_discovery_rule",
            "delay": "5s",
            "status": "0",
            "trapper_hosts": "",
            "templateid": "0",
            "valuemapid": "0",
            "params": "",
            "ipmi_sensor": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "interfaceid": "5",
            "description": "",
            "lifetime": "30d",
            "jmx_endpoint": "",
            "master_itemid": "0",
            "timeout": "3s",
            "url": "http://127.0.0.1/discoverer.php",
            "query_fields": [
                {
                    "mode": "json"
                },
                {
                    "elements": "2"
                }
            ],
            "posts": "",
            "status_codes": "200",
            "follow_redirects": "1",
            "post_type": "0",
            "http_proxy": "",
            "headers": {
                "X-Type": "api",
                "Authorization": "Bearer mF_A.B5f-2.1JcM"
            },
            "retrieve_mode": "0",
            "request_method": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "allow_traps": "0",
```

```
            "uuid": "",
            "state": "0",
            "error": "",
            "parameters": []
        }
    ],
    "id": 1
}
```

Retrieve LLD rule with overrides

Retrieve one LLD rule that has various override settings.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.get",
    "params": {
        "output": ["name"],
        "itemids": "30980",
        "selectOverrides": ["name", "step", "stop", "filter", "operations"]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "name": "Discover database host",
            "overrides": [
                {
                    "name": "Discover MySQL host",
                    "step": "1",
                    "stop": "1",
                    "filter": {
                        "evaltype": "2",
                        "formula": "",
                        "conditions": [
                            {
                                "macro": "{#UNIT.NAME}",
                                "operator": "8",
                                "value": "^mysqld\\.service$",
                                "formulaid": "A"
                            },
                            {
                                "macro": "{#UNIT.NAME}",
                                "operator": "8",
                                "value": "^mariadb\\.service$",
                                "formulaid": "B"
                            }
                        ],
                        "eval_formula": "A or B"
                    },
                    "operations": [
                        {
                            "operationobject": "3",
                            "operator": "2",
                            "value": "Database host",
                            "opstatus": {
                                "status": "0"
                            },
```

```
                            "optag": [
                                {
                                    "tag": "Database",
                                    "value": "MySQL"
                                }
                            ],
                            "optemplate": [
                                {
                                    "templateid": "10170"
                                }
                            ]
                        }
                    ]
                },
                {
                    "name": "Discover PostgreSQL host",
                    "step": "2",
                    "stop": "1",
                    "filter": {
                        "evaltype": "0",
                        "formula": "",
                        "conditions": [
                            {
                                "macro": "{#UNIT.NAME}",
                                "operator": "8",
                                "value": "^postgresql\\.service$",
                                "formulaid": "A"
                            }
                        ],
                        "eval_formula": "A"
                    },
                    "operations": [
                        {
                            "operationobject": "3",
                            "operator": "2",
                            "value": "Database host",
                            "opstatus": {
                                "status": "0"
                            },
                            "optag": [
                                {
                                    "tag": "Database",
                                    "value": "PostgreSQL"
                                }
                            ],
                            "optemplate": [
                                {
                                    "templateid": "10263"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Graph prototype
- Host

- Item prototype
- LLD rule filter
- Trigger prototype

Source

CDiscoveryRule::get() in *ui/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.update

Description

`object discoveryrule.update(object/array lldRules)`

This method allows to update existing LLD rules.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` LLD rule properties to be updated.

The `itemid` property must be defined for each LLD rule, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard LLD rule properties, the method accepts the following parameters.

Parameter	Type	Description
filter	object	LLD rule filter to replace the current filter.
preprocessing	array	LLD rule preprocessing options to replace the existing preprocessing options. Parameter behavior: - *read-only* for inherited objects
lld_macro_paths	array	LLD rule lld_macro_path options to replace the existing lld_macro_path options. Parameter behavior: - *read-only* for inherited objects
overrides	array	LLD rule overrides options to replace the existing overrides options. Parameter behavior: - *read-only* for inherited objects

Return values

`(object)` Returns an object containing the IDs of the updated LLD rules under the `itemids` property.

Examples

Adding a filter to an LLD rule

Add a filter so that the contents of the *{#FSTYPE}* macro would match the *@File systems for discovery* regexp.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "22450",
        "filter": {
            "evaltype": 1,
            "conditions": [
                {
```

```
                    "macro": "{#FSTYPE}",
                    "value": "@File systems for discovery"
                }
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22450"
        ]
    },
    "id": 1
}
```

Adding LLD macro paths

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "22450",
        "lld_macro_paths": [
            {
                "lld_macro": "{#MACRO1}",
                "path": "$.json.path"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "22450"
        ]
    },
    "id": 1
}
```

Disable trapping

Disable LLD trapping for discovery rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "28336",
        "allow_traps": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 1
}
```

Updating LLD rule preprocessing options

Update an LLD rule with preprocessing rule "JSONPath".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "44211",
        "preprocessing": [
            {
                "type": 12,
                "params": "$.path.to.json",
                "error_handler": 2,
                "error_handler_params": "5"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

Updating LLD rule script

Update an LLD rule script with a different script and remove unnecessary parameters that were used by previous script.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.update",
    "params": {
        "itemid": "23865",
        "parameters": [],
        "script": "Zabbix.log(3, 'Log test');\nreturn 1;"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
```

```
        "itemids": [
            "23865"
        ]
    },
    "id": 1
}
```

Source

CDiscoveryRule::update() in *ui/include/classes/api/services/CDiscoveryRule.php*.

Maintenance

This class is designed to work with maintenances.

Object references:

- Maintenance
- Time period

Available methods:

- maintenance.create - creating new maintenances
- maintenance.delete - deleting maintenances
- maintenance.get - retrieving maintenances
- maintenance.update - updating maintenances

> Maintenance object

The following objects are directly related to the `maintenance` API.

Maintenance

The maintenance object has the following properties.

Property	Type	Description
maintenanceid	string	ID of the maintenance. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the maintenance. Property behavior: - *required* for create operations
active_since	timestamp	Time when the maintenance becomes active. The given value will be rounded down to minutes. Property behavior: - *required* for create operations
active_till	timestamp	Time when the maintenance stops being active. The given value will be rounded down to minutes. Property behavior: - *required* for create operations
description	string	Description of the maintenance.
maintenance_type	integer	Type of maintenance. Possible values: 0 - *(default)* with data collection; 1 - without data collection.

Property	Type	Description
tags_evaltype	integer	Problem tag evaluation method. Possible values: 0 - *(default)* And/Or; 2 - Or.

Time period

The time period object is used to define periods when the maintenance must come into effect. It has the following properties.

Property	Type	Description
period	integer	Duration of the maintenance period in seconds. The given value will be rounded down to minutes. Default: 3600.
timeperiod_type	integer	Type of time period. Possible values: 0 - *(default)* one time only; 2 - daily; 3 - weekly; 4 - monthly.
start_date	timestamp	Date when the maintenance period must come into effect. The given value will be rounded down to minutes. Default: current date. Property behavior: - *supported* if `timeperiod_type` is set to ”one time only”
start_time	integer	Time of day when the maintenance starts in seconds. The given value will be rounded down to minutes. Default: 0. Property behavior: - *supported* if `timeperiod_type` is set to ”daily”, ”weekly”, or ”monthly”

Property	Type	Description
every	integer	For daily and weekly periods `every` defines the day or week intervals at which the maintenance must come into effect. Default value if `timeperiod_type` is set to ”daily” or ”weekly”: `1`. For monthly periods when `day` is set, the `every` property defines the day of the month when the maintenance must come into effect. Default value if `timeperiod_type` is set to ”monthly” and `day` is set: `1`. For monthly periods when `dayofweek` is set, the `every` property defines the week of the month when the maintenance must come into effect. Possible values if `timeperiod_type` is set to ”monthly” and `dayofweek` is set: 1 - *(default)* first week; 2 - second week; 3 - third week; 4 - fourth week; 5 - last week. Property behavior: - *supported* if `timeperiod_type` is set to ”daily”, ”weekly”, or ”monthly”
dayofweek	integer	Days of the week when the maintenance must come into effect. Possible bitmap values: 1 - Monday; 2 - Tuesday; 4 - Wednesday; 8 - Thursday; 16 - Friday; 32 - Saturday; 64 - Sunday. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 21 for Monday, Wednesday, and Friday). Property behavior: - *required* if `timeperiod_type` is set to ”weekly” or if `timeperiod_type` is set to ”monthly” and `day` is not set
day	integer	Day of the month when the maintenance must come into effect. Property behavior: - *required* if `timeperiod_type` is set to ”monthly” and `dayofweek` is not set

Property	Type	Description
month	integer	Months when the maintenance must come into effect. Possible bitmap values: 1 - January; 2 - February; 4 - March; 8 - April; 16 - May; 32 - June; 64 - July; 128 - August; 256 - September; 512 - October; 1024 - November; 2048 - December. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 585 for January, April, July, and October). Property behavior: - *required* if `timeperiod_type` is set to ”monthly”

Problem tag

The problem tag object is used to define which problems must be suppressed when the maintenance comes into effect. Tags can only be specified if `maintenance_type` of Maintenance object is set to ”with data collection”. It has the following properties.

Property	Type	Description
tag	string	Problem tag name. Property behavior: - *required*
operator	integer	Condition operator. Possible values: 0 - Equals; 2 - *(default)* Contains.
value	string	Problem tag value.

maintenance.create

Description

`object maintenance.create(object/array maintenances)`

This method allows to create new maintenances.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Maintenances to create.

Additionally to the standard maintenance properties, the method accepts the following parameters.

Parameter	Type	Description
groups	object/array	Host groups that will undergo maintenance. The host groups must have the `groupid` property defined. Parameter behavior: - *required* if `hosts` is not set
hosts	object/array	Hosts that will undergo maintenance. The hosts must have only the `hostid` property defined. Parameter behavior: - *required* if `groups` is not set
timeperiods	object/array	Maintenance time periods. Parameter behavior: - *required*
tags	object/array	Problem tags. Define what problems must be suppressed. If no tags are given, all active maintenance host problems will be suppressed. Parameter behavior: - *supported* if `maintenance_type` of Maintenance object is set to "with data collection"
groupids (deprecated)	array	This parameter is deprecated, please use `groups` instead. IDs of the host groups that will undergo maintenance.
hostids (deprecated)	array	This parameter is deprecated, please use `hosts` instead. IDs of the hosts that will undergo maintenance.

Return values

`(object)` Returns an object containing the IDs of the created maintenances under the `maintenanceids` property. The order of the returned IDs matches the order of the passed maintenances.

Examples

Creating a maintenance

Create a maintenance with data collection for host group with ID "2" and with problem tags **service:mysqld** and **error**. It must be active from 22.01.2013 till 22.01.2014, come in effect each Sunday at 18:00 and last for one hour.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.create",
    "params": {
        "name": "Sunday maintenance",
        "active_since": 1358844540,
        "active_till": 1390466940,
        "tags_evaltype": 0,
        "groups": [
            {"groupid": "2"}
        ],
        "timeperiods": [
            {
                "period": 3600,
                "timeperiod_type": 3,
                "start_time": 64800,
                "every": 1,
                "dayofweek": 64
            }
        ],
```

```
        "tags": [
            {
                "tag": "service",
                "operator": "0",
                "value": "mysqld"
            },
            {
                "tag": "error",
                "operator": "2",
                "value": ""
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Time period

Source

CMaintenance::create() in *ui/include/classes/api/services/CMaintenance.php*.

maintenance.delete

Description

`object maintenance.delete(array maintenanceIds)`

This method allows to delete maintenance periods.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the maintenance periods to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted maintenance periods under the `maintenanceids` property.

Examples

Deleting multiple maintenance periods

Delete two maintenance periods.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.delete",
    "params": [
        "3",
```

```
            "1"
        ],
        "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3",
            "1"
        ]
    },
    "id": 1
}
```

Source

CMaintenance::delete() in *ui/include/classes/api/services/CMaintenance.php*.

maintenance.get

Description

`integer/array maintenance.get(object parameters)`

The method allows to retrieve maintenances according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
groupids	string/array	Return only maintenances that are assigned to the given host groups.
hostids	string/array	Return only maintenances that are assigned to the given hosts.
maintenanceids	string/array	Return only maintenances with the given IDs.
selectHostGroups	query	Return a hostgroups property with host groups assigned to the maintenance.
selectHosts	query	Return a hosts property with hosts assigned to the maintenance.
selectTags	query	Return a tags property with problem tags of the maintenance.
selectTimeperiods	query	Return a timeperiods property with time periods of the maintenance.
sortfield	string/array	Sort the result by the given properties. Possible values: `maintenanceid`, `name`, `maintenance_type`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Parameter	Type	Description
selectGroups (deprecated)	query	This parameter is deprecated, please use `selectHostGroups` instead. Return a groups property with host groups assigned to the maintenance.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving maintenances

Retrieve all configured maintenances, and the data about the assigned host groups, defined time periods and problem tags.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.get",
    "params": {
        "output": "extend",
        "selectHostGroups": "extend",
        "selectTimeperiods": "extend",
        "selectTags": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "maintenanceid": "3",
            "name": "Sunday maintenance",
            "maintenance_type": "0",
            "description": "",
            "active_since": "1358844540",
            "active_till": "1390466940",
            "tags_evaltype": "0",
            "hostgroups": [
                {
                    "groupid": "4",
                    "name": "Zabbix servers",
                    "flags": "0",
                    "uuid": "6f6799aa69e844b4b3918f779f2abf08"
                }
            ],
            "timeperiods": [
                {
                    "timeperiod_type": "3",
                    "every": "1",
                    "month": "0",
                    "dayofweek": "1",
                    "day": "0",
                    "start_time": "64800",
                    "period": "3600",
                    "start_date": "2147483647"
                }
            ],
```

```
            "tags": [
                {
                    "tag": "service",
                    "operator": "0",
                    "value": "mysqld",
                },
                {
                    "tag": "error",
                    "operator": "2",
                    "value": ""
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Host
- Host group
- Time period

Source

CMaintenance::get() in *ui/include/classes/api/services/CMaintenance.php*.

maintenance.update

Description

`object maintenance.update(object/array maintenances)`

This method allows to update existing maintenances.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Maintenance properties to be updated.

The `maintenanceid` property must be defined for each maintenance, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard maintenance properties, the method accepts the following parameters.

Parameter	Type	Description
groups	object/array	Host groups to replace the current groups. The host groups must have the `groupid` property defined. Parameter behavior: - *required* if `hosts` is not set
hosts	object/array	Hosts to replace the current hosts. The hosts must have only the `hostid` property defined. Parameter behavior: - *required* if `groups` is not set
timeperiods	object/array	Maintenance time periods to replace the current periods.

Parameter	Type	Description
tags	object/array	Problem tags to replace the current tags. Parameter behavior: - *supported* if `maintenance_type` of Maintenance object is set to "with data collection"
groupids (deprecated)	array	This parameter is deprecated, please use `groups` instead. IDs of the host groups that will undergo maintenance.
hostids (deprecated)	array	This parameter is deprecated, please use `hosts` instead. IDs of the hosts that will undergo maintenance.

Return values

`(object)` Returns an object containing the IDs of the updated maintenances under the `maintenanceids` property.

Examples

Assigning different hosts

Replace the hosts currently assigned to maintenance with two different ones.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "maintenance.update",
    "params": {
        "maintenanceid": "3",
        "hosts": [
            {"hostid": "10085"},
            {"hostid": "10084"}
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "maintenanceids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Time period

Source

CMaintenance::update() in *ui/include/classes/api/services/CMaintenance.php*.

Map

This class is designed to work with maps.

Object references:

- Map
- Map element
- Map link
- Map URL
- Map user

- Map user group
- Map shape
- Map line

Available methods:

- map.create - create new maps
- map.delete - delete maps
- map.get - retrieve maps
- map.update - update maps

> Map object

The following objects are directly related to the `map` API.

Map

The map object has the following properties.

Property	Type	Description
sysmapid	string	ID of the map. Property behavior: - *read-only* - *required* for update operations
height	integer	Height of the map in pixels. Property behavior: - *required* for create operations
name	string	Name of the map. Property behavior: - *required* for create operations
width	integer	Width of the map in pixels. Property behavior: - *required* for create operations
backgroundid	string	ID of the image used as the background for the map.
expand_macros	integer	Whether to expand macros in labels when configuring the map. Possible values: 0 - *(default)* do not expand macros; 1 - expand macros.
expandproblem	integer	Whether the problem trigger will be displayed for elements with a single problem. Possible values: 0 - always display the number of problems; 1 - *(default)* display the problem trigger if there's only one problem.
grid_align	integer	Whether to enable grid aligning. Possible values: 0 - disable grid aligning; 1 - *(default)* enable grid aligning.
grid_show	integer	Whether to show the grid on the map. Possible values: 0 - do not show the grid; 1 - *(default)* show the grid.

Property	Type	Description
grid_size	integer	Size of the map grid in pixels. Supported values: 20, 40, 50, 75 and 100. Default: 50.
highlight	integer	Whether icon highlighting is enabled. Possible values: 0 - highlighting disabled; 1 - *(default)* highlighting enabled.
iconmapid	string	ID of the icon map used on the map.
label_format	integer	Whether to enable advanced labels. Possible values: 0 - *(default)* disable advanced labels; 1 - enable advanced labels.
label_location	integer	Location of the map element label. Possible values: 0 - *(default)* bottom; 1 - left; 2 - right; 3 - top.
label_string_host	string	Custom label for host elements. Property behavior: - *required* if `label_type_host` is set to ”custom”
label_string_hostgroup	string	Custom label for host group elements. Property behavior: - *required* if `label_type_hostgroup` is set to ”custom”
label_string_image	string	Custom label for image elements. Property behavior: - *required* if `label_type_image` is set to ”custom”
label_string_map	string	Custom label for map elements. Property behavior: - *required* if `label_type_map` is set to ”custom”
label_string_trigger	string	Custom label for trigger elements. Property behavior: - *required* if `label_type_trigger` is set to ”custom”
label_type	integer	Map element label type. Possible values: 0 - label; 1 - IP address; 2 - *(default)* element name; 3 - status only; 4 - nothing.
label_type_host	integer	Label type for host elements. Possible values: 0 - label; 1 - IP address; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom.

Property	Type	Description
label_type_hostgroup	integer	Label type for host group elements. Possible values: 0 - label; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom.
label_type_image	integer	Label type for host group elements. Possible values: 0 - label; 2 - *(default)* element name; 4 - nothing; 5 - custom.
label_type_map	integer	Label type for map elements. Possible values: 0 - label; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom.
label_type_trigger	integer	Label type for trigger elements. Possible values: 0 - label; 2 - *(default)* element name; 3 - status only; 4 - nothing; 5 - custom.
markelements	integer	Whether to highlight map elements that have recently changed their status. Possible values: 0 - *(default)* do not highlight elements; 1 - highlight elements.
severity_min	integer	Minimum severity of the triggers that will be displayed on the map. Refer to the trigger `severity` property for a list of supported trigger severities.
show_unack	integer	How problems should be displayed. Possible values: 0 - *(default)* display the count of all problems; 1 - display only the count of unacknowledged problems; 2 - display the count of acknowledged and unacknowledged problems separately.
userid	string	Map owner user ID.
private	integer	Type of map sharing. Possible values: 0 - public map; 1 - *(default)* private map.
show_suppressed	integer	Whether suppressed problems are shown. Possible values: 0 - *(default)* hide suppressed problems; 1 - show suppressed problems.

Map element

The map element object defines an object displayed on a map. It has the following properties.

Property	Type	Description
selementid	string	ID of the map element. Property behavior: - *read-only*
elements	array	Element data object. Property behavior: - *required* if `elementtype` is set to ”host”, ”map”, ”trigger” or ”host group”
elementtype	integer	Type of map element. Possible values: 0 - host; 1 - map; 2 - trigger; 3 - host group; 4 - image. Property behavior: - *required*
iconid_off	string	ID of the image used to display the element in default state. Property behavior: - *required*
areatype	integer	How separate host group hosts should be displayed. Possible values: 0 - *(default)* the host group element will take up the whole map; 1 - the host group element will have a fixed size.
elementsubtype	integer	How a host group element should be displayed on a map. Possible values: 0 - *(default)* display the host group as a single element; 1 - display each host in the group separately.
evaltype	integer	Map element tag filtering condition evaluation method. Possible values: 0 - *(default)* AND / OR; 2 - OR.
height	integer	Height of the fixed size host group element in pixels. Default: 200.
iconid_disabled	string	ID of the image used to display disabled map elements. Property behavior: - *supported* if `elementtype` is set to ”host”, ”map”, ”trigger”, or ”host group”
iconid_maintenance	string	ID of the image used to display map elements in maintenance. Property behavior: - *supported* if `elementtype` is set to ”host”, ”map”, ”trigger”, or ”host group”
iconid_on	string	ID of the image used to display map elements with problems. Property behavior: - *supported* if `elementtype` is set to ”host”, ”map”, ”trigger”, or ”host group”
label	string	Label of the element.

Property	Type	Description
label_location	integer	Location of the map element label. Possible values: -1 - *(default)* default location; 0 - bottom; 1 - left; 2 - right; 3 - top.
permission	integer	Type of permission level. Possible values: -1 - none; 2 - read only; 3 - read-write.
sysmapid	string	ID of the map that the element belongs to. Property behavior: - *read-only*
urls	array	Map element URLs. The map element URL object is described in detail below.
use_iconmap	integer	Whether icon mapping must be used for host elements. Possible values: 0 - do not use icon mapping; 1 - *(default)* use icon mapping.
viewtype	integer	Host group element placing algorithm. Possible values: 0 - *(default)* grid.
width	integer	Width of the fixed size host group element in pixels. Default: 200.
x	integer	X-coordinates of the element in pixels. Default: 0.
y	integer	Y-coordinates of the element in pixels. Default: 0.

Map element Host

The map element Host object defines one host element.

Property	Type	Description
hostid	string	Host ID

Map element Host group

The map element Host group object defines one host group element.

Property	Type	Description
groupid	string	Host group ID

Map element Map

The map element Map object defines one map element.

Property	Type	Description
sysmapid	string	Map ID

Map element Trigger

The map element Trigger object defines one or more trigger elements.

Property	Type	Description
triggerid	string	Trigger ID

Map element tag

The map element tag object has the following properties.

Property	Type	Description
tag	string	Map element tag name. Property behavior: - *required*
operator	integer	Map element tag condition operator. Possible values: 0 - *(default)* Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist.
value	string	Map element tag value.

Map element URL

The map element URL object defines a clickable link that will be available for a specific map element. It has the following properties:

Property	Type	Description
sysmapelementurlid	string	ID of the map element URL. Property behavior: - *read-only*
name	string	Link caption. Property behavior: - *required*
url	string	Link URL. Property behavior: - *required*
selementid	string	ID of the map element that the URL belongs to.

Map link

The map link object defines a link between two map elements. It has the following properties.

Property	Type	Description
linkid	string	ID of the map link. Property behavior: - *read-only*

Property	Type	Description
selementid1	string	ID of the first map element linked on one end. Property behavior: - *required*
selementid2	string	ID of the first map element linked on the other end. Property behavior: - *required*
color	string	Line color as a hexadecimal color code. Default: 000000.
drawtype	integer	Link line draw style. Possible values: 0 - *(default)* line; 2 - bold line; 3 - dotted line; 4 - dashed line.
label	string	Link label.
linktriggers	array	Map link triggers to use as link status indicators. The map link trigger object is described in detail below.
permission	integer	Type of permission level. Possible values: -1 - none; 2 - read only; 3 - read-write.
sysmapid	string	ID of the map the link belongs to.

Map link trigger

The map link trigger object defines a map link status indicator based on the state of a trigger. It has the following properties:

Property	Type	Description
linktriggerid	string	ID of the map link trigger. Property behavior: - *read-only*
triggerid	string	ID of the trigger used as a link indicator. Property behavior: - *required*
color	string	Indicator color as a hexadecimal color code. Default: DD0000.
drawtype	integer	Indicator draw style. Possible values: 0 - *(default)* line; 2 - bold line; 3 - dotted line; 4 - dashed line.
linkid	string	ID of the map link that the link trigger belongs to.

Map URL

The map URL object defines a clickable link that will be available for all elements of a specific type on the map. It has the following properties:

Property	Type	Description
sysmapurlid	string	ID of the map URL. Property behavior: - *read-only*
name	string	Link caption. Property behavior: - *required*
url	string	Link URL. Property behavior: - *required*
elementtype	integer	Type of map element for which the URL will be available. Refer to the map element `type` property for a list of supported types. Default: 0.
sysmapid	string	ID of the map that the URL belongs to.

Map user

List of map permissions based on users. It has the following properties:

Property	Type	Description
sysmapuserid	string	ID of the map user. Property behavior: - *read-only*
userid	string	User ID. Property behavior: - *required*
permission	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write. Property behavior: - *required*

Map user group

List of map permissions based on user groups. It has the following properties:

Property	Type	Description
sysmapusrgrpid	string	ID of the map user group. Property behavior: - *read-only*
usrgrpid	string	User group ID. Property behavior: - *required*

Property	Type	Description
permission	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write. Property behavior: - *required*

Map shapes

The map shape object defines a geometric shape (with or without text) displayed on a map. It has the following properties:

Property	Type	Description
sysmap_shapeid	string	ID of the map shape element. Property behavior: - *read-only*
type	integer	Type of map shape element. Possible values: 0 - rectangle; 1 - ellipse. Property is required when new shapes are created. Property behavior: - *required*
x	integer	X-coordinates of the shape in pixels. Default: 0.
y	integer	Y-coordinates of the shape in pixels. Default: 0.
width	integer	Width of the shape in pixels. Default: 200.
height	integer	Height of the shape in pixels. Default: 200.
text	string	Text of the shape.
font	integer	Font of the text within shape. Possible values: 0 - Georgia, serif 1 - “Palatino Linotype”, “Book Antiqua”, Palatino, serif 2 - “Times New Roman”, Times, serif 3 - Arial, Helvetica, sans-serif 4 - “Arial Black”, Gadget, sans-serif 5 - “Comic Sans MS”, cursive, sans-serif 6 - Impact, Charcoal, sans-serif 7 - “Lucida Sans Unicode”, “Lucida Grande”, sans-serif 8 - Tahoma, Geneva, sans-serif 9 - “Trebuchet MS”, Helvetica, sans-serif 10 - Verdana, Geneva, sans-serif 11 - “Courier New”, Courier, monospace 12 - “Lucida Console”, Monaco, monospace Default: 9.

Property	Type	Description
font_size	integer	Font size in pixels. Default: 11.
font_color	string	Font color. Default: 000000.
text_halign	integer	Horizontal alignment of text. Possible values: 0 - center; 1 - left; 2 - right. Default: 0.
text_valign	integer	Vertical alignment of text. Possible values: 0 - middle; 1 - top; 2 - bottom. Default: 0.
border_type	integer	Type of the border. Possible values: 0 - none; 1 - ——————; 2 - ┅┅; 3 - – – –. Default: 0.
border_width	integer	Width of the border in pixels. Default: 0.
border_color	string	Border color. Default: 000000.
background_color	string	Background color (fill color). Default: `(empty)`.
zindex	integer	Value used to order all shapes and lines (z-index). Default: 0.

Map lines

The map line object defines a line displayed on a map. It has the following properties:

Property	Type	Description
sysmap_shapeid	string	ID of the map shape element. Property behavior: - *read-only*
x1	integer	X-coordinates of the line point 1 in pixels. Default: 0.
y1	integer	Y-coordinates of the line point 1 in pixels. Default: 0.

Property	Type	Description
x2	integer	X-coordinates of the line point 2 in pixels. Default: 200.
y2	integer	Y-coordinates of the line point 2 in pixels. Default: 200.
line_type	integer	Type of the lines. Possible values: 0 - none; 1 - ——————; 2 - ┅┅; 3 - – – –. Default: 0.
line_width	integer	Width of the lines in pixels. Default: 0.
line_color	string	Line color. Default: 000000.
zindex	integer	Value used to order all shapes and lines (z-index). Default: 0.

map.create

Description

`object map.create(object/array maps)`

This method allows to create new maps.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Maps to create.

Additionally to the standard map properties, the method accepts the following parameters.

Parameter	Type	Description
links	array	Map links to be created on the map.
selements	array	Map elements to be created on the map.
urls	array	Map URLs to be created on the map.
users	array	Map user shares to be created on the map.
userGroups	array	Map user group shares to be created on the map.
shapes	array	Map shapes to be created on the map.
lines	array	Map lines to be created on the map.

Note:
To create map links you'll need to set a map element `selementid` to an arbitrary value and then use this value to reference this element in the links `selementid1` or `selementid2` properties. When the element is created, this value will be replaced with the correct ID generated by Zabbix. See example.

Return values

`(object)` Returns an object containing the IDs of the created maps under the `sysmapids` property. The order of the returned IDs matches the order of the passed maps.

Examples

Create an empty map

Create a map with no elements.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map",
        "width": 600,
        "height": 600
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "8"
        ]
    },
    "id": 1
}
```

Create a host map

Create a map with two host elements and a link between them. Note the use of temporary "selementid1" and "selementid2" values in the map link object to refer to map elements.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Host map",
        "width": 600,
        "height": 600,
        "selements": [
            {
                "selementid": "1",
                "elements": [
                    {"hostid": "1033"}
                ],
                "elementtype": 0,
                "iconid_off": "2"
            },

            {
                "selementid": "2",
                "elements": [
                    {"hostid": "1037"}
                ],
                "elementtype": 0,
                "iconid_off": "2"
            }
        ],
        "links": [
            {
                "selementid1": "1",
                "selementid2": "2"
```

```
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

Create a trigger map

Create a map with trigger element, which contains two triggers.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Trigger map",
        "width": 600,
        "height": 600,
        "selements": [
            {
                "elements": [
                    {"triggerid": "12345"},
                    {"triggerid": "67890"}
                ],
                "elementtype": 2,
                "iconid_off": "2"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "10"
        ]
    },
    "id": 1
}
```

Map sharing

Create a map with two types of sharing (user and user group).

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map sharing",
```

```
        "width": 600,
        "height": 600,
        "users": [
            {
                "userid": "4",
                "permission": "3"
            }
        ],
        "userGroups": [
            {
                "usrgrpid": "7",
                "permission": "2"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

Map shapes

Create a map with map name title.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Host map",
        "width": 600,
        "height": 600,
        "shapes": [
            {
                "type": 0,
                "x": 0,
                "y": 0,
                "width": 600,
                "height": 11,
                "text": "{MAP.NAME}"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "10"
        ]
    },
```

```
    "id": 1
}
```

Map lines

Create a map line.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.create",
    "params": {
        "name": "Map API lines",
        "width": 500,
        "height": 500,
        "lines": [
            {
                "x1": 30,
                "y1": 10,
                "x2": 100,
                "y2": 50,
                "line_type": 1,
                "line_width": 10,
                "line_color": "009900"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "11"
        ]
    },
    "id": 1
}
```

See also

- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shape
- Map line

Source

CMap::create() in *ui/include/classes/api/services/CMap.php*.

map.delete

Description

`object map.delete(array mapIds)`

This method allows to delete maps.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the maps to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted maps under the `sysmapids` property.

Examples

Delete multiple maps

Delete two maps.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.delete",
    "params": [
        "12",
        "34"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "12",
            "34"
        ]
    },
    "id": 1
}
```

Source

CMap::delete() in *ui/include/classes/api/services/CMap.php*.

map.get

Description

`integer/array map.get(object parameters)`

The method allows to retrieve maps according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
sysmapids	string/array	Returns only maps with the given IDs.
userids	string/array	Returns only maps that belong to the given user IDs.

Parameter	Type	Description
expandUrls	flag	Adds global map URLs to the corresponding map elements and expands macros in all map element URLs.
selectIconMap	query	Returns an iconmap property with the icon map used on the map.
selectLinks	query	Returns a links property with the map links between elements.
selectSelements	query	Returns a selements property with the map elements.
selectUrls	query	Returns a urls property with the map URLs.
selectUsers	query	Returns a users property with users that the map is shared with.
selectUserGroups	query	Returns a userGroups property with user groups that the map is shared with.
selectShapes	query	Returns a shapes property with the map shapes.
selectLines	query	Returns a lines property with the map lines.
sortfield	string/array	Sort the result by the given properties. Possible values: `name`, `width`, `height`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve a map

Retrieve all data about map "3".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.get",
    "params": {
        "output": "extend",
        "selectSelements": "extend",
        "selectLinks": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "selectShapes": "extend",
        "selectLines": "extend",
        "sysmapids": "3"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
```

```
        "selements": [
            {
                "selementid": "10",
                "sysmapid": "3",
                "elementtype": "4",
                "evaltype": "0",
                "iconid_off": "1",
                "iconid_on": "0",
                "label": "Zabbix server",
                "label_location": "3",
                "x": "11",
                "y": "141",
                "iconid_disabled": "0",
                "iconid_maintenance": "0",
                "elementsubtype": "0",
                "areatype": "0",
                "width": "200",
                "height": "200",
                "tags": [
                    {
                        "tag": "service",
                        "value": "mysqld",
                        "operator": "0"
                    }
                ],
                "viewtype": "0",
                "use_iconmap": "1",
                "urls": [],
                "elements": []
            },
            {
                "selementid": "11",
                "sysmapid": "3",
                "elementtype": "4",
                "evaltype": "0",
                "iconid_off": "1",
                "iconid_on": "0",
                "label": "Web server",
                "label_location": "3",
                "x": "211",
                "y": "191",
                "iconid_disabled": "0",
                "iconid_maintenance": "0",
                "elementsubtype": "0",
                "areatype": "0",
                "width": "200",
                "height": "200",
                "viewtype": "0",
                "use_iconmap": "1",
                "tags": [],
                "urls": [],
                "elements": []
            },
            {
                "selementid": "12",
                "sysmapid": "3",
                "elementtype": "0",
                "evaltype": "0",
                "iconid_off": "185",
                "iconid_on": "0",
                "label": "{HOST.NAME}\r\n{HOST.CONN}",
                "label_location": "0",
```

```
                "x": "111",
                "y": "61",
                "iconid_disabled": "0",
                "iconid_maintenance": "0",
                "elementsubtype": "0",
                "areatype": "0",
                "width": "200",
                "height": "200",
                "viewtype": "0",
                "use_iconmap": "0",
                "tags": [],
                "urls": [],
                "elements": [
                    {
                        "hostid": "10084"
                    }
                ]
            }
        ],
        "links": [
            {
                "linkid": "23",
                "sysmapid": "3",
                "selementid1": "10",
                "selementid2": "11",
                "drawtype": "0",
                "color": "00CC00",
                "label": "",
                "linktriggers": []
            }
        ],
        "users": [
            {
                "sysmapuserid": "1",
                "userid": "2",
                "permission": "2"
            }
        ],
        "userGroups": [
            {
                "sysmapusrgrpid": "1",
                "usrgrpid": "7",
                "permission": "2"
            }
        ],
        "shapes":[
            {
                "sysmap_shapeid":"1",
                "type":"0",
                "x":"0",
                "y":"0",
                "width":"680",
                "height":"15",
                "text":"{MAP.NAME}",
                "font":"9",
                "font_size":"11",
                "font_color":"000000",
                "text_halign":"0",
                "text_valign":"0",
                "border_type":"0",
                "border_width":"0",
                "border_color":"000000",
```

```
                    "background_color":"",
                    "zindex":"0"
                }
            ],
            "lines":[
                {
                    "sysmap_shapeid":"2",
                    "x1": 30,
                    "y1": 10,
                    "x2": 100,
                    "y2": 50,
                    "line_type": 1,
                    "line_width": 10,
                    "line_color": "009900",
                    "zindex":"1"
                }
            ],
            "sysmapid": "3",
            "name": "Local network",
            "width": "400",
            "height": "400",
            "backgroundid": "0",
            "label_type": "2",
            "label_location": "3",
            "highlight": "1",
            "expandproblem": "1",
            "markelements": "0",
            "show_unack": "0",
            "grid_size": "50",
            "grid_show": "1",
            "grid_align": "1",
            "label_format": "0",
            "label_type_host": "2",
            "label_type_hostgroup": "2",
            "label_type_trigger": "2",
            "label_type_map": "2",
            "label_type_image": "2",
            "label_string_host": "",
            "label_string_hostgroup": "",
            "label_string_trigger": "",
            "label_string_map": "",
            "label_string_image": "",
            "iconmapid": "0",
            "expand_macros": "0",
            "severity_min": "0",
            "userid": "1",
            "private": "1",
            "show_suppressed": "1"
        }
    ],
    "id": 1
}
```


See also

- Icon map
- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shapes
- Map lines

Source

CMap::get() in *ui/include/classes/api/services/CMap.php*.

map.update

Description

`object map.update(object/array maps)`

This method allows to update existing maps.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Map properties to be updated.

The `mapid` property must be defined for each map, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard map properties, the method accepts the following parameters.

Parameter	Type	Description
links	array	Map links to replace the existing links.
selements	array	Map elements to replace the existing elements.
urls	array	Map URLs to replace the existing URLs.
users	array	Map user shares to replace the existing elements.
userGroups	array	Map user group shares to replace the existing elements.
shapes	array	Map shapes to replace the existing shapes.
lines	array	Map lines to replace the existing lines.

> **Note:**
> To create map links between new map elements you'll need to set an element's `selementid` to an arbitrary value and then use this value to reference this element in the links `selementid1` or `selementid2` properties. When the element is created, this value will be replaced with the correct ID generated by Zabbix. See example for map.create.

Return values

`(object)` Returns an object containing the IDs of the updated maps under the `sysmapids` property.

Examples

Resize a map

Change the size of the map to 1200x1200 pixels.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.update",
    "params": {
        "sysmapid": "8",
        "width": 1200,
        "height": 1200
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
```

```
        "sysmapids": [
            "8"
        ]
    },
    "id": 1
}
```

Change map owner

Available only for admins and super admins.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "map.update",
    "params": {
        "sysmapid": "9",
        "userid": "1"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 1
}
```

See also

- Map element
- Map link
- Map URL
- Map user
- Map user group
- Map shapes
- Map lines

Source

CMap::update() in *ui/include/classes/api/services/CMap.php*.

Media type

This class is designed to work with media types.

Object references:

- Media type

Available methods:

- mediatype.create - creating new media types
- mediatype.delete - deleting media types
- mediatype.get - retrieving media types
- mediatype.update - updating media types

> Media type object

The following objects are directly related to the `mediatype` API.

Media type

The media type object has the following properties.

Property	Type	Description
mediatypeid	string	ID of the media type. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the media type. Property behavior: - *required* for create operations
type	integer	Transport used by the media type. Possible values: 0 - Email; 1 - Script; 2 - SMS; 4 - Webhook. Property behavior: - *required* for create operations
exec_path	string	For script media types `exec_path` contains the name of the executed script. Property behavior: - *required* if `type` is set to ”Script”
gsm_modem	string	Serial device name of the GSM modem. Property behavior: - *required* if `type` is set to ”SMS”
passwd	string	Authentication password. Property behavior: - *supported* if `type` is set to ”Email”
provider	integer	Email provider. Possible values: 0 - *(default)* Generic SMTP; 1 - Gmail; 2 - Gmail relay; 3 - Office365; 4 - Office365 relay. Property behavior: - *supported* if `type` is set to ”Email”
smtp_email	string	Email address from which notifications will be sent. Property behavior: - *required* if `type` is set to ”Email”
smtp_helo	string	SMTP HELO. Property behavior: - *required* if `type` is set to ”Email”
smtp_server	string	SMTP server. Property behavior: - *required* if `type` is set to ”Email”
smtp_port	integer	SMTP server port to connect to.

Property	Type	Description
smtp_security	integer	SMTP connection security level to use. Possible values: 0 - None; 1 - STARTTLS; 2 - SSL/TLS.
smtp_verify_host	integer	SSL verify host for SMTP. Possible values: 0 - No; 1 - Yes.
smtp_verify_peer	integer	SSL verify peer for SMTP. Possible values: 0 - No; 1 - Yes.
smtp_authentication	integer	SMTP authentication method to use. Possible values: 0 - None; 1 - Normal password.
status	integer	Whether the media type is enabled. Possible values: 0 - *(default)* Enabled; 1 - Disabled.
username	string	User name. Property behavior: - *supported* if `type` is set to ”Email”
maxsessions	integer	The maximum number of alerts that can be processed in parallel. Possible values if `type` is set to ”SMS”: *(default)* 1. Possible values if `type` is set to ”Email”, ”Script”, or ”Webhook”: 0-100.
maxattempts	integer	The maximum number of attempts to send an alert. Possible values: 1-100. Default value: 3.
attempt_interval	string	The interval between retry attempts. Accepts seconds and time unit with suffix. Possible values: 0-1h. Default value: 10s.
content_type	integer	Message format. Possible values: 0 - plain text; 1 - *(default)* html.
script	text	Media type webhook script javascript body.
timeout	string	Media type webhook script timeout. Accepts seconds and time unit with suffix. Possible values: 1-60s. Default: 30s.

Property	Type	Description
process_tags	integer	Defines should the webhook script response to be interpreted as tags and these tags should be added to associated event. Possible values: 0 - *(default)* Ignore webhook script response; 1 - Process webhook script response as tags.
show_event_menu	integer	Show media type entry in `problem.get` and `event.get` property `urls`. Possible values: 0 - *(default)* Do not add `urls` entry; 1 - Add media type to `urls` property.
event_menu_url	string	Define `url` property of media type entry in `urls` property of `problem.get` and `event.get`.
event_menu_name	string	Define `name` property of media type entry in `urls` property of `problem.get` and `event.get`.
parameters	array	Array of webhook or script input parameters.
description	text	Media type description.

Webhook parameters

Parameters passed to a webhook script when it is being called have the following properties.

Property	Type	Description
name	string	Parameter name. Property behavior: - *required*
value	string	Parameter value, supports macros. Supported macros are described on the Supported macros page.

Script parameters

Parameters passed to a script when it is being called have the following properties.

Property	Type	Description
sortorder	integer	The order in which the parameters will be passed to the script as command-line arguments, starting with 0 as the first one. Property behavior: - *required*
value	string	Parameter value, supports macros. Supported macros are described on the Supported macros page.

Message template

The message template object defines a template that will be used as a default message for action operations to send a notification. It has the following properties.

Property	Type	Description
eventsource	integer	Event source. Possible values: 0 - triggers; 1 - discovery; 2 - autoregistration; 3 - internal; 4 - services. Property behavior: - *required*
recovery	integer	Operation mode. Possible values: 0 - operations; 1 - recovery operations; 2 - update operations. Property behavior: - *required*
subject	string	Message subject.
message	string	Message text.

mediatype.create

Description

`object mediatype.create(object/array mediaTypes)`

This method allows to create new media types.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Media types to create.

Additionally to the standard media type properties, the method accepts the following parameters.

Parameter	Type	Description
parameters	array	Script or webhook parameters to be created for the media type.
message_templates	array	Message templates to be created for the media type.

Return values

`(object)` Returns an object containing the IDs of the created media types under the `mediatypeids` property. The order of the returned IDs matches the order of the passed media types.

Examples

Creating an email media type

Create a new email media type with a custom SMTP port and message templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.create",
    "params": {
        "type": "0",
```

```
        "name": "Email",
        "smtp_server": "mail.example.com",
        "smtp_helo": "example.com",
        "smtp_email": "zabbix@example.com",
        "smtp_port": "587",
        "content_type": "1",
        "message_templates": [
            {
                "eventsource": "0",
                "recovery": "0",
                "subject": "Problem: {EVENT.NAME}",
                "message": "Problem \"{EVENT.NAME}\" on host \"{HOST.NAME}\" started at {EVENT.TIME}."
            },
            {
                "eventsource": "0",
                "recovery": "1",
                "subject": "Resolved in {EVENT.DURATION}: {EVENT.NAME}",
                "message": "Problem \"{EVENT.NAME}\" on host \"{HOST.NAME}\" has been resolved at {EVENT.R
            },
            {
                "eventsource": "0",
                "recovery": "2",
                "subject": "Updated problem in {EVENT.AGE}: {EVENT.NAME}",
                "message": "{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem \"{EVENT.NAME}\" on host \"{HOST
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "7"
        ]
    },
    "id": 1
}
```

Creating a script media type

Create a new script media type with a custom value for the number of attempts and the interval between them.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.create",
    "params": {
        "type": "1",
        "name": "Push notifications",
        "exec_path": "push-notification.sh",
        "maxattempts": "5",
        "attempt_interval": "11s",
        "parameters": [
            {
                "sortorder": "0",
                "value": "{ALERT.SENDTO}"
            },
            {
                "sortorder": "1",
                "value": "{ALERT.SUBJECT}"
```

```
            },
            {
                "sortorder": "2",
                "value": "{ALERT.MESSAGE}"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "8"
        ]
    },
    "id": 1
}
```

Creating a webhook media type

Create a new webhook media type.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.create",
    "params": {
        "type": "4",
        "name": "Webhook",
        "script": "var Webhook = {\r\n    token: null,\r\n    to: null,\r\n    subject: null,\r\n    messa
        "parameters": [
            {
                "name": "Message",
                "value": "{ALERT.MESSAGE}"
            },
            {
                "name": "Subject",
                "value": "{ALERT.SUBJECT}"
            },
            {
                "name": "To",
                "value": "{ALERT.SENDTO}"
            },
            {
                "name": "Token",
                "value": "<Token>"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "9"
        ]
    },
```

```
    "id": 1
}
```

Source

CMediaType::create() in *ui/include/classes/api/services/CMediaType.php*.

mediatype.delete

Description

`object mediatype.delete(array mediaTypeIds)`

This method allows to delete media types.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the media types to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted media types under the `mediatypeids` property.

Examples

Deleting multiple media types

Delete two media types.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.delete",
    "params": [
        "3",
        "5"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "3",
            "5"
        ]
    },
    "id": 1
}
```

Source

CMediaType::delete() in *ui/include/classes/api/services/CMediaType.php*.

mediatype.get

Description

`integer/array mediatype.get(object parameters)`

The method allows to retrieve media types according to the given parameters.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Note:
Since Zabbix 6.4.19, when requesting user-related information of media types, *Admin* type users may retrieve only data about their own user. For an example, see Retrieving media types as *Admin*.

Parameter	Type	Description
mediatypeids	string/array	Return only media types with the given IDs.
mediaids	string/array	Return only media types used by the given media.
userids	string/array	Return only media types used by the given users.
selectMessageTemplates	query	Return a message_templates property with an array of media type messages. Parameter behavior: - *supported* for *Super admin* type users (since Zabbix 6.4.19)
selectUsers	query	Return a users property with the users that use the media type.
sortfield	string/array	Sort the result by the given properties. Possible values: `mediatypeid`.
filter	object	Return only those results that exactly match the given filter. Accepts an object, where the keys are property names, and the values are either a single value or an array of values to match against. Supported properties for *Super admin* type users: all Media type object properties, except properties of `text` data type. Supported properties for *Admin* type users (since Zabbix 6.4.19): `mediatypeid`, `name`, `type`, `status`, `maxattempts`.
output	query	Media type object properties to be returned. Since Zabbix 6.4.19, *Admin* type users may retrieve only the following Media type object properties: `mediatypeid`, `name`, `type`, `status`, `maxattempts`. For an example, see Retrieving media types as *Admin*. Default: `extend`.
search	object	Return results that match the given pattern (case-insensitive). Accepts an object, where the keys are property names, and the values are strings to search for. If no additional options are given, this will perform a `LIKE "%…%"` search. Supported properties for *Super admin* type users: all Media type object properties of `string` and `text` data type. Supported properties for *Admin* type users (since Zabbix 6.4.19): `name`, `description`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
limit	integer	
preservekeys	boolean	
searchByAny	boolean	

Parameter	Type	Description
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving media types

Retrieve all configured media types. The following example returns two media types:

- email media type;
- SMS media type.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.get",
    "params": {
        "output": "extend",
        "selectMessageTemplates": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mediatypeid": "1",
            "type": "0",
            "name": "Email",
            "smtp_server": "mail.example.com",
            "smtp_helo": "example.com",
            "smtp_email": "zabbix@example.com",
            "exec_path": "",
            "gsm_modem": "",
            "username": "",
            "passwd": "",
            "status": "0",
            "smtp_port": "25",
            "smtp_security": "0",
            "smtp_verify_peer": "0",
            "smtp_verify_host": "0",
            "smtp_authentication": "0",
            "maxsessions": "1",
            "maxattempts": "3",
            "attempt_interval": "10s",
            "content_type": "0",
            "script": "",
            "timeout": "30s",
            "process_tags": "0",
            "show_event_menu": "1",
            "event_menu_url": "",
            "event_menu_name": "",
            "description": "",
```

```
            "provider": "0",
            "message_templates": [
                {
                    "eventsource": "0",
                    "recovery": "0",
                    "subject": "Problem: {EVENT.NAME}",
                    "message": "Problem started at {EVENT.TIME} on {EVENT.DATE}\r\nProblem name: {EVENT.NA
                },
                {
                    "eventsource": "0",
                    "recovery": "1",
                    "subject": "Resolved: {EVENT.NAME}",
                    "message": "Problem has been resolved at {EVENT.RECOVERY.TIME} on {EVENT.RECOVERY.DATE
                },
                {
                    "eventsource": "0",
                    "recovery": "2",
                    "subject": "Updated problem: {EVENT.NAME}",
                    "message": "{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem at {EVENT.UPDATE.DATE} {EVEN
                },
                {
                    "eventsource": "1",
                    "recovery": "0",
                    "subject": "Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}",
                    "message": "Discovery rule: {DISCOVERY.RULE.NAME}\r\n\r\nDevice IP: {DISCOVERY.DEVICE.
                },
                {
                    "eventsource": "2",
                    "recovery": "0",
                    "subject": "Autoregistration: {HOST.HOST}",
                    "message": "Host name: {HOST.HOST}\r\nHost IP: {HOST.IP}\r\nAgent port: {HOST.PORT}"
                }
            ],
            "parameters": []
        },
        {
            "mediatypeid": "3",
            "type": "2",
            "name": "SMS",
            "smtp_server": "",
            "smtp_helo": "",
            "smtp_email": "",
            "exec_path": "",
            "gsm_modem": "/dev/ttyS0",
            "username": "",
            "passwd": "",
            "status": "0",
            "smtp_port": "25",
            "smtp_security": "0",
            "smtp_verify_peer": "0",
            "smtp_verify_host": "0",
            "smtp_authentication": "0",
            "maxsessions": "1",
            "maxattempts": "3",
            "attempt_interval": "10s",
            "content_type": "1",
            "script": "",
            "timeout": "30s",
            "process_tags": "0",
            "show_event_menu": "1",
            "event_menu_url": "",
            "event_menu_name": "",
```

```
            "description": "",
            "provider": "0",
            "message_templates": [
                {
                    "eventsource": "0",
                    "recovery": "0",
                    "subject": "",
                    "message": "{EVENT.SEVERITY}: {EVENT.NAME}\r\nHost: {HOST.NAME}\r\n{EVENT.DATE} {EVENT
                },
                {
                    "eventsource": "0",
                    "recovery": "1",
                    "subject": "",
                    "message": "RESOLVED: {EVENT.NAME}\r\nHost: {HOST.NAME}\r\n{EVENT.DATE} {EVENT.TIME}"
                },
                {
                    "eventsource": "0",
                    "recovery": "2",
                    "subject": "",
                    "message": "{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem at {EVENT.UPDATE.DATE} {EVEN
                },
                {
                    "eventsource": "1",
                    "recovery": "0",
                    "subject": "",
                    "message": "Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPADDRESS}"
                },
                {
                    "eventsource": "2",
                    "recovery": "0",
                    "subject": "",
                    "message": "Autoregistration: {HOST.HOST}\r\nHost IP: {HOST.IP}\r\nAgent port: {HOST.P
                }
            ],
            "parameters": []
        }
    ],
    "id": 1
}
```


Retrieving media types as *Admin*

As an *Admin* type user, retrieve all media types that are enabled, with users that use these media types. The following example returns two media types:

- email media type with one user (since Zabbix 6.4.19, only *Admin* type user's own user);
- SMS media type with no users.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "mediatype.get",
    "params": {
        "output": "extend",
        "filter": {
            "status": 0
        },
        "selectUsers": "extend"
    },
    "id": 1
}
```


Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mediatypeid": "1",
            "type": "0",
            "name": "Email",
            "status": "0",
            "description": "",
            "maxattempts": "3",
            "users": [
                {
                    "userid": "3",
                    "username": "database-admin",
                    "name": "John",
                    "surname": "Doe",
                    "url": "",
                    "autologin": "0",
                    "autologout": "0",
                    "lang": "default",
                    "refresh": "30s",
                    "theme": "default",
                    "attempt_failed": "0",
                    "attempt_ip": "",
                    "attempt_clock": "0",
                    "rows_per_page": "50",
                    "timezone": "default",
                    "roleid": "2",
                    "userdirectoryid": "0",
                    "ts_provisioned": "0"
                }
            ]
        },
        {
            "mediatypeid": "3",
            "type": "2",
            "name": "SMS",
            "status": "0",
            "description": "",
            "maxattempts": "3",
            "users": []
        }
    ],
    "id": 1
}
```


Retrieve script and webhook media types

The following example returns three media types:

- script media type with parameters;
- script media type without parameters;
- webhook media type with parameters.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "mediatype.get",
    "params": {
        "output": ["mediatypeid", "name", "parameters"],
        "filter": {
            "type": [1, 4]
        }
    },
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "mediatypeid": "10",
            "name": "Script with parameters",
            "parameters": [
                {
                    "sortorder": "0",
                    "value": "{ALERT.SENDTO}"
                },
                {
                    "sortorder": "1",
                    "value": "{EVENT.NAME}"
                },
                {
                    "sortorder": "2",
                    "value": "{ALERT.MESSAGE}"
                },
                {
                    "sortorder": "3",
                    "value": "Zabbix alert"
                }
            ]
        },
        {
            "mediatypeid": "13",
            "name": "Script without parameters",
            "parameters": []
        },
        {
            "mediatypeid": "11",
            "name": "Webhook",
            "parameters": [
                {
                    "name": "alert_message",
                    "value": "{ALERT.MESSAGE}"
                },
                {
                    "name": "event_update_message",
                    "value": "{EVENT.UPDATE.MESSAGE}"
                },
                {
                    "name": "host_name",
                    "value": "{HOST.NAME}"
                },
                {
                    "name": "trigger_description",
                    "value": "{TRIGGER.DESCRIPTION}"
                },
                {
                    "name": "trigger_id",
                    "value": "{TRIGGER.ID}"
                },
                {
                    "name": "alert_source",
                    "value": "Zabbix"
                }
```

```
            ]
        }
    ],
    "id": 1
}
```

See also

- User

Source

CMediaType::get() in *ui/include/classes/api/services/CMediaType.php*.

mediatype.update

Description

`object mediatype.update(object/array mediaTypes)`

This method allows to update existing media types.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Media type properties to be updated.

The `mediatypeid` property must be defined for each media type, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard media type properties, the method accepts the following parameters.

Parameter	Type	Description
parameters	array	Script or webhook parameters to replace the current parameters.
message_templates	array	Message templates to replace the current message templates.

Return values

`(object)` Returns an object containing the IDs of the updated media types under the `mediatypeids` property.

Examples

Enabling a media type

Enable a media type, that is, set its status to ”0”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.update",
    "params": {
        "mediatypeid": "6",
        "status": "0"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "6"
```

```
        ]
    },
    "id": 1
}
```

Source

CMediaType::update() in *ui/include/classes/api/services/CMediaType.php*.

Module

This class is designed to work with frontend modules.

Object references:

- Module

Available methods:

- module.create - installing new modules
- module.delete - uninstalling modules
- module.get - retrieving modules
- module.update - updating modules

> Module object

The following objects are directly related to the `module` API.

Module

The module object has the following properties.

Property	Type	Description
moduleid	string	ID of the module as stored in the database. Property behavior: - *read-only* - *required* for update operations
id	string	Unique module ID as defined by a developer in the manifest.json file of the module. Possible values for built-in modules: see property ”type” description in Dashboard widget. Property behavior: - *required* for create operations
relative_path	string	Path to the directory of the module relative to the directory of the Zabbix frontend. Possible values: `widgets/*` - for built-in widget modules; `modules/*` - for third-party modules. Property behavior: - *required* for create operations
status	integer	Whether the module is enabled or disabled. Possible values: 0 - *(default)* Disabled; 1 - Enabled.
config	object	Module configuration.

module.create

Description

`object module.create(object/array modules)`

This method allows to install new frontend modules.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

> **Attention:**
> Module files must be unpacked manually in the correct subdirectories, matching the `relative_path` property of the modules.

Parameters

`(object/array)` Modules to create.

The method accepts modules with the standard module properties.

Return values

`(object)` Returns an object containing the IDs of the installed modules under the `moduleids` property. The order of the returned IDs matches the order of the passed modules.

Examples

Installing a module

Install a module with the status "Enabled".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "module.create",
    "params": {
        "id": "example_module",
        "relative_path": "modules/example_module",
        "status": 1
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "moduleids": [
            "25"
        ]
    },
    "id": 1
}
```

See also

- Module
- Frontend modules

Source

CModule::create() in *ui/include/classes/api/services/CModule.php*.

module.delete

Description

`object module.delete(array moduleids)`

This method allows to uninstall modules.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

> **Attention:**
> Module files must be removed manually.

Parameters

`(array)` IDs of the modules to uninstall.

Return values

`(object)` Returns an object containing the IDs of the uninstalled modules under the `moduleids` property.

Examples

Uninstalling multiple modules

Uninstall modules "27" and "28".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "module.delete",
    "params": [
        "27",
        "28"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "moduleids": [
            "27",
            "28"
        ]
    },
    "id": 1
}
```

Source

CModule::delete() in *ui/include/classes/api/services/CModule.php*.

module.get

Description

`integer/array module.get(object parameters)`

The method allows to retrieve modules according to the given parameters.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
moduleids	string/array	Return only modules with the given IDs.
sortfield	string/array	Sort the result by the given properties. Possible values: `moduleid`, `relative_path`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the Reference commentary page.
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving a module by ID

Retrieve all data about modules "1", "2", and "25".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "module.get",
    "params": {
        "output": "extend",
        "moduleids": [
            "1",
            "2",
            "25"
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "moduleid": "1",
            "id": "actionlog",
            "relative_path": "widgets/actionlog",
```

```
            "status": "1",
            "config": []
        },
        {
            "moduleid": "2",
            "id": "clock",
            "relative_path": "widgets/clock",
            "status": "1",
            "config": []
        },
        {
            "moduleid": "25",
            "id": "example",
            "relative_path": "modules/example_module",
            "status": "1",
            "config": []
        }
    ],
    "id": 1
}
```

See also

- Module
- Dashboard widget
- Frontend modules

Source

CModule::get() in *ui/include/classes/api/services/CModule.php*.

module.update

Description

`object module.update(object/array modules)`

This method allows to update existing modules.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Module properties to be updated.

The `moduleid` property must be defined for each module, all other properties are optional. Only the specified properties will be updated.

The method accepts modules with the standard module properties.

Return values

`(object)` Returns an object containing the IDs of the updated modules under the `moduleids` property.

Examples

Disabling a module

Disable module "25".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "module.update",
    "params": {
        "moduleid": "25",
```

```
        "status": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "moduleids": [
            "25"
        ]
    },
    "id": 1
}
```

See also

- Module
- Frontend modules

Source

CModule::update() in *ui/include/classes/api/services/CModule.php*.

Problem

This class is designed to work with problems.

Object references:

- Problem

Available methods:

- problem.get - retrieving problems

> Problem object

The following objects are directly related to the `problem` API.

Problem

Note:
Problems are created by the Zabbix server and cannot be modified via the API.

The problem object has the following properties.

Property	Type	Description
eventid	string	ID of the problem event.
source	integer	Type of the problem event. Possible values: 0 - event created by a trigger; 3 - internal event; 4 - event created on service status update.

Property	Type	Description
object	integer	Type of object that is related to the problem event. Possible values if `source` is set to ”event created by a trigger”: 0 - trigger. Possible values if `source` is set to ”internal event”: 0 - trigger; 4 - item; 5 - LLD rule. Possible values if `source` is set to ”event created on service status update”: 6 - service.
objectid	string	ID of the related object.
clock	timestamp	Time when the problem event was created.
ns	integer	Nanoseconds when the problem event was created.
r_eventid	string	Recovery event ID.
r_clock	timestamp	Time when the recovery event was created.
r_ns	integer	Nanoseconds when the recovery event was created.
cause_eventid	string	Cause event ID.
correlationid	string	Correlation rule ID if this event was recovered by global correlation rule.
userid	string	User ID if the problem was manually closed.
name	string	Resolved problem name.
acknowledged	integer	Acknowledge state for problem. Possible values: 0 - not acknowledged; 1 - acknowledged.
severity	integer	Problem current severity. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
suppressed	integer	Whether the problem is suppressed. Possible values: 0 - problem is in normal state; 1 - problem is suppressed.
opdata	string	Operational data with expanded macros.
urls	array	Active media type URLs.

Problem tag

The problem tag object has the following properties.

Property	Type	Description
tag	string	Problem tag name.
value	string	Problem tag value.

Media type URL

The media type URL object has the following properties.

Property	Type	Description
name	string	Media type defined URL name.
url	string	Media type defined URL value.

Results will contain entries only for active media types with enabled event menu entry. Macro used in properties will be expanded, but if one of the properties contains an unexpanded macro, both properties will be excluded from results. For supported macros, see *Supported macros*.

problem.get

Description

`integer/array problem.get(object parameters)`

The method allows to retrieve problems according to the given parameters.

This method is for retrieving unresolved problems. It is also possible, if specified, to additionally retrieve recently resolved problems. The period that determines how old is "recently" is defined in *Administration* → *General*. Problems that were resolved prior to that period are not kept in the problem table. To retrieve problems that were resolved further back in the past, use the event.get method.

Attention:
This method may return problems of a deleted entity if these problems have not been removed by the housekeeper yet.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
eventids	string/array	Return only problems with the given IDs.
groupids	string/array	Return only problems created by objects that belong to the given host groups.
hostids	string/array	Return only problems created by objects that belong to the given hosts.
objectids	string/array	Return only problems created by the given objects.
source	integer	Return only problems with the given type. Refer to the problem event object page for a list of supported event types. Default: 0 - problem created by a trigger.
object	integer	Return only problems created by objects of the given type. Refer to the problem event object page for a list of supported object types. Default: 0 - trigger.
acknowledged	boolean	`true` - return acknowledged problems only; `false` - unacknowledged only.
suppressed	boolean	`true` - return only suppressed problems; `false` - return problems in the normal state.
symptom	boolean	`true` - return only symptom problem events; `false` - return only cause problem events.
severities	integer/array	Return only problems with given event severities. Applies only if object is trigger.

Parameter	Type	Description
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	array of objects	Return only problems with given tags. Exact match by tag and case-insensitive search by value and operator. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all problems. Possible operator types: 0 - (default) Like; 1 - Equal; 2 - Not like; 3 - Not equal 4 - Exists; 5 - Not exists.
recent	boolean	`true` - return PROBLEM and recently RESOLVED problems (depends on Display OK triggers for N seconds) Default: `false` - UNRESOLVED problems only
eventid_from	string	Return only problems with IDs greater or equal to the given ID.
eventid_till	string	Return only problems with IDs less or equal to the given ID.
time_from	timestamp	Return only problems that have been created after or at the given time.
time_till	timestamp	Return only problems that have been created before or at the given time.
selectAcknowledges	query	Return an `acknowledges` property with the problem updates. Problem updates are sorted in reverse chronological order. The problem update object has the following properties: `acknowledgeid` - `(string)` update's ID; `userid` - `(string)` ID of the user that updated the event; `eventid` - `(string)` ID of the updated event; `clock` - `(timestamp)` time when the event was updated; `message` - `(string)` text of the message; `action` - `(integer)`type of update action (see event.acknowledge); `old_severity` - `(integer)` event severity before this update action; `new_severity` - `(integer)` event severity after this update action; `suppress_until` - `(timestamp)` time till event will be suppressed; `taskid` - `(string)` ID of task if current event is undergoing a rank change; Supports `count`.
selectTags	query	Return a tags property with the problem tags. Output format: `[{"tag": "<tag>", "value": "<value>"}, ...]`.
selectSuppressionData	query	Return a `suppression_data` property with the list of active maintenances and manual suppressions: `maintenanceid` - `(string)` ID of the maintenance; `userid` - `(string)` ID of user who suppressed the problem; `suppress_until` - `(integer)` time until the problem is suppressed.
sortfield	string/array	Sort the result by the given properties. Possible values: `eventid`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	

Parameter	Type	Description
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving trigger problem events

Retrieve recent events from trigger "15112."

Request:

```
{
    "jsonrpc": "2.0",
    "method": "problem.get",
    "params": {
        "output": "extend",
        "selectAcknowledges": "extend",
        "selectTags": "extend",
        "selectSuppressionData": "extend",
        "objectids": "15112",
        "recent": "true",
        "sortfield": ["eventid"],
        "sortorder": "DESC"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "eventid": "1245463",
            "source": "0",
            "object": "0",
            "objectid": "15112",
            "clock": "1472457242",
            "ns": "209442442",
            "r_eventid": "1245468",
            "r_clock": "1472457285",
            "r_ns": "125644870",
            "correlationid": "0",
            "userid": "1",
            "name": "Zabbix agent on localhost is unreachable for 5 minutes",
            "acknowledged": "1",
            "severity": "3",
            "cause_eventid": "0",
            "opdata": "",
            "acknowledges": [
                {
                    "acknowledgeid": "14443",
                    "userid": "1",
                    "eventid": "1245463",
                    "clock": "1472457281",
```

```
                    "message": "problem solved",
                    "action": "6",
                    "old_severity": "0",
                    "new_severity": "0",
                    "suppress_until": "1472511600",
                    "taskid": "0"
                }
            ],
            "suppression_data": [
                {
                    "maintenanceid": "15",
                    "suppress_until": "1472511600",
                    "userid": "0"
                }
            ],
            "suppressed": "1",
            "tags": [
                {
                    "tag": "test tag",
                    "value": "test value"
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Alert
- Item
- Host
- LLD rule
- Trigger

Source

CEvent::get() in *ui/include/classes/api/services/CProblem.php*.

Proxy

This class is designed to work with proxies.

Object references:

- Proxy
- Proxy interface

Available methods:

- proxy.create - create new proxies
- proxy.delete - delete proxies
- proxy.get - retrieve proxies
- proxy.update - update proxies

> Proxy object

The following objects are directly related to the `proxy` API.

Proxy

The proxy object has the following properties.

Property	Type	Description
proxyid	string	ID of the proxy. Property behavior: - *read-only* - *required* for update operations
host	string	Name of the proxy. Property behavior: - *required* for create operations
status	integer	Type of proxy. Possible values: 5 - active proxy; 6 - passive proxy. Property behavior: - *required* for create operations
description	text	Description of the proxy.
lastaccess	timestamp	Time when the proxy last connected to the server. Property behavior: - *read-only*
tls_connect	integer	Connections to host. Possible values: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate.
tls_accept	integer	Connections from host. Possible bitmap values: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 6 for PSK and certificate).
tls_issuer	string	Certificate issuer.
tls_subject	string	Certificate subject.
tls_psk_identity	string	PSK identity; must be paired with only one PSK (across autoregistration, hosts, and proxies). Do not include sensitive information in the PSK identity, as it is sent unencrypted over the network to inform the receiver which PSK to use. Property behavior: - *write-only* - *required* if `tls_connect` is set to ”PSK”, or `tls_accept` contains the ”PSK” bit
tls_psk	string	Pre-shared key (PSK); must be at least 32 hex digits. Property behavior: - *write-only* - *required* if `tls_connect` is set to ”PSK”, or `tls_accept` contains the ”PSK” bit
proxy_address	string	Comma-delimited IP addresses or DNS names of active Zabbix proxy.

Property	Type	Description
auto_compress	integer	Indicates if communication between Zabbix server and proxy is compressed. Possible values: 0 - No compression; 1 - Compression enabled. Property behavior: - *read-only*
version	integer	Version of proxy. Three-part Zabbix version number, where two decimal digits are used for each part, e.g., 50401 for version 5.4.1, 60200 for version 6.2.0, etc. 0 - Unknown proxy version. Property behavior: - *read-only*
compatibility	integer	Version of proxy compared to Zabbix server version. Possible values: 0 - Undefined; 1 - Current version (proxy and server have the same major version); 2 - Outdated version (proxy version is older than server version, but is partially supported); 3 - Unsupported version (proxy version is older than server previous LTS release version or server major version is older than proxy major version). Property behavior: - *read-only*

Proxy interface

The proxy interface object defines the interface used to connect to a passive proxy. It has the following properties.

Property	Type	Description
dns	string	DNS name to connect to. Can be empty if connections are made via IP address. Property behavior: - *required* if `useip` is set to ”connect using DNS name”
ip	string	IP address to connect to. Can be empty if connections are made via DNS names. Property behavior: - *required* if `useip` is set to ”connect using IP address”
port	string	Port number to connect to. Property behavior: - *required*

Property	Type	Description
useip	integer	Whether the connection should be made via IP address. Possible values: 0 - connect using DNS name; 1 - connect using IP address. Property behavior: - *required*

proxy.create

Description

`object proxy.create(object/array proxies)`

This method allows to create new proxies.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Proxies to create.

Additionally to the standard proxy properties, the method accepts the following parameters.

Parameter	Type	Description
hosts	array	Hosts to be monitored by the proxy. If a host is already monitored by a different proxy, it will be reassigned to the current proxy. The hosts must have the `hostid` property defined.
interface	object	Host interface to be created for the passive proxy. Parameter behavior: - *required* if `status` of Proxy object is set to "passive proxy"

Return values

`(object)` Returns an object containing the IDs of the created proxies under the `proxyids` property. The order of the returned IDs matches the order of the passed proxies.

Examples

Create an active proxy

Create an action proxy "Active proxy" and assign a host to be monitored by it.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.create",
    "params": {
        "host": "Active proxy",
        "status": "5",
        "hosts": [
            {
                "hostid": "10279"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10280"
        ]
    },
    "id": 1
}
```

Create a passive proxy

Create a passive proxy "Passive proxy" and assign two hosts to be monitored by it.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.create",
    "params": {
        "host": "Passive proxy",
        "status": "6",
        "interface": {
            "ip": "127.0.0.1",
            "dns": "",
            "useip": "1",
            "port": "10051"
        },
        "hosts": [
            {
                "hostid": "10192"
            },
            {
                "hostid": "10139"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10284"
        ]
    },
    "id": 1
}
```

See also

- Host
- Proxy interface

Source

CProxy::create() in *ui/include/classes/api/services/CProxy.php*.

proxy.delete

Description

`object proxy.delete(array proxies)`

This method allows to delete proxies.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of proxies to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted proxies under the `proxyids` property.

Examples

Delete multiple proxies

Delete two proxies.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.delete",
    "params": [
        "10286",
        "10285"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10286",
            "10285"
        ]
    },
    "id": 1
}
```

Source

CProxy::delete() in *ui/include/classes/api/services/CProxy.php*.

proxy.get

Description

`integer/array proxy.get(object parameters)`

The method allows to retrieve proxies according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
proxyids	string/array	Return only proxies with the given IDs.

Parameter	Type	Description
selectHosts	query	Return a hosts property with the hosts monitored by the proxy.
selectInterface	query	Return an interface property with the proxy interface used by a passive proxy.
sortfield	string/array	Sort the result by the given properties. Possible values: `hostid`, `host`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve all proxies

Retrieve all configured proxies and their interfaces.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.get",
    "params": {
        "output": "extend",
        "selectInterface": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "host": "Active proxy",
            "status": "5",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "proxy_address": "",
            "auto_compress": "0",
            "version": "60400",
            "compatibility": "1",
            "proxyid": "30091",
            "interface": []
        },
```

```
        {
            "host": "Passive proxy",
            "status": "6",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "proxy_address": "",
            "auto_compress": "0",
            "lastaccess": "0",
            "version": "0",
            "compatibility": "0",
            "proxyid": "30092",
            "interface": {
                "interfaceid": "30109",
                "hostid": "30092",
                "main": "1",
                "type": "0",
                "useip": "1",
                "ip": "127.0.0.1",
                "dns": "",
                "port": "10051",
                "available": "0",
                "error": "",
                "errors_from": "0",
                "disable_until": "0",
                "details": []
            }
        }
    ],
    "id": 1
}
```


See also

- Host
- Proxy interface

Source

CProxy::get() in *ui/include/classes/api/services/CProxy.php*.

proxy.update

Description

`object proxy.update(object/array proxies)`

This method allows to update existing proxies.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Proxy properties to be updated.

The `proxyid` property must be defined for each proxy, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard proxy properties, the method accepts the following parameters.

Parameter	Type	Description
hosts	array	Hosts to be monitored by the proxy. If a host is already monitored by a different proxy, it will be reassigned to the current proxy. The hosts must have the `hostid` property defined.
interface	object	Host interface to replace the existing interface for the passive proxy. Parameter behavior: - *supported* if `status` of Proxy object is set to "passive proxy"

Return values

`(object)` Returns an object containing the IDs of the updated proxies under the `proxyids` property.

Examples

Change hosts monitored by a proxy

Update the proxy to monitor the two given hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.update",
    "params": {
        "proxyid": "10293",
        "hosts": [
            {
                "hostid": "10294"
            },
            {
                "hostid": "10295"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10293"
        ]
    },
    "id": 1
}
```

Change proxy status

Change the proxy to an active proxy and rename it to "Active proxy".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "proxy.update",
    "params": {
        "proxyid": "10293",
        "host": "Active proxy",
        "status": "5"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "proxyids": [
            "10293"
        ]
    },
    "id": 1
}
```

See also

- Host
- Proxy interface

Source

CProxy::update() in *ui/include/classes/api/services/CProxy.php*.

Regular expression

This class is designed to work with global regular expressions.

Object references:

- Regular expression

Available methods:

- regexp.create - creating new regular expressions
- regexp.delete - deleting regular expressions
- regexp.get - retrieving regular expressions
- regexp.update - updating regular expressions

> Regular expression object

The following objects are directly related to the `regexp` API.

Regular expression

The global regular expression object has the following properties.

Property	Type	Description
regexpid	string	ID of the regular expression. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the regular expression. Property behavior: - *required* for create operations
test_string	string	Test string.

Expressions

The expressions object has the following properties.

Property	Type	Description
expression	string	Regular expression. Property behavior: - *required*
expression_type	integer	Type of Regular expression. Possible values: 0 - Character string included; 1 - Any character string included; 2 - Character string not included; 3 - Result is TRUE; 4 - Result is FALSE. Property behavior: - *required*
exp_delimiter	string	Expression delimiter. Default value: `","`. Possible values: `","` or `"."`, or `"/"`. Property behavior: - *supported* if `expression_type` is set to ”Any character string included”
case_sensitive	integer	Case sensitivity. Default value: 0. Possible values: 0 - Case insensitive; 1 - Case sensitive.

regexp.create

Description

`object regexp.create(object/array regularExpressions)`

This method allows to create new global regular expressions.

> **Note:**
> This method is only available to *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Regular expressions to create.

Additionally to the standard properties, the method accepts the following parameters.

Parameter	Type	Description
expressions	array	Expressions options. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the created regular expressions under the `regexpids` property.

Examples

Creating a new global regular expression.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "regexp.create",
    "params": {
      "name": "Storage devices for SNMP discovery",
      "test_string": "/boot",
      "expressions": [
        {
          "expression": "^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$",
          "expression_type": "4",
          "case_sensitive": "1"
        }
      ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "regexpids": [
            "16"
        ]
    },
    "id": 1
}
```

Source

CRegexp::create() in *ui/include/classes/api/services/CRegexp.php*.

regexp.delete

Description

`object regexp.delete(array regexpids)`

This method allows to delete global regular expressions.

> **Note:**
> This method is only available to *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the regular expressions to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted regular expressions under the `regexpids` property.

Examples

Deleting multiple global regular expressions.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "regexp.delete",
    "params": [
        "16",
        "17"
```

```
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "regexpids": [
            "16",
            "17"
        ]
    },
    "id": 1
}
```

Source

CRegexp::delete() in *ui/include/classes/api/services/CRegexp.php*.

regexp.get

Description

`integer/array regexp.get(object parameters)`

The method allows to retrieve global regular expressions according to the given parameters.

Note:
This method is available only to *Super Admin*. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
regexpids	string/array	Return only regular expressions with the given IDs.
selectExpressions	query	Return a expressions property.
sortfield	string/array	Sort the result by the given properties. Possible values: `regexpid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving global regular expressions.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "regexp.get",
    "params": {
        "output": ["regexpid", "name"],
        "selectExpressions": ["expression", "expression_type"],
        "regexpids": [1, 2],
        "preservekeys": true
    },
    "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "1": {
      "regexpid": "1",
      "name": "File systems for discovery",
      "expressions": [
        {
          "expression": "^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|
          "expression_type": "3"
        }
      ]
    },
    "2": {
      "regexpid": "2",
      "name": "Network interfaces for discovery",
      "expressions": [
        {
          "expression": "^Software Loopback Interface",
          "expression_type": "4"
        },
        {
          "expression": "^(In)?[Ll]oop[Bb]ack[0-9._]*$",
          "expression_type": "4"
        },
        {
          "expression": "^NULL[0-9.]*$",
          "expression_type": "4"
        },
        {
          "expression": "^[Ll]o[0-9.]*$",
          "expression_type": "4"
        },
        {
          "expression": "^[Ss]ystem$",
          "expression_type": "4"
        },
        {
          "expression": "^Nu[0-9.]*$",
          "expression_type": "4"
        }
      ]
    }
  },
  "id": 1
```

```
}
```

Source

CRegexp::get() in *ui/include/classes/api/services/CRegexp.php*.

regexp.update

Description

`object regexp.update(object/array regularExpressions)`

This method allows to update existing global regular expressions.

> **Note:**
> This method is only available to *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Regular expression properties to be updated.

The `regexpid` property must be defined for each object, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard properties, the method accepts the following parameters.

Parameter	Type	Description
expressions	array	Expressions options.

Return values

`(object)` Returns an object containing the IDs of the updated regular expressions under the `regexpids` property.

Examples

Updating global regular expression for file systems discovery.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "regexp.update",
    "params": {
      "regexpid": "1",
      "name": "File systems for discovery",
      "test_string": "",
      "expressions": [
        {
          "expression": "^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|zfs)$",
          "expression_type": "3",
          "exp_delimiter": ",",
          "case_sensitive": "0"
        },
        {
          "expression": "^(ntfs|fat32|fat16)$",
          "expression_type": "3",
          "exp_delimiter": ",",
          "case_sensitive": "0"
        }
      ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "regexpids": [
            "1"
        ]
    },
    "id": 1
}
```

Source

CRegexp::update() in *ui/include/classes/api/services/CRegexp.php*.

Report

This class is designed to work with scheduled reports.

Object references:

- Report
- Users
- User groups

Available methods:

- report.create - create new scheduled reports
- report.delete - delete scheduled reports
- report.get - retrieve scheduled reports
- report.update - update scheduled reports

> Report object

The following objects are directly related to the `report` API.

Report

The report object has the following properties:

Property	Type	Description
reportid	string	ID of the report. Property behavior: - *read-only* - *required* for update operations
userid	string	ID of the user who created the report. Property behavior: - *required* for create operations
name	string	Unique name of the report. Property behavior: - *required* for create operations
dashboardid	string	ID of the dashboard that the report is based on. Property behavior: - *required* for create operations

Property	Type	Description
period	integer	Period for which the report will be prepared. Possible values: 0 - *(default)* previous day; 1 - previous week; 2 - previous month; 3 - previous year.
cycle	integer	Period repeating schedule. Possible values: 0 - *(default)* daily; 1 - weekly; 2 - monthly; 3 - yearly.
start_time	integer	Time of the day, in seconds, when the report will be prepared for sending. Default: 0.
weekdays	integer	Days of the week for sending the report. Possible bitmap values: 1 - Monday; 2 - Tuesday; 4 - Wednesday; 8 - Thursday; 16 - Friday; 32 - Saturday; 64 - Sunday. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 21 for Monday, Wednesday, and Friday). Default: 0. Property behavior: - *required* if `cycle` is set to "weekly".
active_since	string	On which date to start. Possible values: empty string - *(default)* not specified (stored as 0); specific date in YYYY-MM-DD format (stored as a timestamp of the beginning of a day (00:00:00)).
active_till	string	On which date to end. Possible values: empty string - *(default)* not specified (stored as 0); specific date in YYYY-MM-DD format (stored as a timestamp of the end of a day (23:59:59)).
subject	string	Report message subject.
message	string	Report message text.
status	integer	Whether the report is enabled or disabled. Possible values: 0 - Disabled; 1 - *(default)* Enabled.
description	text	Description of the report.

Property	Type	Description
state	integer	State of the report. Possible values: 0 - *(default)* report was not yet processed; 1 - report was generated and successfully sent to all recipients; 2 - report generating failed; ”info” contains error information; 3 - report was generated, but sending to some (or all) recipients failed; ”info” contains error information. Property behavior: - *read-only*
lastsent	timestamp	Unix timestamp of the last successfully sent report. Property behavior: - *read-only*
info	string	Error description or additional information. Property behavior: - *read-only*

Users

The users object has the following properties:

Property	Type	Description
userid	string	ID of user to send the report to. Property behavior: - *required*
access_userid	string	ID of user on whose behalf the report will be generated. 0 - *(default)* Generate report by recipient.
exclude	integer	Whether to exclude the user from mailing list. Possible values: 0 - *(default)* Include; 1 - Exclude.

User groups

The user groups object has the following properties:

Property	Type	Description
usrgrpid	string	ID of user group to send the report to. Property behavior: - *required*
access_userid	string	ID of user on whose behalf the report will be generated. 0 - *(default)* Generate report by recipient.

report.create

Description

`object report.create(object/array reports)`

This method allows to create new scheduled reports.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Scheduled reports to create.

Additionally to the standard scheduled report properties, the method accepts the following parameters.

Parameter	Type	Description
users	object/array	Users to send the report to. Parameter behavior: - *required* if `user_groups` is not set
user_groups	object/array	User groups to send the report to. Parameter behavior: - *required* if `users` is not set

Return values

`(object)` Returns an object containing the IDs of the created scheduled reports under the `reportids` property. The order of the returned IDs matches the order of the passed scheduled reports.

Examples

Creating a scheduled report

Create a weekly report that will be prepared for the previous week every Monday-Friday at 12:00 from 2021-04-01 to 2021-08-31.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "report.create",
    "params": {
        "userid": "1",
        "name": "Weekly report",
        "dashboardid": "1",
        "period": "1",
        "cycle": "1",
        "start_time": "43200",
        "weekdays": "31",
        "active_since": "2021-04-01",
        "active_till": "2021-08-31",
        "subject": "Weekly report",
        "message": "Report accompanying text",
        "status": "1",
        "description": "Report description",
        "users": [
            {
                "userid": "1",
                "access_userid": "1",
                "exclude": "0"
            },
            {
                "userid": "2",
                "access_userid": "0",
                "exclude": "1"
            }
        ],
        "user_groups": [
            {
                "usrgrpid": "7",
```

```
                "access_userid": "0"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "reportids": [
            "1"
        ]
    },
    "id": 1
}
```

See also

- Users
- User groups

Source

CReport::create() in *ui/include/classes/api/services/CReport.php*.

report.delete

Description

`object report.delete(array reportids)`

This method allows to delete scheduled reports.

> **Note:**
> This method is only available to *Admin* and *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the scheduled reports to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted scheduled reports under the `reportids` property.

Examples

Deleting multiple scheduled reports

Delete two scheduled reports.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "report.delete",
    "params": [
        "1",
        "2"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "reportids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

Source

CReport::delete() in *ui/include/classes/api/services/CReport.php*.

report.get

Description

`integer/array report.get(object parameters)`

The method allows to retrieve scheduled reports according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
reportids	string/array	Return only scheduled reports with the given report IDs.
expired	boolean	If set to `true` returns only expired scheduled reports, if `false` - only active scheduled reports.
selectUsers	query	Return a users property the report is configured to be sent to.
selectUserGroups	query	Return a user_groups property the report is configured to be sent to.
sortfield	string/array	Sort the result by the given properties. Possible values: `reportid`, `name`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving report data

Request:

```
{
    "jsonrpc": "2.0",
    "method": "report.get",
    "params": [
        "output": "extend",
        "selectUsers": "extend",
        "selectUserGroups": "extend",
        "reportids": ["1", "2"]
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "reportid": "1",
            "userid": "1",
            "name": "Weekly report",
            "dashboardid": "1",
            "period": "1",
            "cycle": "1",
            "start_time": "43200",
            "weekdays": "31",
            "active_since": "2021-04-01",
            "active_till": "2021-08-31",
            "subject": "Weekly report",
            "message": "Report accompanying text",
            "status": "1",
            "description": "Report description",
            "state": "1",
            "lastsent": "1613563219",
            "info": "",
            "users": [
                {
                    "userid": "1",
                    "access_userid": "1",
                    "exclude": "0"
                },
                {
                    "userid": "2",
                    "access_userid": "0",
                    "exclude": "1"
                }
            ],
            "user_groups": [
                {
                    "usrgrpid": "7",
                    "access_userid": "0"
                }
            ]
        },
        {
            "reportid": "2",
            "userid": "1",
            "name": "Monthly report",
            "dashboardid": "2",
            "period": "2",
            "cycle": "2",
            "start_time": "0",
```

```
            "weekdays": "0",
            "active_since": "2021-05-01",
            "active_till": "",
            "subject": "Monthly report",
            "message": "Report accompanying text",
            "status": "1",
            "description": "",
            "state": "0",
            "lastsent": "0",
            "info": "",
            "users": [
                {
                    "userid": "1",
                    "access_userid": "1",
                    "exclude": "0"
                }
            ],
            "user_groups": []
        }
    ],
    "id": 1
}
```


See also

- Users
- User groups

Source

CReport::get() in *ui/include/classes/api/services/CReport.php*.

report.update

Description

`object report.update(object/array reports)`

This method allows to update existing scheduled reports.

> **Note:**
> This method is only available to *Admin* and *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Scheduled report properties to be updated.

The `reportid` property must be defined for each scheduled report, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard scheduled report properties the method accepts the following parameters.

Parameter	Type	Description
users	object/array	Users to replace the current users assigned to the scheduled report. Parameter behavior: - *required* if `user_groups` is not set
user_groups	object/array	User groups to replace the current user groups assigned to the scheduled report. Parameter behavior: - *required* if `users` is not set

Return values

(object) Returns an object containing the IDs of the updated scheduled reports under the reportids property.

Examples

Disabling scheduled report

Request:

```
{
    "jsonrpc": "2.0",
    "method": "report.update",
    "params": {
        "reportid": "1",
        "status": "0"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "reportids": [
            "1"
        ]
    },
    "id": 1
}
```

See also

- Users
- User groups

Source

CReport::update() in *ui/include/classes/api/services/CReport.php*.

Role

This class is designed to work with user roles.

Object references:

- Role
- Role rules
- UI element
- Service
- Service tag
- Module
- Action

Available methods:

- role.create - create new user roles
- role.delete - delete user roles
- role.get - retrieve user roles
- role.update - update user roles

> Role object

The following objects are directly related to the role API.

Role

The role object has the following properties:

Property	Type	Description
roleid	string	ID of the role. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the role. Property behavior: - *required* for create operations
type	integer	User type. Possible values: 1 - *(default)* User; 2 - Admin; 3 - Super admin. Property behavior: - *required* for create operations
readonly	integer	Whether the role is readonly. Possible values: 0 - *(default)* No; 1 - Yes. Property behavior: - *read-only*

Role rules

The role rules object has the following properties:

Property	Type	Description
ui	array	Array of the UI element objects.
ui.default_access	integer	Whether access to new UI elements is enabled. Possible values: 0 - Disabled; 1 - *(default)* Enabled.
services.read.mode	integer	Read-only access to services. Possible values: 0 - Read-only access to the services, specified by the `services.read.list` or matched by the `services.read.tag` properties; 1 - *(default)* Read-only access to all services.
services.read.list	array	Array of Service objects. The specified services, including child services, will be granted a read-only access to the user role. Read-only access will not override read-write access to the services. Property behavior: - *supported* if `services.read.mode` is set to "0"

Property	Type	Description
services.read.tag	object	Array of Service tag object. The tag matched services, including child services, will be granted a read-only access to the user role. Read-only access will not override read-write access to the services. Property behavior: - *supported* if `services.read.mode` is set to "0"
services.write.mode	integer	Read-write access to services. Possible values: 0 - *(default)* Read-write access to the services, specified by the `services.write.list` or matched by the `services.write.tag` properties; 1 - Read-write access to all services.
services.write.list	array	Array of Service objects. The specified services, including child services, will be granted a read-write access to the user role. Read-write access will override read-only access to the services. Property behavior: - *supported* if `services.write.mode` is set to "0"
services.write.tag	object	Array of Service tag object. The tag matched services, including child services, will be granted a read-write access to the user role. Read-write access will override read-only access to the services. Property behavior: - *supported* if `services.write.mode` is set to "0"
modules	array	Array of the module objects.
modules.default_access	integer	Whether access to new modules is enabled. Possible values: 0 - Disabled; 1 - *(default)* Enabled.
api.access	integer	Whether access to API is enabled. Possible values: 0 - Disabled; 1 - *(default)* Enabled.
api.mode	integer	Mode for treating API methods listed in the `api` property. Possible values: 0 - *(default)* Deny list; 1 - Allow list.
api	array	Array of API methods.
actions	array	Array of the action objects.
actions.default_access	integer	Whether access to new actions is enabled. Possible values: 0 - Disabled; 1 - *(default)* Enabled.

UI element

The UI element object has the following properties:

Property	Type	Description
name	string	Name of the UI element. Possible values if `type` of Role object is set to ”User”, ”Admin”, or ”Super admin”: `monitoring.dashboard` - *Dashboards*; `monitoring.problems` - *Monitoring → Problems*; `monitoring.hosts` - *Monitoring → Hosts*; `monitoring.latest_data` - *Monitoring → Latest data*; `monitoring.maps` - *Monitoring → Maps*; `services.services` - *Services → Services*; `services.sla_report` - *Services → SLA report*; `inventory.overview` - *Inventory → Overview*; `inventory.hosts` - *Inventory → Hosts*; `reports.availability_report` - *Reports → Availability report*; `reports.top_triggers` - *Reports → Triggers top 100*. Possible values if `type` of Role object is set to ”Admin” or ”Super admin”: `monitoring.discovery` - *Monitoring → Discovery*; `services.sla` - *Services → SLA*; `reports.scheduled_reports` - *Reports → Scheduled reports*; `reports.notifications` - *Reports → Notifications*; `configuration.template_groups` - *Data collection → Template groups*; `configuration.host_groups` - *Data collection → Host groups*; `configuration.templates` - *Data collection → Templates*; `configuration.hosts` - *Data collection → Hosts*; `configuration.maintenance` - *Data collection → Maintenance*; `configuration.discovery` - *Data collection → Discovery*; `configuration.trigger_actions` - *Alerts → Actions → Trigger actions*; `configuration.service_actions` - *Alerts → Actions → Service actions*; `configuration.discovery_actions` - *Alerts → Actions → Discovery actions*; `configuration.autoregistration_actions` - *Alerts → Actions → Autoregistration actions*; `configuration.internal_actions` - *Alerts → Actions → Internal actions*. Possible values if `type` of Role object is set to ”Super admin”: `reports.system_info` - *Reports → System information*; `reports.audit` - *Reports → Audit log*; `reports.action_log` - *Reports → Action log*; `configuration.event_correlation` - *Data collection → Event correlation*; `administration.media_types` - *Alerts → Media types*; `administration.scripts` - *Alerts → Scripts*; `administration.user_groups` - *Users → User groups*; `administration.user_roles` - *Users → User roles*; `administration.users` - *Users → Users*; `administration.api_tokens` - *Users → API tokens*; `administration.authentication` - *Users → Authentication*; `administration.general` - *Administration → General*; `administration.audit_log` - *Administration → Audit log*; `administration.housekeeping` - *Administration → Housekeeping*; `administration.proxies` - *Administration → Proxies*; `administration.macros` - *Administration → Macros*; `administration.queue` - *Administration → Queue*. Property behavior: *- required*

Property	Type	Description
status	integer	Whether access to the UI element is enabled. Possible values: 0 - Disabled; 1 - *(default)* Enabled.

Service

Property	Type	Description
serviceid	string	ID of the Service. Property behavior: - *required*

Service tag

Property	Type	Description
tag	string	Tag name. If empty string is specified, the service tag will not be used for service matching. Property behavior: - *required*
value	string	Tag value. If no value or empty string is specified, only the tag name will be used for service matching.

Module

The module object has the following properties:

Property	Type	Description
moduleid	string	ID of the module. Property behavior: - *required*
status	integer	Whether access to the module is enabled. Possible values: 0 - Disabled; 1 - *(default)* Enabled.

Action

The action object has the following properties:

Property	Type	Description
name	string	Name of the action. Possible values if `type` of Role object is set to ”User”, ”Admin”, or ”Super admin: `edit_dashboards` - Create and edit dashboards; `edit_maps` - Create and edit maps; `add_problem_comments` - Add problem comments; `change_severity` - Change problem severity; `acknowledge_problems` - Acknowledge problems; `suppress_problems` - Suppress problems; `close_problems` - Close problems; `execute_scripts` - Execute scripts; `manage_api_tokens` - Manage API tokens. Possible values if `type` of Role object is set to ”Admin” or ”Super admin”: `edit_maintenance` - Create and edit maintenances; `manage_scheduled_reports` - Manage scheduled reports, `manage_sla` - Manage SLA. Possible values if `type` of Role object is set to ”User” or ”Admin”: `invoke_execute_now` - allows to execute item checks for users that have only read permissions on host. Property behavior: - *required*
status	integer	Whether access to perform the action is enabled. Possible values: 0 - Disabled; 1 - *(default)* Enabled.

role.create

Description

`object role.create(object/array roles)`

This method allows to create new roles.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Roles to create.

Additionally to the standard role properties, the method accepts the following parameters.

Parameter	Type	Description
rules	array	Role rules to be created for the role.

Return values

`(object)` Returns an object containing the IDs of the created roles under the `roleids` property. The order of the returned IDs matches the order of the passed roles.

Examples

Creating a role

Create a role with type ”User” and denied access to two UI elements.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "role.create",
    "params": {
        "name": "Operator",
        "type": "1",
        "rules": {
            "ui": [
                {
                    "name": "monitoring.hosts",
                    "status": "0"
                },
                {
                    "name": "monitoring.maps",
                    "status": "0"
                }
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": [
            "5"
        ]
    },
    "id": 1
}
```

See also

- Role rules
- UI element
- Module
- Action

Source

CRole::create() in *ui/include/classes/api/services/CRole.php*.

role.delete

Description

`object role.delete(array roleids)`

This method allows to delete roles.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the roles to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted roles under the `roleids` property.

Examples

Deleting multiple user roles

Delete two user roles.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "role.delete",
    "params": [
        "4",
        "5"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

Source

CRole::delete() in *ui/include/classes/api/services/CRole.php*.

role.get

Description

`integer/array role.get(object parameters)`

The method allows to retrieve roles according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
roleids	string/array	Return only roles with the given IDs.
selectRules	query	Return role rules in the rules property.
selectUsers	query	Select users this role is assigned to.
sortfield	string/array	Sort the result by the given properties. Possible values: `roleid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	

Parameter	Type	Description
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving role data

Retrieve "Super admin role" role data and its access rules.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "role.get",
    "params": {
        "output": "extend",
        "selectRules": "extend",
        "roleids": "3"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "roleid": "3",
            "name": "Super admin role",
            "type": "3",
            "readonly": "1",
            "rules": {
                "ui": [
                    {
                        "name": "monitoring.dashboard",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.problems",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.hosts",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.latest_data",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.maps",
                        "status": "1"
                    },
                    {
                        "name": "services.services",
```

```
            "status": "1"
        },
        {
            "name": "services.sla_report",
            "status": "1"
        },
        {
            "name": "inventory.overview",
            "status": "1"
        },
        {
            "name": "inventory.hosts",
            "status": "1"
        },
        {
            "name": "reports.availability_report",
            "status": "1"
        },
        {
            "name": "reports.top_triggers",
            "status": "1"
        },
        {
            "name": "monitoring.discovery",
            "status": "1"
        },
        {
            "name": "services.sla",
            "status": "1"
        },
        {
            "name": "reports.scheduled_reports",
            "status": "1"
        },
        {
            "name": "reports.notifications",
            "status": "1"
        },
        {
            "name": "configuration.template_groups",
            "status": "1"
        },
        {
            "name": "configuration.host_groups",
            "status": "1"
        },
        {
            "name": "configuration.templates",
            "status": "1"
        },
        {
            "name": "configuration.hosts",
            "status": "1"
        },
        {
            "name": "configuration.maintenance",
            "status": "1"
        },
        {
            "name": "configuration.discovery",
            "status": "1"
        },
```

```
                {
                    "name": "configuration.trigger_actions",
                    "status": "1"
                },
                {
                    "name": "configuration.service_actions",
                    "status": "1"
                },
                {
                    "name": "configuration.discovery_actions",
                    "status": "1"
                },
                {
                    "name": "configuration.autoregistration_actions",
                    "status": "1"
                },
                {
                    "name": "configuration.internal_actions",
                    "status": "1"
                },
                {
                    "name": "reports.system_info",
                    "status": "1"
                },
                {
                    "name": "reports.audit",
                    "status": "1"
                },
                {
                    "name": "reports.action_log",
                    "status": "1"
                },
                {
                    "name": "configuration.event_correlation",
                    "status": "1"
                },
                {
                    "name": "administration.media_types",
                    "status": "1"
                },
                {
                    "name": "administration.scripts",
                    "status": "1"
                },
                {
                    "name": "administration.user_groups",
                    "status": "1"
                },
                {
                    "name": "administration.user_roles",
                    "status": "1"
                },
                {
                    "name": "administration.users",
                    "status": "1"
                },
                {
                    "name": "administration.api_tokens",
                    "status": "1"
                },
                {
                    "name": "administration.authentication",
```

```
                "status": "1"
            },
            {
                "name": "administration.general",
                "status": "1"
            },
            {
                "name": "administration.audit_log",
                "status": "1"
            },
            {
                "name": "administration.housekeeping",
                "status": "1"
            },
            {
                "name": "administration.proxies",
                "status": "1"
            },
            {
                "name": "administration.macros",
                "status": "1"
            },
            {
                "name": "administration.queue",
                "status": "1"
            }
        ],
        "ui.default_access": "1",
        "services.read.mode": "1",
        "services.read.list": [],
        "services.read.tag": {
            "tag": "",
            "value": ""
        },
        "services.write.mode": "1",
        "services.write.list": [],
        "services.write.tag": {
            "tag": "",
            "value": ""
        },
        "modules": [],
        "modules.default_access": "1",
        "api.access": "1",
        "api.mode": "0",
        "api": [],
        "actions": [
            {
                "name": "edit_dashboards",
                "status": "1"
            },
            {
                "name": "edit_maps",
                "status": "1"
            },
            {
                "name": "acknowledge_problems",
                "status": "1"
            },
            {
                "name": "suppress_problems",
                "status": "1"
            },
```

```
                    {
                        "name": "close_problems",
                        "status": "1"
                    },
                    {
                        "name": "change_severity",
                        "status": "1"
                    },
                    {
                        "name": "add_problem_comments",
                        "status": "1"
                    },
                    {
                        "name": "execute_scripts",
                        "status": "1"
                    },
                    {
                        "name": "manage_api_tokens",
                        "status": "1"
                    },
                    {
                        "name": "edit_maintenance",
                        "status": "1"
                    },
                    {
                        "name": "manage_scheduled_reports",
                        "status": "1"
                    },
                    {
                        "name": "manage_sla",
                        "status": "1"
                    },
                    {
                        "name": "invoke_execute_now",
                        "status": "1"
                    }
                ],
                "actions.default_access": "1"
            }
        }
    ],
    "id": 1
}
```


See also

- Role rules
- User

Source

CRole::get() in *ui/include/classes/api/services/CRole.php*.

role.update

Description

`object role.update(object/array roles)`

This method allows to update existing roles.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Role properties to be updated.

The `roleid` property must be defined for each role, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard role properties the method accepts the following parameters.

Parameter	Type	Description
rules	array	Access rules to update for the role.

Return values

`(object)` Returns an object containing the IDs of the updated roles under the `roleids` property.

Examples

Disabling ability to execute scripts

Update role with ID ”5”, disable ability to execute scripts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "role.update",
    "params": [
        {
            "roleid": "5",
            "rules": {
                "actions": [
                    {
                        "name": "execute_scripts",
                        "status": "0"
                    }
                ]
            }
        }
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": [
            "5"
        ]
    },
    "id": 1
}
```

Limiting access to API

Update role with ID ”5”, deny to call any ”create”, ”update” or ”delete” methods.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "role.update",
    "params": [
        {
            "roleid": "5",
            "rules": {
                "api.access": "1",
```

```
                "api.mode": "0",
                "api": ["*.create", "*.update", "*.delete"]
            }
        }
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "roleids": [
            "5"
        ]
    },
    "id": 1
}
```

Source

CRole::update() in *ui/include/classes/api/services/CRole.php*.

Script

This class is designed to work with scripts.

Object references:

- Script
- Webhook parameters
- Debug
- Log entry

Available methods:

- script.create - create new scripts
- script.delete - delete scripts
- script.execute - run scripts
- script.get - retrieve scripts
- script.getscriptsbyhosts - retrieve scripts for hosts
- script.update - update scripts

> Script object

The following objects are directly related to the `script` API.

Script

The script object has the following properties.

Property	Type	Description
scriptid	string	ID of the script. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the script. Property behavior: - *required* for create operations

Property	Type	Description
type	integer	Script type. Possible values if `scope` is set to ”action operation”: 0 - Script; 1 - IPMI; 2 - SSH; 3 - TELNET; 5 - Webhook. Possible values if `scope` is set to ”manual host action” or ”manual event action”: 6 - URL. Property behavior: - *required* for create operations
command	string	Command to run. Property behavior: - *required* if `type` is set to ”Script”, ”IPMI”, ”SSH”, ”TELNET”, or ”Webhook”
scope	integer	Script scope. Possible values: 1 - action operation; 2 - manual host action; 4 - manual event action. Property behavior: - *required* for create operations
execute_on	integer	Where to run the script. Possible values: 0 - run on Zabbix agent; 1 - run on Zabbix server; 2 - *(default)* run on Zabbix server (proxy). Property behavior: - *supported* if `type` is set to ”Script”
menu_path	string	Folders separated by slash that form a menu like navigation in frontend when clicked on host or event. Property behavior: - *supported* if `scope` is set to ”manual host action” or ”manual event action”
authtype	integer	Authentication method used for SSH script type. Possible values: 0 - password; 1 - public key. Property behavior: - *supported* if `type` is set to ”SSH”
username	string	User name used for authentication. Property behavior: - *required* if `type` is set to ”SSH” or ”TELNET”
password	string	Password used for SSH scripts with password authentication and TELNET scripts. Property behavior: - *supported* if `type` is set to ”SSH” and `authtype` is set to ”password”, or `type` is set to ”TELNET”

Property	Type	Description
publickey	string	Name of the public key file used for SSH scripts with public key authentication. Property behavior: - *required* if `type` is set to ”SSH” and `authtype` is set to ”public key”
privatekey	string	Name of the private key file used for SSH scripts with public key authentication. Property behavior: - *required* if `type` is set to ”SSH” and `authtype` is set to ”public key”
port	string	Port number used for SSH and TELNET scripts. Property behavior: - *supported* if `type` is set to ”SSH” or ”TELNET”
groupid	string	ID of the host group that the script can be run on. If set to ”0”, the script will be available on all host groups. Default: 0.
usrgrpid	string	ID of the user group that will be allowed to run the script. If set to ”0”, the script will be available for all user groups. Default: 0. Property behavior: - *supported* if `scope` is set to ”manual host action” or ”manual event action”
host_access	integer	Host permissions needed to run the script. Possible values: 2 - *(default)* read; 3 - write. Property behavior: - *supported* if `scope` is set to ”manual host action” or ”manual event action”
confirmation	string	Confirmation pop up text. The pop up will appear when trying to run the script from the Zabbix frontend. Property behavior: - *supported* if `scope` is set to ”manual host action” or ”manual event action”
timeout	string	Webhook script execution timeout in seconds. Time suffixes are supported (e.g., 30s, 1m). Possible values: 1-60s. Default: 30s. Property behavior: - *required* if `type` is set to ”Webhook”
parameters	array	Array of webhook input parameters. Property behavior: - *supported* if `type` is set to ”Webhook”
description	string	Description of the script.
url	string	User defined URL. Property behavior: - *required* if `type` is set to ”URL”

Property	Type	Description
new_window	integer	Open URL in a new window. Possible values: 0 - No; 1 - *(default)* Yes. Property behavior: - *supported* if `type` is set to "URL"

Webhook parameters

Parameters passed to webhook script when it is called have the following properties.

Property	Type	Description
name	string	Parameter name. Property behavior: - *required*
value	string	Parameter value. Supports macros.

Debug

Debug information of executed webhook script. The debug object has the following properties.

Property	Type	Description
logs	array	Array of log entries.
ms	string	Script execution duration in milliseconds.

Log entry

The log entry object has the following properties.

Property	Type	Description
level	integer	Log level.
ms	string	The time elapsed in milliseconds since the script was run before log entry was added.
message	string	Log message.

script.create

Description

`object script.create(object/array scripts)`

This method allows to create new scripts.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Scripts to create.

The method accepts scripts with the standard script properties.

Return values

`(object)` Returns an object containing the IDs of the created scripts under the `scriptids` property. The order of the returned IDs matches the order of the passed scripts.

Examples

Create a webhook script

Create a webhook script that sends HTTP request to external service.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "Webhook script",
        "command": "try {\n var request = new HttpRequest(),\n response,\n data;\n\n request.addHeader('Co
        "type": 5,
        "timeout": "40s",
        "parameters": [
            {
                "name": "token",
                "value": "{$WEBHOOK.TOKEN}"
            },
            {
                "name": "host",
                "value": "{HOST.HOST}"
            },
            {
                "name": "v",
                "value": "2.2"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "3"
        ]
    },
    "id": 1
}
```

Create an SSH script

Create an SSH script with public key authentication that can be executed on a host and has a context menu.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "SSH script",
        "command": "my script command",
        "type": 2,
        "username": "John",
        "publickey": "pub.key",
        "privatekey": "priv.key",
        "password": "secret",
        "port": "12345",
        "scope": 2,
        "menu_path": "All scripts/SSH",
        "usrgrpid": "7",
        "groupid": "4"
    },
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "5"
        ]
    },
    "id": 1
}
```

Create a custom script

Create a custom script that will reboot a server. The script will require write access to the host and will display a configuration message before running in the frontend.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "Reboot server",
        "command": "reboot server 1",
        "confirmation": "Are you sure you would like to reboot the server?",
        "scope": 2,
        "type": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "4"
        ]
    },
    "id": 1
}
```

Create a URL type script

Create a URL type script that for host scope and remains in same window and has confirmation text.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.create",
    "params": {
        "name": "URL script",
        "type": 6,
        "scope": 2,
        "url": "http://zabbix/ui/zabbix.php?action=host.edit&hostid={HOST.ID}",
        "confirmation": "Edit host {HOST.NAME}?",
        "new_window": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "56"
        ]
    },
    "id": 1
}
```

Source

CScript::create() in *ui/include/classes/api/services/CScript.php*.

script.delete

Description

`object script.delete(array scriptIds)`

This method allows to delete scripts.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the scripts to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted scripts under the `scriptids` property.

Examples

Delete multiple scripts

Delete two scripts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.delete",
    "params": [
        "3",
        "4"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "3",
            "4"
        ]
    },
    "id": 1
}
```

Source

CScript::delete() in *ui/include/classes/api/services/CScript.php*.

script.execute

Description

`object script.execute(object parameters)`

This method allows to run a script on a host or event. Except for URL type scripts. Those are not executable.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the ID of the script to run and either the ID of the host or the ID of the event.

Parameter	Type	Description
scriptid	string	ID of the script to run. Parameter behavior: - *required*
hostid	string	ID of the host to run the script on. Parameter behavior: - *required* if `eventid` is not set
eventid	string	ID of the event to run the script on. Parameter behavior: - *required* if `hostid` is not set

Return values

`(object)` Returns the result of script execution.

Property	Type	Description
response	string	Whether the script was run successfully. Possible value - `success`.
value	string	Script output.
debug	object	Contains a debug object if a webhook script is executed. For other script types, it contains empty object.

Examples

Run a webhook script

Run a webhook script that sends HTTP request to external service.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.execute",
    "params": {
        "scriptid": "4",
        "hostid": "30079"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "response": "success",
        "value": "{\"status\":\"sent\",\"timestamp\":\"1611235391\"}",
        "debug": {
            "logs": [
                {
                    "level": 3,
                    "ms": 480,
                    "message": "[Webhook Script] HTTP status: 200."
                }
            ],
            "ms": 495
        }
    },
    "id": 1
}
```


Run a custom script

Run a "ping" script on a host.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "script.execute",
    "params": {
        "scriptid": "1",
        "hostid": "30079"
    },
    "id": 1
}
```


Response:


```
{
    "jsonrpc": "2.0",
    "result": {
        "response": "success",
        "value": "PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.\n64 bytes from 127.0.0.1: icmp_req=1 tt
        "debug": []
    },
    "id": 1
}
```


Source

CScript::execute() in *ui/include/classes/api/services/CScript.php*.

script.get

Description

`integer/array script.get(object parameters)`

The method allows to retrieve scripts according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
groupids	string/array	Return only scripts that can be run on the given host groups.
hostids	string/array	Return only scripts that can be run on the given hosts.
scriptids	string/array	Return only scripts with the given IDs.
usrgrpids	string/array	Return only scripts that can be run by users in the given user groups.
selectHostGroups	query	Return a hostgroups property with host groups that the script can be run on.
selectHosts	query	Return a hosts property with hosts that the script can be run on.
selectActions	query	Return a actions property with actions that the script is associated with.
sortfield	string/array	Sort the result by the given properties. Possible values: `scriptid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectGroups (deprecated)	query	This parameter is deprecated, please use `selectHostGroups` instead. Return a groups property with host groups that the script can be run on.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve all scripts

Retrieve all configured scripts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.get",
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "scriptid": "1",
            "name": "Ping",
            "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
```

```
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "parameters": []
        },
        {
            "scriptid": "2",
            "name": "Traceroute",
            "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "parameters": []
        },
        {
            "scriptid": "3",
            "name": "Detect operating system",
            "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "7",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
```

```
            "url": "",
            "new_window": "1",
            "parameters": []
        },
        {
            "scriptid": "4",
            "name": "Webhook",
            "command": "try {\n var request = new HttpRequest(),\n response,\n data;\n\n request.addHeader
            "host_access": "2",
            "usrgrpid": "7",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "5",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "2",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "parameters": [
                {
                    "name": "token",
                    "value": "{$WEBHOOK.TOKEN}"
                },
                {
                    "name": "host",
                    "value": "{HOST.HOST}"
                },
                {
                    "name": "v",
                    "value": "2.2"
                }
            ]
        },
        {
            "scriptid": "5",
            "name": "URL",
            "command": "",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "Go to {HOST.NAME}?",
            "type": "6",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "http://zabbix/ui/zabbix.php?action=latest.view&hostids[]={HOST.ID}",
```

```
            "new_window": "0",
            "parameters": []
        }
    ],
    "id": 1
}
```

See also

- Host
- Host group

Source

CScript::get() in *ui/include/classes/api/services/CScript.php*.

script.getscriptsbyevents

Description

`object script.getscriptsbyevents(array eventIds)`

This method allows to retrieve scripts available to the given events.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(string/array)` IDs of events to return scripts for.

Return values

`(object)` Returns an object with event IDs as properties and arrays of available scripts as values.

Note:
The method will automatically expand macros in the `confirmation` text and `url`.

Examples

Retrieve scripts by event IDs

Retrieve all scripts available to events ”632” and ”614”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.getscriptsbyevents",
    "params": [
        "632",
        "614"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "632": [
            {
                "scriptid": "3",
                "name": "Detect operating system",
                "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
                "host_access": "2",
```

```
            "usrgrpid": "7",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "parameters": []
        },
        {
            "scriptid": "1",
            "name": "Ping",
            "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "parameters": []
        },
        {
            "scriptid": "2",
            "name": "Traceroute",
            "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
```

```
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "parameters": []
        }
    ],
    "614": [
        {
            "scriptid": "3",
            "name": "Detect operating system",
            "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "7",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "parameters": []
        },
        {
            "scriptid": "1",
            "name": "Ping",
            "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
            "description": "",
            "confirmation": "",
            "type": "0",
            "execute_on": "1",
            "timeout": "30s",
            "scope": "4",
            "port": "",
            "authtype": "0",
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "menu_path": "",
            "url": "",
            "new_window": "1",
            "parameters": []
        },
        {
            "scriptid": "2",
            "name": "Traceroute",
            "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
            "host_access": "2",
            "usrgrpid": "0",
            "groupid": "0",
```

```
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "4",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "parameters": []
            }
        ]
    },
    "id": 1
}
```


Source

CScript::getScriptsByEvents() in *ui/include/classes/api/services/CScript.php*.

script.getscriptsbyhosts

Description

`object script.getscriptsbyhosts(array hostIds)`

This method allows to retrieve scripts available on the given hosts.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(string/array)` IDs of hosts to return scripts for.

Return values

`(object)` Returns an object with host IDs as properties and arrays of available scripts as values.

> **Note:**
> The method will automatically expand macros in the `confirmation` text and `url`.

Examples

Retrieve scripts by host IDs

Retrieve all scripts available on hosts ”30079” and ”30073”.

Request:


```
{
    "jsonrpc": "2.0",
    "method": "script.getscriptsbyhosts",
    "params": [
        "30079",
        "30073"
    ],
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": {
        "30079": [
            {
                "scriptid": "3",
                "name": "Detect operating system",
                "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "7",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "2",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "parameters": []
            },
            {
                "scriptid": "1",
                "name": "Ping",
                "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "2",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "parameters": []
            },
            {
                "scriptid": "2",
                "name": "Traceroute",
                "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
                "description": "",
                "confirmation": "",
```

```
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "2",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "parameters": []
            }
        ],
        "30073": [
            {
                "scriptid": "3",
                "name": "Detect operating system",
                "command": "sudo /usr/bin/nmap -O {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "7",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "2",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "parameters": []
            },
            {
                "scriptid": "1",
                "name": "Ping",
                "command": "/bin/ping -c 3 {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "2",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
```

```
                "new_window": "1",
                "parameters": []
            },
            {
                "scriptid": "2",
                "name": "Traceroute",
                "command": "/usr/bin/traceroute {HOST.CONN} 2>&1",
                "host_access": "2",
                "usrgrpid": "0",
                "groupid": "0",
                "description": "",
                "confirmation": "",
                "type": "0",
                "execute_on": "1",
                "timeout": "30s",
                "scope": "2",
                "port": "",
                "authtype": "0",
                "username": "",
                "password": "",
                "publickey": "",
                "privatekey": "",
                "menu_path": "",
                "url": "",
                "new_window": "1",
                "parameters": []
            }
        ]
    },
    "id": 1
}
```


Source

CScript::getScriptsByHosts() in *ui/include/classes/api/services/CScript.php*.

script.update

Description

`object script.update(object/array scripts)`

This method allows to update existing scripts.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Script properties to be updated.

The `scriptid` property must be defined for each script, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged. An exception is `type` property change from 5 (Webhook) to other: the `parameters` property will be cleaned.

Return values

`(object)` Returns an object containing the IDs of the updated scripts under the `scriptids` property.

Examples

Change script command

Change the command of the script to "/bin/ping -c 10 {HOST.CONN} 2>&1".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "script.update",
    "params": {
        "scriptid": "1",
        "command": "/bin/ping -c 10 {HOST.CONN} 2>&1"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "1"
        ]
    },
    "id": 1
}
```

Source

CScript::update() in *ui/include/classes/api/services/CScript.php*.

Service

This class is designed to work with IT infrastructure/business services.

Object references:

- Service
- Status rule
- Service tag
- Service alarm
- Problem tag

Available methods:

- service.create - creating new services
- service.delete - deleting services
- service.get - retrieving services
- service.update - updating services

> Service object

The following objects are directly related to the `service` API.

Service

The service object has the following properties.

Property	Type	Description
serviceid	string	ID of the service. Property behavior: - *read-only* - *required* for update operations

Property	Type	Description
algorithm	integer	Status calculation rule. Only applicable if child services exist. Possible values: 0 - set status to OK; 1 - most critical if all children have problems; 2 - most critical of child services. Property behavior: - *required* for create operations
name	string	Name of the service. Property behavior: - *required* for create operations
sortorder	integer	Position of the service used for sorting. Possible values: 0-999. Property behavior: - *required* for create operations
weight	integer	Service weight. Possible values: 0-1000000. Default: 0.
propagation_rule	integer	Status propagation rule. Possible values: 0 - *(default)* propagate service status as is - without any changes; 1 - increase the propagated status by a given `propagation_value` (by 1 to 5 severities); 2 - decrease the propagated status by a given `propagation_value` (by 1 to 5 severities); 3 - ignore this service - the status is not propagated to the parent service at all; 4 - set fixed service status using a given `propagation_value`. Property behavior: - *required* if `propagation_value` is set
propagation_value	integer	Status propagation value. Possible values if `propagation_rule` is set to "0" or "3": 0 - Not classified. Possible values if `propagation_rule` is set to "1" or "2": 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster. Possible values if `propagation_rule` is set to "4": -1 - OK; 0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster. Property behavior: - *required* if `propagation_rule` is set

Property	Type	Description
status	integer	Whether the service is in OK or problem state. If the service is in problem state, `status` is equal either to: - the severity of the most critical problem; - the highest status of a child service in problem state. If the service is in OK state, `status` is equal to: -1. Property behavior: - *read-only*
description	string	Description of the service.
uuid	string	Universal unique identifier, used for linking imported services to already existing ones. Auto-generated, if not given.
created_at	integer	Unix timestamp when service was created.
readonly	boolean	Access to the service. Possible values: 0 - Read-write; 1 - Read-only. Property behavior: - *read-only*

Status rule

The status rule object has the following properties.

Property	Type	Description
type	integer	Condition for setting (New status) status. Possible values: 0 - if at least (N) child services have (Status) status or above; 1 - if at least (N%) of child services have (Status) status or above; 2 - if less than (N) child services have (Status) status or below; 3 - if less than (N%) of child services have (Status) status or below; 4 - if weight of child services with (Status) status or above is at least (W); 5 - if weight of child services with (Status) status or above is at least (N%); 6 - if weight of child services with (Status) status or below is less than (W); 7 - if weight of child services with (Status) status or below is less than (N%). Where: - N (W) is `limit_value`; - (Status) is `limit_status`; - (New status) is `new_status`. Property behavior: - *required*
limit_value	integer	Limit value. Possible values: - for N and W: 1-100000; - for N%: 1-100. Property behavior: - *required*

Property	Type	Description
limit_status	integer	Limit status. Possible values: -1 - OK; 0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster. Property behavior: - *required*
new_status	integer	New status value. Possible values: 0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster. Property behavior: - *required*

Service tag

The service tag object has the following properties.

Property	Type	Description
tag	string	Service tag name. Property behavior: - *required*
value	string	Service tag value.

Service alarm

Note:
Service alarms cannot be directly created, updated or deleted via the Zabbix API.

The service alarm objects represent a service's state change. It has the following properties.

Property	Type	Description
clock	timestamp	Time when the service state change has happened.
value	integer	Status of the service. Refer to the service status property for a list of possible values.

Problem tag

Problem tags allow linking services with problem events. The problem tag object has the following properties.

Property	Type	Description
tag	string	Problem tag name. Property behavior: - *required*
operator	integer	Mapping condition operator. Possible values: 0 - *(default)* equals; 2 - like.
value	string	Problem tag value.

service.create

Description

`object service.create(object/array services)`

This method allows to create new services.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` services to create.

Additionally to the standard service properties, the method accepts the following parameters.

Parameter	Type	Description
children	array	Child services to be linked to the service. The children must have the `serviceid` property defined.
parents	array	Parent services to be linked to the service. The parents must have the `serviceid` property defined.
tags	array	Service tags to be created for the service.
problem_tags	array	Problem tags to be created for the service.
status_rules	array	Status rules to be created for the service.

Return values

`(object)` Returns an object containing the IDs of the created services under the `serviceids` property. The order of the returned IDs matches the order of the passed services.

Examples

Creating a service

Create a service that will be switched to problem state, if at least one child has a problem.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.create",
    "params": {
        "name": "Server 1",
        "algorithm": 1,
        "sortorder": 1
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "5"
        ]
    },
    "id": 1
}
```

Source

CService::create() in *ui/include/classes/api/services/CService.php*.

service.delete

Description

`object service.delete(array serviceIds)`

This method allows to delete services.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the services to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted services under the `serviceids` property.

Examples

Deleting multiple services

Delete two services.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.delete",
    "params": [
        "4",
        "5"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

Source

CService::delete() in *ui/include/classes/api/services/CService.php*.

service.get

Description

`integer/array service.get(object parameters)`

The method allows to retrieve services according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
serviceids	string/array	Return only services with the given IDs.
parentids	string/array	Return only services that are linked to the given parent services.
deep_parentids	flag	Return all direct and indirect child services. Used together with `parentids`.
childids	string/array	Return only services that are linked to the given child services.
evaltype	integer	Rules for tag searching. Possible values: 0 - *(default)* And/Or; 2 - Or.
tags	object/array	Return only services with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all services. Possible operator values: 0 - *(default)* Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist.
problem_tags	object/array	Return only services with given problem tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all services. Possible operator values: 0 - *(default)* Contains; 1 - Equals; 2 - Does not contain; 3 - Does not equal; 4 - Exists; 5 - Does not exist.
without_problem_tags	flag	Return only services without problem tags.
slaids	string/array	Return only services that are linked to the specific SLA(s).
selectChildren	query	Return a `children` property with the child services. Supports `count`.

Parameter	Type	Description
selectParents	query	Return a `parents` property with the parent services. Supports `count`.
selectTags	query	Return a `tags` property with service tags. Supports `count`.
selectProblemEvents	query	Return a `problem_events` property with an array of problem event objects. The problem event object has the following properties: `eventid` - `(string)` Event ID; `severity` - `(string)` Current event severity; `name` - `(string)` Resolved event name. Supports `count`.
selectProblemTags	query	Return a `problem_tags` property with problem tags. Supports `count`.
selectStatusRules	query	Return a `status_rules` property with status rules. Supports `count`.
selectStatusTimeline	object/array	Return a `status_timeline` property containing service state changes for given periods. Format `[{"period_from": "<period_from>", "period_to": "<period_to>"}, ...]` - `period_from` being a starting date (inclusive; integer timestamp) and `period_to` being an ending date (exclusive; integer timestamp) for the period you're interested in. Returns an array of entries containing a `start_value` property and an alarms array for the state changes within specified periods.
sortfield	string/array	Sort the result by the given properties. Possible values: `serviceid`, `name`, `status`, `sortorder`, `created_at`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving all services

Retrieve all data about all services and their dependencies.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.get",
    "params": {
        "output": "extend",
        "selectChildren": "extend",
        "selectParents": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "serviceid": "1",
            "name": "My Service - 0001",
            "status": "-1",
            "algorithm": "2",
            "sortorder": "0",
            "weight": "0",
            "propagation_rule": "0",
            "propagation_value": "0",
            "description": "My Service Description 0001.",
            "uuid": "dfa4daeaea754e3a95c04d6029182681",
            "created_at": "946684800",
            "readonly": false,
            "parents": [],
            "children": []
        },
        {
            "serviceid": "2",
            "name": "My Service - 0002",
            "status": "-1",
            "algorithm": "2",
            "sortorder": "0",
            "weight": "0",
            "propagation_rule": "0",
            "propagation_value": "0",
            "description": "My Service Description 0002.",
            "uuid": "20ea0d85212841219130abeaca28c065",
            "created_at": "946684800",
            "readonly": false,
            "parents": [],
            "children": []
        }
    ],
    "id": 1
}
```

Source

CService::get() in *ui/include/classes/api/services/CService.php*.

service.update

Description

`object service.update(object/array services)`

This method allows to update existing services.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` service properties to be updated.

The `serviceid` property must be defined for each service, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard service properties, the method accepts the following parameters.

Parameter	Type	Description
children	array	Child services to replace the current service children. The children must have the `serviceid` property defined.
parents	array	Parent services to replace the current service parents. The parents must have the `serviceid` property defined.
tags	array	Service tags to replace the current service tags.
problem_tags	array	Problem tags to replace the current problem tags.
status_rules	array	Status rules to replace the current status rules.

Return values

`(object)` Returns an object containing the IDs of the updated services under the `serviceids` property.

Examples

Setting the parent for a service

Make service with ID "3" to be the parent for service with ID "5".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "serviceid": "5",
        "parents": [
            {
                "serviceid": "3"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "5"
        ]
    },
    "id": 1
}
```

Adding a scheduled downtime

Add a downtime for service with ID "4" scheduled weekly from Monday 22:00 till Tuesday 10:00.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "serviceid": "4",
        "times": [
            {
                "type": "1",
                "ts_from": "165600",
                "ts_to": "201600"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "serviceids": [
            "4"
        ]
    },
    "id": 1
}
```

Source

CService::update() in *ui/include/classes/api/services/CService.php*.

Settings

This class is designed to work with common administration settings.

Object references:

- Settings

Available methods:

- settings.get - retrieve settings
- settings.update - update settings

> Settings object

The following objects are directly related to the `settings` API.

Settings

The settings object has the following properties.

Property	Type	Description
default_lang	string	System language by default. Default: `en_US`.
default_timezone	string	System time zone by default. Default: `system` - system default. For the full list of supported time zones please refer to PHP documentation.

Property	Type	Description
default_theme	string	Default theme. Possible values: `blue-theme` - *(default)* Blue; `dark-theme` - Dark; `hc-light` - High-contrast light; `hc-dark` - High-contrast dark.
search_limit	integer	Limit for search and filter results. Default: 1000.
max_overview_table_size	integer	Max number of columns and rows in Data overview and Trigger overview dashboard widgets. Default: 50.
max_in_table	integer	Max count of elements to show inside table cell. Default: 50.
server_check_interval	integer	Show warning if Zabbix server is down. Possible values: 0 - Do not show warning; 10 - *(default)* Show warning.
work_period	string	Working time. Default: 1-5,09:00-18:00.
show_technical_errors	integer	Show technical errors (PHP/SQL) to non-Super admin users and to users that are not part of user groups with debug mode enabled. Possible values: 0 - *(default)* Do not technical errors; 1 - Show technical errors.
history_period	string	Max period to display history data in Latest data, Web, and Data overview dashboard widgets. Accepts seconds and time unit with suffix. Default: 24h.
period_default	string	Time filter default period. Accepts seconds and time unit with suffix with month and year support (30s, 1m, 2h, 1d, 1M, 1y). Default: 1h.
max_period	string	Max period for time filter. Accepts seconds and time unit with suffix with month and year support (30s, 1m, 2h, 1d, 1M, 1y). Default: 2y.
severity_color_0	string	Color for ”Not classified” severity as a hexadecimal color code. Default: 97AAB3.
severity_color_1	string	Color for ”Information” severity as a hexadecimal color code. Default: 7499FF.
severity_color_2	string	Color for ”Warning” severity as a hexadecimal color code. Default: FFC859.
severity_color_3	string	Color for ”Average” severity as a hexadecimal color code. Default: FFA059.
severity_color_4	string	Color for ”High” severity as a hexadecimal color code. Default: E97659.

Property	Type	Description
severity_color_5	string	Color for ”Disaster” severity as a hexadecimal color code. Default: E45959.
severity_name_0	string	Name for ”Not classified” severity. Default: Not classified.
severity_name_1	string	Name for ”Information” severity. Default: Information.
severity_name_2	string	Name for ”Warning” severity. Default: Warning.
severity_name_3	string	Name for ”Average” severity. Default: Average.
severity_name_4	string	Name for ”High” severity. Default: High.
severity_name_5	string	Name for ”Disaster” severity. Default: Disaster.
custom_color	integer	Use custom event status colors. Possible values: 0 - *(default)* Do not use custom event status colors; 1 - Use custom event status colors.
ok_period	string	Display OK triggers period. Accepts seconds and time unit with suffix. Default: 5m.
blink_period	string	On status change triggers blink period. Accepts seconds and time unit with suffix. Default: 2m.
problem_unack_color	string	Color for unacknowledged PROBLEM events as a hexadecimal color code. Default: CC0000.
problem_ack_color	string	Color for acknowledged PROBLEM events as a hexadecimal color code. Default: CC0000.
ok_unack_color	string	Color for unacknowledged RESOLVED events as a hexadecimal color code. Default: 009900.
ok_ack_color	string	Color for acknowledged RESOLVED events as a hexadecimal color code. Default: 009900.
problem_unack_style	integer	Blinking for unacknowledged PROBLEM events. Possible values: 0 - Do not show blinking; 1 - *(default)* Show blinking.
problem_ack_style	integer	Blinking for acknowledged PROBLEM events. Possible values: 0 - Do not show blinking; 1 - *(default)* Show blinking.

Property	Type	Description
ok_unack_style	integer	Blinking for unacknowledged RESOLVED events. Possible values: 0 - Do not show blinking; 1 - *(default)* Show blinking.
ok_ack_style	integer	Blinking for acknowledged RESOLVED events. Possible values: 0 - Do not show blinking; 1 - *(default)* Show blinking.
url	string	Frontend URL.
discovery_groupid	integer	ID of the host group to which will be automatically placed discovered hosts.
default_inventory_mode	integer	Default host inventory mode. Possible values: -1 - *(default)* Disabled; 0 - Manual; 1 - Automatic.
alert_usrgrpid	integer	ID of the user group to which will be sending database down alarm message. If set to ”0”, the alarm message will not be sent.
snmptrap_logging	integer	Log unmatched SNMP traps. Possible values: 0 - Do not log unmatched SNMP traps; 1 - *(default)* Log unmatched SNMP traps.
login_attempts	integer	Number of failed login attempts after which login form will be blocked. Default: 5.
login_block	string	Time interval during which login form will be blocked if number of failed login attempts exceeds defined in login_attempts field. Accepts seconds and time unit with suffix. Default: 30s.
validate_uri_schemes	integer	Validate URI schemes. Possible values: 0 - Do not validate; 1 - *(default)* Validate.
uri_valid_schemes	string	Valid URI schemes. Default: http,https,ftp,file,mailto,tel,ssh.
x_frame_options	string	X-Frame-Options HTTP header. Default: SAMEORIGIN.
iframe_sandboxing_enabled	integer	Use iframe sandboxing. Possible values: 0 - Do not use; 1 - *(default)* Use.
iframe_sandboxing_exceptions	string	Iframe sandboxing exceptions.
connect_timeout	string	Connection timeout with Zabbix server. Default: 3s.
socket_timeout	string	Network default timeout. Default: 3s.
media_type_test_timeout	string	Network timeout for media type test. Default: 65s.

Property	Type	Description
item_test_timeout	string	Network timeout for item tests. Default: 60s.
script_timeout	string	Network timeout for script execution. Default: 60s.
report_test_timeout	string	Network timeout for scheduled report test. Default: 60s.
auditlog_enabled	integer	Enable audit logging. Possible values: 0 - Disable; 1 - *(default)* Enable.
ha_failover_delay	string	Failover delay in seconds. Default: 1m.
geomaps_tile_provider	string	Property behavior: - *read-only* Geomap tile provider. Possible values: `OpenStreetMap.Mapnik` - *(default)* OpenStreetMap Mapnik; `OpenTopoMap` - OpenTopoMap; `Stamen.TonerLite` - Stamen Toner Lite; `Stamen.Terrain` - Stamen Terrain; `USGS.USTopo` - USGS US Topo; `USGS.USImagery` - USGS US Imagery. Supports empty string to specify custom values of `geomaps_tile_url`, `geomaps_max_zoom` and `geomaps_attribution`.
geomaps_tile_url	string	Geomap tile URL. Property behavior: - *supported* if `geomaps_tile_provider` is set to empty string
geomaps_max_zoom	integer	Geomap max zoom level. Possible values: 0-30. Property behavior: - *supported* if `geomaps_tile_provider` is set to empty string
geomaps_attribution	string	Geomap attribution text. Property behavior: - *supported* if `geomaps_tile_provider` is set to empty string
vault_provider	integer	Vault provider. Possible values: 0 - *(default)* HashiCorp Vault; 1 - CyberArk Vault.

settings.get

Description

`object settings.get(object parameters)`

The method allows to retrieve settings object according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports only one parameter.

Parameter	Type	Description
output	query	This parameter being common for all `get` methods described in the reference commentary.

Return values

`(object)` Returns settings object.

Examples

Request:

```
{
    "jsonrpc": "2.0",
    "method": "settings.get",
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "default_theme": "blue-theme",
        "search_limit": "1000",
        "max_in_table": "50",
        "server_check_interval": "10",
        "work_period": "1-5,09:00-18:00",
        "show_technical_errors": "0",
        "history_period": "24h",
        "period_default": "1h",
        "max_period": "2y",
        "severity_color_0": "97AAB3",
        "severity_color_1": "7499FF",
        "severity_color_2": "FFC859",
        "severity_color_3": "FFA059",
        "severity_color_4": "E97659",
        "severity_color_5": "E45959",
        "severity_name_0": "Not classified",
        "severity_name_1": "Information",
        "severity_name_2": "Warning",
        "severity_name_3": "Average",
        "severity_name_4": "High",
        "severity_name_5": "Disaster",
        "custom_color": "0",
        "ok_period": "5m",
        "blink_period": "2m",
        "problem_unack_color": "CC0000",
        "problem_ack_color": "CC0000",
        "ok_unack_color": "009900",
        "ok_ack_color": "009900",
        "problem_unack_style": "1",
```

```
        "problem_ack_style": "1",
        "ok_unack_style": "1",
        "ok_ack_style": "1",
        "discovery_groupid": "5",
        "default_inventory_mode": "-1",
        "alert_usrgrpid": "7",
        "snmptrap_logging": "1",
        "default_lang": "en_US",
        "default_timezone": "system",
        "login_attempts": "5",
        "login_block": "30s",
        "validate_uri_schemes": "1",
        "uri_valid_schemes": "http,https,ftp,file,mailto,tel,ssh",
        "x_frame_options": "SAMEORIGIN",
        "iframe_sandboxing_enabled": "1",
        "iframe_sandboxing_exceptions": "",
        "max_overview_table_size": "50",
        "connect_timeout": "3s",
        "socket_timeout": "3s",
        "media_type_test_timeout": "65s",
        "script_timeout": "60s",
        "item_test_timeout": "60s",
        "url": "",
        "report_test_timeout": "60s",
        "auditlog_enabled": "1",
        "ha_failover_delay": "1m",
        "geomaps_tile_provider": "OpenStreetMap.Mapnik",
        "geomaps_tile_url": "",
        "geomaps_max_zoom": "0",
        "geomaps_attribution": "",
        "vault_provider": "0"
    },
    "id": 1
}
```


Source

CSettings::get() in *ui/include/classes/api/services/CSettings.php*.

settings.update

Description

`object settings.update(object settings)`

This method allows to update existing common settings.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Settings properties to be updated.

Return values

`(array)` Returns an array with the names of updated parameters.

Examples

Request:


```
{
    "jsonrpc": "2.0",
    "method": "settings.update",
```

```
    "params": {
        "login_attempts": "1",
        "login_block": "1m"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        "login_attempts",
        "login_block"
    ],
    "id": 1
}
```

Source

CSettings::update() in *ui/include/classes/api/services/CSettings.php*.

SLA

This class is designed to work with SLA (Service Level Agreement) objects used to estimate the performance of IT infrastructure and business services.

Object references:

- SLA
- SLA schedule
- SLA excluded downtime
- SLA service tag

Available methods:

- sla.create - creating new SLAs
- sla.delete - deleting SLAs
- sla.get - retrieving SLAs
- sla.getsli - retrieving Service Level Indicator (SLI) data for SLAs
- sla.update - updating SLAs

> SLA object

The following objects are directly related to the `sla` (Service Level Agreement) API.

SLA

The SLA object has the following properties.

Property	Type	Description
slaid	string	ID of the SLA. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the SLA. Property behavior: - *required* for create operations

Property	Type	Description
period	integer	Reporting period of the SLA. Possible values: 0 - daily; 1 - weekly; 2 - monthly; 3 - quarterly; 4 - annually. Property behavior: - *required* for create operations
slo	float	Minimum acceptable Service Level Objective expressed as a percent. If the Service Level Indicator (SLI) drops lower, the SLA is considered to be in problem/unfulfilled state. Possible values: 0-100 (up to 4 fractional digits). Property behavior: - *required* for create operations
effective_date	integer	Effective date of the SLA. Possible values: date timestamp in UTC.
timezone	string	Reporting time zone, for example: `Europe/London`, `UTC`. For the full list of supported time zones please refer to PHP documentation. Property behavior: - *required* for create operations
status	integer	Status of the SLA. Possible values: 0 - *(default)* disabled SLA; 1 - enabled SLA.
description	string	Description of the SLA.

SLA Schedule

The SLA schedule object defines periods where the connected service(s) are scheduled to be in working order. It has the following properties.

Property	Type	Description
period_from	integer	Starting time of the recurrent weekly period of time (inclusive). Possible values: number of seconds (counting from Sunday). Property behavior: - *required*
period_to	integer	Ending time of the recurrent weekly period of time (exclusive). Possible values: number of seconds (counting from Sunday). Property behavior: - *required*

SLA excluded downtime

The excluded downtime object defines periods where the connected service(s) are scheduled to be out of working order, without affecting SLI, e.g., undergoing planned maintenance. It has the following properties.

Property	Type	Description
name	string	Name of the excluded downtime. Property behavior: - *required*
period_from	integer	Starting time of the excluded downtime (inclusive). Possible values: timestamp. Property behavior: - *required*
period_to	integer	Ending time of the excluded downtime (exclusive). Possible values: timestamp. Property behavior: - *required*

SLA service tag

The SLA service tag object links services to include in the calculations for the SLA. It has the following properties.

Property	Type	Description
tag	string	SLA service tag name. Property behavior: - *required*
operator	integer	SLA service tag operator. Possible values: 0 - *(default)* equals; 2 - contains.
value	string	SLA service tag value.

sla.create

Description

`object sla.create(object/array SLAs)`

This method allows to create new SLA objects.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` SLA objects to create.

Additionally to the standard SLA properties, the method accepts the following parameters.

Parameter	Type	Description
service_tags	array	SLA service tags to be created for the SLA. Parameter behavior: - *required*
schedule	array	SLA schedule to be created for the SLA. Specifying an empty parameter will be interpreted as a 24x7 schedule. Default: 24x7 schedule.
excluded_downtimes	array	SLA excluded downtimes to be created for the SLA.

Return values

(object) Returns an object containing the IDs of the created SLAs under the slaids property. The order of the returned IDs matches the order of the passed SLAs.

Examples

Creating an SLA

Instruct to create an SLA entry for: * tracking uptime for SQL-engine related services; * custom schedule of all weekdays excluding last hour on Saturday; * an effective date of the last day of the year 2022; * with 1 hour and 15 minutes long planned downtime starting at midnight on the 4th of July; * SLA weekly report calculation will be on; * the minimum acceptable SLO will be 99.9995%.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "sla.create",
    "params": [
        {
            "name": "Database Uptime",
            "slo": "99.9995",
            "period": "1",
            "timezone": "America/Toronto",
            "description": "Provide excellent uptime for main database engines.",
            "effective_date": 1672444800,
            "status": 1,
            "schedule": [
                {
                    "period_from": 0,
                    "period_to": 601200
                }
            ],
            "service_tags": [
                {
                    "tag": "Database",
                    "operator": "0",
                    "value": "MySQL"
                },
                {
                    "tag": "Database",
                    "operator": "0",
                    "value": "PostgreSQL"
                }
            ],
            "excluded_downtimes": [
                {
                    "name": "Software version upgrade rollout",
                    "period_from": "1648760400",
                    "period_to": "1648764900"
                }
            ]
        }
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "5"
        ]
    },
    "id": 1
```

```
}
```

Source

CSla::create() in *ui/include/classes/api/services/CSla.php*.

sla.delete

Description

`object sla.delete(array slaids)`

This method allows to delete SLA entries.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the SLAs to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted SLAs under the `slaids` property.

Examples

Deleting multiple SLAs

Delete two SLA entries.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "sla.delete",
    "params": [
        "4",
        "5"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "4",
            "5"
        ]
    },
    "id": 1
}
```

Source

CSla::delete() in *ui/include/classes/api/services/CSla.php*.

sla.get

Description

`integer/array sla.get(object parameters)`

The method allows to retrieve SLA objects according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
slaids	string/array	Return only SLAs with the given IDs.
serviceids	string/array	Return only SLAs matching the specific services.
selectSchedule	query	Return a `schedule` property with SLA schedules. Supports `count`.
selectExcludedDowntimes	query	Return an `excluded_downtimes` property with SLA excluded downtimes. Supports `count`.
selectServiceTags	query	Return a `service_tags` property with SLA service tags. Supports `count`.
sortfield	string/array	Sort the result by the given properties. Possible values: `slaid`, `name`, `period`, `slo`, `effective_date`, `timezone`, `status`, `description`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving all SLAs

Retrieve all data about all SLAs and their properties.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "sla.get",
    "params": {
        "output": "extend",
        "selectSchedule": ["period_from", "period_to"],
        "selectExcludedDowntimes": ["name", "period_from", "period_to"],
        "selectServiceTags": ["tag", "operator", "value"],
        "preservekeys": true
```

```
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "1": {
            "slaid": "1",
            "name": "Database Uptime",
            "period": "1",
            "slo": "99.9995",
            "effective_date": "1672444800",
            "timezone": "America/Toronto",
            "status": "1",
            "description": "Provide excellent uptime for main SQL database engines.",
            "service_tags": [
                {
                    "tag": "Database",
                    "operator": "0",
                    "value": "MySQL"
                },
                {
                    "tag": "Database",
                    "operator": "0",
                    "value": "PostgreSQL"
                }
            ],
            "schedule": [
                {
                    "period_from": "0",
                    "period_to": "601200"
                }
            ],
            "excluded_downtimes": [
                {
                    "name": "Software version upgrade rollout",
                    "period_from": "1648760400",
                    "period_to": "1648764900"
                }
            ]
        }
    },
    "id": 1
}
```

Source

CSla:get() in *ui/include/classes/api/services/CSla.php*.

sla.getsli

Description

`object sla.getsli(object parameters)`

This method allows to calculate the Service Level Indicator (SLI) data for a Service Level Agreement (SLA).

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the SLA ID, reporting periods and, optionally, the IDs of the services - to calculate the SLI for.

Parameter	Type	Description
slaid	string	ID of the SLA to return availability information for. Parameter behavior: - *required*
period_from	timestamp	Starting date (inclusive) to report the SLI for. Possible values: timestamp.
period_to	timestamp	Ending date (exclusive) to report the SLI for. Possible values: timestamp.
periods	array	Preferred number of periods to report. Possible values: 1-100
serviceids	string/array	IDs of services to return the SLI for.

Partitioning of periods

The following table demonstrates the arrangement of returned period slices based on combinations of parameters.

Note:
The returned periods will not precede the first available period based on the effective date of the SLA and will not exceed the current period.

Parameters			Description
period_from	**period_to**	**periods**	
-	-	-	Return the last 20 periods.
-	-	specified	Return the last periods specified by the **periods** parameter.
-	specified	-	Return the last 20 periods before the specified **period_to** date.
-	specified	specified	Return the last periods specified by the **periods** parameter before the specified **period_to** date.
specified	-	-	Return the first 20 periods starting with the specified **period_from** date.
specified	-	specified	Return the first periods specified by the **periods** parameter starting with the specified **period_from** date.
specified	specified	-	Return up to 100 periods within the specified date range.
specified	specified	specified	Return periods specified by the **periods** parameter within the specified date range.

Return values

`(object)` Returns the results of the calculation.

Property	Type	Description
periods	array	List of the reported periods. Each reported period is represented as an object consisting of: - `period_from` - Starting date of the reported period (timestamp). - `period_to` - Ending date of the reported period (timestamp). Periods are sorted by `period_from` field ascending.

Property	Type	Description
serviceids	array	List of service IDs in the reported periods. The sorting order of the list is not defined. Even if `serviceids` parameter was passed to the `sla.getsli` method.
sli	array	SLI data (as a **two-dimensional array**) for each reported period and service. The index of the `periods` property is used as the **first** dimension of the `sli` property. The index of the `serviceids` property is used as the **second** dimension of the `sli` property.

SLI data

The SLI data returned for each reported period and service consists of:

Property	Type	Description
uptime	integer	Amount of time service spent in an *OK* state during scheduled uptime, less the excluded downtimes.
downtime	integer	Amount of time service spent in a *not OK* state during scheduled uptime, less the excluded downtimes.
sli	float	SLI (per cent of total uptime), based on uptime and downtime.
error_budget	integer	Error budget (in seconds), based on the SLI and the SLO.
excluded_downtimes	array	Array of excluded downtimes in this reporting period. Each object will contain the following parameters: - `name` - Name of the excluded downtime. - `period_from` - Starting date and time (inclusive) of the excluded downtime. - `period_to` - Ending date and time (exclusive) of the excluded downtime. Excluded downtimes are sorted by period_from field ascending.

Examples

Calculating SLI

Retrieve SLI data on services with IDs ”50”, ”60” and ”70” that are linked to the SLA with ID ”5”. Retrieve data for 3 periods starting from Nov 01, 2021.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "sla.getsli",
    "params": {
        "slaid": "5",
        "serviceids": [
            50,
            60,
            70
        ],
        "periods": 3,
        "period_from": "1635724800"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "periods": [
            {
                "period_from": 1635724800,
                "period_to": 1638316800
            },
            {
                "period_from": 1638316800,
                "period_to": 1640995200
            },
            {
                "period_from": 1640995200,
                "period_to": 1643673600
            }
        ],
        "serviceids": [
            50,
            60,
            70
        ],
        "sli": [
            [
                {
                    "uptime": 1186212,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": [
                        {
                            "name": "Excluded Downtime - 1",
                            "period_from": 1637836212,
                            "period_to": 1638316800
                        }
                    ]
                },
                {
                    "uptime": 1186212,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": [
                        {
                            "name": "Excluded Downtime - 1",
                            "period_from": 1637836212,
                            "period_to": 1638316800
                        }
                    ]
                },
                {
                    "uptime": 1186212,
                    "downtime": 0,
                    "sli": 100,
                    "error_budget": 0,
                    "excluded_downtimes": [
                        {
                            "name": "Excluded Downtime - 1",
                            "period_from": 1637836212,
                            "period_to": 1638316800
                        }
                    ]
```

```
            }
        ],
        [
            {
                "uptime": 1147548,
                "downtime": 0,
                "sli": 100,
                "error_budget": 0,
                "excluded_downtimes": [
                    {
                        "name": "Excluded Downtime - 1",
                        "period_from": 1638439200,
                        "period_to": 1639109652
                    }
                ]
            },
            {
                "uptime": 1147548,
                "downtime": 0,
                "sli": 100,
                "error_budget": 0,
                "excluded_downtimes": [
                    {
                        "name": "Excluded Downtime - 1",
                        "period_from": 1638439200,
                        "period_to": 1639109652
                    }
                ]
            },
            {
                "uptime": 1147548,
                "downtime": 0,
                "sli": 100,
                "error_budget": 0,
                "excluded_downtimes": [
                    {
                        "name": "Excluded Downtime - 1",
                        "period_from": 1638439200,
                        "period_to": 1639109652
                    }
                ]
            }
        ],
        [
            {
                "uptime": 1674000,
                "downtime": 0,
                "sli": 100,
                "error_budget": 0,
                "excluded_downtimes": []
            },
            {
                "uptime": 1674000,
                "downtime": 0,
                "sli": 100,
                "error_budget": 0,
                "excluded_downtimes": []
            },
            {
                "uptime": 1674000,
                "downtime": 0,
                "sli": 100,
```

```
                    "error_budget": 0,
                    "excluded_downtimes": []
                }
            ]
        ]
    },
    "id": 1
}
```

Source

CSla::getSli() in *ui/include/classes/api/services/CSla.php*

sla.update

Description

`object sla.update(object/array slaids)`

This method allows to update existing SLA entries.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` SLA properties to be updated.

The `slaid` property must be defined for each SLA, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard SLA properties, the method accepts the following parameters.

Parameter	Type	Description
service_tags	array	SLA service tags to replace the current SLA service tags.
schedule	array	SLA schedule to replace the current one. Specifying parameter as empty will be interpreted as a 24x7 schedule.
excluded_downtimes	array	SLA excluded downtimes to replace the current ones.

Return values

`(object)` Returns an object containing the IDs of the updated SLAs under the `slaids` property.

Examples

Updating service tags

Make SLA with ID "5" to be calculated at monthly intervals for NoSQL related services, without changing its schedule or excluded downtimes; set SLO to 95%.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "sla.update",
    "params": [
        {
            "slaid": "5",
            "name": "NoSQL Database engines",
            "slo": "95",
            "period": 2,
            "service_tags": [
                {
                    "tag": "Database",
```

```
                    "operator": "0",
                    "value": "Redis"
                },
                {
                    "tag": "Database",
                    "operator": "0",
                    "value": "MongoDB"
                }
            ]
        }
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "5"
        ]
    },
    "id": 1
}
```

Changing the schedule of an SLA

Switch the SLA with ID ”5” to a 24x7 schedule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "slaid": "5",
        "schedule": []
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "5"
        ]
    },
    "id": 1
}
```

Changing the excluded downtimes for an SLA

Add a planned 4 hour long RAM upgrade downtime on the 6th of April, 2022, while keeping (needs to be defined anew) a previously existing software upgrade planned on the 4th of July for the SLA with ID ”5”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "service.update",
    "params": {
        "slaid": "5",
        "excluded_downtimes": [
            {
```

```
                "name": "Software version upgrade rollout",
                "period_from": "1648760400",
                "period_to": "1648764900"
            },
            {
                "name": "RAM upgrade",
                "period_from": "1649192400",
                "period_to": "1649206800"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "slaids": [
            "5"
        ]
    },
    "id": 1
}
```

Source

CSla::update() in *ui/include/classes/api/services/CSla.php*.

Task

This class is designed to work with tasks (such as checking items or low-level discovery rules without config reload).

Object references:

- Task
- 'Execute now' request object
- 'Diagnostic information' request object
- Statistic request object
- Statistic result object

Available methods:

- task.create - creating new tasks
- task.get - retrieving tasks

> Task object

The following objects are directly related to the `task` API.

The task object has the following properties:

Property	Type	Description
taskid	string	ID of the task. Property behavior: - *read-only*

Property	Type	Description
type	integer	Type of the task. Possible values: 1 - Diagnostic information; 2 - Refresh proxy configuration; 6 - Execute now. Property behavior: - *required*
status	integer	Status of the task. Possible values: 1 - new task; 2 - task in progress; 3 - task is completed; 4 - task is expired. Property behavior: - *read-only*
clock	timestamp	Time when the task was created. Property behavior: - *read-only*
ttl	integer	The time in seconds after which task expires. Property behavior: - *read-only*
proxy_hostid	string	ID of the proxy about which diagnostic information statistic is collected. Property behavior: - *supported* if `type` is set to ”Diagnostic information” or ”Refresh proxy configuration”
request	object	Task request object according to the task type: Object of ’Execute now’ task is described in detail below; Object of ’Refresh proxy configuration’ task is described in detail below; Object of ’Diagnostic information’ task is described in detail below. Property behavior: - *required*
result	object	Result object of the diagnostic information task. May contain NULL if result is not yet ready. Result object is described in detail below. Property behavior: - *read-only*

’Execute now’ request object

The ’Execute now’ task request object has the following properties.

Property	Type	Description
itemid	string	ID of item and low-level discovery rules.

’Refresh proxy configuration’ request object

The ’Refresh proxy configuration’ task request object has the following properties.

Property	Type	Description
proxy_hostids	array	Proxy IDs.

'Diagnostic information' request object

The diagnostic information task request object has the following properties. Statistic request object for all types of properties is described in detail below.

Property	Type	Description
historycache	object	History cache statistic request. Available on server and proxy.
valuecache	object	Items cache statistic request. Available on server.
preprocessing	object	Preprocessing manager statistic request. Available on server and proxy.
alerting	object	Alert manager statistic request. Available on server.
lld	object	LLD manager statistic request. Available on server.

Statistic request object

Statistic request object is used to define what type of information should be collected about server/proxy internal processes. It has the following properties.

Property	Type	Description
stats	query	Statistic object properties to be returned. The list of available fields for each type of diagnostic information statistic are described in detail below. Default: `extend` will return all available statistic fields.
top	object	Object to sort and limit returned statistic values. The list of available fields for each type of diagnostic information statistic are described in detail below. Example: { "source.alerts": 10 }

List of statistic fields available for each type of diagnostic information request

Following statistic fields can be requested for each type of diagnostic information request property.

Diagnostic type	Available fields	Description
historycache	items	Number of cached items.
	values	Number of cached values.
	memory	Shared memory statistics (free space, number of used chunks, number of free chunks, max size of free chunk).
	memory.data	History data cache shared memory statistics.
	memory.index	History index cache shared memory statistics.
valuecache	items	Number of cached items.
	values	Number of cached values.
	memory	Shared memory statistics (free space, number of used chunks, number of free chunks, max size of free chunk).
	mode	Value cache mode.
preprocessing	values	Number of queued values.
	preproc.values	Number of queued values with preprocessing steps.
alerting	alerts	Number of queued alerts.
lld	rules	Number of queued rules.
	values	Number of queued values.

List of sorting fields available for each type of diagnostic information request

Following statistic fields can be used to sort and limit requested information.

Diagnostic type	Available fields	Type
historycache	values	integer
valuecache	values	integer
	request.values	integer
preprocessing	values	integer

Diagnostic type	Available fields	Type
alerting	media.alerts	integer
	source.alerts	integer
lld	values	integer

Statistic result object

Statistic result object is retrieved in `result` field of task object.

Property	Type	Description
status	integer	Status of the task result. Possible values: -1 - error occurred during performing task; 0 - task result is created. Property behavior: - *read-only*
data	string/object	Results according the statistic request object of particular diagnostic information task. Contains error message string if error occurred during performing task.

task.create

Description

`object task.create(object/array tasks)`

This method allows to create a new task (such as collect diagnostic data or check items or low-level discovery rules without config reload).

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` A task to create.

The method accepts tasks with the standard task properties.

Note that 'Execute now' tasks can be created only for the following types of items/discovery rules:

- Zabbix agent
- SNMPv1/v2/v3 agent
- Simple check
- Internal check
- External check
- Database monitor
- HTTP agent
- IPMI agent
- SSH agent
- TELNET agent
- Calculated check
- JMX agent
- Dependent item

If item or discovery rule is of type "Dependent item", then top level master item must be of type:

- Zabbix agent
- SNMPv1/v2/v3 agent
- Simple check
- Internal check
- External check

- Database monitor
- HTTP agent
- IPMI agent
- SSH agent
- TELNET agent
- Calculated check
- JMX agent

Return values

`(object)` Returns an object containing the IDs of the created tasks under the `taskids` property. One task is created for each item and low-level discovery rule. The order of the returned IDs matches the order of the passed `itemids`.

Examples

Creating a task

Create a task `Execute now` for two items. One is an item, the other is a low-level discovery rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": [
        {
            "type": 6,
            "request": {
                "itemid": "10092"
            }
        },
        {
            "type": 6,
            "request": {
                "itemid": "10093"
            }
        }
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

Create a task `Refresh proxy configuration` for two proxies.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": [
        {
            "type": 2,
            "request": {
                "proxy_hostids": ["10459", "10460"]
            }
        }
    ],
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "1"
        ]
    },
    "id": 1
}
```

Create a task `diagnostic information`.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "task.create",
    "params": [
        {
            "type": 1,
            "request": {
                "alerting": {
                    "stats": [
                        "alerts"
                    ],
                    "top": {
                        "media.alerts": 10
                    }
                },
                "lld": {
                    "stats": "extend",
                    "top": {
                        "values": 5
                    }
                }
            },
            "proxy_hostid": 0
        }
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Task
- 'Execute now' request object
- 'Diagnostic information' request object
- Statistic request object

Source

CTask::create() in *ui/include/classes/api/services/CTask.php*.

task.get

Description

`integer/array task.get(object parameters)`

The method allows to retrieve tasks according to the given parameters. Method returns details only about 'diagnostic information' tasks.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
taskids	string/array	Return only tasks with the given IDs.
output	query	These parameters being common for all `get` methods are described in detail in the reference commentary.
preservekeys	boolean	

Return values

`(integer/array)` Returns an array of objects.

Examples

Retrieve task by ID

Retrieve all the data about the task with the ID "1".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "task.get",
    "params": {
        "output": "extend",
        "taskids": "1"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "taskid": "1",
            "type": "7",
            "status": "3",
            "clock": "1601039076",
            "ttl": "3600",
            "proxy_hostid": null,
            "request": {
                "alerting": {
                    "stats": [
                        "alerts"
                    ],
                    "top": {
```

```
                        "media.alerts": 10
                    }
                },
                "lld": {
                    "stats": "extend",
                    "top": {
                        "values": 5
                    }
                }
            },
            "result": {
                "data": {
                    "alerting": {
                        "alerts": 0,
                        "top": {
                            "media.alerts": []
                        },
                        "time": 0.000663
                    },
                    "lld": {
                        "rules": 0,
                        "values": 0,
                        "top": {
                            "values": []
                        },
                        "time": 0.000442
                    }
                },
                "status": "0"
            }
        }
    ],
    "id": 1
}
```


See also

- Task
- Statistic result object

Source

CTask::get() in *ui/include/classes/api/services/CTask.php*.

Template

This class is designed to work with templates.

Object references:

- Template

Available methods:

- template.create - creating new templates
- template.delete - deleting templates
- template.get - retrieving templates
- template.massadd - adding related objects to templates
- template.massremove - removing related objects from templates
- template.massupdate - replacing or removing related objects from templates
- template.update - updating templates

> Template object

The following objects are directly related to the `template` API.

Template

The template object has the following properties.

Property	Type	Description
templateid	string	ID of the template. Property behavior: - *read-only* - *required* for update operations
host	string	Technical name of the template. Property behavior: - *required* for create operations
description	text	Description of the template.
name	string	Visible name of the template. Default: `host` property value.
uuid	string	Universal unique identifier, used for linking imported templates to already existing ones. Auto-generated, if not given.
vendor_name	string	Template vendor name. For create operations, both `vendor_name` and `vendor_version` should be either set or left empty. For update operations, `vendor_version` can be left empty if it has a value in the database.
vendor_version	string	Template vendor version. For create operations, both `vendor_name` and `vendor_version` should be either set or left empty. For update operations, `vendor_name` can be left empty if it has a value in the database.

Template tag

The template tag object has the following properties.

Property	Type	Description
tag	string	Template tag name. Property behavior: - *required*
value	string	Template tag value.

template.create

Description

`object template.create(object/array templates)`

This method allows to create new templates.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Templates to create.

Additionally to the standard template properties, the method accepts the following parameters.

Parameter	Type	Description
groups	object/array	Template groups to add the template to. The template groups must have the `groupid` property defined. Parameter behavior: - *required*
tags	object/array	Template tags.
templates	object/array	Templates to be linked to the template. The templates must have the `templateid` property defined.
macros	object/array	User macros to be created for the template.

Return values

(`object`) Returns an object containing the IDs of the created templates under the `templateids` property. The order of the returned IDs matches the order of the passed templates.

Examples

Creating a template

Create a template with tags and link two templates to this template.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.create",
    "params": {
        "host": "Linux template",
        "groups": {
            "groupid": 1
        },
        "templates": [
            {
                "templateid": "11115"
            },
            {
                "templateid": "11116"
            }
        ],
        "tags": [
            {
                "tag": "Host name",
                "value": "{HOST.NAME}"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "11117"
        ]
    },
    "id": 1
}
```

Source

CTemplate::create() in *ui/include/classes/api/services/CTemplate.php*.

template.delete

Description

`object template.delete(array templateIds)`

This method allows to delete templates.

Deleting a template will cause deletion of all template entities (items, triggers, graphs, etc.). To leave template entities with the hosts, but delete the template itself, first unlink the template from required hosts using one of these methods: template.update, template.massupdate, host.update, host.massupdate.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the templates to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted templates under the `templateids` property.

Examples

Deleting multiple templates

Delete two templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.delete",
    "params": [
        "13",
        "32"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "13",
            "32"
        ]
    },
    "id": 1
}
```

Source

CTemplate::delete() in *ui/include/classes/api/services/CTemplate.php*.

template.get

Description

`integer/array template.get(object parameters)`

The method allows to retrieve templates according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
templateids	string/array	Return only templates with the given template IDs.
groupids	string/array	Return only templates that belong to the given template groups.
parentTemplateids	string/array	Return only templates that the given template is linked to.
hostids	string/array	Return only templates that are linked to the given hosts/templates.
graphids	string/array	Return only templates that contain the given graphs.
itemids	string/array	Return only templates that contain the given items.
triggerids	string/array	Return only templates that contain the given triggers.
with_items	flag	Return only templates that have items.
with_triggers	flag	Return only templates that have triggers.
with_graphs	flag	Return only templates that have graphs.
with_httptests	flag	Return only templates that have web scenarios.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	object/array	Return only templates with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all templates. Possible operator values: 0 - (default) Contains; 1 - Equals; 2 - Not like; 3 - Not equal 4 - Exists; 5 - Not exists.
selectTags	query	Return template tags in the `tags` property.
selectHosts	query	Return the hosts that are linked to the template in the `hosts` property. Supports `count`.
selectTemplateGroups	query	Return the template groups that the template belongs to in the `templategroups` property.
selectTemplates	query	Return templates to which the given template is linked in the `templates` property. Supports `count`.
selectParentTemplates	query	Return templates that are linked to the given template in the `parentTemplates` property. Supports `count`.
selectHttpTests	query	Return the web scenarios from the template in the `httpTests` property. Supports `count`.
selectItems	query	Return items from the template in the `items` property. Supports `count`.

Parameter	Type	Description
selectDiscoveries	query	Return low-level discoveries from the template in the `discoveries` property. Supports `count`.
selectTriggers	query	Return triggers from the template in the `triggers` property. Supports `count`.
selectGraphs	query	Return graphs from the template in the `graphs` property. Supports `count`.
selectMacros	query	Return the macros from the template in the `macros` property..
selectDashboards	query	Return dashboards from the template in the `dashboards` property. Supports `count`.
selectValueMaps	query	Return a `valuemaps` property with template value maps.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectTemplates` - results will be sorted by `name`; `selectHosts` - sorted by `host`; `selectParentTemplates` - sorted by `host`; `selectItems` - sorted by `name`; `selectDiscoveries` - sorted by `name`; `selectTriggers` - sorted by `description`; `selectGraphs` - sorted by `name`; `selectDashboards` - sorted by `name`.
sortfield	string/array	Sort the result by the given properties. Possible values: `hostid`, `host`, `name`, `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectGroups (deprecated)	query	This parameter is deprecated, please use `selectTemplateGroups` instead. Return the template groups that the template belongs to in the groups property.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving templates by name

Retrieve all data about two templates named ”Linux” and ”Windows”.

Request:

```
{
    "jsonrpc": "2.0",
```

```
    "method": "template.get",
    "params": {
        "output": "extend",
        "filter": {
            "host": [
                "Linux",
                "Windows"
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "proxy_hostid": "0",
            "host": "Linux",
            "status": "3",
            "disable_until": "0",
            "error": "",
            "available": "0",
            "errors_from": "0",
            "lastaccess": "0",
            "ipmi_authtype": "0",
            "ipmi_privilege": "2",
            "ipmi_username": "",
            "ipmi_password": "",
            "ipmi_disable_until": "0",
            "ipmi_available": "0",
            "snmp_disable_until": "0",
            "snmp_available": "0",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "ipmi_errors_from": "0",
            "snmp_errors_from": "0",
            "ipmi_error": "",
            "snmp_error": "",
            "jmx_disable_until": "0",
            "jmx_available": "0",
            "jmx_errors_from": "0",
            "jmx_error": "",
            "name": "Linux",
            "flags": "0",
            "templateid": "10001",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": "",
            "uuid": "282ffe33afc74cccaf1524d9aa9dc502"
        },
        {
            "proxy_hostid": "0",
            "host": "Windows",
            "status": "3",
```

```
            "disable_until": "0",
            "error": "",
            "available": "0",
            "errors_from": "0",
            "lastaccess": "0",
            "ipmi_authtype": "0",
            "ipmi_privilege": "2",
            "ipmi_username": "",
            "ipmi_password": "",
            "ipmi_disable_until": "0",
            "ipmi_available": "0",
            "snmp_disable_until": "0",
            "snmp_available": "0",
            "maintenanceid": "0",
            "maintenance_status": "0",
            "maintenance_type": "0",
            "maintenance_from": "0",
            "ipmi_errors_from": "0",
            "snmp_errors_from": "0",
            "ipmi_error": "",
            "snmp_error": "",
            "jmx_disable_until": "0",
            "jmx_available": "0",
            "jmx_errors_from": "0",
            "jmx_error": "",
            "name": "Windows",
            "flags": "0",
            "templateid": "10081",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "tls_psk_identity": "",
            "tls_psk": "",
            "uuid": "522d17e1834049be879287b7c0518e5d"
        }
    ],
    "id": 1
}
```


Retrieving template groups

Retrieve template groups that the template ”Linux by Zabbix agent” is a member of.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": ["hostid"],
        "selectTemplateGroups": "extend",
        "filter": {
            "host": [
                "Linux by Zabbix agent"
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "templateid": "10001",
            "templategroups": [
                {
                    "groupid": "10",
                    "name": "Templates/Operating systems",
                    "uuid": "846977d1dfed4968bc5f8bdb363285bc"
                }
            ]
        }
    ],
    "id": 1
}
```


Retrieving hosts by template

Retrieve hosts that have the ”10001” (*Linux by Zabbix agent*) template linked to them.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": "templateid",
        "templateids": "10001",
        "selectHosts": ["hostid", "name"]
    },
    "id": 1
}
```

Response:


```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "templateid": "10001",
            "hosts": [
                {
                    "hostid": "10084",
                    "name": "Zabbix server"
                },
                {
                    "hostid": "10603",
                    "name": "Host 1"
                },
                {
                    "hostid": "10604",
                    "name": "Host 2"
                }
            ]
        }
    ],
    "id": 1
}
```


Searching by template tags

Retrieve templates that have tag ”Host name” equal to ”{HOST.NAME}”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.get",
    "params": {
        "output": ["hostid"],
        "selectTags": "extend",
        "evaltype": 0,
        "tags": [
            {
                "tag": "Host name",
                "value": "{HOST.NAME}",
                "operator": 1
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10402",
            "tags": [
                {
                    "tag": "Host name",
                    "value": "{HOST.NAME}"
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Template group
- Template
- User macro
- Host interface

Source

CTemplate::get() in *ui/include/classes/api/services/CTemplate.php*.

template.massadd

Description

`object template.massadd(object parameters)`

This method allows to simultaneously add multiple related objects to the given templates.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the templates to update and the objects to add to the templates.

The method accepts the following parameters.

Parameter	Type	Description
templates	object/array	Templates to be updated. The templates must have the `templateid` property defined. Parameter behavior: - *required*
groups	object/array	Template groups to add the given templates to. The template groups must have the `groupid` property defined.
macros	object/array	User macros to be created for the given templates.
templates_link	object/array	Templates to link to the given templates. The templates must have the `templateid` property defined.

Return values

`(object)` Returns an object containing the IDs of the updated templates under the `templateids` property.

Examples

Link a group to templates

Add template group "2" to two templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10085"
            },
            {
                "templateid": "10086"
            }
        ],
        "groups": [
            {
                "groupid": "2"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

Link two templates to a template

Link templates "10106" and "10104" to template "10073".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10073"
            }
        ],
        "templates_link": [
            {
                "templateid": "10106"
            },
            {
                "templateid": "10104"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10073"
        ]
    },
    "id": 1
}
```

See also

- template.update
- Host
- Template group
- User macro

Source

CTemplate::massAdd() in *ui/include/classes/api/services/CTemplate.php*.

template.massremove

Description

`object template.massremove(object parameters)`

This method allows to remove related objects from multiple templates.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the templates to update and the objects that should be removed.

Parameter	Type	Description
templateids	string/array	IDs of the templates to be updated. Parameter behavior: - *required*

Parameter	Type	Description
groupids	string/array	Template groups from which to remove the given templates.
macros	string/array	User macros to delete from the given templates.
templateids_clear	string/array	Templates to unlink and clear from the given templates (upstream).
templateids_link	string/array	Templates to unlink from the given templates (upstream).

Return values

`(object)` Returns an object containing the IDs of the updated templates under the `templateids` property.

Examples

Removing templates from a group

Remove two templates from group ”2”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": [
            "10085",
            "10086"
        ],
        "groupids": "2"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

Unlinking templates from a host

Unlink templates ”10106” and ”10104” from template ”10085”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": "10085",
        "templateids_link": [
            "10106",
            "10104"
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
```

```
        "templateids": [
            "10085"
        ]
    },
    "id": 1
}
```

See also

- template.update
- User macro

Source

CTemplate::massRemove() in *ui/include/classes/api/services/CTemplate.php*.

template.massupdate

Description

`object template.massupdate(object parameters)`

This method allows to simultaneously replace or remove related objects and update properties on multiple templates.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the templates to update and the objects to replace for the templates.

The method accepts the following parameters.

Parameter	Type	Description
templates	object/array	Templates to be updated. The templates must have the `templateid` property defined. Parameter behavior: - *required*
groups	object/array	Template groups to replace the current template groups the templates belong to. The template groups must have the `groupid` property defined.
macros	object/array	User macros to replace all of the current user macros on the given templates.
templates_clear	object/array	Templates to unlink and clear from the given templates. The templates must have the `templateid` property defined.
templates_link	object/array	Templates to replace the currently linked templates. The templates must have the `templateid` property defined.

Return values

`(object)` Returns an object containing the IDs of the updated templates under the `templateids` property.

Examples

Unlinking a template

Unlink and clear template ”10091” from the given templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massupdate",
    "params": {
        "templates": [
            {
                "templateid": "10085"
            },
            {
                "templateid": "10086"
            }
        ],
        "templates_clear": [
            {
                "templateid": "10091"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

Replacing user macros

Replace all user macros with the given user macro on multiple templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massupdate",
    "params": {
        "templates": [
            {
                "templateid": "10074"
            },
            {
                "templateid": "10075"
            },
            {
                "templateid": "10076"
            },
            {
                "templateid": "10077"
            }
        ],
        "macros": [
            {
                "macro": "{$AGENT.TIMEOUT}",
                "value": "5m",
                "description": "Timeout after which agent is considered unavailable. Works only for agents
            }
        ]
```

```
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10074",
            "10075",
            "10076",
            "10077"
        ]
    },
    "id": 1
}
```

See also

- template.update
- template.massadd
- Template group
- User macro

Source

CTemplate::massUpdate() in *ui/include/classes/api/services/CTemplate.php*.

template.update

Description

`object template.update(object/array templates)`

This method allows to update existing templates.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Template properties to be updated.

The `templateid` property must be defined for each template, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.

Additionally to the standard template properties, the method accepts the following parameters.

Parameter	Type	Description
groups	object/array	Template groups to replace the current template groups the templates belong to. The template groups must have the `groupid` property defined.
tags	object/array	Template tags to replace the current template tags.
macros	object/array	User macros to replace the current user macros on the given templates.
templates	object/array	Templates to replace the currently linked templates. Templates that are not passed are only unlinked. The templates must have the `templateid` property defined.
templates_clear	object/array	Templates to unlink and clear from the given templates. The templates must have the `templateid` property defined.

Return values

(object) Returns an object containing the IDs of the updated templates under the templateids property.

Examples

Renaming a template

Rename the template to "Template OS Linux".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "name": "Template OS Linux"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

Updating template tags

Replace all template tags with a new one.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.update",
    "params": {
        "templateid": "10086",
        "tags": [
            {
                "tag": "Host name",
                "value": "{HOST.NAME}"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}
```

Source

CTemplate::update() in *ui/include/classes/api/services/CTemplate.php*.

Template dashboard

This class is designed to work with template dashboards.

Object references:

- Template dashboard
- Template dashboard page
- Template dashboard widget
- Template dashboard widget field

Available methods:

- templatedashboard.create - creating new template dashboards
- templatedashboard.delete - deleting template dashboards
- templatedashboard.get - retrieving template dashboards
- templatedashboard.update - updating template dashboards

> Template dashboard object

The following objects are directly related to the `templatedashboard` API.

Template dashboard

The template dashboard object has the following properties.

Property	Type	Description
dashboardid	string	ID of the template dashboard. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the template dashboard. Property behavior: - *required* for create operations
templateid	string	ID of the template the dashboard belongs to. Property behavior: - *constant* - *required* for create operations
display_period	integer	Default page display period (in seconds). Possible values: 10, 30, 60, 120, 600, 1800, 3600. Default: 30.
auto_start	integer	Auto start slideshow. Possible values: 0 - do not auto start slideshow; 1 - *(default)* auto start slideshow.
uuid	string	Universal unique identifier, used for linking imported template dashboards to already existing ones. Auto-generated, if not given.

Template dashboard page

The template dashboard page object has the following properties.

Property	Type	Description
dashboard_pageid	string	ID of the dashboard page. Property behavior: - *read-only*
name	string	Dashboard page name. Default: empty string.
display_period	integer	Dashboard page display period (in seconds). Possible values: 0, 10, 30, 60, 120, 600, 1800, 3600. Default: 0 (will use the default page display period).
widgets	array	Array of the template dashboard widget objects.

Template dashboard widget

The template dashboard widget object has the following properties.

Property	Type	Description
widgetid	string	ID of the dashboard widget. Property behavior: - *read-only*
type	string	Type of the dashboard widget. Possible values: clock - Clock; graph - Graph (classic); graphprototype - Graph prototype; item - Item value; plaintext - Plain text; url - URL. Property behavior: - *required*
name	string	Custom widget name.
x	integer	A horizontal position from the left side of the dashboard. Valid values range from 0 to 23.
y	integer	A vertical position from the top of the dashboard. Valid values range from 0 to 62.
width	integer	The widget width. Valid values range from 1 to 24.
height	integer	The widget height. Valid values range from 2 to 32.
view_mode	integer	The widget view mode. Possible values: 0 - (default) default widget view; 1 - with hidden header;
fields	array	Array of the template dashboard widget field objects.

Template dashboard widget field

The template dashboard widget field object has the following properties.

Property	Type	Description
type	integer	Type of the widget field. Possible values: 0 - Integer; 1 - String; 4 - Item; 5 - Item prototype; 6 - Graph; 7 - Graph prototype. Property behavior: - *required*
name	string	Widget field name. Property behavior: - *required*
value	mixed	Widget field value depending on type. Property behavior: - *required*

templatedashboard.create

Description

`object templatedashboard.create(object/array templateDashboards)`

This method allows to create new template dashboards.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Template dashboards to create.

Additionally to the standard template dashboard properties, the method accepts the following parameters.

Parameter	Type	Description
pages	array	Template dashboard pages to be created for the dashboard. Dashboard pages will be ordered in the same order as specified. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the created template dashboards under the `dashboardids` property. The order of the returned IDs matches the order of the passed template dashboards.

Examples

Creating a template dashboard

Create a template dashboard named “Graphs” with one Graph widget on a single dashboard page.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.create",
    "params": {
```

```
        "templateid": "10318",
        "name": "Graphs",
        "pages": [
            {
                "widgets": [
                    {
                        "type": "graph",
                        "x": 0,
                        "y": 0,
                        "width": 12,
                        "height": 5,
                        "view_mode": 0,
                        "fields": [
                            {
                                "type": 6,
                                "name": "graphid",
                                "value": "1123"
                            }
                        ]
                    }
                ]

            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "32"
        ]
    },
    "id": 1
}
```

See also

- Template dashboard page
- Template dashboard widget
- Template dashboard widget field

Source

CTemplateDashboard::create() in *ui/include/classes/api/services/CTemplateDashboard.php*.

templatedashboard.delete

Description

`object templatedashboard.delete(array templateDashboardIds)`

This method allows to delete template dashboards.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the template dashboards to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted template dashboards under the `dashboardids` property.

Examples

Deleting multiple template dashboards

Delete two template dashboards.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.delete",
    "params": [
        "45",
        "46"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "45",
            "46"
        ]
    },
    "id": 1
}
```

Source

CTemplateDashboard::delete() in *ui/include/classes/api/services/CTemplateDashboard.php*.

templatedashboard.get

Description

`integer/array templatedashboard.get(object parameters)`

The method allows to retrieve template dashboards according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
dashboardids	string/array	Return only template dashboards with the given IDs.
templateids	string/array	Return only template dashboards that belong to the given templates.
selectPages	query	Return a pages property with template dashboard pages, correctly ordered.
sortfield	string/array	Sort the result by the given properties. Possible values: `dashboardid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	

Parameter	Type	Description
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving template dashboards

Retrieve all template dashboards with widgets for a specified template.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.get",
    "params": {
        "output": "extend",
        "selectPages": "extend",
        "templateids": "10001"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "dashboardid": "23",
            "name": "Docker overview",
            "templateid": "10001",
            "display_period": "30",
            "auto_start": "1",
            "uuid": "6dfcbe0bc5ad400ea9c1c2dd7649282f",
            "pages": [
                {
                    "dashboard_pageid": "1",
                    "name": "",
                    "display_period": "0",
                    "widgets": [
                        {
                            "widgetid": "220",
                            "type": "graph",
                            "name": "",
                            "x": "0",
                            "y": "0",
                            "width": "12",
                            "height": "5",
                            "view_mode": "0",
                            "fields": [
```

```
                {
                    "type": "6",
                    "name": "graphid",
                    "value": "1125"
                }
            ]
        },
        {
            "widgetid": "221",
            "type": "graph",
            "name": "",
            "x": "12",
            "y": "0",
            "width": "12",
            "height": "5",
            "view_mode": "0",
            "fields": [
                {
                    "type": "6",
                    "name": "graphid",
                    "value": "1129"
                }
            ]
        },
        {
            "widgetid": "222",
            "type": "graph",
            "name": "",
            "x": "0",
            "y": "5",
            "width": "12",
            "height": "5",
            "view_mode": "0",
            "fields": [
                {
                    "type": "6",
                    "name": "graphid",
                    "value": "1128"
                }
            ]
        },
        {
            "widgetid": "223",
            "type": "graph",
            "name": "",
            "x": "12",
            "y": "5",
            "width": "12",
            "height": "5",
            "view_mode": "0",
            "fields": [
                {
                    "type": "6",
                    "name": "graphid",
                    "value": "1126"
                }
            ]
        },
        {
            "widgetid": "224",
            "type": "graph",
            "name": "",
```

```
                            "x": "0",
                            "y": "10",
                            "width": "12",
                            "height": "5",
                            "view_mode": "0",
                            "fields": [
                                {
                                    "type": "6",
                                    "name": "graphid",
                                    "value": "1127"
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Template dashboard page
- Template dashboard widget
- Template dashboard widget field

Source

CTemplateDashboard::get() in *ui/include/classes/api/services/CTemplateDashboard.php*.

templatedashboard.update

Description

`object templatedashboard.update(object/array templateDashboards)`

This method allows to update existing template dashboards.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Template dashboard properties to be updated.

The `dashboardid` property must be specified for each dashboard, all other properties are optional. Only the specified properties will be updated.

Additionally to the standard template dashboard properties, the method accepts the following parameters.

Parameter	Type	Description
pages	array	Template dashboard pages to replace the existing dashboard pages. Dashboard pages are updated by the `dashboard_pageid` property. New dashboard pages will be created for objects without `dashboard_pageid` property and the existing dashboard pages will be deleted if not reused. Dashboard pages will be ordered in the same order as specified. Only the specified properties of the dashboard pages will be updated. At least one dashboard page object is required for `pages` property.

Return values

(object) Returns an object containing the IDs of the updated template dashboards under the dashboardids property.

Examples

Renaming a template dashboard

Rename a template dashboard to "Performance graphs".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.update",
    "params": {
        "dashboardid": "23",
        "name": "Performance graphs"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "23"
        ]
    },
    "id": 1
}
```

Updating template dashboard pages

Rename the first dashboard page, replace widgets on the second dashboard page and add a new page as the third one. Delete all other dashboard pages.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templatedashboard.update",
    "params": {
        "dashboardid": "2",
        "pages": [
            {
                "dashboard_pageid": 1,
                "name": "Renamed Page"
            },
            {
                "dashboard_pageid": 2,
                "widgets": [
                    {
                        "type": "clock",
                        "x": 0,
                        "y": 0,
                        "width": 4,
                        "height": 3
                    }
                ]
            },
            {
                "display_period": 60
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 1
}
```

See also

- Template dashboard widget
- Template dashboard widget field

Source

CTemplateDashboard::update() in *ui/include/classes/api/services/CTemplateDashboard.php*.

Template group

This class is designed to work with template groups.

Object references:

- Template group

Available methods:

- templategroup.create - creating new template groups
- templategroup.delete - deleting template groups
- templategroup.get - retrieving template groups
- templategroup.massadd - adding related objects to template groups
- templategroup.massremove - removing related objects from template groups
- templategroup.massupdate - replacing or removing related objects from template groups
- templategroup.propagate - propagating permissions to template groups' subgroups
- templategroup.update - updating template groups

> Template group object

The following objects are directly related to the `templategroup` API.

Template group

The template group object has the following properties.

Property	Type	Description
groupid	string	ID of the template group. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the template group. Property behavior: - *required* for create operations
uuid	string	Universal unique identifier, used for linking imported template groups to already existing ones. Auto-generated, if not given.

templategroup.create

Description

`object templategroup.create(object/array templateGroups)`

This method allows to create new template groups.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Template groups to create. The method accepts template groups with the standard template group properties.

Return values

`(object)` Returns an object containing the IDs of the created template groups under the `groupids` property. The order of the returned IDs matches the order of the passed template groups.

Examples

Creating a template group

Create a template group called "Templates/Databases".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.create",
    "params": {
        "name": "Templates/Databases"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107820"
        ]
    },
    "id": 1
}
```

Source

CTemplateGroup::create() in *ui/include/classes/api/services/CTemplateGroup.php*.

templategroup.delete

Description

`object templategroup.delete(array templateGroupIds)`

This method allows to delete template groups.

A template group can not be deleted if it contains templates that belong to this group only.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the template groups to delete.

Return values

(object) Returns an object containing the IDs of the deleted template groups under the groupids property.

Examples

Deleting multiple template groups

Delete two template groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.delete",
    "params": [
        "107814",
        "107815"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "107814",
            "107815"
        ]
    },
    "id": 1
}
```

Source

CTemplateGroup::delete() in *ui/include/classes/api/services/CTemplateGroup.php*.

templategroup.get

Description

integer/array templategroup.get(object parameters)

The method allows to retrieve template groups according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
graphids	string/array	Return only template groups that contain templates with the given graphs.
groupids	string/array	Return only template groups with the given template group IDs.

Parameter	Type	Description
templateids	string/array	Return only template groups that contain the given templates.
triggerids	string/array	Return only template groups that contain templates with the given triggers.
with_graphs	flag	Return only template groups that contain templates with graphs.
with_graph_prototypes	flag	Return only template groups that contain templates with graph prototypes.
with_httptests	flag	Return only template groups that contain templates with web checks.
with_items	flag	Return only template groups that contain templates with items. Overrides the `with_simple_graph_items` parameters.
with_item_prototypes	flag	Return only template groups that contain templates with item prototypes. Overrides the `with_simple_graph_item_p` parameter.
with_simple_graph_item_prototypes	flag	Return only template groups that contain templates with item prototypes, which are enabled for creation and have numeric type of information.
with_simple_graph_items	flag	Return only template groups that contain templates with numeric items.
with_templates	flag	Return only template groups that contain templates.

Parameter	Type	Description
with_triggers	flag	Return only template groups that contain templates with triggers.
selectTemplates	query	Return a templates property with the templates that belong to the template group. Supports `count`.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectTemplates` - results will be sorted by `template`.
sortfield	string/array	Sort the result by the given properties. Possible values: `groupid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving data by name

Retrieve all data about two template groups named ”Templates/Databases” and ”Templates/Modules”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.get",
    "params": {
        "output": "extend",
        "filter": {
            "name": [
                "Templates/Databases",
                "Templates/Modules"
            ]
        }
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "groupid": "13",
            "name": "Templates/Databases",
            "uuid": "748ad4d098d447d492bb935c907f652f"
        },
        {
            "groupid": "8",
            "name": "Templates/Modules",
            "uuid": "57b7ae836ca64446ba2c296389c009b7"
        }
    ],
    "id": 1
}
```

See also

- Template

Source

CTemplateGroup::get() in *ui/include/classes/api/services/CTemplateGroup.php*.

templategroup.massadd

Description

`object templategroup.massadd(object parameters)`

This method allows to simultaneously add multiple related objects to all the given template groups.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the template groups to update and the objects to add to all the template groups.

The method accepts the following parameters.

Parameter	Type	Description
groups	object/array	Template groups to be updated. The template groups must have the `groupid` property defined. Parameter behavior: - *required*
templates	object/array	Templates to add to all template groups. The templates must have the `templateid` property defined. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the updated template groups under the `groupids` property.

Examples

Adding templates to template groups

Add two templates to template groups with IDs 12 and 13.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.massadd",
    "params": {
        "groups": [
            {
                "groupid": "12"
            },
            {
                "groupid": "13"
            }
        ],
        "templates": [
            {
                "templateid": "10486"
            },
            {
                "templateid": "10487"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "12",
            "13"
        ]
    },
    "id": 1
}
```

See also

- Template

Source

CTemplateGroup::massAdd() in *ui/include/classes/api/services/CTemplateGroup.php*.

templategroup.massremove

Description

`object templategroup.massremove(object parameters)`

This method allows to remove related objects from multiple template groups.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the template groups to update and the objects that should be removed.

Parameter	Type	Description
groupids	string/array	IDs of the template groups to be updated. Parameter behavior: - *required*
templateids	string/array	Templates to remove from all template groups. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the updated template groups under the `groupids` property.

Examples

Removing templates from template groups

Remove two templates from the given template groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.massremove",
    "params": {
        "groupids": [
            "5",
            "6"
        ],
        "templateids": [
            "30050",
            "30001"
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "5",
            "6"
```

```
        ]
    },
    "id": 1
}
```

Source

CTemplateGroup::massRemove() in *ui/include/classes/api/services/CTemplateGroup.php*.

templategroup.massupdate

Description

`object templategroup.massupdate(object parameters)`

This method allows to replace templates with the specified ones in multiple template groups.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters containing the IDs of the template groups to update and the objects that should be updated.

Parameter	Type	Description
groups	object/array	Template groups to be updated. The template groups must have the `groupid` property defined. Parameter behavior: - *required*
templates	object/array	Templates to replace the current template on the given template groups. All other template, except the ones mentioned, will be excluded from template groups. The templates must have the `templateid` property defined. Parameter behavior: - *required*

Return values

`(object)` Returns an object containing the IDs of the updated template groups under the `groupids` property.

Examples

Replacing templates in a template group

Replace all templates in a template group to ones mentioned templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.massupdate",
    "params": {
        "groups": [
            {
                "groupid": "8"
            }
        ],
        "templates": [
            {
```

```
                "templateid": "40050"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "8",
        ]
    },
    "id": 1
}
```

See also

- templategroup.update
- templategroup.massadd
- Template

Source

CTemplateGroup::massUpdate() in *ui/include/classes/api/services/CTemplateGroup.php*.

templategroup.propagate

Description

`object templategroup.propagate(object parameters)`

This method allows to apply permissions to all template groups' subgroups.

Note:
This method is only available to *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
groups	object/array	Template groups to propagate. The template groups must have the `groupid` property defined. Parameter behavior: *- required*
permissions	boolean	Set `true` if need to propagate permissions. Parameter behavior: *- required*

Return values

`(object)` Returns an object containing the IDs of the propagated template groups under the `groupids` property.

Examples

Propagating template group permissions to its subgroups.

Propagate template group permissions to its subgroups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.propagate",
    "params": {
        "groups": [
            {
                "groupid": "15"
            }
        ],
        "permissions": true
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "15",
        ]
    },
    "id": 1
}
```

See also

- templategroup.update
- templategroup.massadd
- Template

Source

CTemplateGroup::propagate() in *ui/include/classes/api/services/CTemplateGroup.php*.

templategroup.update

Description

`object templategroup.update(object/array templateGroups)`

This method allows to update existing template groups.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Template group properties to be updated.

The `groupid` property must be defined for each template group, all other properties are optional. Only the given properties will be updated, all others will remain unchanged.

Return values

`(object)` Returns an object containing the IDs of the updated template groups under the `groupids` property.

Examples

Renaming a template group

Rename a template group to "Templates/Databases"

Request:

```
{
    "jsonrpc": "2.0",
    "method": "templategroup.update",
    "params": {
        "groupid": "7",
        "name": "Templates/Databases"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "groupids": [
            "7"
        ]
    },
    "id": 1
}
```

Source

CTemplateGroup::update() in *ui/include/classes/api/services/CTemplateGroup.php*.

Token

This class is designed to work with tokens.

Object references:

- Token

Available methods:

- token.create - create new tokens
- token.delete - delete tokens
- token.get - retrieve tokens
- token.update - update tokens
- token.generate - generate tokens

> Token object

The following objects are directly related to the `token` API.

Token

The token object has the following properties.

Property	Type	Description
tokenid	string	ID of the token. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the token. Property behavior: - *required* for create operations
description	text	Description of the token.

Property	Type	Description
userid	string	A user the token has been assigned to. Default: *current user*. Property behavior: - *constant*
lastaccess	timestamp	Most recent date and time the token was authenticated. ”0” if the token has never been authenticated. Property behavior: - *read-only*
status	integer	Token status. Possible values: 0 - *(default)* enabled token; 1 - disabled token.
expires_at	timestamp	Token expiration date and time. ”0” for never-expiring tokens.
created_at	timestamp	Token creation date and time. Property behavior: - *read-only*
creator_userid	string	The creator user of the token. Property behavior: - *read-only*

token.create

Description

`object token.create(object/array tokens)`

This method allows to create new tokens.

> **Note:**
> The *Manage API tokens* permission is required for the user role to manage tokens for other users.

> **Attention:**
> A token created by this method also has to be generated before it is usable.

Parameters

`(object/array)` Tokens to create.

The method accepts tokens with the standard token properties.

Return values

`(object)` Returns an object containing the IDs of the created tokens under the `tokenids` property. The order of the returned IDs matches the order of the passed tokens.

Examples

Create a token

Create an enabled token that never expires and authenticates user of ID 2.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "token.create",
```

```
    "params": {
        "name": "Your token",
        "userid": "2"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "tokenids": [
            "188"
        ]
    },
    "id": 1
}
```

Create a disabled token that expires at January 21st, 2021. This token will authenticate current user.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "token.create",
    "params": {
        "name": "Your token",
        "status": "1",
        "expires_at": "1611238072"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "tokenids": [
            "189"
        ]
    },
    "id": 1
}
```

Source

CToken::create() in *ui/include/classes/api/services/CToken.php*.

token.delete

Description

`object token.delete(array tokenids)`

This method allows to delete tokens.

> **Note:**
> The *Manage API tokens* permission is required for the user role to manage tokens for other users.

Parameters

`(array)` IDs of the tokens to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted tokens under the `tokenids` property.

Examples

Delete multiple tokens

Delete two tokens.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "token.delete",
    "params": [
        "188",
        "192"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "tokenids": [
            "188",
            "192"
        ]
    },
    "id": 1
}
```

Source

CToken::delete() in *ui/include/classes/api/services/CToken.php*.

token.generate

Description

`object token.generate(array tokenids)`

This method allows to generate tokens.

Note:
The *Manage API tokens* permission is required for the user role to manage tokens for other users.

Attention:
A token can be generated by this method only if it has been created.

Parameters

`(array)` IDs of the tokens to generate.

Return values

`(array)` Returns an array of objects containing the ID of the generated token under the `tokenid` property and generated authorization string under `token` property.

Property	Type	Description
tokenid	string	ID of the token.
token	string	The generated authorization string for this token.

Examples

Generate multiple tokens

Generate two tokens.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "token.generate",
    "params": [
        "1",
        "2"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "tokenid": "1",
            "token": "bbcfce79a2d95037502f7e9a534906d3466c9a1484beb6ea0f4e7be28e8b8ce2"
        },
        {
            "tokenid": "2",
            "token": "fa1258a83d518eabd87698a96bd7f07e5a6ae8aeb8463cae33d50b91dd21bd6d"
        }
    ],
    "id": 1
}
```

Source

CToken::generate() in *ui/include/classes/api/services/CToken.php*.

token.get

Description

`integer/array token.get(object parameters)`

The method allows to retrieve tokens according to the given parameters.

> **Note:**
> Only *Super admin* user type is allowed to view tokens for other users.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
tokenids	string/array	Return only tokens with the given IDs.
userids	string/array	Return only tokens created for the given users.
token	string	Return only tokens created for the given *Auth token*.
valid_at	timestamp	Return only tokens, which are valid (not expired) at the given date and time.
expired_at	timestamp	Return only tokens, which are expired (not valid) at the given date and time.
sortfield	string/array	Sort the result by the given properties. Possible values: `tokenid`, `name`, `lastaccess`, `status`, `expires_at`, `created_at`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
excludeSearch	boolean	
filter	object	

Parameter	Type	Description
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve a token

Retrieve all data for the token with ID "2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "token.get",
    "params": {
        "output": "extend",
        "tokenids": "2"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "tokenid": "1",
            "name": "The Token",
            "description": "",
            "userid": "1",
            "lastaccess": "0",
            "status": "0",
            "expires_at": "1609406220",
            "created_at": "1611239454",
            "creator_userid": "1"
        }
    ],
    "id": 1
}
```

Source

CToken::get() in *ui/include/classes/api/services/CToken.php*.

token.update

Description

`object token.update(object/array tokens)`

This method allows to update existing tokens.

> **Note:**
> The *Manage API tokens* permission is required for the user role to manage tokens for other users.

Parameters

`(object/array)` Token properties to be updated.

The `tokenid` property must be defined for each token, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

The method accepts tokens with the standard token properties.

Return values

`(object)` Returns an object containing the IDs of the updated tokens under the `tokenids` property.

Examples

Remove token expiry

Remove expiry date from token.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "token.update",
    "params": {
        "tokenid": "2",
        "expires_at": "0"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "tokenids": [
            "2"
        ]
    },
    "id": 1
}
```

Source

CToken::update() in *ui/include/classes/api/services/CToken.php*.

Trend

This class is designed to work with trend data.

Object references:

- Trend

Available methods:

- trend.get - retrieving trends

> Trend object

The following objects are directly related to the `trend` API.

Note:
Trend objects differ depending on the item's type of information. They are created by the Zabbix server and cannot be modified via the API.

Float trend

The float trend object has the following properties.

Property	Type	Description
clock	timestamp	Timestamp of an hour for which the value was calculated. For example, timestamp of "04:00:00" means values calculated for period "04:00:00-04:59:59".
itemid	integer	ID of the related item.
num	integer	Number of values that were available for the hour.
value_min	float	Hourly minimum value.
value_avg	float	Hourly average value.
value_max	float	Hourly maximum value.

Integer trend

The integer trend object has the following properties.

Property	Type	Description
clock	timestamp	Timestamp of an hour for which the value was calculated. For example, timestamp of "04:00:00" means values calculated for period "04:00:00-04:59:59".
itemid	integer	ID of the related item.
num	integer	Number of values that were available for the hour.
value_min	integer	Hourly minimum value.
value_avg	integer	Hourly average value.
value_max	integer	Hourly maximum value.

trend.get

Description

`integer/array trend.get(object parameters)`

The method allows to retrieve trend data according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
itemids	string/array	Return only trends with the given item IDs.
time_from	timestamp	Return only values that have been collected after or at the given time.
time_till	timestamp	Return only values that have been collected before or at the given time.
countOutput	boolean	Count the number of retrieved objects.
limit	integer	Limit the amount of retrieved objects.
output	query	Set Trend object properties to be returned.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving item trend data

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trend.get",
    "params": {
        "output": [
            "itemid",
            "clock",
            "num",
            "value_min",
            "value_avg",
            "value_max",
        ],
        "itemids": [
            "23715"
        ],
        "limit": "1"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "itemid": "23715",
            "clock": "1446199200",
            "num": "60",
            "value_min": "0.165",
            "value_avg": "0.2168",
            "value_max": "0.35",
        }
    ],
    "id": 1
}
```

Source

CTrend::get() in *ui/include/classes/api/services/CTrend.php*.

Trigger

This class is designed to work with triggers.

Object references:

- Trigger

Available methods:

- trigger.create - creating new triggers
- trigger.delete - deleting triggers
- trigger.get - retrieving triggers
- trigger.update - updating triggers

> Trigger object

The following objects are directly related to the `trigger` API.

Trigger

The trigger object has the following properties.

Property	Type	Description
triggerid	string	ID of the trigger. Property behavior: - *read-only* - *required* for update operations
description	string	Name of the trigger. Property behavior: - *required* for create operations
expression	string	Reduced trigger expression. Property behavior: - *required* for create operations
event_name	string	Event name generated by the trigger.
opdata	string	Operational data.
comments	string	Additional description of the trigger.
error	string	Error text if there have been any problems when updating the state of the trigger. Property behavior: - *read-only*
flags	integer	Origin of the trigger. Possible values: 0 - *(default)* a plain trigger; 4 - a discovered trigger. Property behavior: - *read-only*
lastchange	timestamp	Time when the trigger last changed its state. Property behavior: - *read-only*
priority	integer	Severity of the trigger. Possible values: 0 - *(default)* not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
state	integer	State of the trigger. Possible values: 0 - *(default)* trigger state is up to date; 1 - current trigger state is unknown. Property behavior: - *read-only*

Property	Type	Description
status	integer	Whether the trigger is enabled or disabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
templateid	string	ID of the parent template trigger. Property behavior: - *read-only*
type	integer	Whether the trigger can generate multiple problem events. Possible values: 0 - *(default)* do not generate multiple events; 1 - generate multiple events.
url	string	URL associated with the trigger.
url_name	string	Label for the URL associated with the trigger.
value	integer	Whether the trigger is in OK or problem state. Possible values: 0 - *(default)* OK; 1 - problem. Property behavior: - *read-only*
recovery_mode	integer	OK event generation mode. Possible values: 0 - *(default)* Expression; 1 - Recovery expression; 2 - None.
recovery_expression	string	Reduced trigger recovery expression.
correlation_mode	integer	OK event closes. Possible values: 0 - *(default)* All problems; 1 - All problems if tag values match.
correlation_tag	string	Tag for matching.
manual_close	integer	Allow manual close. Possible values: 0 - *(default)* No; 1 - Yes.
uuid	string	Universal unique identifier, used for linking imported triggers to already existing ones. Auto-generated, if not given. Property behavior: - *supported* if the trigger belongs to a template

Trigger tag

The trigger tag object has the following properties.

Property	Type	Description
tag	string	Trigger tag name. Property behavior: - *required*
value	string	Trigger tag value.

trigger.create

Description

`object trigger.create(object/array triggers)`

This method allows to create new triggers.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Triggers to create.

Additionally to the standard trigger properties the method accepts the following parameters.

Parameter	Type	Description
dependencies	array	Triggers that the trigger is dependent on. The triggers must have the `triggerid` property defined.
tags	array	Trigger tags.

> **Attention:**
> The trigger expression has to be given in its expanded form.

Return values

`(object)` Returns an object containing the IDs of the created triggers under the `triggerids` property. The order of the returned IDs matches the order of the passed triggers.

Examples

Creating a trigger

Create a trigger with a single trigger dependency.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.create",
    "params": [
        {
            "description": "Processor load is too high on {HOST.NAME}",
            "expression": "last(/Linux server/system.cpu.load[percpu,avg1])>5",
            "dependencies": [
                {
                    "triggerid": "17367"
                }
            ]
        },
        {
            "description": "Service status",
            "expression": "length(last(/Linux server/log[/var/log/system,Service .* has stopped]))<>0",
            "dependencies": [
                {
                    "triggerid": "17368"
                }
            ],
            "tags": [
                {
                    "tag": "service",
                    "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}"
                },
                {
                    "tag": "error",
```

```
                    "value": ""
                }
            ]
        }
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17369",
            "17370"
        ]
    },
    "id": 1
}
```

Source

CTrigger::create() in *ui/include/classes/api/services/CTrigger.php*.

trigger.delete

Description

`object trigger.delete(array triggerIds)`

This method allows to delete triggers.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the triggers to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted triggers under the `triggerids` property.

Examples

Delete multiple triggers

Delete two triggers.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.delete",
    "params": [
        "12002",
        "12003"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
```

```
            "12002",
            "12003"
        ]
    },
    "id": 1
}
```

Source

CTrigger::delete() in *ui/include/classes/api/services/CTrigger.php*.

trigger.get

Description

`integer/array trigger.get(object parameters)`

The method allows to retrieve triggers according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
triggerids	string/array	Return only triggers with the given IDs.
groupids	string/array	Return only triggers that belong to hosts or templates from the given host groups or template groups.
templateids	string/array	Return only triggers that belong to the given templates.
hostids	string/array	Return only triggers that belong to the given hosts.
itemids	string/array	Return only triggers that contain the given items.
functions	string/array	Return only triggers that use the given functions. Refer to the supported function page for a list of supported functions.
group	string	Return only triggers that belong to hosts or templates from the host group or template group with the given name.
host	string	Return only triggers that belong to host with the given technical name.
inherited	boolean	If set to `true` return only triggers inherited from a template.
templated	boolean	If set to `true` return only triggers that belong to templates.
dependent	boolean	If set to `true` return only triggers that have dependencies. If set to `false` return only triggers that do not have dependencies.
monitored	flag	Return only enabled triggers that belong to monitored hosts and contain only enabled items.
active	flag	Return only enabled triggers that belong to monitored hosts.
maintenance	boolean	If set to `true` return only enabled triggers that belong to hosts in maintenance.
withUnacknowledgedEvents	flag	Return only triggers that have unacknowledged events.
withAcknowledgedEvents	flag	Return only triggers with all events acknowledged.
withLastEventUnacknowledged	flag	Return only triggers with the last event unacknowledged.
skipDependent	flag	Skip triggers in a problem state that are dependent on other triggers. Note that the other triggers are ignored if disabled, have disabled items or disabled item hosts.
lastChangeSince	timestamp	Return only triggers that have changed their state after the given time.
lastChangeTill	timestamp	Return only triggers that have changed their state before the given time.
only_true	flag	Return only triggers that have recently been in a problem state.
min_severity	integer	Return only triggers with severity greater or equal than the given severity.

Parameter	Type	Description
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	array of objects	Return only triggers with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all triggers. Possible operator types: 0 - (default) Like; 1 - Equal; 2 - Not like; 3 - Not equal 4 - Exists; 5 - Not exists.
expandComment	flag	Expand macros in the trigger description.
expandDescription	flag	Expand macros in the name of the trigger.
expandExpression	flag	Expand functions and macros in the trigger expression.
selectHostGroups	query	Return the host groups that the trigger belongs to in the hostgroups property.
selectHosts	query	Return the hosts that the trigger belongs to in the hosts property.
selectItems	query	Return items contained by the trigger in the items property.
selectFunctions	query	Return functions used in the trigger in the `functions` property. The function objects represent the functions used in the trigger expression and has the following properties: `functionid` - `(string)` ID of the function; `itemid` - `(string)` ID of the item used in the function; `function` - `(string)` name of the function; `parameter` - `(string)` parameter passed to the function. Query parameter is replaced by $ symbol in returned string.
selectDependencies	query	Return triggers that the trigger depends on in the `dependencies` property.
selectDiscoveryRule	query	Return the low-level discovery rule that created the trigger.
selectLastEvent	query	Return the last significant trigger event in the lastEvent property.
selectTags	query	Return the trigger tags in tags property.
selectTemplateGroups	query	Return the template groups that the trigger belongs to in the templategroups property.
selectTriggerDiscovery	query	Return the trigger discovery object in the `triggerDiscovery` property. The trigger discovery objects link the trigger to a trigger prototype from which it was created. It has the following properties: `parent_triggerid` - `(string)` ID of the trigger prototype from which the trigger has been created.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: `host` - technical name of the host that the trigger belongs to; `hostid` - ID of the host that the trigger belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectHosts` - results will be sorted by `host`.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties. Possible values: `triggerid`, `description`, `status`, `priority`, `lastchange`, `hostname`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectGroups (deprecated)	query	This parameter is deprecated, please use `selectHostGroups` or `selectTemplateGroups` instead. Return the host groups and template groups that the trigger belongs to in the `groups` property.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving data by trigger ID

Retrieve all data and the functions used in trigger "14062".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "triggerids": "14062",
        "output": "extend",
        "selectFunctions": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "14062",
            "expression": "{13513}<10m",
            "description": "{HOST.NAME} has been restarted (uptime < 10m)",
            "url": "",
            "status": "0",
            "value": "0",
            "priority": "2",
            "lastchange": "0",
            "comments": "The host uptime is less than 10 minutes",
            "error": "",
            "templateid": "10016",
            "type": "0",
```

```
            "state": "0",
            "flags": "0",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0",
            "opdata": "",
            "event_name": "",
            "uuid": "",
            "url_name": "",
            "functions": [
                {
                    "functionid": "13513",
                    "itemid": "24350",
                    "triggerid": "14062",
                    "parameter": "$",
                    "function": "last"
                }
            ]
        }
    ],
    "id": 1
}
```

Retrieving triggers in problem state

Retrieve the ID, name and severity of all triggers in problem state and sort them by severity in descending order.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": [
            "triggerid",
            "description",
            "priority"
        ],
        "filter": {
            "value": 1
        },
        "sortfield": "priority",
        "sortorder": "DESC"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "13907",
            "description": "Zabbix self-monitoring processes < 100% busy",
            "priority": "4"
        },
        {
            "triggerid": "13824",
            "description": "Zabbix discoverer processes more than 75% busy",
            "priority": "3"
        }
    ],
```

```
    "id": 1
}
```

Retrieving a specific trigger with tags

Retrieve a specific trigger with tags.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.get",
    "params": {
        "output": [
            "triggerid",
            "description"
        ],
        "selectTags": "extend",
        "triggerids": [
            "17578"
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "17370",
            "description": "Service status",
            "tags": [
                {
                    "tag": "service",
                    "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}"
                },
                {
                    "tag": "error",
                    "value": ""
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Discovery rule
- Item
- Host
- Host group
- Template group

Source

CTrigger::get() in *ui/include/classes/api/services/CTrigger.php*.

trigger.update

Description

`object trigger.update(object/array triggers)`

This method allows to update existing triggers.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Trigger properties to be updated.

The `triggerid` property must be defined for each trigger, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard trigger properties the method accepts the following parameters.

Parameter	Type	Description
dependencies	array	Triggers that the trigger is dependent on. The triggers must have the `triggerid` property defined.
tags	array	Trigger tags.

Attention:
The trigger expression has to be given in its expanded form.

Return values

`(object)` Returns an object containing the IDs of the updated triggers under the `triggerids` property.

Examples

Enabling a trigger

Enable a trigger, that is, set its status to "0".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

Replacing triggers tags

Replace tags for trigger.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "13938",
```

```
        "tags": [
            {
                "tag": "service",
                "value": "{{ITEM.VALUE}.regsub(\"Service (.*) has stopped\", \"\\1\")}"
            },
            {
                "tag": "error",
                "value": ""
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

Replacing dependencies

Replace dependencies for trigger.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": {
        "triggerid": "22713",
        "dependencies": [
            {
                "triggerid": "22712"
            },
            {
                "triggerid": "22772"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "22713"
        ]
    },
    "id": 1
}
```

Source

CTrigger::update() in *ui/include/classes/api/services/CTrigger.php*.

Trigger prototype

This class is designed to work with trigger prototypes.

Object references:

- Trigger prototype

Available methods:

- triggerprototype.create - creating new trigger prototypes
- triggerprototype.delete - deleting trigger prototypes
- triggerprototype.get - retrieving trigger prototypes
- triggerprototype.update - updating trigger prototypes

> Trigger prototype object

The following objects are directly related to the `triggerprototype` API.

Trigger prototype

The trigger prototype object has the following properties.

Property	Type	Description
triggerid	string	ID of the trigger prototype. Property behavior: - *read-only* - *required* for update operations
description	string	Name of the trigger prototype. Property behavior: - *required* for create operations
expression	string	Reduced trigger expression. Property behavior: - *required* for create operations
event_name	string	Event name generated by the trigger.
opdata	string	Operational data.
comments	string	Additional comments to the trigger prototype.
priority	integer	Severity of the trigger prototype. Possible values: 0 - *(default)* not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
status	integer	Whether the trigger prototype is enabled or disabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
templateid	string	ID of the parent template trigger prototype. Property behavior: - *read-only*
type	integer	Whether the trigger prototype can generate multiple problem events. Possible values: 0 - *(default)* do not generate multiple events; 1 - generate multiple events.
url	string	URL associated with the trigger prototype.
url_name	string	Label for the URL associated with the trigger prototype.

Property	Type	Description
recovery_mode	integer	OK event generation mode. Possible values: 0 - *(default)* Expression; 1 - Recovery expression; 2 - None.
recovery_expression	string	Reduced trigger recovery expression.
correlation_mode	integer	OK event closes. Possible values: 0 - *(default)* All problems; 1 - All problems if tag values match.
correlation_tag	string	Tag for matching.
manual_close	integer	Allow manual close. Possible values: 0 - *(default)* No; 1 - Yes.
discover	integer	Trigger prototype discovery status. Possible values: 0 - *(default)* new triggers will be discovered; 1 - new triggers will not be discovered and existing triggers will be marked as lost.
uuid	string	Universal unique identifier, used for linking imported trigger prototypes to already existing ones. Auto-generated, if not given. Property behavior: - *supported* if the trigger prototype belongs to a template

Trigger prototype tag

The trigger prototype tag object has the following properties.

Property	Type	Description
tag	string	Trigger prototype tag name. Property behavior: - *required*
value	string	Trigger prototype tag value.

triggerprototype.create

Description

`object triggerprototype.create(object/array triggerPrototypes)`

This method allows to create new trigger prototypes.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Trigger prototypes to create.

Additionally to the standard trigger prototype properties the method accepts the following parameters.

Parameter	Type	Description
dependencies	array	Triggers and trigger prototypes that the trigger prototype is dependent on. The triggers must have the `triggerid` property defined.
tags	array	Trigger prototype tags.

Attention:
The trigger expression has to be given in its expanded form and must contain at least one item prototype.

Return values

`(object)` Returns an object containing the IDs of the created trigger prototypes under the `triggerids` property. The order of the returned IDs matches the order of the passed trigger prototypes.

Examples

Creating a trigger prototype

Create a trigger prototype to detect when a file system has less than 20% free disk space.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.create",
    "params": {
        "description": "Free disk space is less than 20% on volume {#FSNAME}",
        "expression": "last(/Zabbix server/vfs.fs.size[{#FSNAME},pfree])<20",
        "tags": [
            {
                "tag": "volume",
                "value": "{#FSNAME}"
            },
            {
                "tag": "type",
                "value": "{#FSTYPE}"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17372"
        ]
    },
    "id": 1
}
```

Source

CTriggerPrototype::create() in *ui/include/classes/api/services/CTriggerPrototype.php*.

triggerprototype.delete

Description

`object triggerprototype.delete(array triggerPrototypeIds)`

This method allows to delete trigger prototypes.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the trigger prototypes to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted trigger prototypes under the `triggerids` property.

Examples

Deleting multiple trigger prototypes

Delete two trigger prototypes.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.delete",
    "params": [
        "12002",
        "12003"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "12002",
            "12003"
        ]
    },
    "id": 1
}
```

Source

CTriggerPrototype::delete() in *ui/include/classes/api/services/CTriggerPrototype.php*.

triggerprototype.get

Description

`integer/array triggerprototype.get(object parameters)`

The method allows to retrieve trigger prototypes according to the given parameters.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
active	flag	Return only enabled trigger prototypes that belong to monitored hosts.
discoveryids	string/array	Return only trigger prototypes that belong to the given LLD rules.

Parameter	Type	Description
functions	string/array	Return only triggers that use the given functions. Refer to the Supported functions page for a list of supported functions.
group	string	Return only trigger prototypes that belong to hosts or templates from the host groups or template groups with the given name.
groupids	string/array	Return only trigger prototypes that belong to hosts or templates from the given host groups or template groups.
host	string	Return only trigger prototypes that belong to hosts with the given name.
hostids	string/array	Return only trigger prototypes that belong to the given hosts.
inherited	boolean	If set to `true` return only trigger prototypes inherited from a template.
maintenance	boolean	If set to `true` return only enabled trigger prototypes that belong to hosts in maintenance.
min_severity	integer	Return only trigger prototypes with severity greater or equal than the given severity.
monitored	flag	Return only enabled trigger prototypes that belong to monitored hosts and contain only enabled items.
templated	boolean	If set to `true` return only trigger prototypes that belong to templates.
templateids	string/array	Return only trigger prototypes that belong to the given templates.
triggerids	string/array	Return only trigger prototypes with the given IDs.
expandExpression	flag	Expand functions and macros in the trigger expression.
selectDependencies	query	Return trigger prototypes and triggers that the trigger prototype depends on in the `dependencies` property.
selectDiscoveryRule	query	Return the LLD rule that the trigger prototype belongs to.
selectFunctions	query	Return functions used in the trigger prototype in the `functions` property. The function objects represent the functions used in the trigger expression and has the following properties: `functionid` - `(string)` ID of the function; `itemid` - `(string)` ID of the item used in the function; `function` - `(string)` name of the function; `parameter` - `(string)` parameter passed to the function. Query parameter is replaced by $ symbol in returned string.
selectHostGroups	query	Return the host groups that the trigger prototype belongs to in the hostgroups property.
selectHosts	query	Return the hosts that the trigger prototype belongs to in the hosts property.
selectItems	query	Return items and item prototypes used the trigger prototype in the items property.
selectTags	query	Return the trigger prototype tags in tags property.
selectTemplateGroups	query	Return the template groups that the trigger prototype belongs to in the templategroups property.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: `host` - technical name of the host that the trigger prototype belongs to; `hostid` - ID of the host that the trigger prototype belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: `selectHosts` - results will be sorted by `host`.
sortfield	string/array	Sort the result by the given properties. Possible values: `triggerid`, `description`, `status`, `priority`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	

Parameter	Type	Description
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectGroups (deprecated)	query	This parameter is deprecated, please use `selectHostGroups` or `selectTemplateGroups` instead. Return the host groups and template groups that the trigger prototype belongs to in the `groups` property.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieve trigger prototypes from an LLD rule

Retrieve all trigger prototypes and their functions from an LLD rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": "extend",
        "selectFunctions": "extend",
        "discoveryids": "22450"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "13272",
            "expression": "{12598}<20",
            "description": "Free inodes is less than 20% on volume {#FSNAME}",
            "url": "",
            "status": "0",
            "value": "0",
            "priority": "2",
            "lastchange": "0",
            "comments": "",
            "error": "",
            "templateid": "0",
            "type": "0",
            "state": "0",
            "flags": "2",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0",
```

```
            "opdata": "",
            "discover": "0",
            "event_name": "",
            "uuid": "6ce467d05e8745409a177799bed34bb3",
            "url_name": "",
            "functions": [
                {
                    "functionid": "12598",
                    "itemid": "22454",
                    "triggerid": "13272",
                    "parameter": "$",
                    "function": "last"
                }
            ]
        },
        {
            "triggerid": "13266",
            "expression": "{13500}<20",
            "description": "Free disk space is less than 20% on volume {#FSNAME}",
            "url": "",
            "status": "0",
            "value": "0",
            "priority": "2",
            "lastchange": "0",
            "comments": "",
            "error": "",
            "templateid": "0",
            "type": "0",
            "state": "0",
            "flags": "2",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0",
            "opdata": "",
            "discover": "0",
            "event_name": "",
            "uuid": "74a1fc62bfe24b7eabe4e244c70dc384",
            "url_name": "",
            "functions": [
                {
                    "functionid": "13500",
                    "itemid": "22686",
                    "triggerid": "13266",
                    "parameter": "$",
                    "function": "last"
                }
            ]
        }
    ],
    "id": 1
}
```


Retrieving a specific trigger prototype with tags

Request:


```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.get",
    "params": {
        "output": [
            "triggerid",
```

```
            "description"
        ],
        "selectTags": "extend",
        "triggerids": [
            "17373"
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "triggerid": "17373",
            "description": "Free disk space is less than 20% on volume {#FSNAME}",
            "tags": [
                {
                    "tag": "volume",
                    "value": "{#FSNAME}"
                },
                {
                    "tag": "type",
                    "value": "{#FSTYPE}"
                }
            ]
        }
    ],
    "id": 1
}
```

See also

- Discovery rule
- Item
- Host
- Host group
- Template group

Source

CTriggerPrototype::get() in *ui/include/classes/api/services/CTriggerPrototype.php*.

triggerprototype.update

Description

`object triggerprototype.update(object/array triggerPrototypes)`

This method allows to update existing trigger prototypes.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Trigger prototype properties to be updated.

The `triggerid` property must be defined for each trigger prototype, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard trigger prototype properties the method accepts the following parameters.

Parameter	Type	Description
dependencies	array	Triggers and trigger prototypes that the trigger prototype is dependent on. The triggers must have the `triggerid` property defined.
tags	array	Trigger prototype tags.

Attention:
The trigger expression has to be given in its expanded form and must contain at least one item prototype.

Return values

`(object)` Returns an object containing the IDs of the updated trigger prototypes under the `triggerids` property.

Examples

Enabling a trigger prototype

Enable a trigger prototype, that is, set its status to "0".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
        "triggerid": "13938",
        "status": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "13938"
        ]
    },
    "id": 1
}
```

Replacing trigger prototype tags

Replace tags for one trigger prototype.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "triggerprototype.update",
    "params": {
        "triggerid": "17373",
        "tags": [
            {
                "tag": "volume",
                "value": "{#FSNAME}"
            },
            {
                "tag": "type",
                "value": "{#FSTYPE}"
            }
        ]
    },
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "17373"
        ]
    },
    "id": 1
}
```

Source

CTriggerPrototype::update() in *ui/include/classes/api/services/CTriggerPrototype.php*.

User

This class is designed to work with users.

Object references:

- User

Available methods:

- user.checkauthentication - checking and prolonging user sessions
- user.create - creating new users
- user.delete - deleting users
- user.get - retrieving users
- user.login - logging in to the API
- user.logout - logging out of the API
- user.unblock - unblocking users
- user.update - updating users

> User object

The following objects are directly related to the `user` API.

User

The user object has the following properties.

Property	Type	Description
userid	string	ID of the user. Property behavior: - *read-only* - *required* for update operations

Property	Type	Description
username	string	User's name. Property behavior: - *required* for create operations - *read-only* for provisioned users if the user is linked to a user directory (`userdirectoryid` is set to a valid value that is not "0"), and user directory provisioning status is enabled (`provision_status` of User directory object is set to "1"), and authentication status of all LDAP or SAML provisioning is enabled (`ldap_jit_status` of Authentication object is set to "Enabled for configured LDAP IdPs" or `saml_jit_status` of Authentication object is set to "Enabled for configured SAML IdPs")
passwd	string	User's password. The value of this parameter can be an empty string if the user is linked to a user directory. Property behavior: - *write-only*
roleid	string	Role ID of the user. Note that users without a role can log into Zabbix only using LDAP or SAML authentication, provided their LDAP/SAML information matches the user group mappings configured in Zabbix.
attempt_clock	timestamp	Time of the last unsuccessful login attempt. Property behavior: - *read-only*
attempt_failed	integer	Recent failed login attempt count. Property behavior: - *read-only*
attempt_ip	string	IP address from where the last unsuccessful login attempt came from. Property behavior: - *read-only*
autologin	integer	Whether to enable auto-login. Possible values: 0 - *(default)* auto-login disabled; 1 - auto-login enabled.
autologout	string	User session life time. Accepts seconds and time unit with suffix. If set to 0s, the session will never expire. Default: 15m.
lang	string	Language code of the user's language, for example, `en_US`. Default: `default` - system default.
name	string	Name of the user.
refresh	string	Automatic refresh period. Accepts seconds or time unit with suffix (e.g., 30s, 90s, 1m, 1h). Default: 30s.
rows_per_page	integer	Amount of object rows to show per page. Default: 50.
surname	string	Surname of the user.

Property	Type	Description
theme	string	User's theme. Possible values: `default` - *(default)* system default; `blue-theme` - Blue; `dark-theme` - Dark.
ts_provisioned	timestamp	Time when the latest provisioning operation was made. Property behavior: - *read-only*
url	string	URL of the page to redirect the user to after logging in.
userdirectoryid	string	ID of the user directory that the user in linked to. Used for provisioning (creating or updating), as well as to login a user that is linked to a user directory. For login operations the value of this property will have priority over the `userdirectoryid` property of user groups that the user belongs to. Default: 0.
timezone	string	User's time zone, for example, `Europe/London`, `UTC`. Default: `default` - system default. For the full list of supported time zones please refer to PHP documentation.

Media

The media object has the following properties.

Property	Type	Description
mediatypeid	string	ID of the media type used by the media. Property behavior: - *required*
sendto	string/array	Address, user name or other identifier of the recipient. If `type` of Media type is set to "Email", values are represented as array. For other types of Media types, value is represented as a string. Property behavior: - *required*
active	integer	Whether the media is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.

Property	Type	Description
severity	integer	Trigger severities to send notifications about. Possible bitmap values: 1 - Not classified; 2 - Information; 4 - Warning; 8 - Average; 16 - High; 32 - Disaster. This is a bitmask field; any sum of possible bitmap values is acceptable (for example, 48 for Average, High, and Disaster). Default: 63.
period	string	Time when the notifications can be sent as a time period or user macros separated by a semicolon. Default: 1-7,00:00-24:00.

user.checkAuthentication

Description

`object user.checkAuthentication`

This method checks and prolongs the user session.

Attention:
Calling the `user.checkAuthentication` method using the parameter `sessionid` prolongs the user session by default.

Parameters

The method accepts the following parameters.

Parameter	Type	Description
extend	boolean	Whether to prolong the user session. Default value: "true". Setting the value to "false" allows to check the user session without prolonging it.
sessionid	string	Parameter behavior: - *supported* if `sessionid` is set User authentication token. Parameter behavior: - *required* if `token` is not set
secret	string	Random 32 characters string. Is generated on user login.
token	string	User API token. Parameter behavior: - *required* if `sessionid` is not set

Return values

`(object)` Returns an object containing information about the user.

Additionally to the standard user properties, the following information is returned.

Property	Type	Description
auth_type	integer	Default authentication for the user. Refer to the `authentication_type` property of the Authentication object for a list of possible values.
debug_mode	integer	Whether debug mode is enabled or disabled for the user. Refer to the `debug_mode` property of the User group object for a list of possible values.
deprovisioned	boolean	Whether the user belongs to a deprovisioned users group.
gui_access	string	User's authentication method to the frontend. Refer to the `gui_access` property of the User group object for a list of possible values.
secret	string	Random 32 characters string. Is generated on user login. Property `secret` is not returned if the user session is checked using an API token.
sessionid	string	Authentication token, which must be used in the following API requests. Property `sessionid` is not returned if the user session is checked using an API token.
type	integer	User type. Refer to the `type` property of the Role object for a list of possible values.
userip	string	IP address of the user.

Examples

Check authentication using authentication token

Check and prolong a user session using the user authentication token, and return additional information about the user.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.checkAuthentication",
    "params": {
        "sessionid": "673b8ba11562a35da902c66cf5c23fa2"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userid": "1",
        "username": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919038",
        "rows_per_page": "50",
```

```
        "timezone": "Europe/Riga",
        "roleid": "3",
        "userdirectoryid": "0",
        "ts_provisioned": "0",
        "type": 3,
        "userip": "127.0.0.1",
        "debug_mode": 0,
        "gui_access": "0",
        "deprovisioned": false,
        "auth_type": 0,
        "sessionid": "673b8ba11562a35da902c66cf5c23fa2",
        "secret": "0e329b933e46984e49a5c1051ecd0751"
    },
    "id": 1
}
```

Check authentication using API token

Check a user session using the user API token, and return additional information about the user.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.checkAuthentication",
    "params": {
        "token": "00aff470e07c12d707e50d98cfe39edef9e6ec349c14728dbdfbc8ddc5ea3eae"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userid": "1",
        "username": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919338",
        "rows_per_page": "50",
        "timezone": "Europe/Riga",
        "roleid": "3",
        "userdirectoryid": "0",
        "ts_provisioned": "0",
        "type": 3,
        "userip": "127.0.0.1",
        "debug_mode": 0,
        "gui_access": "1",
        "deprovisioned": false,
        "auth_type": 0
    },
    "id": 1
}
```

Source

CUser::checkAuthentication() in *ui/include/classes/api/services/CUser.php*.

user.create

Description

`object user.create(object/array users)`

This method allows to create new users.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

> **Note:**
> The strength of user password is validated according the password policy rules defined by Authentication API. See Authentication API for more information.

Parameters

`(object/array)` Users to create.

Additionally to the standard user properties, the method accepts the following parameters.

Parameter	Type	Description
usrgrps	array	User groups to add the user to. The user groups must have the `usrgrpid` property defined.
medias	array	User media to be created.

Return values

`(object)` Returns an object containing the IDs of the created users under the `userids` property. The order of the returned IDs matches the order of the passed users.

Examples

Creating a user

Create a new user, add him to a user group and create a new media for him.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.create",
    "params": {
        "username": "John",
        "passwd": "Doe123",
        "roleid": "5",
        "usrgrps": [
            {
                "usrgrpid": "7"
            }
        ],
        "medias": [
            {
                "mediatypeid": "1",
                "sendto": [
                    "support@company.com"
                ],
                "active": 0,
                "severity": 63,
                "period": "1-7,00:00-24:00"
            }
```

```
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "12"
        ]
    },
    "id": 1
}
```

See also

- Authentication
- Media
- User group
- Role

Source

CUser::create() in *ui/include/classes/api/services/CUser.php*.

user.delete

Description

`object user.delete(array users)`

This method allows to delete users.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of users to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted users under the `userids` property.

Examples

Deleting multiple users

Delete two users.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.delete",
    "params": [
        "1",
        "5"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "userids": [
            "1",
            "5"
        ]
    },
    "id": 1
}
```

Source

CUser::delete() in *ui/include/classes/api/services/CUser.php*.

user.get

Description

`integer/array user.get(object parameters)`

The method allows to retrieve users according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
mediaids	string/array	Return only users that use the given media.
mediatypeids	string/array	Return only users that use the given media types.
userids	string/array	Return only users with the given IDs.
usrgrpids	string/array	Return only users that belong to the given user groups.
getAccess	flag	Adds additional information about user permissions. Adds the following properties for each user: `gui_access` - `(integer)` user's frontend authentication method. Refer to the `gui_access` property of the user group object for a list of possible values. `debug_mode` - `(integer)` indicates whether debug is enabled for the user. Possible values: 0 - debug disabled, 1 - debug enabled. `users_status` - `(integer)` indicates whether the user is disabled. Possible values: 0 - user enabled, 1 - user disabled.
selectMedias	query	Return media used by the user in the medias property.
selectMediatypes	query	Return media types used by the user in the mediatypes property.
selectUsrgrps	query	Return user groups that the user belongs to in the usrgrps property.
selectRole	query	Return user role in the role property.
sortfield	string/array	Sort the result by the given properties. Possible values: `userid`, `username`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	

Parameter	Type	Description
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving users

Retrieve all of the configured users.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "userid": "1",
            "username": "Admin",
            "name": "Zabbix",
            "surname": "Administrator",
            "url": "",
            "autologin": "1",
            "autologout": "0",
            "lang": "en_US",
            "refresh": "0s",
            "theme": "default",
            "attempt_failed": "0",
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "50",
            "timezone": "default",
            "roleid": "3",
            "userdirectoryid": "0",
            "ts_provisioned": "0"
        },
        {
            "userid": "2",
            "username": "guest",
            "name": "",
            "surname": "",
            "url": "",
            "autologin": "0",
            "autologout": "15m",
            "lang": "default",
            "refresh": "30s",
            "theme": "default",
            "attempt_failed": "0",
            "attempt_ip": "",
```

```
            "attempt_clock": "0",
            "rows_per_page": "50",
            "timezone": "default",
            "roleid": "4",
            "userdirectoryid": "0",
            "ts_provisioned": "0"
        },
        {
            "userid": "3",
            "username": "user",
            "name": "Zabbix",
            "surname": "User",
            "url": "",
            "autologin": "0",
            "autologout": "0",
            "lang": "ru_RU",
            "refresh": "15s",
            "theme": "dark-theme",
            "attempt_failed": "0",
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "100",
            "timezone": "default",
            "roleid": "1",
            "userdirectoryid": "0",
            "ts_provisioned": "0"
        }
    ],
    "id": 1
}
```


Retrieving user data

Retrieve data of a user with ID "12".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.get",
    "params": {
        "output": ["userid", "username"],
        "selectRole": "extend",
        "userids": "12"
    },
    "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
      {
          "userid": "12",
          "username": "John",
          "role": {
              "roleid": "5",
              "name": "Operator",
              "type": "1",
              "readonly": "0"
          }
      }
  ],
  "id": 1
```

```
}
```

See also

- Media
- Media type
- User group
- Role

Source

CUser::get() in *ui/include/classes/api/services/CUser.php*.

user.login

Description

`string/object user.login(object parameters)`

This method allows to log in to the API and generate an authentication token.

Warning:
When using this method, you also need to do user.logout to prevent the generation of a large number of open session records.

Attention:
This method is only available to unauthenticated users and must be called without the `auth` parameter in the JSON-RPC request.

Parameters

`(object)` Parameters containing the user name and password.

The method accepts the following parameters.

Parameter	Type	Description
password	string	User password. Parameter behavior: - *required*
username	string	User name. Parameter behavior: - *required*
userData	flag	Return information about the authenticated user.

Return values

`(string/object)` If the `userData` parameter is used, returns an object containing information about the authenticated user.

Additionally to the standard user properties, the following information is returned:

Property	Type	Description
auth_type	integer	Default authentication for the user. Refer to the `authentication_type` property of the Authentication object for a list of possible values.
debug_mode	integer	Whether debug mode is enabled or disabled for the user. Refer to the `debug_mode` property of the User group object for a list of possible values.
deprovisioned	boolean	Whether the user belongs to a deprovisioned users group.
gui_access	string	User's authentication method to the frontend. Refer to the `gui_access` property of the User group object for a list of possible values.

Property	Type	Description
secret	string	Random 32 characters string. Is generated on user login.
sessionid	string	Authentication token, which must be used in the following API requests.
type	integer	User type. Refer to the `type` property of the Role object for a list of possible values.
userip	string	IP address of the user.

Note:
If a user has been successfully authenticated after one or more failed attempts, the method will return the current values for the `attempt_clock`, `attempt_failed` and `attempt_ip` properties and then reset them.

If the `userData` parameter is not used, the method returns an authentication token.

Note:
The generated authentication token should be remembered and used in the `auth` parameter of the following JSON-RPC requests. It is also required when using HTTP authentication.

Examples

Authenticating a user

Authenticate a user.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "username": "Admin",
        "password": "zabbix"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

Requesting authenticated user's information

Authenticate and return additional information about the user.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "username": "Admin",
        "password": "zabbix",
        "userData": true
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "userid": "1",
        "username": "Admin",
        "name": "Zabbix",
        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919038",
        "rows_per_page": "50",
        "timezone": "Europe/Riga",
        "roleid": "3",
        "userdirectoryid": "0",
        "type": 3,
        "userip": "127.0.0.1",
        "debug_mode": 0,
        "gui_access": "0",
        "deprovisioned": false,
        "auth_type": 0,
        "sessionid": "5b56eee8be445e98f0bd42b435736e42",
        "secret": "cd0ba923319741c6586f3d866423a8f4"
    },
    "id": 1
}
```


See also

- user.logout

Source

CUser::login() in *ui/include/classes/api/services/CUser.php*.

user.logout

Description

`string/object user.logout(array)`

This method allows to log out of the API and invalidates the current authentication token.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` The method accepts an empty array.

Return values

`(boolean)` Returns `true` if the user has been logged out successfully.

Examples

Logging out

Log out from the API.

Request:


```
{
    "jsonrpc": "2.0",
```

```
    "method": "user.logout",
    "params": [],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- user.login

Source

CUser::login() in *ui/include/classes/api/services/CUser.php*.

user.provision

Description

`object user.provision(object/array users)`

This method allows to provision LDAP users.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of users to provision.

Return values

`(object)` Returns an object containing the IDs of the provisioned users under the `userids` property.

Examples

Provisioning multiple users

Provision two users.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.provision",
    "params": [
        "1",
        "5"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1",
            "5"
        ]
    },
```

```
    "id": 1
}
```

Source

CUser::provision() in *ui/include/classes/api/services/CUser.php*.

user.unblock

Description

`object user.unblock(array userids)`

This method allows to unblock users.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of users to unblock.

Return values

`(object)` Returns an object containing the IDs of the unblocked users under the `userids` property.

Examples

Unblocking multiple users

Unblock two users.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.unblock",
    "params": [
        "1",
        "5"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1",
            "5"
        ]
    },
    "id": 1
}
```

Source

CUser::unblock() in *ui/include/classes/api/services/CUser.php*.

user.update

Description

`object user.update(object/array users)`

This method allows to update existing users.

Note:
This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Note:
The strength of user password is validated according the password policy rules defined by Authentication API. See Authentication API for more information.

Parameters

`(object/array)` User properties to be updated.

The `userid` property must be defined for each user, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard user properties, the method accepts the following parameters.

Parameter	Type	Description
current_passwd	string	User's current password. The value of this parameter can be an empty string if the user is linked to a user directory. Parameter behavior: - *write-only* - *required* if `passwd` of User object is set and user changes own user password
usrgrps	array	User groups to replace existing user groups. The user groups must have the `usrgrpid` property defined.
medias	array	User media to replace existing media.

Return values

`(object)` Returns an object containing the IDs of the updated users under the `userids` property.

Examples

Renaming a user

Rename a user to John Doe.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.update",
    "params": {
        "userid": "1",
        "name": "John",
        "surname": "Doe"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "1"
        ]
    },
    "id": 1
}
```

Changing user role

Change a role of a user.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "user.update",
    "params": {
        "userid": "12",
        "roleid": "6"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userids": [
            "12"
        ]
    },
    "id": 1
}
```

See also

- Authentication

Source

CUser::update() in *ui/include/classes/api/services/CUser.php*.

User directory

This class is designed to work with user directories.

Object references:

- User directory

Available methods:

- userdirectory.create - create new user directory
- userdirectory.delete - delete user directory
- userdirectory.get - retrieve user directory
- userdirectory.update - update user directory
- userdirectory.test - test user directory connection

> User directory object

The following objects are directly related to the `userdirectory` API.

User directory

The user directory object has the following properties.

Property	Type	Description
userdirectoryid	string	ID of the user directory. If a user directory is deleted, the value of the User object property `userdirectoryid` is set to ”0” for all users that are linked to the deleted user directory. Property behavior: - *read-only* - *required* for update operations
idp_type	integer	Type of the authentication protocol used by the identity provider for the user directory. Note that only one user directory of type SAML can exist. Possible values: 1 - User directory of type LDAP; 2 - User directory of type SAML. Property behavior: - *required* for create operations
group_name	string	LDAP/SAML user directory attribute that contains the group name used to map groups between the LDAP/SAML user directory and Zabbix. Example: *cn* Property behavior: - *required* if `provision_status` is set to ”Enabled” and `saml_jit_status` of Authentication object is set to ”Enabled for configured SAML IdPs”
user_username	string	LDAP/SAML user directory attribute (also SCIM attribute if `scim_status` is set to ”SCIM provisioning is enabled”) that contains the user’s name which is used as the value for the User object property `name` when the user is provisioned. Examples: *cn*, *commonName*, *displayName*, *name*
user_lastname	string	LDAP/SAML user directory attribute (also SCIM attribute if `scim_status` is set to ”SCIM provisioning is enabled”) that contains the user’s last name which is used as the value for the User object property `surname` when the user is provisioned. Examples: *sn*, *surname*, *lastName*
provision_status	integer	Provisioning status of the user directory. Possible values: 0 - *(default)* Disabled (provisioning of users created by this user directory is disabled); 1 - Enabled (provisioning of users created by this user directory is enabled; additionally, the status of LDAP or SAML provisioning (`ldap_jit_status` or `saml_jit_status` of Authentication object) must be enabled).
provision_groups	array	Array of provisioning groups mappings objects for mapping LDAP/SAML user group pattern to Zabbix user group and user role. Property behavior: - *required* if `provision_status` is set to ”Enabled”
provision_media	array	Array of media type mappings objects for mapping user’s LDAP/SAML media attributes (e.g., email) to Zabbix user media for sending notifications.
LDAP-specific properties:		

Property	Type	Description
name	string	Unique name of the user directory. Property behavior: - *required* if `idp_type` is set to "User directory of type LDAP"
host	string	Host name, IP or URI of the LDAP server. URI must contain schema (`ldap://` or `ldaps://`), host, and port (optional). Examples: *host.example.com* *127.0.0.1* *ldap://ldap.example.com:389* Property behavior: - *required* if `idp_type` is set to "User directory of type LDAP"
port	integer	Port of the LDAP server. Property behavior: - *required* if `idp_type` is set to "User directory of type LDAP"
base_dn	string	LDAP user directory base path to user accounts. Examples: *ou=Users,dc=example,dc=org* *ou=Users,ou=system* (for OpenLDAP) *DC=company,DC=com* (for Microsoft Active Directory) *uid=%{user},dc=example,dc=com* (for direct user binding; placeholder "*%{user}*" is mandatory) Property behavior: - *required* if `idp_type` is set to "User directory of type LDAP"
search_attribute	string	LDAP user directory attribute by which to identify the user account from the information provided in the login request. Examples: *uid* (for OpenLDAP) *sAMAccountName* (for Microsoft Active Directory) Property behavior: - *required* if `idp_type` is set to "User directory of type LDAP"
bind_dn	string	LDAP server account for binding and searching over the LDAP server. For direct user binding and anonymous binding, `bind_dn` must be empty. Examples: *uid=ldap_search,ou=system* (for OpenLDAP) *CN=ldap_search,OU=user_group,DC=company,DC=com* (for Microsoft Active Directory) *CN=Admin,OU=Users,OU=Zabbix,DC=zbx,DC=local* Property behavior: - *supported* if `idp_type` is set to "User directory of type LDAP"
bind_password	string	LDAP password of the account for binding and searching over the LDAP server. For direct user binding and anonymous binding, `bind_password` must be empty. Property behavior: - *supported* if `idp_type` is set to "User directory of type LDAP"

Property	Type	Description
description	string	Description of the user directory. Property behavior: - *supported* if `idp_type` is set to "User directory of type LDAP"
group_basedn	string	LDAP user directory base path to groups; used to configure a user membership check in the LDAP user directory. Ignored when provisioning a user if `group_membership` is set. Example: *ou=Groups,dc=example,dc=com* Property behavior: - *supported* if `idp_type` is set to "User directory of type LDAP"
group_filter	string	Filter string for retrieving LDAP user directory groups that the user is a member of; used to configure a user membership check in the LDAP user directory. Ignored when provisioning a user if `group_membership` is set. Supported `group_filter` placeholders: *%{attr}* - search attribute (replaced by the `search_attribute` property value); *%{groupattr}* - group attribute (replaced by the `group_member` property value); *%{host}* - host name, IP or URI of the LDAP server (replaced by the `host` property value); *%{user}* - Zabbix user username. Default: *(%{groupattr}=%{user})* Examples: - *(member=uid=%{ref},ou=Users,dc=example,dc=com)* will match "User1" if an LDAP group object contains the "*member*" attribute with the value "*uid=User1,ou=Users,dc=example,dc=com*", and will return the group that "User1" is a member of; - *(%{groupattr}=cn=%{ref},ou=Users,ou=Zabbix,DC=example,DC=com)* will match "User1" if an LDAP group object contains the attribute specified in the `group_member` property with the value "*cn=User1,ou=Users,ou=Zabbix,DC=example,DC=com*", and will return the group that "User1" is a member of. Property behavior: - *supported* if `idp_type` is set to "User directory of type LDAP"
group_member	string	LDAP user directory attribute that contains information about the group members; used to configure a user membership check in the LDAP user directory. Ignored when provisioning a user if `group_membership` is set. Property behavior: - *supported* if `idp_type` is set to "User directory of type LDAP"
group_membership	string	LDAP user directory attribute that contains information about the groups that a user belongs to. Example: *memberOf* Property behavior: - *supported* if `idp_type` is set to "User directory of type LDAP"

Property	Type	Description
search_filter	string	Custom filter string used to locate and authenticate a user in an LDAP user directory based on the information provided in the login request. Supported `search_filter` placeholders: *%{attr}* - search attribute name (e.g., *uid*, *sAMAccountName*); *%{user}* - Zabbix user username. Default: *(%{attr}=%{user})* Property behavior: - *supported* if `idp_type` is set to "User directory of type LDAP"
start_tls	integer	LDAP server configuration option that allows the communication with the LDAP server to be secured using Transport Layer Security (TLS). Note that `start_tls` must be set to "Disabled" for hosts using the `ldaps://` protocol. Possible values: 0 - *(default)* Disabled; 1 - Enabled. Property behavior: - *supported* if `idp_type` is set to "User directory of type LDAP"
user_ref_attr	string	LDAP user directory attribute used to reference a user object. The value of `user_ref_attr` is used to get values from the specified attribute in the user directory and place them instead of the *%{ref}* placeholder in the `group_filter` string. Examples: *cn*, *uid*, *member*, *uniqueMember* Property behavior: - *supported* if `idp_type` is set to "User directory of type LDAP"
SAML-specific properties:		
idp_entityid	string	URI that identifies the identity provider and is used to communicate with the identity provider in SAML messages. Example: *https://idp.example.com/idp* Property behavior: - *required* if `idp_type` is set to "User directory of type SAML"
sp_entityid	string	URL or any string that identifies the identity provider's service provider. Examples: *https://idp.example.com/sp* *zabbix* Property behavior: - *required* if `idp_type` is set to "User directory of type SAML"
username_attribute	string	SAML user directory attribute (also SCIM attribute if `scim_status` is set to "SCIM provisioning is enabled") that contains the user's username which is compared with the value of the User object property `username` when authenticating. Examples: *uid*, *userprincipalname*, *samaccountname*, *username*, *userusername*, *urn:oid:0.9.2342.19200300.100.1.1*, *urn:oid:1.3.6.1.4.1.5923.1.1.1.13*, *urn:oid:0.9.2342.19200300.100.1.44* Property behavior: - *required* if `idp_type` is set to "User directory of type SAML"

Property	Type	Description
sso_url	string	URL of the identity provider's SAML single sign-on service, to which Zabbix will send the SAML authentication requests. Example: *http://idp.example.com/idp/sso/saml* Property behavior: - *required* if `idp_type` is set to "User directory of type SAML"
slo_url	string	URL of the identity provider's SAML single log-out service, to which Zabbix will send the SAML logout requests. Example: *https://idp.example.com/idp/slo/saml* Property behavior: - *supported* if `idp_type` is set to "User directory of type SAML"
encrypt_nameid	integer	Whether the SAML name ID should be encrypted. Possible values: 0 - *(default)* Do not encrypt name ID; 1 - Encrypt name ID. Property behavior: - *supported* if `idp_type` is set to "User directory of type SAML"
encrypt_assertions	integer	Whether the SAML assertions should be encrypted. Possible values: 0 - *(default)* Do not encrypt assertions; 1 - Encrypt assertions. Property behavior: - *supported* if `idp_type` is set to "User directory of type SAML"
nameid_format	string	Name ID format of the SAML identity provider's service provider. Examples: *urn:oasis:names:tc:SAML:2.0:nameid-format:persistent* *urn:oasis:names:tc:SAML:2.0:nameid-format:transient* *urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos* *urn:oasis:names:tc:SAML:2.0:nameid-format:entity* Property behavior: - *supported* if `idp_type` is set to "User directory of type SAML"
scim_status	integer	Whether SCIM provisioning for SAML is enabled or disabled. Possible values: 0 - *(default)* SCIM provisioning is disabled; 1 - SCIM provisioning is enabled. Property behavior: - *supported* if `idp_type` is set to "User directory of type SAML"
sign_assertions	integer	Whether the SAML assertions should be signed with a SAML signature. Possible values: 0 - *(default)* Do not sign assertions; 1 - Sign assertions. Property behavior: - *supported* if `idp_type` is set to "User directory of type SAML"

Property	Type	Description
sign_authn_requests	integer	Whether the SAML AuthN requests should be signed with a SAML signature. Possible values: 0 - *(default)* Do not sign AuthN requests; 1 - Sign AuthN requests. Property behavior: - *supported* if `idp_type` is set to "User directory of type SAML"
sign_messages	integer	Whether the SAML messages should be signed with a SAML signature. Possible values: 0 - *(default)* Do not sign messages; 1 - Sign messages. Property behavior: - *supported* if `idp_type` is set to "User directory of type SAML"
sign_logout_requests	integer	Whether the SAML logout requests should be signed with a SAML signature. Possible values: 0 - *(default)* Do not sign logout requests; 1 - Sign logout requests. Property behavior: - *supported* if `idp_type` is set to "User directory of type SAML"
sign_logout_responses	integer	Whether the SAML logout responses should be signed with a SAML signature. Possible values: 0 - *(default)* Do not sign logout responses; 1 - Sign logout responses. Property behavior: - *supported* if `idp_type` is set to "User directory of type SAML"

Media type mappings

The media type mappings object has the following properties.

Property	Type	Description
name	string	Visible name in the list of media type mappings. Property behavior: - *required*
mediatypeid	string	ID of the media type to be created; used as the value for the Media object property `mediatypeid`. Property behavior: - *required*
attribute	string	LDAP/SAML user directory attribute (also SCIM attribute if `scim_status` is set to "SCIM provisioning is enabled") that contains the user's media (e.g., *user@example.com*) which is used as the value for the Media object property `sendto`. If present in data received from the LDAP/SAML identity provider, and the value is not empty, this will trigger media creation for the provisioned user. Property behavior: - *required*

Provisioning groups mappings

The provisioning groups mappings has the following properties.

Property	Type	Description
name	string	Full name of a group (e.g., *Zabbix administrators*) in LDAP/SAML user directory (also SCIM if `scim_status` is set to "SCIM provisioning is enabled"). Supports the wildcard character "*". Unique across all provisioning groups mappings. Property behavior: - *required*
roleid	string	ID of the user role to assign to the user. If multiple provisioning groups mappings are matched, the role of the highest user type (*User*, *Admin*, or *Super admin*) is assigned to the user. If there are multiple roles with the same user type, the first role (sorted in alphabetical order) is assigned to the user. Property behavior: - *required*
user_groups	array	Array of Zabbix user group ID objects. Each object has the following properties: `usrgrpid` - `(integer)` ID of Zabbix user group to assign to the user. If multiple provisioning groups mappings are matched, Zabbix user groups of all matched mappings is assigned to the user. Property behavior: - *required*

userdirectory.create

Description

`object userdirectory.create(object/array userDirectory)`

This method allows to create new user directories.

> **Note:**
> This method is only available to *Super admin* user type.

Parameters

`(object/array)` User directories to create.

The method accepts user directories with the standard user directory properties.

Return values

`(object)` Returns an object containing the IDs of the created user directories under the `userdirectoryids` property. The order of the returned IDs matches the order of the passed user directories.

Examples

Creating a user directory

Create a user directory to authenticate users with StartTLS over LDAP. Note that to authenticate users over LDAP, LDAP authentication must be enabled.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.create",
    "params": {
        "idp_type": "1",
        "name": "LDAP API server #1",
        "host": "ldap://local.ldap",
```

```
        "port": "389",
        "base_dn": "ou=Users,dc=example,dc=org",
        "bind_dn": "cn=ldap_search,dc=example,dc=org",
        "bind_password": "ldapsecretpassword",
        "search_attribute": "uid",
        "start_tls": "1"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userdirectoryids": [
            "3"
        ]
    },
    "id": 1
}
```

Creating a user directory (JIT provisioning enabled)

Create a user directory to authenticate users over LDAP (with JIT provisioning enabled). Note that to authenticate users over LDAP, LDAP authentication must be enabled.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.create",
    "params": {
            "idp_type": "1",
            "name": "AD server",
            "provision_status": "1",
            "description": "",
            "host": "host.example.com",
            "port": "389",
            "base_dn": "DC=zbx,DC=local",
            "search_attribute": "sAMAccountName",
            "bind_dn": "CN=Admin,OU=Users,OU=Zabbix,DC=zbx,DC=local",
            "start_tls": "0",
            "search_filter": "",
            "group_basedn": "OU=Zabbix,DC=zbx,DC=local",
            "group_name": "CN",
            "group_member": "member",
            "group_filter": "(%{groupattr}=CN=%{ref},OU=Users,OU=Zabbix,DC=zbx,DC=local)",
            "group_membership": "",
            "user_username": "givenName",
            "user_lastname": "sn",
            "user_ref_attr": "CN",
            "provision_media": [
                {
                    "name": "example.com",
                    "mediatypeid": "1",
                    "attribute": "user@example.com"
                }
            ],
            "provision_groups": [
                {
                    "name": "*",
                    "roleid": "4",
                    "user_groups": [
                        {
```

```
                            "usrgrpid": "8"
                        }
                    ]
                },
                {
                    "name": "Zabbix administrators",
                    "roleid": "2",
                    "user_groups": [
                        {
                            "usrgrpid": "7"
                        },
                        {
                            "usrgrpid": "8"
                        }
                    ]
                }
            ]
        },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userdirectoryids": [
            "2"
        ]
    },
    "id": 1
}
```

Source

CUserDirectory::create() in *ui/include/classes/api/services/CUserDirectory.php*.

userdirectory.delete

Description

`object userdirectory.delete(array userDirectoryIds)`

This method allows to delete user directories. User directory cannot be deleted when it is directly used for at least one user group. Default LDAP user directory cannot be deleted when `authentication.ldap_configured` is set to `1` or when there are more user directories left.

> **Note:**
> This method is only available to *Super admin* user type.

Parameters

`(array)` IDs of the user directories to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted user directories under the `userdirectoryids` property.

Examples

Deleting multiple user directories

Delete two user directories.

Request:

```
{
    "jsonrpc": "2.0",
```

```
    "method": "userdirectory.delete",
    "params": [
        "2",
        "12"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userdirectoryids": [
            "2",
            "12"
        ]
    },
    "id": 1
}
```

Source

CUserDirectory::delete() in *ui/include/classes/api/services/CUserDirectory.php*.

userdirectory.get

Description

`integer/array userdirectory.get(object parameters)`

The method allows to retrieve user directories according to the given parameters.

Note:
This method is only available to *Super admin* user types.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
userdirectoryids	string/array	Return only user directories with the given IDs.
selectUsrgrps	query	Return a `usrgrps` property with user groups associated with a user directory. Supports `count`.
selectProvisionMedia	query	Return a `provision_media` property with media type mappings associated with a user directory.
selectProvisionGroups	query	Return a `provision_groups` property with provisioning groups mappings associated with a user directory.
sortfield	string/array	Sort the result by the given properties. Possible values: `name`.
filter	object	Return only those results that exactly match the given filter. Accepts an object, where the keys are property names, and the values are either a single value or an array of values. Supported keys: `userdirectoryid`, `idp_type`, `provision_status`.

Parameter	Type	Description
search	object	Return results that match the given pattern (case-insensitive). br> Accepts an object, where the keys are property names, and the values are strings to search for. If no additional options are given, this will perform a `LIKE "%…%"` search. Supported properties: `name`, `description`. User directory of type SAML will have an empty value for both `name` and `description` fields. Both fields can be changed with `userdirectory.update` operation.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving user directories

Retrieve all user directories with additional properties that display media type mappings and provisioning groups mappings associated with each user directory.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.get",
    "params": {
        "output": "extend",
        "selectProvisionMedia": "extend",
        "selectProvisionGroups": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
       {
            "userdirectoryid": "1",
            "idp_type": "2",
            "name": "",
            "provision_status": "1",
            "description": "",
            "group_name": "groups",
            "user_username": "",
            "user_lastname": "",
            "idp_entityid": "http://example.com/simplesaml/saml2/idp/metadata.php",
```

```
            "sso_url": "http://example.com/simplesaml/saml2/idp/SSOService.php",
            "slo_url": "",
            "username_attribute": "uid",
            "sp_entityid": "zabbix",
            "nameid_format": "",
            "sign_messages": "0",
            "sign_assertions": "0",
            "sign_authn_requests": "0",
            "sign_logout_requests": "0",
            "sign_logout_responses": "0",
            "encrypt_nameid": "0",
            "encrypt_assertions": "0",
            "scim_status": "1",
            "provision_media": [
                {
                    "name": "example.com",
                    "mediatypeid": "1",
                    "attribute": "user@example.com"
                }
            ],
            "provision_groups": [
                {
                    "name": "*",
                    "roleid": "1",
                    "user_groups": [
                        {
                            "usrgrpid": "13"
                        }
                    ]
                }
            ]
        },
        {
            "userdirectoryid": "2",
            "idp_type": "1",
            "name": "AD server",
            "provision_status": "1",
            "description": "",
            "host": "host.example.com",
            "port": "389",
            "base_dn": "DC=zbx,DC=local",
            "search_attribute": "sAMAccountName",
            "bind_dn": "CN=Admin,OU=Users,OU=Zabbix,DC=zbx,DC=local",
            "start_tls": "0",
            "search_filter": "",
            "group_basedn": "OU=Zabbix,DC=zbx,DC=local",
            "group_name": "CN",
            "group_member": "member",
            "group_filter": "(%{groupattr}=CN=%{ref},OU=Users,OU=Zabbix,DC=zbx,DC=local)",
            "group_membership": "",
            "user_username": "givenName",
            "user_lastname": "sn",
            "user_ref_attr": "CN",
            "provision_media": [
                {
                    "name": "example.com",
                    "mediatypeid": "1",
                    "attribute": "user@example.com"
                }
            ],
            "provision_groups": [
                {
```

```
                    "name": "*",
                    "roleid": "4",
                    "user_groups": [
                        {
                            "usrgrpid": "8"
                        }
                    ]
                },
                {
                    "name": "Zabbix administrators",
                    "roleid": "2",
                    "user_groups": [
                        {
                            "usrgrpid": "7"
                        },
                        {
                            "usrgrpid": "8"
                        }
                    ]
                }
            ]
        },
        {
            "userdirectoryid": "3",
            "idp_type": "1",
            "name": "LDAP API server #1",
            "provision_status": "0",
            "description": "",
            "host": "ldap://local.ldap",
            "port": "389",
            "base_dn": "ou=Users,dc=example,dc=org",
            "search_attribute": "uid",
            "bind_dn": "cn=ldap_search,dc=example,dc=org",
            "start_tls": "1",
            "search_filter": "",
            "group_basedn": "",
            "group_name": "",
            "group_member": "",
            "group_filter": "",
            "group_membership": "",
            "user_username": "",
            "user_lastname": "",
            "user_ref_attr": "",
            "provision_media": [],
            "provision_groups": []
        }
    ],
    "id": 1
}
```


See also

- User group

Source

CUserDirectory::get() in *ui/include/classes/api/services/CUserDirectory.php*.

userdirectory.test

Description

`object userdirectory.test(array userDirectory)`

This method allows to test user directory connection settings.

Note:
This method also allows to test what configured data matches the user directory settings for user provisioning (e.g., what user role, user groups, user medias will be assigned to the user). For this type of test the API request should be made for a user directory that has `provision_status` set to enabled.

Note:
This method is only available to *Super admin* user type.

Parameters

`(object)` User directory properties.

Since `userdirectory.get` API does not return `bind_password` field, `userdirectoryid` and/or `bind_password` should be supplied.
Additionally to the standard user directory properties, the method accepts the following parameters.

Parameter	Type	Description
test_username	string	Username to test in user directory.
test_password	string	Username associated password to test in user directory.

Return values

`(bool)` Returns true on success.

Examples

Test user directory for existing user

Test user directory "3" for "user1".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.test",
    "params": {
        "userdirectoryid": "3",
        "host": "127.0.0.1",
        "port": "389",
        "base_dn": "ou=Users,dc=example,dc=org",
        "search_attribute": "uid",
        "bind_dn": "cn=ldap_search,dc=example,dc=org",
        "bind_password": "password",
        "test_username": "user1",
        "test_password": "password"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Test user directory for non-existing user

Test user directory "3" for non-existing "user2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.test",
    "params": {
```

```
        "userdirectoryid": "3",
        "host": "127.0.0.1",
        "port": "389",
        "base_dn": "ou=Users,dc=example,dc=org",
        "search_attribute": "uid",
        "bind_dn": "cn=ldap_search,dc=example,dc=org",
        "test_username": "user2",
        "test_password": "password"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "error": {
        "code": -32500,
        "message": "Application error.",
        "data": "Incorrect user name or password or account is temporarily blocked."
    },
    "id": 1
}
```

Test user directory for user provisioning

Test userdirectory ”3” for what configured data matches the user directory settings for ”user3” provisioning (e.g., what user role, user groups, user medias will be assigned to the user).

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.test",
    "params": {
        "userdirectoryid": "2",
        "host": "host.example.com",
        "port": "389",
        "base_dn": "DC=zbx,DC=local",
        "search_attribute": "sAMAccountName",
        "bind_dn": "CN=Admin,OU=Users,OU=Zabbix,DC=zbx,DC=local",
        "test_username": "user3",
        "test_password": "password"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "username": "user3",
        "name": "John",
        "surname": "Doe",
        "medias": [],
        "usrgrps": [
            {
                "usrgrpid": "8"
            },
            {
                "usrgrpid": "7"
            }
        ],
        "roleid": "2",
        "userdirectoryid": "2"
```

```
    },
    "id": 1
}
```

Source

CUserDirectory::test() in *ui/include/classes/api/services/CUserDirectory.php*.

userdirectory.update

Description

`object userdirectory.update(object/array userDirectory)`

This method allows to update existing user directories.

> **Note:**
> This method is only available to *Super admin* user type.

Parameters

`(object/array)` User directory properties to be updated.

The `userdirectoryid` property must be defined for each user directory, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Return values

`(object)` Returns an object containing the IDs of the updated user directories under the `userdirectoryids` property.

Examples

Update bind password for user directory

Set new bind password for a user directory.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.update",
    "params": {
        "userdirectoryid": "3",
        "bind_password": "newldappassword"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userdirectoryids": [
            "3"
        ]
    },
    "id": 1
}
```

Update mappings for user directory

Update provisioning groups mappings and media type mappings for user directory ”2”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.update",
    "params": {
```

```
        "userdirectoryid": "2",
        "provision_media": [
                {
                    "name": "example.com",
                    "mediatypeid": "1",
                    "attribute": "admin@example.com"
                }
            ],
            "provision_groups": [
                {
                    "name": "Zabbix administrators",
                    "roleid": "2",
                    "user_groups": [
                        {
                            "usrgrpid": "7"
                        },
                        {
                            "usrgrpid": "8"
                        },
                        {
                            "usrgrpid": "11"
                        }
                    ]
                }
            ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "userdirectoryids": [
            "2"
        ]
    },
    "id": 1
}
```

Source

CUserDirectory::update() in *ui/include/classes/api/services/CUserDirectory.php*.

User group

This class is designed to work with user groups.

Object references:

- User group

Available methods:

- usergroup.create - creating new user groups
- usergroup.delete - deleting user groups
- usergroup.get - retrieving user groups
- usergroup.update - updating user groups

> User group object

The following objects are directly related to the `usergroup` API.

User group

The user group object has the following properties.

Property	Type	Description
usrgrpid	string	ID of the user group. Property behavior: - *read-only* - *required* for update operations
name	string	Name of the user group. Property behavior: - *required* for create operations
debug_mode	integer	Whether debug mode is enabled or disabled. Possible values: 0 - *(default)* disabled; 1 - enabled.
gui_access	integer	Frontend authentication method of the users in the group. Possible values: 0 - *(default)* use the system default authentication method; 1 - use internal authentication; 2 - use LDAP authentication; 3 - disable access to the frontend.
users_status	integer	Whether the user group is enabled or disabled. For deprovisioned users, the user group cannot be enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
userdirectoryid	string	User directory used for authentication. Property behavior: - *supported* if `gui_access` is set to ”use the system default authentication method” or ”use LDAP authentication”

Permission

The permission object has the following properties.

Property	Type	Description
id	string	ID of the host group or template group to add permission to. Property behavior: - *required*
permission	integer	Access level to the host group or template group. Possible values: 0 - access denied; 2 - read-only access; 3 - read-write access. Property behavior: - *required* for create operations

Tag-based permission

The tag-based permission object has the following properties.

Property	Type	Description
groupid	string	ID of the host group to add permission to. Property behavior: - *required*
tag	string	Tag name.
value	string	Tag value.

usergroup.create

Description

`object usergroup.create(object/array userGroups)`

This method allows to create new user groups.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` User groups to create.

Additionally to the standard user group properties, the method accepts the following parameters.

Parameter	Type	Description
hostgroup_rights	object/array	Host group permissions to assign to the user group.
templategroup_rights	object/array	Template group permissions to assign to the user group.
tag_filters	array	Tag-based permissions to assign to the user group.
users	object/array	Users to add to the user group. The user must have the `userid` property defined.
rights (deprecated)	object/array	This parameter is deprecated, please use `hostgroup_rights` or `templategroup_rights` instead. Permissions to assign to the user group.

Return values

`(object)` Returns an object containing the IDs of the created user groups under the `usrgrpids` property. The order of the returned IDs matches the order of the passed user groups.

Examples

Creating a user group

Create a user group *Operation managers* with denied access to host group "2", and add a user to it.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.create",
    "params": {
        "name": "Operation managers",
        "hostgroup_rights": {
            "id": "2",
            "permission": 0
        },
        "users": [
            {
                "userid": "12"
            }
        ]
```

```
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "20"
        ]
    },
    "id": 1
}
```

See also

- Permission

Source

CUserGroup::create() in *ui/include/classes/api/services/CUserGroup.php*.

usergroup.delete

Description

`object usergroup.delete(array userGroupIds)`

This method allows to delete user groups.

Attention:
Deprovisioned users group (the user group specified for `disabled_usrgrpid` in Authentication) cannot be deleted.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the user groups to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted user groups under the `usrgrpids` property.

Examples

Deleting multiple user groups

Delete two user groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.delete",
    "params": [
        "20",
        "21"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "usrgrpids": [
            "20",
            "21"
        ]
    },
    "id": 1
}
```

Source

CUserGroup::delete() in *ui/include/classes/api/services/CUserGroup.php*.

usergroup.get

Description

`integer/array usergroup.get(object parameters)`

The method allows to retrieve user groups according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
status	integer	Return only user groups with the given status. Refer to the user group page for a list of supported statuses.
userids	string/array	Return only user groups that contain the given users.
usrgrpids	string/array	Return only user groups with the given IDs.
selectTagFilters	query	Return user group tag based permissions in the `tag_filters` property. It has the following properties: `groupid` - `(string)` ID of the host group; `tag` - `(string)` tag name; `value` - `(string)` tag value.
selectUsers	query	Return the users from the user group in the `users` property.
selectHostGroupRights	query	Return user group host group rights in the `hostgroup_rights` property. It has the following properties: `permission` - `(integer)` access level to the host group; `id` - `(string)` ID of the host group. Refer to the user group page for a list of access levels to host groups.
selectTemplateGroupRights	query	Return user group template group rights in the `templategroup_rights` property. It has the following properties: `permission` - `(integer)` access level to the template group; `id` - `(string)` ID of the template group. Refer to the user group page for a list of access levels to template groups.
limitSelects	integer	Limits the number of records returned by subselects.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties. Possible values: `usrgrpid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectRights (deprecated)	query	This parameter is deprecated, please use `selectHostGroupRights` or `selectTemplateGroupRights` instead. Return user group rights in the rights property. It has the following properties: `permission` - `(integer)` access level to the host group; `id` - `(string)` ID of the host group. Refer to the user group page for a list of access levels to host groups.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving enabled user groups

Retrieve all enabled user groups.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.get",
    "params": {
        "output": "extend",
        "status": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "usrgrpid": "7",
            "name": "Zabbix administrators",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1",
            "userdirectoryid": "0"
        },
        {
```

```
            "usrgrpid": "8",
            "name": "Guests",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0"
        },
        {
            "usrgrpid": "11",
            "name": "Enabled debug mode",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "1",
            "userdirectoryid": "0"
        },
        {
            "usrgrpid": "12",
            "name": "No access to the frontend",
            "gui_access": "2",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0"
        },
        {
            "usrgrpid": "14",
            "name": "Read only",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0"
        },
        {
            "usrgrpid": "18",
            "name": "Deny",
            "gui_access": "0",
            "users_status": "0",
            "debug_mode": "0",
            "userdirectoryid": "0"
        }
    ],
    "id": 1
}
```


See also

- User

Source

CUserGroup::get() in *ui/include/classes/api/services/CUserGroup.php*.

usergroup.update

Description

`object usergroup.update(object/array userGroups)`

This method allows to update existing user groups.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

(object/array) User group properties to be updated.

The usrgrpid property must be defined for each user group, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard user group properties, the method accepts the following parameters.

Parameter	Type	Description
hostgroup_rights	object/array	Host group permissions to replace the current permissions assigned to the user group.
templategroup_rights	object/array	Template group permissions to replace the current permissions assigned to the user group.
tag_filters	array	Tag-based permissions to replace the current permissions assigned to the user group.
users	object/array	Users to replace the current users assigned to the user group. The user must have only the userid property defined.
rights (deprecated)	object/array	This parameter is deprecated, please use hostgroup_rights or templategroup_rights instead. Permissions to assign to the user group.

Return values

(object) Returns an object containing the IDs of the updated user groups under the usrgrpids property.

Examples

Enabling a user group and updating permissions

Enable a user group and provide read-write access for it to host groups "2" and "4".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usergroup.update",
    "params": {
        "usrgrpid": "17",
        "users_status": "0",
        "hostgroup_rights": [
            {
                "id": "2",
                "permission": 3
            },
            {
                "id": "4",
                "permission": 3
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "17"
        ]
    },
    "id": 1
}
```

See also

- Permission

Source

CUserGroup::update() in *ui/include/classes/api/services/CUserGroup.php*.

User macro

This class is designed to work with host-level and global user macros.

Object references:

- Global macro
- Host macro

Available methods:

- usermacro.create - creating new host macros
- usermacro.createglobal - creating new global macros
- usermacro.delete - deleting host macros
- usermacro.deleteglobal - deleting global macros
- usermacro.get - retrieving host and global macros
- usermacro.update - updating host macros
- usermacro.updateglobal - updating global macros

> User macro object

The following objects are directly related to the `usermacro` API.

Global macro

The global macro object has the following properties.

Property	Type	Description
globalmacroid	string	ID of the global macro. Property behavior: - *read-only* - *required* for update operations
macro	string	Macro string.
value	string	Value of the macro. Property behavior: - *required* for create operations
type	integer	Type of macro. Property behavior: - *write-only* if `type` is set to "Secret macro" - *required* for create operations Possible values: 0 - *(default)* Text macro; 1 - Secret macro; 2 - Vault secret.
description	string	Description of the macro.

Host macro

The host macro object defines a macro available on a host, host prototype or template. It has the following properties.

Property	Type	Description
hostmacroid	string	ID of the host macro. Property behavior: - *read-only* - *required* for update operations
hostid	string	ID of the host that the macro belongs to. Property behavior: - *constant* - *required* for create operations
macro	string	Macro string. Property behavior: - *required* for create operations
value	string	Value of the macro. Property behavior: - *write-only* if `type` is set to "Secret macro" - *required* for create operations
type	integer	Type of macro. Possible values: 0 - *(default)* Text macro; 1 - Secret macro; 2 - Vault secret.
description	string	Description of the macro.
automatic	integer	Defines whether the macro is controlled by discovery rule. Possible values: 0 - *(default)* Macro is managed by user; 1 - Macro is managed by discovery rule. User is not allowed to create automatic macro. To update automatic macro, it must be converted to manual.

usermacro.create

Description

`object usermacro.create(object/array hostMacros)`

This method allows to create new host macros.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Host macros to create.

The method accepts host macros with the standard host macro properties.

Return values

`(object)` Returns an object containing the IDs of the created host macros under the `hostmacroids` property. The order of the returned IDs matches the order of the passed host macros.

Examples

Creating a host macro

Create a host macro "{$SNMP_COMMUNITY}" with the value "public" on host "10198".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.create",
    "params": {
        "hostid": "10198",
        "macro": "{$SNMP_COMMUNITY}",
        "value": "public"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "11"
        ]
    },
    "id": 1
}
```

Source

CUserMacro::create() in *ui/include/classes/api/services/CUserMacro.php*.

usermacro.createglobal

Description

`object usermacro.createglobal(object/array globalMacros)`

This method allows to create new global macros.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Global macros to create.

The method accepts global macros with the standard global macro properties.

Return values

`(object)` Returns an object containing the IDs of the created global macros under the `globalmacroids` property. The order of the returned IDs matches the order of the passed global macros.

Examples

Creating a global macro

Create a global macro "{$SNMP_COMMUNITY}" with value "public".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.createglobal",
    "params":  {
        "macro": "{$SNMP_COMMUNITY}",
        "value": "public"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "6"
        ]
    },
    "id": 1
}
```

Source

CUserMacro::createGlobal() in *ui/include/classes/api/services/CUserMacro.php*.

usermacro.delete

Description

`object usermacro.delete(array hostMacroIds)`

This method allows to delete host macros.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the host macros to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted host macros under the `hostmacroids` property.

Examples

Deleting multiple host macros

Delete two host macros.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.delete",
    "params": [
        "32",
        "11"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "32",
            "11"
        ]
    },
    "id": 1
}
```

Source

CUserMacro::delete() in *ui/include/classes/api/services/CUserMacro.php*.

usermacro.deleteglobal

Description

`object usermacro.deleteglobal(array globalMacroIds)`

This method allows to delete global macros.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the global macros to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted global macros under the `globalmacroids` property.

Examples

Deleting multiple global macros

Delete two global macros.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.deleteglobal",
    "params": [
        "32",
        "11"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "32",
            "11"
        ]
    },
    "id": 1
}
```

Source

CUserMacro::deleteGlobal() in *ui/include/classes/api/services/CUserMacro.php*.

usermacro.get

Description

`integer/array usermacro.get(object parameters)`

The method allows to retrieve host and global macros according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
globalmacro	flag	Return global macros instead of host macros.
globalmacroids	string/array	Return only global macros with the given IDs.
groupids	string/array	Return only host macros that belong to hosts or templates from the given host groups or template groups.
hostids	string/array	Return only macros that belong to the given hosts or templates.
hostmacroids	string/array	Return only host macros with the given IDs.
inherited	boolean	If set to `true` return only host prototype user macros inherited from a template.
selectHostGroups	query	Return host groups that the host macro belongs to in the hostgroups property. Used only when retrieving host macros.
selectHosts	query	Return hosts that the host macro belongs to in the hosts property. Used only when retrieving host macros.
selectTemplateGroups	query	Return template groups that the template macro belongs to in the templategroups property. Used only when retrieving template macros.
selectTemplates	query	Return templates that the host macro belongs to in the templates property. Used only when retrieving host macros.
sortfield	string/array	Sort the result by the given properties. Possible values: `macro`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	
selectGroups (deprecated)	query	This parameter is deprecated, please use `selectHostGroups` or `selectTemplateGroups` instead. Return host groups and template groups that the host macro belongs to in the `groups` property. Used only when retrieving host macros.

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving host macros for a host

Retrieve all host macros defined for host "10198".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.get",
    "params": {
        "output": "extend",
        "hostids": "10198"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostmacroid": "9",
            "hostid": "10198",
            "macro": "{$INTERFACE}",
            "value": "eth0",
            "description": "",
            "type": "0",
            "automatic": "0"
        },
        {
            "hostmacroid": "11",
            "hostid": "10198",
            "macro": "{$SNMP_COMMUNITY}",
            "value": "public",
            "description": "",
            "type": "0",
            "automatic": "0"
        }
    ],
    "id": 1
}
```

Retrieving global macros

Retrieve all global macros.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.get",
    "params": {
        "output": "extend",
        "globalmacro": true
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "globalmacroid": "6",
            "macro": "{$SNMP_COMMUNITY}",
            "value": "public",
            "description": "",
            "type": "0"
        }
    ],
```

```
    "id": 1
}
```

Source

CUserMacro::get() in *ui/include/classes/api/services/CUserMacro.php*.

usermacro.update

Description

`object usermacro.update(object/array hostMacros)`

This method allows to update existing host macros.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Host macro properties to be updated.

The `hostmacroid` property must be defined for each host macro, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Return values

`(object)` Returns an object containing the IDs of the updated host macros under the `hostmacroids` property.

Examples

Changing the value of a host macro

Change the value of a host macro to "public".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.update",
    "params": {
        "hostmacroid": "1",
        "value": "public"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "1"
        ]
    },
    "id": 1
}
```

Change macro value that was created by discovery rule

Convert discovery rule created "automatic" macro to "manual" and change its value to "new-value".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.update",
    "params": {
        "hostmacroid": "1",
```

```
        "value": "new-value",
        "automatic": "0"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "1"
        ]
    },
    "id": 1
}
```

Source

CUserMacro::update() in *ui/include/classes/api/services/CUserMacro.php*.

usermacro.updateglobal

Description

`object usermacro.updateglobal(object/array globalMacros)`

This method allows to update existing global macros.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Global macro properties to be updated.

The `globalmacroid` property must be defined for each global macro, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Return values

`(object)` Returns an object containing the IDs of the updated global macros under the `globalmacroids` property.

Examples

Changing the value of a global macro

Change the value of a global macro to "public".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "usermacro.updateglobal",
    "params": {
        "globalmacroid": "1",
        "value": "public"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "globalmacroids": [
            "1"
```

```
        ]
    },
    "id": 1
}
```

Source

CUserMacro::updateGlobal() in *ui/include/classes/api/services/CUserMacro.php*.

Value map

This class is designed to work with value maps.

Object references:

- Value map

Available methods:

- valuemap.create - creating new value maps
- valuemap.delete - deleting value maps
- valuemap.get - retrieving value maps
- valuemap.update - updating value maps

> Value map object

The following objects are directly related to the `valuemap` API.

Value map

The value map object has the following properties.

Property	Type	Description
valuemapid	string	ID of the value map. Property behavior: - *read-only* - *required* for update operations
hostid	id	ID of the host or template that the value map belongs to. Property behavior: - *constant* - *required* for create operations
name	string	Name of the value map. Property behavior: - *required* for create operations
mappings	array	Value mappings for current value map. The mapping object is described in detail below. Property behavior: - *required* for create operations
uuid	string	Universal unique identifier, used for linking imported value maps to already existing ones. Auto-generated, if not given. Property behavior: - *supported* if the value map belongs to a template

Value mappings

The value mappings object defines value mappings of the value map. It has the following properties.

Property	Type	Description
type	integer	Mapping match type. Possible values: 0 - *(default)* mapping will be applied if value is equal; 1 - mapping will be applied if value is greater or equal[1]; 2 - mapping will be applied if value is less or equal[1]; 3 - mapping will be applied if value is in range (ranges are inclusive; multiple ranges, separated by comma character, can be defined)[1]; 4 - mapping will be applied if value matches a regular expression[2]; 5 - if no matches are found, mapping will not be applied, and the default value will be used. If `type` is set to ”0”, ”1”, ”2”, ”3”, ”4”, then `value` cannot be empty. If `type` is set to ”5”, then `value` must be empty.
value	string	Original value. Property behavior: - *required* if `type` is set to ”1”, ”2”, ”3”, ”4” - *supported* if `type` is set to ”5”
newvalue	string	Value to which the original value is mapped to. Property behavior: - *required*

[1] supported only for items having value type ”numeric unsigned”, ”numeric float”.
[2] supported only for items having value type ”character”.

valuemap.create

Description

`object valuemap.create(object/array valuemaps)`

This method allows to create new value maps.

Note:
This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Value maps to create.

The method accepts value maps with the standard value map properties.

Return values

`(object)` Returns an object containing the IDs of the created value maps the `valuemapids` property. The order of the returned IDs matches the order of the passed value maps.

Examples

Creating a value map

Create one value map with two mappings.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.create",
    "params": {
        "hostid": "50009",
```

```
        "name": "Service state",
        "mappings": [
            {
                "type": "1",
                "value": "1",
                "newvalue": "Up"
            },
            {
                "type": "5",
                "newvalue": "Down"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "1"
        ]
    },
    "id": 1
}
```

Source

CValueMap::create() in *ui/include/classes/api/services/CValueMap.php*.

valuemap.delete

Description

`object valuemap.delete(array valuemapids)`

This method allows to delete value maps.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the value maps to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted value maps under the `valuemapids` property.

Examples

Deleting multiple value maps

Delete two value maps.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.delete",
    "params": [
        "1",
        "2"
    ],
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "1",
            "2"
        ]
    },
    "id": 1
}
```

Source

CValueMap::delete() in *ui/include/classes/api/services/CValueMap.php*.

valuemap.get

Description

`integer/array valuemap.get(object parameters)`

The method allows to retrieve value maps according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
valuemapids	string/array	Return only value maps with the given IDs.
selectMappings	query	Return the value mappings for current value map in the mappings property. Supports `count`.
sortfield	string/array	Sort the result by the given properties. Possible values: `valuemapid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving value maps

Retrieve all configured value maps.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.get",
    "params": {
        "output": "extend"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "valuemapid": "4",
            "name": "APC Battery Replacement Status"
        },
        {
            "valuemapid": "5",
            "name": "APC Battery Status"
        },
        {
            "valuemapid": "7",
            "name": "Dell Open Manage System Status"
        }
    ],
    "id": 1
}
```

Retrieve one value map with its mappings.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.get",
    "params": {
        "output": "extend",
        "selectMappings": "extend",
        "valuemapids": ["4"]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "valuemapid": "4",
            "name": "APC Battery Replacement Status",
            "mappings": [
                {
                    "type": "0",
                    "value": "1",
```

```
                    "newvalue": "unknown"
                },
                {
                    "type": "0",
                    "value": "2",
                    "newvalue": "notInstalled"
                },
                {
                    "type": "0",
                    "value": "3",
                    "newvalue": "ok"
                },
                {
                    "type": "0",
                    "value": "4",
                    "newvalue": "failed"
                },
                {
                    "type": "0",
                    "value": "5",
                    "newvalue": "highTemperature"
                },
                {
                    "type": "0",
                    "value": "6",
                    "newvalue": "replaceImmediately"
                },
                {
                    "type": "0",
                    "value": "7",
                    "newvalue": "lowCapacity"
                }
            ]
        }
    ],
    "id": 1
}
```


Source

CValueMap::get() in *ui/include/classes/api/services/CValueMap.php*.

valuemap.update

Description

`object valuemap.update(object/array valuemaps)`

This method allows to update existing value maps.

> **Note:**
> This method is only available to *Super admin* user type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Value map properties to be updated.

The `valuemapid` property must be defined for each value map, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Return values

`(object)` Returns an object containing the IDs of the updated value maps under the `valuemapids` property.

Examples

Changing value map name

Change value map name to ”Device status”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.update",
    "params": {
        "valuemapid": "2",
        "name": "Device status"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "2"
        ]
    },
    "id": 1
}
```

Changing mappings for one value map.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.update",
    "params": {
        "valuemapid": "2",
        "mappings": [
            {
                "type": "0",
                "value": "0",
                "newvalue": "Online"
            },
            {
                "type": "0",
                "value": "1",
                "newvalue": "Offline"
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "valuemapids": [
            "2"
        ]
    },
    "id": 1
}
```

Source

CValueMap::update() in *ui/include/classes/api/services/CValueMap.php*.

Web scenario

This class is designed to work with web scenarios.

Object references:

- Web scenario
- Scenario step

Available methods:

- httptest.create - creating new web scenarios
- httptest.delete - deleting web scenarios
- httptest.get - retrieving web scenarios
- httptest.update - updating web scenarios

> Web scenario object

The following objects are directly related to the `webcheck` API.

Web scenario

The web scenario object has the following properties.

Property	Type	Description
httptestid	string	ID of the web scenario. Property behavior: - *read-only* - *required* for update operations
hostid	string	ID of the host that the web scenario belongs to. Property behavior: - *constant* - *required* for create operations
name	string	Name of the web scenario. Property behavior: - *required* for create operations
agent	string	User agent string that will be used by the web scenario. Default: Zabbix
authentication	integer	Authentication method that will be used by the web scenario. Possible values: 0 - *(default)* none; 1 - basic HTTP authentication; 2 - NTLM authentication.
delay	string	Execution interval of the web scenario. Accepts seconds, time unit with suffix, or a user macro. Default: 1m.
headers	array	HTTP headers that will be sent when performing a request.
http_password	string	Password used for basic HTTP or NTLM authentication.
http_proxy	string	Proxy that will be used by the web scenario given as *http://[username[:password]@]proxy.example.com[:port]*.
http_user	string	User name used for basic HTTP or NTLM authentication.
retries	integer	Number of times a web scenario will try to execute each step before failing. Default: 1.

Property	Type	Description
ssl_cert_file	string	Name of the SSL certificate file used for client authentication (must be in PEM format).
ssl_key_file	string	Name of the SSL private key file used for client authentication (must be in PEM format).
ssl_key_password	string	SSL private key password.
status	integer	Whether the web scenario is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
templateid	string	ID of the parent template web scenario. Property behavior: - *read-only*
variables	array	Web scenario variables.
verify_host	integer	Whether to validate that the host name for the connection matches the one in the host's certificate. Possible values: 0 - *(default)* skip host verification; 1 - verify host.
verify_peer	integer	Whether to validate that the host's certificate is authentic. Possible values: 0 - *(default)* skip peer verification; 1 - verify peer.
uuid	string	Global unique identifier, used for linking imported web scenarios to already existing ones. Auto-generated, if not given. Property behavior: - *supported* if the web scenario belongs to a template

Web scenario tag

The web scenario tag object has the following properties.

Property	Type	Description
tag	string	Web scenario tag name. Property behavior: - *required*
value	string	Web scenario tag value.

Scenario step

The scenario step object defines a specific web scenario check. It has the following properties.

Property	Type	Description
name	string	Name of the scenario step. Property behavior: - *required*
no	integer	Sequence number of the step in a web scenario. Property behavior: - *required*
url	string	URL to be checked. Property behavior: - *required*

Property	Type	Description
follow_redirects	integer	Whether to follow HTTP redirects. Possible values: 0 - don't follow redirects; 1 - *(default)* follow redirects.
headers	array	HTTP headers that will be sent when performing a request. Scenario step headers will overwrite headers specified for the web scenario.
posts	string/array	HTTP POST variables as a string (raw post data) or as an array of HTTP fields (form field data).
required	string	Text that must be present in the response.
retrieve_mode	integer	Part of the HTTP response that the scenario step must retrieve. Possible values: 0 - *(default)* only body; 1 - only headers; 2 - headers and body.
status_codes	string	Ranges of required HTTP status codes, separated by commas.
timeout	string	Request timeout in seconds. Accepts seconds, time unit with suffix, or a user macro. Default: 15s. Maximum: 1h. Minimum: 1s.
variables	array	Scenario step variables.
query_fields	array	Query fields - array of HTTP fields that will be added to URL when performing a request.

HTTP field

The HTTP field object defines the name and value that is used to specify the web scenario variables, HTTP headers, and POST fields or query fields. It has the following properties.

Property	Type	Description
name	string	Name of header/variable/POST or GET field. Property behavior: - *required*
value	string	Value of header/variable/POST or GET field. Property behavior: - *required*

httptest.create

Description

`object httptest.create(object/array webScenarios)`

This method allows to create new web scenarios.

Note:
Creating a web scenario will automatically create a set of web monitoring items.

Note:
This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Web scenarios to create.

Additionally to the standard web scenario properties, the method accepts the following parameters.

Parameter	Type	Description
steps	array	Web scenario steps. Parameter behavior: - *required*
tags	array	Web scenario tags.

Return values

(object) Returns an object containing the IDs of the created web scenarios under the `httptestids` property. The order of the returned IDs matches the order of the passed web scenarios.

Examples

Creating a web scenario

Create a web scenario to monitor the company home page. The scenario will have two steps, to check the home page and the "About" page and make sure they return the HTTP status code 200.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.create",
    "params": {
        "name": "Homepage check",
        "hostid": "10085",
        "steps": [
            {
                "name": "Homepage",
                "url": "http://example.com",
                "status_codes": "200",
                "no": 1
            },
            {
                "name": "Homepage / About",
                "url": "http://example.com/about",
                "status_codes": "200",
                "no": 2
            }
        ]
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "5"
        ]
    },
    "id": 1
}
```

See also

- Scenario step

Source

CHttpTest::create() in *ui/include/classes/api/services/CHttpTest.php*.

httptest.delete

Description

`object httptest.delete(array webScenarioIds)`

This method allows to delete web scenarios.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(array)` IDs of the web scenarios to delete.

Return values

`(object)` Returns an object containing the IDs of the deleted web scenarios under the `httptestids` property.

Examples

Deleting multiple web scenarios

Delete two web scenarios.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.delete",
    "params": [
        "2",
        "3"
    ],
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "2",
            "3"
        ]
    },
    "id": 1
}
```

Source

CHttpTest::delete() in *ui/include/classes/api/services/CHttpTest.php*.

httptest.get

Description

`integer/array httptest.get(object parameters)`

The method allows to retrieve web scenarios according to the given parameters.

> **Note:**
> This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object)` Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
groupids	string/array	Return only web scenarios that belong to the given host groups.
hostids	string/array	Return only web scenarios that belong to the given hosts.
httptestids	string/array	Return only web scenarios with the given IDs.
inherited	boolean	If set to `true` return only web scenarios inherited from a template.
monitored	boolean	If set to `true` return only enabled web scenarios that belong to monitored hosts.
templated	boolean	If set to `true` return only web scenarios that belong to templates.
templateids	string/array	Return only web scenarios that belong to the given templates.
expandName	flag	Expand macros in the name of the web scenario.
expandStepName	flag	Expand macros in the names of scenario steps.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	array of objects	Return only web scenarios with given tags. Exact match by tag and case-sensitive or case-insensitive search by tag value depending on operator value. Format: `[{"tag": "<tag>", "value": "<value>", "operator": "<operator>"}, ...]`. An empty array returns all web scenarios. Possible operator types: 0 - (default) Like; 1 - Equal; 2 - Not like; 3 - Not equal 4 - Exists; 5 - Not exists.
selectHosts	query	Return the hosts that the web scenario belongs to as an array in the `hosts` property.
selectSteps	query	Return web scenario steps in the `steps` property. Supports `count`.
selectTags	query	Return web scenario tags in the `tags` property.
sortfield	string/array	Sort the result by the given properties. Possible values: `httptestid`, `name`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

`(integer/array)` Returns either:

- an array of objects;
- the count of retrieved objects, if the `countOutput` parameter has been used.

Examples

Retrieving a web scenario

Retrieve all data about web scenario "4".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.get",
    "params": {
        "output": "extend",
        "selectSteps": "extend",
        "httptestids": "9"
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "httptestid": "9",
            "name": "Homepage check",
            "delay": "1m",
            "status": "0",
            "variables": [],
            "agent": "Zabbix",
            "authentication": "0",
            "http_user": "",
            "http_password": "",
            "hostid": "10084",
            "templateid": "0",
            "http_proxy": "",
            "retries": "1",
            "ssl_cert_file": "",
            "ssl_key_file": "",
            "ssl_key_password": "",
            "verify_peer": "0",
            "verify_host": "0",
            "headers": [],
            "steps": [
                {
                    "httpstepid": "36",
                    "httptestid": "9",
                    "name": "Homepage",
                    "no": "1",
                    "url": "http://example.com",
                    "timeout": "15s",
                    "posts": "",
                    "required": "",
                    "status_codes": "200",
                    "variables": [
                        {
                            "name":"{var}",
                            "value":"12"
                        }
                    ],
                    "follow_redirects": "1",
                    "retrieve_mode": "0",
                    "headers": [],
                    "query_fields": []
                },
                {
                    "httpstepid": "37",
                    "httptestid": "9",
```

```
                    "name": "Homepage / About",
                    "no": "2",
                    "url": "http://example.com/about",
                    "timeout": "15s",
                    "posts": "",
                    "required": "",
                    "status_codes": "200",
                    "variables": [],
                    "follow_redirects": "1",
                    "retrieve_mode": "0",
                    "headers": [],
                    "query_fields": []
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Host
- Scenario step

Source

CHttpTest::get() in *ui/include/classes/api/services/CHttpTest.php*.

httptest.update

Description

`object httptest.update(object/array webScenarios)`

This method allows to update existing web scenarios.

> **Note:**
> This method is only available to *Admin* and *Super admin* user types. Permissions to call the method can be revoked in user role settings. See User roles for more information.

Parameters

`(object/array)` Web scenario properties to be updated.

The `httptestid` property must be defined for each web scenario, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the standard web scenario properties, the method accepts the following parameters.

Parameter	Type	Description
steps	array	Scenario steps to replace existing steps.
tags	array	Web scenario tags.

Return values

`(object)` Returns an object containing the IDs of the updated web scenarios under the `httptestid` property.

Examples

Enabling a web scenario

Enable a web scenario, that is, set its status to ”0”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "httptest.update",
    "params": {
        "httptestid": "5",
        "status": 0
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "httptestids": [
            "5"
        ]
    },
    "id": 1
}
```

See also

- Scenario step

Source

CHttpTest::update() in *ui/include/classes/api/services/CHttpTest.php*.

Appendix 1. Reference commentary

Notation Data types

The Zabbix API supports the following data types as input:

Type	Description
boolean	A boolean value, accepts either `true` or `false`.
flag	The value is considered to be `true` if it is passed and not equal to `null`; otherwise, it is considered to be `false`.
integer	A whole number.
float	A floating point number.
string	A text string.
text	A longer text string.
timestamp	A Unix timestamp.
array	An ordered sequence of values, that is, a plain array.
object	An associative array.
query	A value which defines, what data should be returned. Can be defined as an array of property names to return only specific properties, or as one of the predefined values: `extend` - returns all object properties; `count` - returns the number of retrieved records, supported only by certain subselects.

Attention:
Zabbix API always returns values as strings or arrays only.

Property behavior

Some of the object properties are marked with short labels to describe their behavior. The following labels are used:

- *read-only* - the value of the property is set automatically and cannot be defined or changed by the user, even in some specific conditions (e.g., *read-only* for inherited objects or discovered objects);
- *write-only* - the value of the property can be set, but cannot be accessed after;
- *constant* - the value of the property can be set when creating an object, but cannot be changed after;
- *supported* - the value of the property is not required to be set, but is allowed to be set in some specific conditions (e.g., *supported* if `type` is set to ”Simple check”, ”External check”, ”SSH agent”, ”TELNET agent”, or ”HTTP agent”);
- *required* - the value of the property is required to be set for all operations (except get operations) or in some specific conditions (e.g., *required* for create operations; *required* if `operationtype` is set to ”global script” and `opcommand_hst` is not set).

Note:
For update operations a property is considered as ”set” when setting it during the update operation.

Properties that are not marked with labels are optional.

Parameter behavior

Some of the operation parameters are marked with short labels to describe their behavior for the operation. The following labels are used:

- *read-only* - the value of the parameter is set automatically and cannot be defined or changed by the user, even in some specific conditions (e.g., *read-only* for inherited objects or discovered objects);
- *write-only* - the value of the parameter can be set, but cannot be accessed after;
- *supported* - the value of the parameter is not required to be set, but is allowed to be set in some specific conditions (e.g., *supported* if `status` of Proxy object is set to ”passive proxy”);
- *required* - the value of the parameter is required to be set.

Parameters that are not marked with labels are optional.

Reserved ID value ”0” Reserved ID value ”0” can be used to filter elements and to remove referenced objects. For example, to remove a referenced proxy from a host, proxy_hostid should be set to 0 (”proxy_hostid”: ”0”) or to filter hosts monitored by server option proxyids should be set to 0 (”proxyids”: ”0”).

Common ”get” method parameters The following parameters are supported by all `get` methods:

Parameter	Type	Description
countOutput	boolean	Return the number of records in the result instead of the actual data.
editable	boolean	If set to `true`, return only objects that the user has write permissions to. Default: `false`.
excludeSearch	boolean	Return results that do not match the criteria given in the `search` parameter.
filter	object	Return only those results that exactly match the given filter. Accepts an object, where the keys are property names, and the values are either a single value or an array of values to match against. Does not support properties of `text` data type.
limit	integer	Limit the number of records returned.
output	query	Object properties to be returned. Default: `extend`.
preservekeys	boolean	Use IDs as keys in the resulting array.
search	object	Return results that match the given pattern (case-insensitive). Accepts an object, where the keys are property names, and the values are strings to search for. If no additional options are given, this will perform a `LIKE "%…%"` search. Supports only properties of `string` and `text` data type.
searchByAny	boolean	If set to `true`, return results that match any of the criteria given in the `filter` or `search` parameter instead of all of them. Default: `false`.

Parameter	Type	Description
searchWildcardsEnabled	boolean	If set to `true`, enables the use of "*" as a wildcard character in the `search` parameter. Default: `false`.
sortfield	string/array	Sort the result by the given properties. Refer to a specific API get method description for a list of properties that can be used for sorting. Macros are not expanded before sorting. If no value is specified, data will be returned unsorted.
sortorder	string/array	Order of sorting. If an array is passed, each value will be matched to the corresponding property given in the `sortfield` parameter. Possible values: `ASC` - *(default)* ascending; `DESC` - descending.
startSearch	boolean	The `search` parameter will compare the beginning of fields, that is, perform a `LIKE "…%"` search instead. Ignored if `searchWildcardsEnabled` is set to `true`.

Examples User permission check

Does the user have permission to write to hosts whose names begin with "MySQL" or "Linux" ?

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "countOutput": true,
        "search": {
            "host": ["MySQL", "Linux"]
        },
        "editable": true,
        "startSearch": true,
        "searchByAny": true
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "0",
    "id": 1
}
```

Note:
Zero result means no hosts with read/write permissions.

Mismatch counting

Count the number of hosts whose names do not contain the substring "ubuntu"

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "countOutput": true,
        "search": {
            "host": "ubuntu"
        },
```

```
        "excludeSearch": true
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "44",
    "id": 1
}
```

Searching for hosts using wildcards

Find hosts whose name contains word ”server” and have interface ports ”10050” or ”10071”. Sort the result by host name in descending order and limit it to 5 hosts.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "host"],
        "selectInterfaces": ["port"],
        "filter": {
            "port": ["10050", "10071"]
        },
        "search": {
            "host": "*server*"
        },
        "searchWildcardsEnabled": true,
        "searchByAny": true,
        "sortfield": "host",
        "sortorder": "DESC",
        "limit": 5
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "50003",
            "host": "WebServer-Tomcat02",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        {
            "hostid": "50005",
            "host": "WebServer-Tomcat01",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        {
            "hostid": "50004",
```

```
            "host": "WebServer-Nginx",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        {
            "hostid": "99032",
            "host": "MySQL server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        },
        {
            "hostid": "99061",
            "host": "Linux server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        }
    ],
    "id": 1
}
```

Searching for hosts using wildcards with ”preservekeys”

If you add the parameter ”preservekeys” to the previous request, the result is returned as an associative array, where the keys are the id of the objects.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid", "host"],
        "selectInterfaces": ["port"],
        "filter": {
            "port": ["10050", "10071"]
        },
        "search": {
            "host": "*server*"
        },
        "searchWildcardsEnabled": true,
        "searchByAny": true,
        "sortfield": "host",
        "sortorder": "DESC",
        "limit": 5,
        "preservekeys": true
    },
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "50003": {
            "hostid": "50003",
```

```
            "host": "WebServer-Tomcat02",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "50005": {
            "hostid": "50005",
            "host": "WebServer-Tomcat01",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "50004": {
            "hostid": "50004",
            "host": "WebServer-Nginx",
            "interfaces": [
                {
                    "port": "10071"
                }
            ]
        },
        "99032": {
            "hostid": "99032",
            "host": "MySQL server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        },
        "99061": {
            "hostid": "99061",
            "host": "Linux server 01",
            "interfaces": [
                {
                    "port": "10050"
                }
            ]
        }
    },
    "id": 1
}
```


Appendix 2. Changes from 6.2 to 6.4

Backward incompatible changes authentication

ZBXNEXT-276 Removed properties `saml_idp_entityid`, `saml_sso_url`, `saml_slo_url`, `saml_username_attribute`, `saml_sp_entityid`, `saml_nameid_format`, `saml_sign_messages`, `saml_sign_assertions`, `saml_sign_authn_requests`, `saml_sign_logout_requests`, `saml_sign_logout_responses`, `saml_encrypt_nameid`, `saml_encrypt_assertions`. These properties are now part of User directory API.
ZBXNEXT-276 Renamed property `ldap_configured` to `ldap_auth_enabled`.
ZBXNEXT-276 Added new properties `ldap_jit_status`, `saml_jit_status`, `jit_provision_interval`, `disabled_usrgrpid`.

drule

ZBXNEXT-7921 Removed property `nextcheck`.

httptest

ZBXNEXT-7921 Removed property `nextcheck`.

item

ZBXNEXT-6980 `item.create`, `item.update`: Added strict validation of the method parameters.

itemprototype

ZBXNEXT-6980 `itemprototype.create`, `itemprototype.update`: Added strict validation of the method parameters.

mediatype

ZBXNEXT-6796 Removed property `exec_params`.
ZBXNEXT-6796 Added new property `parameters` (contains a list of parameter objects for "script" media types).
ZBXNEXT-7912 Added new property `provider`.

proxy

ZBXNEXT-2557 `proxy.get`: Added strict validation of the method parameters.

user

ZBXNEXT-8085 Removed support for deprecated property `alias`.
ZBXNEXT-276 Property `roleid` is no longer mandatory.
ZBXNEXT-276 Added new properties `userdirectoryid`, `ts_provisioned`.
ZBXNEXT-276 Added new method `user.provision`.
ZBXNEXT-276 `user.create`: Parameter `usrgrps` is no longer mandatory.
ZBXNEXT-8085 `user.create`, `user.update`: Removed support for deprecated parameter `user_medias`.
ZBXNEXT-8085 `user.login`: Removed support for deprecated parameter `user`.
ZBXNEXT-3100 `user.update`: Added new parameter `current_passwd`.

userdirectory

ZBXNEXT-276 Added new properties `idp_type`, `provision_status`, `user_username`, `user_lastname`, `user_ref_attr`, `group_membership`, `group_basedn`, `group_name`, `group_member`, `group_filter`, `idp_entityid`, `sp_entityid`, `sso_url`, `slo_url`, `username_attribute`, `nameid_format`, `scim_status`, `encrypt_nameid`, `encrypt_assertions`, `sign_messages`, `sign_assertions`, `sign_authn_requests`, `sign_logout_requests`, `sign_logout_responses`, `provision_media`, `provision_groups`.
ZBXNEXT-276 `userdirectory.get`: Removed support for searching by `base_dn`, `bind_dn`, `host`, `search_attribute`, `search_filter`.
ZBXNEXT-276 `userdirectory.get`: Removed support for filtering by `host`, `name`.
ZBXNEXT-276 `userdirectory.get`: Added support for filtering by `idp_type`, `provision_status`.
ZBXNEXT-276 `userdirectory.get`: Added new parameters `selectProvisionMedia`, `selectProvisionGroups`.

Other changes and bug fixes API

ZBXNEXT-8051 Authorization method changed from `auth` parameter to `Authorization` header.
ZBXNEXT-8051 Deprecated parameter `auth`.

action

ZBXNEXT-7964 Added new property `pause_symptoms`.

auditlog

ZBXNEXT-8105 Added new audit log entry resource type (51 - Connector).

connector

ZBXNEXT-8105 Added new Connector API with methods `connector.create`, `connector.update`, `connector.get`, `connector.delete`.

dashboard

ZBXNEXT-4714 Added new dashboard widget field types (11 - User, 12 - Action, 13 - Media type).

Added new possible value combinations of the dashboard widget field object properties for different widget types:
ZBXNEXT-4714 `actionlog`: Added `"type": 11, "name": "userids", "value": <user ID>`.
ZBXNEXT-4714 `actionlog`: Added `"type": 12, "name": "actionids", "value": <action ID>`.
ZBXNEXT-4714 `actionlog`: Added `"type": 13, "name": "mediatypeids", "value": <media type ID>`.
ZBXNEXT-4714 `actionlog`: Added `"type": 0, "name": "statuses", "value": <0 - In progress; 1 - Sent/Executed; 2 - Failed>`.
ZBXNEXT-4714 `actionlog`: Added `"type": 1, "name": "message", "value": "<any string value>"`.
ZBXNEXT-7661 `item`: Added `"type": 1, "name": "thresholds.color.<N>", "value": "<hexadecimal color code>"`.
ZBXNEXT-7661 `item`: Added `"type": 1, "name": "thresholds.threshold.<N>", "value": "<any string value>"`.
ZBXNEXT-5491 `svggraph`: Added `"type": 1, "name": "ds.data_set_label.<N>", "value": "<any string value>"`.
ZBXNEXT-8165 `tophosts`: Added `"type": 0, "name": "columns.decimal_places.<N>", "value": <valid values range: 0-10>`.

discoveryrule

ZBXNEXT-6406 Property `interfaceid` is no longer mandatory for LLD rule types "Simple check" (3), "External check" (10), "SSH agent" (13), and "TELNET agent" (14).
ZBXNEXT-4428, ZBXNEXT-8246 Added new LLD rule preprocessing types "SNMP walk value" (28), "SNMP walk to JSON" (29).

event

ZBXNEXT-7964 Added new property `cause_eventid`.
ZBXNEXT-7964 `event.acknowledge`: Added new event update actions "change event rank to cause" (128), "change event rank to symptom" (256).
ZBXNEXT-7964 `event.acknowledge`: Added new parameter `cause_eventid`.
ZBXNEXT-7964 `event.get`: Added new parameter `symptom`.

graph

ZBX-7706 `graph.get`: Graph availability does not depend on permissions to the items specified in the graph properties `ymin_itemid` and `ymax_itemid`.

graphprototype

ZBX-7706 `graphprototype.get`: Graph prototype availability does not depend on permissions to the items specified in the graph prototype properties `ymin_itemid` and `ymax_itemid`.

hostinterface

ZBXNEXT-4428 Added new property `max_repetitions`.

hostprototype

ZBXNEXT-4428 Added new property `max_repetitions`.

item

ZBXNEXT-6406 Property `interfaceid` is no longer mandatory for item types "Simple check" (3), "External check" (10), "SSH agent" (13), and "TELNET agent" (14).
ZBXNEXT-4428, ZBXNEXT-8246 Added new item preprocessing types "SNMP walk value" (28), "SNMP walk to JSON" (29).

itemprototype

ZBXNEXT-6406 Property `interfaceid` is no longer mandatory for item prototype types "Simple check" (3), "External check" (10), "SSH agent" (13), and "TELNET agent" (14).
ZBXNEXT-4428, ZBXNEXT-8246 Added new item prototype preprocessing types "SNMP walk value" (28), "SNMP walk to JSON" (29).

module

ZBXNEXT-7469 Added new Module API with methods `module.create`, `module.delete`, `module.get`, `module.update`.

problem

ZBXNEXT-7964 Added new property `cause_eventid`.
ZBXNEXT-7964 `problem.get`: Added new parameter `symptom`.

proxy

ZBXNEXT-2557 Added new properties `version`, `compatibility`.

script

ZBXNEXT-3496 Added new properties `url`, `new_window`.
ZBXNEXT-3496 Added new Script type "URL" (6).
ZBXNEXT-3496 Added new method `script.getscriptsbyevents`.
ZBXNEXT-3496 `script.create`: Property `scope` is now mandatory.
ZBXNEXT-3496 `script.create`: Property `command` is now mandatory for Script types "Script" (0), "IPMI" (1), "SSH" (2), "TELNET" (3), and "Webhook" (5).
ZBXNEXT-3496 `script.getscriptsbyhosts`: Method now automatically resolves macros in properties `confirmation`, `url`.

template

ZBXNEXT-1111 Added new properties `vendor_name`, `vendor_version`.

trigger

ZBXNEXT-7972 Added new property `url_name`.

triggerprototype

ZBXNEXT-7972 Added new property `url_name`.

Zabbix API changes in 6.4

6.4.21 API

ZBX-25732 Added support for `Authorization` HTTP request header and `OPTIONS` HTTP request method in cross-origin requests (CORS).

6.4.19 mediatype

ZBX-25385 `mediatype.get`: Parameter `selectMessageTemplates` is now supported only for *Super admin* type users.
ZBX-25385 `mediatype.get`: *Admin* type users may now retrieve only the following Media type object properties: `mediatypeid`, `name`, `type`, `status`, `maxattempts`.
ZBX-25385 `mediatype.get`: When requesting user-related information of media types, *Admin* type users may now retrieve only data about their own user.

6.4.5 userdirectory

ZBX-22800 `userdirectory.get`: Restored sorting by `name`.

6.4.1 script

ZBX-19466 Changed validation of Script object to be unique by combination of 2 properties: `name` and `menu_path`.

user

ZBXNEXT-8012 `user.checkAuthentication`: Added new parameter `token`.

20 Extensions

Overview Although Zabbix offers a multiplicity of features, there is always room for additional functionality. Extensions are a convenient way of modifying and enhancing the monitoring capabilities of Zabbix without changing its source code.

You can extend Zabbix functionality either by using built-in extension options (trapper items, user parameters, etc.) or by using or creating custom extensions (loadable modules, plugins, etc.).

This section provides an overview with references to all the options for extending Zabbix.

Data collection with custom commands Trapper items

Trapper items are items that accept incoming data instead of querying for it. Trapper items are useful for sending specific data to Zabbix server, for example, periodic availability and performance data in the case of long running user scripts. You can do that by using the Zabbix sender command-line utility or by implementing a JSON-based communication protocol (similar to that used in Zabbix sender) yourself.

External checks

An external check is an item for executing checks by running an executable, for example, a shell script or a binary.

External checks are executed by Zabbix server or proxy (when host is monitored by proxy), and do not require an agent running on the host being monitored.

User parameters

A user parameter is a user-defined command (associated with a user-defined key) that, when executed, can retrieve the data you need from the host where Zabbix agent is running. User parameters are useful for configuring agent or agent 2 items that are not predefined in Zabbix.

`system.run[]` Zabbix agent items

`system.run[]` Zabbix agent item is an item for a user-defined command (associated with a predefined key `system.run[]`, for example, `system.run[myscript.sh]`) that can be executed on the host where Zabbix agent is running.

Note: `system.run[]` items are disabled by default and, if used, must be enabled (allowed) and defined in the Zabbix agent or agent 2 configuration file (`AllowKey` configuration parameter).

> **Attention:**
> User-defined commands in items such as external checks, user parameters and `system.run[]` Zabbix agent items are executed from the OS user that is used to run Zabbix components. To execute these commands, this user must have the necessary permissions.

HTTP agent items

HTTP agent item is an item for executing data requests over HTTP/HTTPS. HTTP agent items are useful for sending requests to HTTP endpoints to retrieve data from services such as *Elasticsearch* and *OpenWeatherMap*, for checking the status of Zabbix API or the status of Apache or Nginx web server, etc.

Script items

A script item is an item for executing user-defined JavaScript code that retrieves data over HTTP/HTTPS. Script items are useful when the functionality provided by HTTP agent items is not enough. For example, in demanding data collection scenarios that require multiple steps or complex logic, a script item can be configured to make an HTTP call, then process the data received, and then pass the transformed value to a second HTTP call.

Note:
HTTP agent items and script items are supported by Zabbix server and proxy, and do not require an agent running on the host being monitored.

Advanced extensions Loadable modules

Loadable modules, written in C, are a versatile and performance-minded option for extending the functionality of Zabbix components (server, proxy, agent) on UNIX platforms. A loadable module is basically a shared library used by Zabbix daemon and loaded on startup. The library should contain certain functions, so that a Zabbix process may detect that the file is indeed a module it can load and work with.

Loadable modules have a number of benefits, including the ability to add new metrics or implement any other logic (for example, Zabbix history data export), great performance, and the option to develop, use and share the functionality they provide. It contributes to trouble-free maintenance and helps to deliver new functionality easier and independently of the Zabbix core code base.

Loadable modules are especially useful in a complex monitoring setup. When monitoring embedded systems, having a large number of monitored parameters or heavy scripts with complex logic or long startup time, extensions such as user parameters, `system.run[]` Zabbix agent items, and external checks will have an impact on performance. Loadable modules offer a way of extending Zabbix functionality without sacrificing performance.

Plugins

Plugins provide an alternative to loadable modules (written in C). However, plugins are a way to extend Zabbix agent 2 only.

A plugin is a *Go* package that defines the structure and implements one or several plugin interfaces (*Exporter*, *Collector*, *Configurator*, *Runner*, *Watcher*). Two types of Zabbix agent 2 plugins are supported:

- Built-in plugins (supported since Zabbix 4.4.0)
- Loadable plugins (supported since Zabbix 6.0.0)

See the list of built-in plugins.

For instructions and tutorials on writing your own plugins, see Developer center.

Alert customization Webhooks

A webhook is a Zabbix media type that provides an option to extend Zabbix alerting capabilities to external software such as helpdesk systems, chats, or messengers. Similarly to script items, webhooks are useful for making HTTP calls using custom JavaScript code, for example, to push notifications to different platforms such as Microsoft Teams, Discord, and Jira. It is also possible to return some data (for example, about created helpdesk tickets) that is then displayed in Zabbix.

Existing webhooks are available in the Zabbix Git repository. For custom webhook development, see Webhook development guidelines.

Alert scripts

An alert script is a Zabbix media type that provides an option to create an alternative way (script) to handle Zabbix alerts. Alert scripts are useful if you are not satisfied with the existing media types for sending alerts in Zabbix.

Frontend customization Custom themes

It is possible to change Zabbix frontend visual appearance by using custom themes. See the instructions on creating and applying your own themes.

Frontend modules

Frontend modules provide an option to extend Zabbix frontend functionality by adding third-party modules or by developing your own. With frontend modules you can add new menu items, their respective views, actions, etc.

Global scripts A global script is a user-defined set of commands that can be executed on a monitoring target (by shell (/bin/sh) interpreter), depending on the configured scope and user permissions. Global scripts can be configured for the following actions:

- Action operation
- Manual host action
- Manual event action

Global scripts are useful in many cases. For example, if configured for action operations or manual host actions, you can use global scripts to automatically or manually execute remote commands such as restarting an application (web server, middleware, CRM,

etc.) or freeing disk space (removing older files, cleaning `/tmp`, etc). Or, another example, if configured for manual event actions, you can use global scripts to manage problem tickets in external systems.

Global scripts can be executed by Zabbix server, proxy or agent.

> **Attention:**
> User-defined commands are executed from the OS user that is used to run Zabbix components. To execute these commands, this user must have the necessary permissions.

Zabbix API Zabbix API is an HTTP-based API that is part of Zabbix frontend. With Zabbix API, you can do any of the following operations:

- Programmatically retrieve and modify the configuration of Zabbix.
- Import and export Zabbix configuration.
- Access Zabbix historical and trend data.
- Configure applications to work with Zabbix.
- Integrate Zabbix with third-party software.
- Automate routine tasks.

Zabbix API consists of a multiplicity of methods that are nominally grouped into separate APIs. Each method performs a specific task. For the available methods, as well as an overview of the functions provided by Zabbix API, see Zabbix API Method reference.

1 Loadable modules

Overview

Loadable modules offer a performance-minded option for extending Zabbix functionality.

You can extend Zabbix functionality in many ways, for example, with user parameters, external checks, and `system.run[]` Zabbix agent items. These work very well, but have one major drawback, namely fork(). Zabbix has to fork a new process every time it handles a user metric, which is not good for performance. It is not a big deal normally, however it could be a serious issue when monitoring embedded systems, having a large number of monitored parameters or heavy scripts with complex logic or long startup time.

Support of loadable modules offers ways for extending Zabbix agent, server and proxy without sacrificing performance.

A loadable module is basically a shared library used by Zabbix daemon and loaded on startup. The library should contain certain functions, so that a Zabbix process may detect that the file is indeed a module it can load and work with.

Loadable modules have a number of benefits. Great performance and ability to implement any logic are very important, but perhaps the most important advantage is the ability to develop, use and share Zabbix modules. It contributes to trouble-free maintenance and helps to deliver new functionality easier and independently of the Zabbix core code base.

Module licensing and distribution in binary form is governed by the GPL license (modules are linking with Zabbix in runtime and are using Zabbix headers; currently the whole Zabbix code is licensed under GPL license). Binary compatibility is not guaranteed by Zabbix.

Module API stability is guaranteed during one Zabbix LTS (Long Term Support) release cycle. Stability of Zabbix API is not guaranteed (technically it is possible to call Zabbix internal functions from a module, but there is no guarantee that such modules will work).

Module API

In order for a shared library to be treated as a Zabbix module, it should implement and export several functions. There are currently six functions in the Zabbix module API, only one of which is mandatory and the other five are optional.

Mandatory interface

The only mandatory function is **zbx_module_api_version()**:

```
int zbx_module_api_version(void);
```

This function should return the API version implemented by this module and in order for the module to be loaded this version must match module API version supported by Zabbix. Version of module API supported by Zabbix is ZBX_MODULE_API_VERSION. So this function should return this constant. Old constant ZBX_MODULE_API_VERSION_ONE used for this purpose is now defined to equal ZBX_MODULE_API_VERSION to preserve source compatibility, but it's usage is not recommended.

Optional interface

The optional functions are **zbx_module_init()**, **zbx_module_item_list()**, **zbx_module_item_timeout()**, **zbx_module_history_write_cbs()** and **zbx_module_uninit()**:

```
int zbx_module_init(void);
```

This function should perform the necessary initialization for the module (if any). If successful, it should return ZBX_MODULE_OK. Otherwise, it should return ZBX_MODULE_FAIL. In the latter case Zabbix will not start.

```
ZBX_METRIC  *zbx_module_item_list(void);
```

This function should return a list of items supported by the module. Each item is defined in a ZBX_METRIC structure, see the section below for details. The list is terminated by a ZBX_METRIC structure with "key" field of NULL.

```
void    zbx_module_item_timeout(int timeout);
```

If module exports **zbx_module_item_list()** then this function is used by Zabbix to specify the timeout settings in Zabbix configuration file that the item checks implemented by the module should obey. Here, the "timeout" parameter is in seconds.

```
ZBX_HISTORY_WRITE_CBS   zbx_module_history_write_cbs(void);
```

This function should return callback functions Zabbix server will use to export history of different data types. Callback functions are provided as fields of ZBX_HISTORY_WRITE_CBS structure, fields can be NULL if module is not interested in the history of certain type.

```
int zbx_module_uninit(void);
```

This function should perform the necessary uninitialization (if any) like freeing allocated resources, closing file descriptors, etc.

All functions are called once on Zabbix startup when the module is loaded, with the exception of zbx_module_uninit(), which is called once on Zabbix shutdown when the module is unloaded.

Defining items

Each item is defined in a ZBX_METRIC structure:

```
typedef struct
{
    char        *key;
    unsigned    flags;
    int     (*function)();
    char        *test_param;
}
ZBX_METRIC;
```

Here, **key** is the item key (e.g., "dummy.random"), **flags** is either CF_HAVEPARAMS or 0 (depending on whether the item accepts parameters or not), **function** is a C function that implements the item (e.g., "zbx_module_dummy_random"), and **test_param** is the parameter list to be used when Zabbix agent is started with the "-p" flag (e.g., "1,1000", can be NULL). An example definition may look like this:

```
static ZBX_METRIC keys[] =
{
    { "dummy.random", CF_HAVEPARAMS, zbx_module_dummy_random, "1,1000" },
    { NULL }
}
```

Each function that implements an item should accept two pointer parameters, the first one of type AGENT_REQUEST and the second one of type AGENT_RESULT:

```
int zbx_module_dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    ...

    SET_UI64_RESULT(result, from + rand() % (to - from + 1));

    return SYSINFO_RET_OK;
}
```

These functions should return SYSINFO_RET_OK, if the item value was successfully obtained. Otherwise, they should return SYSINFO_RET_FAIL. See example "dummy" module below for details on how to obtain information from AGENT_REQUEST and how to set information in AGENT_RESULT.

Providing history export callbacks

Attention:
History export via module is no longer supported by Zabbix proxy since Zabbix 4.0.0.

Module can specify functions to export history data by type: Numeric (float), Numeric (unsigned), Character, Text and Log:

```
typedef struct
{
    void    (*history_float_cb)(const ZBX_HISTORY_FLOAT *history, int history_num);
    void    (*history_integer_cb)(const ZBX_HISTORY_INTEGER *history, int history_num);
    void    (*history_string_cb)(const ZBX_HISTORY_STRING *history, int history_num);
    void    (*history_text_cb)(const ZBX_HISTORY_TEXT *history, int history_num);
    void    (*history_log_cb)(const ZBX_HISTORY_LOG *history, int history_num);
}
ZBX_HISTORY_WRITE_CBS;
```

Each of them should take "history" array of "history_num" elements as arguments. Depending on history data type to be exported, "history" is an array of the following structures, respectively:

```
typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    double      value;
}
ZBX_HISTORY_FLOAT;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    zbx_uint64_t    value;
}
ZBX_HISTORY_INTEGER;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    const char  *value;
}
ZBX_HISTORY_STRING;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    const char  *value;
}
ZBX_HISTORY_TEXT;

typedef struct
{
    zbx_uint64_t    itemid;
    int     clock;
    int     ns;
    const char  *value;
    const char  *source;
    int     timestamp;
```

```
    int     logeventid;
    int     severity;
}
ZBX_HISTORY_LOG;
```

Callbacks will be used by Zabbix server history syncer processes in the end of history sync procedure after data is written into Zabbix database and saved in value cache.

Attention:
In case of internal error in history export module, it is recommended that module is written in such a way that it does not block whole monitoring until it recovers but discards data instead and allows Zabbix server to continue running.

Building modules

Modules are currently meant to be built inside Zabbix source tree, because the module API depends on some data structures that are defined in Zabbix headers.

The most important header for loadable modules is **include/module.h**, which defines these data structures. Other necessary system headers that help **include/module.h** to work properly are **stdlib.h** and **stdint.h**.

With this information in mind, everything is ready for the module to be built. The module should include **stdlib.h**, **stdint.h** and **module.h**, and the build script should make sure that these files are in the include path. See example "dummy" module below for details.

Another useful header is **include/log.h**, which defines **zabbix_log()** function, which can be used for logging and debugging purposes.

Configuration parameters

Zabbix agent, server and proxy support two parameters to deal with modules:

- LoadModulePath – full path to the location of loadable modules
- LoadModule – module(s) to load at startup. The modules must be located in a directory specified by LoadModulePath or the path must precede the module name. If the preceding path is absolute (starts with '/') then LoadModulePath is ignored. It is allowed to include multiple LoadModule parameters.

For example, to extend Zabbix agent we could add the following parameters:

```
LoadModulePath=/usr/local/lib/zabbix/agent/
LoadModule=mariadb.so
LoadModule=apache.so
LoadModule=kernel.so
LoadModule=/usr/local/lib/zabbix/dummy.so
```

Upon agent startup it will load the mariadb.so, apache.so and kernel.so modules from the /usr/local/lib/zabbix/agent directory while dummy.so will be loaded from /usr/local/lib/zabbix. The agent will fail to start if a module is missing, in case of bad permissions or if a shared library is not a Zabbix module.

Frontend configuration

Loadable modules are supported by Zabbix agent, server and proxy. Therefore, item type in Zabbix frontend depends on where the module is loaded. If the module is loaded into the agent, then the item type should be "Zabbix agent" or "Zabbix agent (active)". If the module is loaded into server or proxy, then the item type should be "Simple check".

History export through Zabbix modules does not need any frontend configuration. If the module is successfully loaded by server and provides **zbx_module_history_write_cbs()** function which returns at least one non-NULL callback function then history export will be enabled automatically.

Dummy module

Zabbix includes a sample module written in C language. The module is located under src/modules/dummy:

```
alex@alex:~trunk/src/modules/dummy$ ls -l
-rw-rw-r-- 1 alex alex 9019 Apr 24 17:54 dummy.c
-rw-rw-r-- 1 alex alex   67 Apr 24 17:54 Makefile
-rw-rw-r-- 1 alex alex  245 Apr 24 17:54 README
```

The module is well documented, it can be used as a template for your own modules.

After ./configure has been run in the root of Zabbix source tree as described above, just run **make** in order to build **dummy.so**.

```c
/*
** Zabbix
** Copyright (C) 2001-2020 Zabbix SIA
**
** This program is free software; you can redistribute it and/or modify
** it under the terms of the GNU General Public License as published by
** the Free Software Foundation; either version 2 of the License, or
** (at your option) any later version.
**
** This program is distributed in the hope that it will be useful,
** but WITHOUT ANY WARRANTY; without even the implied warranty of
** MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
** GNU General Public License for more details.
**
** You should have received a copy of the GNU General Public License
** along with this program; if not, write to the Free Software
** Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301, USA.
**/

###include <stdlib.h>
###include <string.h>
###include <time.h>
###include <stdint.h>

###include "module.h"

/* the variable keeps timeout setting for item processing */
static int  item_timeout = 0;

/* module SHOULD define internal functions as static and use a naming pattern different from Zabbix intern
/* symbols (zbx_*) and loadable module API functions (zbx_module_*) to avoid conflicts
static int  dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result);
static int  dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result);
static int  dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result);

static ZBX_METRIC keys[] =
/*  KEY         FLAG        FUNCTION    TEST PARAMETERS */
{
    {"dummy.ping",      0,      dummy_ping, NULL},
    {"dummy.echo",      CF_HAVEPARAMS,  dummy_echo, "a message"},
    {"dummy.random",    CF_HAVEPARAMS,  dummy_random,   "1,1000"},
    {NULL}
};

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_api_version                                           *
 *                                                                            *
 * Purpose: returns version number of the module interface                    *
 *                                                                            *
 * Return value: ZBX_MODULE_API_VERSION - version of module.h module is       *
 *               compiled with, in order to load module successfully Zabbix   *
 *               MUST be compiled with the same version of this header file   *
 *                                                                            *
 ******************************************************************************/
int zbx_module_api_version(void)
{
    return ZBX_MODULE_API_VERSION;
}

/******************************************************************************
 *                                                                            *
```

```
 * Function: zbx_module_item_timeout                                          *
 *                                                                            *
 * Purpose: set timeout value for processing of items                         *
 *                                                                            *
 * Parameters: timeout - timeout in seconds, 0 - no timeout set               *
 *                                                                            *
 ******************************************************************************/
void    zbx_module_item_timeout(int timeout)
{
    item_timeout = timeout;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_item_list                                             *
 *                                                                            *
 * Purpose: returns list of item keys supported by the module                 *
 *                                                                            *
 * Return value: list of item keys                                            *
 *                                                                            *
 ******************************************************************************/
ZBX_METRIC  *zbx_module_item_list(void)
{
    return keys;
}

static int  dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    SET_UI64_RESULT(result, 1);

    return SYSINFO_RET_OK;
}

static int  dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    char    *param;

    if (1 != request->nparam)
    {
        /* set optional error message */
        SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
        return SYSINFO_RET_FAIL;
    }

    param = get_rparam(request, 0);

    SET_STR_RESULT(result, strdup(param));

    return SYSINFO_RET_OK;
}

/******************************************************************************
 *                                                                            *
 * Function: dummy_random                                                     *
 *                                                                            *
 * Purpose: a main entry point for processing of an item                      *
 *                                                                            *
 * Parameters: request - structure that contains item key and parameters      *
 *              request->key - item key without parameters                    *
 *              request->nparam - number of parameters                        *
 *              request->params[N-1] - pointers to item key parameters        *
 *              request->types[N-1] - item key parameters types:              *
```

```
 *                  REQUEST_PARAMETER_TYPE_UNDEFINED (key parameter is empty) *
 *                  REQUEST_PARAMETER_TYPE_ARRAY (array)                      *
 *                  REQUEST_PARAMETER_TYPE_STRING (quoted or unquoted string) *
 *                                                                            *
 *             result - structure that will contain result                    *
 *                                                                            *
 * Return value: SYSINFO_RET_FAIL - function failed, item will be marked      *
 *                                  as not supported by zabbix                *
 *               SYSINFO_RET_OK - success                                     *
 *                                                                            *
 * Comment: get_rparam(request, N-1) can be used to get a pointer to the Nth  *
 *          parameter starting from 0 (first parameter). Make sure it exists  *
 *          by checking value of request->nparam.                             *
 *          In the same manner get_rparam_type(request, N-1) can be used to   *
 *          get a parameter type.                                             *
 *                                                                            *
 ******************************************************************************/
static int  dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    char    *param1, *param2;
    int from, to;

    if (2 != request->nparam)
    {
        /* set optional error message */
        SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
        return SYSINFO_RET_FAIL;
    }

    param1 = get_rparam(request, 0);
    param2 = get_rparam(request, 1);

    /* there is no strict validation of parameters and types for simplicity sake */
    from = atoi(param1);
    to = atoi(param2);

    if (from > to)
    {
        SET_MSG_RESULT(result, strdup("Invalid range specified."));
        return SYSINFO_RET_FAIL;
    }

    SET_UI64_RESULT(result, from + rand() % (to - from + 1));

    return SYSINFO_RET_OK;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_init                                                  *
 *                                                                            *
 * Purpose: the function is called on agent startup                           *
 *          It should be used to call any initialization routines             *
 *                                                                            *
 * Return value: ZBX_MODULE_OK - success                                      *
 *               ZBX_MODULE_FAIL - module initialization failed               *
 *                                                                            *
 * Comment: the module won't be loaded in case of ZBX_MODULE_FAIL             *
 *                                                                            *
 ******************************************************************************/
int zbx_module_init(void)
{
```

```
    /* initialization for dummy.random */
    srand(time(NULL));

    return ZBX_MODULE_OK;
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_uninit                                                *
 *                                                                            *
 * Purpose: the function is called on agent shutdown                          *
 *          It should be used to cleanup used resources if there are any      *
 *                                                                            *
 * Return value: ZBX_MODULE_OK - success                                      *
 *               ZBX_MODULE_FAIL - function failed                            *
 *                                                                            *
 ******************************************************************************/
int zbx_module_uninit(void)
{
    return ZBX_MODULE_OK;
}

/******************************************************************************
 *                                                                            *
 * Functions: dummy_history_float_cb                                          *
 *            dummy_history_integer_cb                                        *
 *            dummy_history_string_cb                                         *
 *            dummy_history_text_cb                                           *
 *            dummy_history_log_cb                                            *
 *                                                                            *
 * Purpose: callback functions for storing historical data of types float,    *
 *          integer, string, text and log respectively in external storage    *
 *                                                                            *
 * Parameters: history     - array of historical data                         *
 *             history_num - number of elements in history array              *
 *                                                                            *
 ******************************************************************************/
static void dummy_history_float_cb(const ZBX_HISTORY_FLOAT *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

static void dummy_history_integer_cb(const ZBX_HISTORY_INTEGER *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

static void dummy_history_string_cb(const ZBX_HISTORY_STRING *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
```

```
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

static void dummy_history_text_cb(const ZBX_HISTORY_TEXT *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

static void dummy_history_log_cb(const ZBX_HISTORY_LOG *history, int history_num)
{
    int i;

    for (i = 0; i < history_num; i++)
    {
        /* do something with history[i].itemid, history[i].clock, history[i].ns, history[i].value, ... */
    }
}

/******************************************************************************
 *                                                                            *
 * Function: zbx_module_history_write_cbs                                     *
 *                                                                            *
 * Purpose: returns a set of module functions Zabbix will call to export      *
 *          different types of historical data                                *
 *                                                                            *
 * Return value: structure with callback function pointers (can be NULL if    *
 *               module is not interested in data of certain types)           *
 *                                                                            *
 ******************************************************************************/
ZBX_HISTORY_WRITE_CBS   zbx_module_history_write_cbs(void)
{
    static ZBX_HISTORY_WRITE_CBS    dummy_callbacks =
    {
        dummy_history_float_cb,
        dummy_history_integer_cb,
        dummy_history_string_cb,
        dummy_history_text_cb,
        dummy_history_log_cb,
    };

    return dummy_callbacks;
}
```

The module exports three new items:

- `dummy.ping` - always returns '1'
- `dummy.echo[param1]` - returns the first parameter as it is, for example, `dummy.echo[ABC]` will return ABC
- `dummy.random[param1, param2]` - returns a random number within the range of param1-param2, for example, `dummy.random[1,1000000]`

Limitations

Support of loadable modules is implemented for the Unix platform only. It means that it does not work for Windows agents.

In some cases a module may need to read module-related configuration parameters from *zabbix_agentd.conf*. It is not supported currently. If you need your module to use some configuration parameters you should probably implement parsing of a module-specific configuration file.

2 Plugins

Overview

Plugins provide an option to extend the monitoring capabilities of Zabbix. Plugins are written in Go programming language and are supported by Zabbix agent 2 only. Plugins provide an alternative to loadable modules (written in C), and other methods for extending Zabbix functionality.

The following features are specific to agent 2 and its plugins:

- support of scheduled and flexible intervals for both passive and active checks;
- task queue management with respect to schedule and task concurrency;
- plugin-level timeouts;
- compatibility check of Zabbix agent 2 and its plugins on start up.

Since Zabbix 6.0.0, plugins don't have to be integrated into the agent 2 directly and can be added as loadable plugins, thus making the creation process of additional plugins for gathering new monitoring metrics easier.

This page lists Zabbix native and loadable plugins, and describes plugin configuration principles from the user perspective.

For instructions and tutorials on writing your own plugins, see Developer center.

For more information on the communication process between Zabbix agent 2 and a loadable plugin, as well as the metrics collection process, see Connection diagram.

Configuring plugins

This section provides common plugin configuration principles and best practices.

All plugins are configured using *Plugins.** parameter, which can either be part of the Zabbix agent 2 configuration file or a plugin's own configuration file. If a plugin uses a separate configuration file, path to this file should be specified in the Include parameter of Zabbix agent 2 configuration file.

A typical plugin parameter has the following structure:

Plugins.<PluginName>.<Parameter>=<Value>

Additionally, there are two specific groups of parameters:

- *Plugins.<PluginName>.Default.<Parameter>=<Value>* used for defining default parameter values.
- *Plugins.<PluginName>.<SessionName>.<Parameter>=<Value>* used for defining separate sets of parameters for different monitoring targets via named sessions.

All parameter names should adhere to the following requirements:

- it is recommended to capitalize the names of your plugins;
- the parameter should be capitalized;
- special characters are not allowed;
- nesting isn't limited by a maximum level;
- the number of parameters is not limited.

For example, to perform active checks that do not have *Scheduling* update interval immediately after the agent restart only for the Uptime plugin, set `Plugins.Uptime.System.ForceActiveChecksOnStart=1` in the configuration file. Similarly, to set custom limit for concurrent checks for the CPU plugin, set the `Plugins.CPU.System.Capacity=N` in the configuration file.

Default values

Since Zabbix 6.4.3, you can set default values for the connection-related parameters (URI, username, password, etc.) in the configuration file in the format:

Plugins.<PluginName>.Default.<Parameter>=<Value>

For example, *Plugins.Mysql.Default.Username=zabbix*, *Plugins.MongoDB.Default.Uri=tcp://127.0.0.1:27017*, etc.

If a value for such parameter is not provided in an item key or in the named session parameters, the plugin will use the default value. If a default parameter is also undefined, hardcoded defaults will be used.

> **Note:**
> If an item key does not have any parameters, Zabbix agent 2 will attempt to collect the metric using values defined in the default parameters section.

Named sessions

Named sessions represent an additional level of plugin parameters and can be used to specify separate sets of authentication parameters for each of the instances being monitored. Each named session parameter should have the following structure:

Plugins.<PluginName>.Sessions.<SessionName>.<Parameter>=<Value>

A session name can be used as a connString item key parameter instead of specifying a URI, username, and/or password separately.

In item keys, the first parameter can be either a connString or a URI. If the first key parameter doesn't match any session name, it will be treated as a URI. Note that embedding credentials into a URI is not supported, use named session parameters instead.

The list of available named session parameters depends on the plugin.

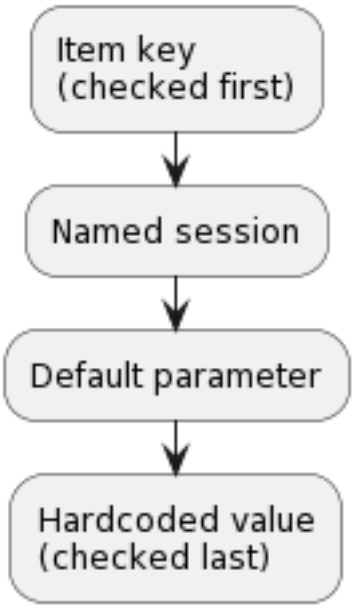
In Zabbix versions 6.4.0-6.4.2, when providing a connString (session name) in key parameters, item key parameters for username and password must be empty. The values will be taken from the session parameters. If an authentication parameter is not specified for the named session, a hardcoded default value will be used.

Since Zabbix 6.4.2, it is possible to override session parameters by specifying new values in the item key parameters (see example).

Since Zabbix 6.4.3, if a parameter is not defined for the named session, Zabbix agent 2 will use the value defined in the default plugin parameter.

Parameter priority

Since version 6.4.3, Zabbix agent 2 plugins search for connection-related parameter values in the following order:



1. The first item key parameter is compared to session names. If no match is found it is treated as an actual value; in this case, step 3 will be skipped. If a match is found, the parameter value (usually, a URI) must be defined in the named session.
2. Other parameters will be taken from the item key if defined.
3. If an item key parameter (for example, password) is empty, plugin will look for the corresponding named session parameter.
4. If the session parameter is also not specified, the value defined in the corresponding default parameter will be used.
5. If all else fails, the plugin will use the hardcoded default value.

Example 1

Monitoring of two instances "MySQL1" and "MySQL2".

Configuration parameters:

```
Plugins.Mysql.Sessions.MySQL1.Uri=tcp://127.0.0.1:3306
Plugins.Mysql.Sessions.MySQL1.User=mysql1_user
Plugins.Mysql.Sessions.MySQL1.Password=unique_password
Plugins.Mysql.Sessions.MySQL2.Uri=tcp://192.0.2.0:3306
Plugins.Mysql.Sessions.MySQL2.User=mysql2_user
Plugins.Mysql.Sessions.MySQL2.Password=different_password
```

As a result of this configuration, each session name may be used as a connString in an item key, e.g., `mysql.ping[MySQL1]` or `mysql.ping[MySQL2]`.

Example 2

Providing some of the parameters in the item key (supported since Zabbix 6.4.2).

Configuration parameters:

```
Plugins.PostgreSQL.Sessions.Session1.Uri=tcp://192.0.2.234:5432
Plugins.PostgreSQL.Sessions.Session1.User=old_username
Plugins.PostgreSQL.Sessions.Session1.Password=session_password
```

Item key: `pgsql.ping[session1,new_username,,postgres]`

As a result of this configuration, the agent will connect to PostgreSQL using the following parameters:

- URI from session parameter: *192.0.2.234:5432*
- Username from the item key: *new_username*
- Password from session parameter (since it is omitted in the item key): *session_password*
- Database name from the item key: *postgres*

Example 3

Collecting a metric using default configuration parameters.

Configuration parameters:

```
Plugins.PostgreSQL.Default.Uri=tcp://192.0.2.234:5432
Plugins.PostgreSQL.Default.User=zabbix
Plugins.PostgreSQL.Default.Password=password
```

Item key: `pgsql.ping[,,,postgres]`

As a result of this configuration, the agent will connect to PostgreSQL using the parameters:

- Default URI: *192.0.2.234:5432*
- Default username: *zabbix*
- Default password: *password*
- Database name from the item key: *postgres*

Connections

Some plugins support gathering metrics from multiple instances simultaneously. Both local and remote instances can be monitored. TCP and Unix-socket connections are supported.

It is recommended to configure plugins to keep connections to instances in an open state. The benefits are reduced network congestion, latency, and CPU and memory usage due to the lower number of connections. The client library takes care of this.

> **Note:**
> Time period for which unused connections should remain open can be determined by *Plugins.<PluginName>.KeepAlive* parameter. Example: *Plugins.Memcached.KeepAlive*

Plugins

All metrics supported by Zabbix agent 2 are collected by plugins.

Built-in

The following plugins for Zabbix agent 2 are available out-of-the-box. Click on the plugin name to go to the plugin repository with additional information.

Plugin name	Description	Supported item keys	Comments
Agent	Metrics of the Zabbix agent being used.	agent.hostname, agent.ping, agent.version	Supported keys have the same parameters as Zabbix agent keys.
Ceph	Ceph monitoring.	ceph.df.details, ceph.osd.stats, ceph.osd.discovery, ceph.osd.dump, ceph.ping, ceph.pool.discovery, ceph.status	

Plugin name	Description	Supported item keys	Comments
CPU	System CPU monitoring (number of CPUs/CPU cores, discovered CPUs, utilization percentage).	system.cpu.discovery, system.cpu.num, system.cpu.util	Supported keys have the same parameters as Zabbix agent keys.
Docker	Monitoring of Docker containers.	docker.container_info, docker.container_stats, docker.containers, docker.containers.discovery, docker.data_usage, docker.images, docker.images.discovery, docker.info, docker.ping	See also: Configuration parameters
File	File metrics collection.	vfs.file.cksum, vfs.file.contents, vfs.file.exists, vfs.file.md5sum, vfs.file.regexp, vfs.file.regmatch, vfs.file.size, vfs.file.time	Supported keys have the same parameters as Zabbix agent keys.
Kernel	Kernel monitoring.	kernel.maxfiles, kernel.maxproc	Supported keys have the same parameters as Zabbix agent keys.
Log	Log file monitoring.	log, log.count, logrt, logrt.count	Supported keys have the same parameters as Zabbix agent keys. See also: Plugin configuration parameters (Unix/Windows)
Memcached	Memcached server monitoring.	memcached.ping, memcached.stats	
Modbus	Reads Modbus data.	modbus.get	Supported keys have the same parameters as Zabbix agent keys.
MQTT	Receives published values of MQTT topics.	mqtt.get	To configure encrypted connection to the MQTT broker, specify the TLS parameters in the agent configuration file as named session or default parameters. Currently, TLS parameters cannot be passed as item key parameters.
MySQL	Monitoring of MySQL and its forks.	mysql.custom.query, mysql.db.discovery, mysql.db.size, mysql.get_status_variables, mysql.ping, mysql.replication.discovery, mysql.replication.get_slave_status, mysql.version	To configure encrypted connection to the database, specify the TLS parameters in the agent configuration file as named session or default parameters. Currently, TLS parameters cannot be passed as item key parameters. `mysql.custom.query` item key is supported since Zabbix 6.4.6.
NetIf	Monitoring of network interfaces.	net.if.collisions, net.if.discovery, net.if.in, net.if.out, net.if.total	Supported keys have the same parameters as Zabbix agent keys.

Plugin name	Description	Supported item keys	Comments
Oracle	Oracle Database monitoring.	oracle.diskgroups.stats, oracle.diskgroups.discovery, oracle.archive.info, oracle.archive.discovery, oracle.cdb.info, oracle.custom.query, oracle.datafiles.stats, oracle.db.discovery, oracle.fra.stats, oracle.instance.info, oracle.pdb.info, oracle.pdb.discovery, oracle.pga.stats, oracle.ping, oracle.proc.stats, oracle.redolog.info, oracle.sga.stats, oracle.sessions.stats, oracle.sys.metrics, oracle.sys.params, oracle.ts.stats, oracle.ts.discovery, oracle.user.info, oracle.version	Install the Oracle Instant Client before using the plugin.
Proc	Process CPU utilization percentage.	proc.cpu.util	Supported key has the same parameters as Zabbix agent key.
Redis	Redis server monitoring.	redis.config, redis.info, redis.ping, redis.slowlog.count	
Smart	S.M.A.R.T. monitoring.	smart.attribute.discovery, smart.disk.discovery, smart.disk.get	Sudo/root access rights to smartctl are required for the user executing Zabbix agent 2. The minimum required smartctl version is 7.1. Supported keys can be used with Zabbix agent 2 only on Linux/Windows, both as a passive and active check. See also: Configuration parameters
SW	Listing of installed packages.	system.sw.packages, system.sw.packages.get	The supported keys have the same parameters as Zabbix agent key.
Swap	Swap space size in bytes/percentage.	system.swap.size	Supported key has the same parameters as Zabbix agent key.
SystemRun	Runs specified command.	system.run	Supported key has the same parameters as Zabbix agent key. See also: Plugin configuration parameters (Unix/Windows)
Systemd	Monitoring of systemd services.	systemd.unit.discovery, systemd.unit.get, systemd.unit.info	
TCP	TCP connection availability check.	net.tcp.port	Supported key has the same parameters as Zabbix agent key.
UDP	Monitoring of the UDP services availability and performance.	net.udp.service, net.udp.service.perf	Supported keys have the same parameters as Zabbix agent keys.
Uname	Retrieval of information about the system.	system.hostname, system.sw.arch, system.uname	Supported keys have the same parameters as Zabbix agent keys.

Plugin name	Description	Supported item keys	Comments
Uptime	System uptime metrics collection.	system.uptime	Supported key has the same parameters as Zabbix agent key.
VFSDev	VFS metrics collection.	vfs.dev.discovery, vfs.dev.read, vfs.dev.write	Supported keys have the same parameters as Zabbix agent keys.
WebCertificate	Monitoring of TLS/SSL website certificates.	web.certificate.get	
WebPage	Web page monitoring.	web.page.get, web.page.perf, web.page.regexp	Supported keys have the same parameters as Zabbix agent keys.
ZabbixAsync	Asynchronous metrics collection.	net.tcp.listen, net.udp.listen, sensor, system.boottime, system.cpu.intr, system.cpu.load, system.cpu.switches, system.hw.cpu, system.hw.macaddr, system.localtime, system.sw.os, system.swap.in, system.swap.out, vfs.fs.discovery	Supported keys have the same parameters as Zabbix agent keys.
ZabbixStats	Zabbix server/proxy internal metrics or number of delayed items in a queue.	zabbix.stats	Supported keys have the same parameters as Zabbix agent keys.
ZabbixSync	Synchronous metrics collection.	net.dns, net.dns.record, net.tcp.service, net.tcp.service.perf, proc.mem, proc.num, system.hw.chassis, system.hw.devices, system.sw.packages, system.users.num, vfs.dir.count, vfs.dir.size, vfs.fs.get, vfs.fs.inode, vfs.fs.size, vm.memory.size.	Supported keys have the same parameters as Zabbix agent keys.

Loadable

Note:
Loadable plugins, when launched with:
 - *-V --version* - print plugin version and license information;
 - *-h --help* - print help information.

Click on the plugin name to go to the plugin repository with additional information.

Plugin name	Description	Supported item keys	Comments
MongoDB	Monitoring of MongoDB servers and clusters (document-based, distributed database).	mongodb.collection.stats, mongodb.collections.discovery, mongodb.collections.usage, mongodb.connpool.stats, mongodb.db.stats, mongodb.db.discovery, mongodb.jumbo_chunks.count, mongodb.oplog.stats, mongodb.ping, mongodb.rs.config, mongodb.rs.status, mongodb.server.status, mongodb.sh.discovery, mongodb.version	Pre-compiled plugin binaries for Windows are available since Zabbix 6.4.4 and are compatible with previous 6.4 versions. To configure encrypted connections to the database, specify the TLS parameters in the agent configuration file as named session parameters. Currently, TLS parameters cannot be passed as item key parameters. See also MongoDB plugin configuration parameters.
MSSQL	Monitoring of MSSQL database.	mssql.availability.group.get, mssql.custom.query, mssql.db.get, mssql.job.status.get, mssql.last.backup.get, mssql.local.db.get, mssql.mirroring.get, mssql.nonlocal.db.get, mssql.perfcounter.get, mssql.ping, mssql.quorum.get, mssql.quorum.member.get, mssql.replica.get, mssql.version	This plugin is supported since Zabbix 6.4.12. Pre-compiled plugin binaries for Windows are available since Zabbix 6.4.13 and are compatible with 6.4.12 version. To configure encrypted connection to the database, specify the TLS parameters in the agent configuration file as named session or default parameters. Currently, TLS parameters cannot be passed as item key parameters. See also MSSQL plugin configuration parameters.
PostgreSQL	Monitoring of PostgreSQL and its forks.	pgsql.autovacuum.count, pgsql.archive, pgsql.bgwriter, pgsql.cache.hit, pgsql.connections, pgsql.custom.query, pgsql.dbstat, pgsql.dbstat.sum, pgsql.db.age, pgsql.db.bloating_tables, pgsql.db.discovery, pgsql.db.size, pgsql.locks, pgsql.oldest.xid, pgsql.ping, pgsql.queries, pgsql.replication.count, pgsql.replication.process, pgsql.replication.process.discovery, pgsql.replication.recovery_role, pgsql.replication.status, pgsql.replication_lag.b, pgsql.replication_lag.sec, pgsql.uptime, pgsql.version, pgsql.wal.stat	Pre-compiled plugin binaries for Windows are available since Zabbix 6.4.4 and are compatible with previous 6.4 versions. To configure encrypted connections to the database, specify the TLS parameters in the agent configuration file as named session or default parameters. Currently, TLS parameters cannot be passed as item key parameters. See also PostgreSQL plugin configuration parameters.

See also: Building loadable plugins.

1 Building loadable plugins

Overview

This page provides the steps required to build a loadable plugin binary from the sources.

If the source tarball is downloaded, it is possible to build the plugin offline, i.e. without the internet connection.

The PostgreSQL plugin is used as an example. Other loadable plugins can be built in a similar way.

Steps

1. Download the plugin sources from Zabbix Cloud Images and Appliances. The official download page will be available soon.

2. Transfer the archive to the machine where you are going to build the plugin.

3. Unarchive the tarball, e.g.:

```
tar xvf zabbix-agent2-plugin-postgresql-1.0.0.tar.gz
```

Make sure to replace "zabbix-agent2-plugin-postgresql-1.0.0.tar.gz" with the name of the downloaded archive.

4. Enter the extracted directory:

```
cd <path to directory>
```

5. Run:

```
make
```

6. The plugin executable may be placed anywhere as long as it is loadable by Zabbix agent 2. Specify the path to the plugin binary in the plugin configuration file, e.g. in postgresql.conf for the PostgreSQL plugin:

```
Plugins.PostgreSQL.System.Path=/path/to/executable/zabbix-agent2-plugin-postgresql
```

7. Path to the plugin configuration file must be specified in the Include parameter of the Zabbix agent 2 configuration file:

```
Include=/path/to/plugin/configuration/file/postgresql.conf
```

Makefile targets

Loadable plugins provided by Zabbix have simple makefiles with the following targets:

Target	Description
make	Build plugin.
make clean	Delete all files that are normally created by building the plugin.
make check	Perform self-tests. A real PostgreSQL database is required.
make style	Check Go code style with 'golangci-lint'.
make format	Format Go code with 'go fmt'.
make dist	Create an archive containing the plugin sources and sources of all packages needed to build the plugin and its self-tests.

3 Frontend modules

Overview

It is possible to enhance Zabbix frontend functionality by adding third-party modules or by developing your own modules without the need to change the source code of Zabbix.

Note that the module code will run with the same privileges as Zabbix source code. This means:

- third-party modules can be harmful. You must trust the modules you are installing;
- Errors in a third-party module code may crash the frontend. If this happens, just remove the module code from the frontend. As soon as you reload Zabbix frontend, you'll see a note saying that some modules are absent. Go to Module administration (in *Administration* → *General* → *Modules*) and click *Scan directory* again to remove non-existent modules from the database.

Installation

Please always read the installation manual for a particular module. It is recommended to install new modules one by one to catch failures easily.

Just before you install a module:

- Make sure you have downloaded the module from a trusted source. Installation of harmful code may lead to consequences, such as data loss
- Different versions of the same module (same ID) can be installed in parallel, but only a single version can be enabled at once

Steps to install a module:

- Unpack your module within its own folder in the `modules` folder of the Zabbix frontend
- Ensure that your module folder contains at least the manifest.json file
- Navigate to Module administration and click the *Scan directory* button
- New module will appear in the list along with its version, author, description and status
- Enable module by clicking on its status

Troubleshooting:

Problem	Solution
Module did not appear in the list	Make sure that the manifest.json file exists in `modules/your-module/` folder of the Zabbix frontend. If it does that means the module does not suit the current Zabbix version. If manifest.json file does not exist, you have probably unpacked in the wrong directory.
Frontend crashed	The module code is not compatible with the current Zabbix version or server configuration. Please delete module files and reload the frontend. You'll see a notice that some modules are absent. Go to Module administration and click *Scan directory* again to remove non-existent modules from the database.
Error message about identical namespace, ID or actions appears	New module tried to register a namespace, ID or actions which are already registered by other enabled modules. Disable the conflicting module (mentioned in error message) prior to enabling the new one.
Technical error messages appear	Report errors to the developer of the module.

Developing modules

For information about developing custom modules, see Developer center.

21 Appendixes

Please use the sidebar to access content in the Appendixes section.

1 Installation and setup

Please use the sidebar to access content in this section.

1 Database creation

Overview

A Zabbix database must be created during the installation of Zabbix server or proxy.

This section provides instructions for creating a Zabbix database. A separate set of instructions is available for each supported database.

UTF-8 is the only encoding supported by Zabbix. It is known to work without any security flaws. Users should be aware that there are known security issues if using some of the other encodings. For switching to UTF-8, see Repairing Zabbix database character set and collation. See also Limits of filtering with utf8mb4 collations.

Note:
If installing from Zabbix Git repository, you need to run the following command prior to proceeding to the next steps:

 `$ make dbschema`

MySQL/MariaDB

Character sets utf8 (aka utf8mb3) and utf8mb4 are supported (with utf8_bin and utf8mb4_bin collation respectively) for Zabbix server/proxy to work properly with MySQL database. It is recommended to use utf8mb4 for new installations.

Deterministic triggers need to be created during the import of schema. On MySQL and MariaDB, this requires `GLOBAL log_bin_trust_function_creators = 1` to be set if binary logging is enabled and there is no superuser privileges and log_bin_trust_function_creators = 1 is not set in MySQL configuration file.

If you are installing from Zabbix **packages**, proceed to the instructions for your platform.

If you are installing Zabbix from sources:

- Create and configure a database and a user.

```
mysql -uroot -p<password>

mysql> create database zabbix character set utf8mb4 collate utf8mb4_bin;
mysql> create user 'zabbix'@'localhost' identified by '<password>';
mysql> grant all privileges on zabbix.* to 'zabbix'@'localhost';
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
mysql> quit;
```

- Import the data into the database and set utf8mb4 character set as default. For a Zabbix proxy database, only `schema.sql` should be imported (no images.sql nor data.sql).

```
cd database/mysql
mysql -uzabbix -p<password> zabbix < schema.sql
#### stop here if you are creating database for Zabbix proxy
mysql -uzabbix -p<password> zabbix < images.sql
mysql -uzabbix -p<password> --default-character-set=utf8mb4 zabbix < data.sql
```

`log_bin_trust_function_creators` can be disabled after the schema has been successfully imported:

```
mysql -uroot -p<password>

mysql> SET GLOBAL log_bin_trust_function_creators = 0;
mysql> quit;
```

PostgreSQL

You need to have database user with permissions to create database objects.

If you are installing from Zabbix **packages**, proceed to the instructions for your platform.

If you are installing Zabbix from sources:

- Create a database user.

The following shell command will create user `zabbix`. Specify a password when prompted and repeat the password (note, you may first be asked for sudo password):

```
sudo -u postgres createuser --pwprompt zabbix
```

- Create a database.

The following shell command will create the database `zabbix` (last parameter) with the previously created user as the owner (`-O zabbix`).

```
sudo -u postgres createdb -O zabbix -E Unicode -T template0 zabbix
```

- Import the initial schema and data (assuming you are in the root directory of Zabbix sources). For a Zabbix proxy database, only `schema.sql` should be imported (no images.sql nor data.sql).

```
cd database/postgresql
cat schema.sql | sudo -u zabbix psql zabbix
#### stop here if you are creating database for Zabbix proxy
```

```
cat images.sql | sudo -u zabbix psql zabbix
cat data.sql | sudo -u zabbix psql zabbix
```

Attention:
The above commands are provided as an example that will work in most of GNU/Linux installations. You can use different commands depending on how your system/database is configured, for example:

 `psql -U <username>`

 If you have any trouble setting up the database, please consult your Database administrator.

TimescaleDB

Instructions for creating and configuring TimescaleDB are provided in a separate section.

Oracle

Instructions for creating and configuring Oracle database are provided in a separate section.

SQLite

Using SQLite is supported for **Zabbix proxy** only!

The database will be automatically created if it does not exist.

Return to the installation section.

2 Repairing Zabbix database character set and collation

MySQL/MariaDB

Historically, MySQL and derivatives used 'utf8' as an alias for utf8mb3 - MySQL's own 3-byte implementation of the standard UTF8, which is 4-byte. Starting from MySQL 8.0.28 and MariaDB 10.6.1, 'utf8mb3' character set is deprecated and at some point its support will be dropped while 'utf8' will become a reference to 'utf8mb4'. Since Zabbix 6.0, 'utf8mb4' is supported. To avoid future problems, it is highly recommended to use 'utf8mb4'. Another advantage of switching to 'utf8mb4' is support of supplementary Unicode characters.

Warning:
As versions before Zabbix 6.0 are not aware of utf8mb4, make sure to first upgrade Zabbix server and DB schema to 6.0.x or later before executing utf8mb4 conversion.

1. Check the database character set and collation.

For example:

```
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------+----------------------+
| @@character_set_database | @@collation_database |
+--------------------------+----------------------+
| latin2                   | latin2_general_ci    |
+--------------------------+----------------------+
```

Or:

```
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------+----------------------+
| @@character_set_database | @@collation_database |
+--------------------------+----------------------+
| utf8                     | utf8_bin             |
+--------------------------+----------------------+
```

As we see, the character set here is not 'utf8mb4' and collation is not 'utf8mb4_bin', so we need to fix them.

2. Stop Zabbix.

3. Create a backup copy of the database!

4. Fix the character set and collation on database level:

```
alter database <your DB name> character set utf8mb4 collate utf8mb4_bin;
```

Fixed values:

```
mysql> SELECT @@character_set_database, @@collation_database;
+--------------------------+----------------------+
| @@character_set_database | @@collation_database |
+--------------------------+----------------------+
| utf8mb4                  | utf8mb4_bin          |
+--------------------------+----------------------+
```

5. Load the script to fix character set and collation on table and column level:

```
mysql <your DB name> < utf8mb4_convert.sql
```

6. Execute the script:

```
                SET @ZABBIX_DATABASE = '<your DB name>';
If MariaDB →    set innodb_strict_mode = OFF;
                CALL zbx_convert_utf8();
If MariaDB →    set innodb_strict_mode = ON;
                drop procedure zbx_convert_utf8;
```

Please note that 'utf8mb4' is expected to consume slightly more disk space.

7. If no errors - you may want to create a database backup copy with the fixed database.

8. Start Zabbix.

3 Database upgrade to primary keys

Overview

This section provides instructions for manually upgrading tables in existing installations to primary keys.

Upgrading to primary keys optimizes how data is indexed and accessed, which may speed up queries and save space. It also improves data management and synchronization in clustered setups, helping with scaling and ensuring the system remains reliable even if some servers fail.

Primary keys are used for all tables in new installations since Zabbix 6.0.

There is no automatic database upgrade to primary keys; however, existing installations may be upgraded manually **after** upgrading Zabbix server to 6.0 or newer.

Instructions are available for:

- MySQL
- PostgreSQL
- TimescaleDB
- Oracle

Attention:
The instructions provided on this page are designed for advanced users and may need to be adjusted for your specific configuration. Upgrading to primary keys can be time-consuming and resource-intensive. Ensure that enough free disk space is available; depending on your database size and stored data, the process may require up to 2.5 times the space currently used by history tables.

Important notes

To perform the database upgrade:

1. Stop Zabbix server.

Stopping Zabbix server for the time of the upgrade is strongly recommended. However, if absolutely necessary, you can perform the upgrade while the server is running (only for MySQL, MariaDB, and PostgreSQL without TimescaleDB).

2. Back up your database.
3. Run the scripts for your database.
4. Start Zabbix server.

Warning:
Run the scripts for the server database only. The proxy will not benefit from this upgrade.

If the database uses partitions, contact the DB administrator or Zabbix Support for help.

CSV files can be removed after a successful upgrade to primary keys.

Optionally, Zabbix frontend may be switched to maintenance mode.

MySQL

Export and import must be performed in tmux/screen to ensure that the session isn't dropped.

See also: Important notes

MySQL 8.0+ with mysqlsh

This method can be used with a running Zabbix server, but it is recommended to stop the server for the time of the upgrade. The MySQL Shell (*mysqlsh*) must be installed and able to connect to the DB.

- Log in to MySQL console as root (recommended) or as any user with FILE privileges.
- Start MySQL with local_infile variable enabled.
- Rename old tables and create new tables by running `history_pk_prepare.sql`.

```
mysql -uzabbix -p<password> zabbix < /usr/share/zabbix-sql-scripts/mysql/history_pk_prepare.sql
```

- Export and import data.

Connect via mysqlsh. If using a socket connection, specifying the path might be required.

```
sudo mysqlsh -uroot -S /run/mysqld/mysqld.sock --no-password -Dzabbix
```

Run (CSVPATH can be changed as needed):

```
CSVPATH="/var/lib/mysql-files";

util.exportTable("history_old", CSVPATH + "/history.csv", { dialect: "csv" });
util.importTable(CSVPATH + "/history.csv", {"dialect": "csv", "table": "history" });

util.exportTable("history_uint_old", CSVPATH + "/history_uint.csv", { dialect: "csv" });
util.importTable(CSVPATH + "/history_uint.csv", {"dialect": "csv", "table": "history_uint" });

util.exportTable("history_str_old", CSVPATH + "/history_str.csv", { dialect: "csv" });
util.importTable(CSVPATH + "/history_str.csv", {"dialect": "csv", "table": "history_str" });

util.exportTable("history_log_old", CSVPATH + "/history_log.csv", { dialect: "csv" });
util.importTable(CSVPATH + "/history_log.csv", {"dialect": "csv", "table": "history_log" });

util.exportTable("history_text_old", CSVPATH + "/history_text.csv", { dialect: "csv" });
util.importTable(CSVPATH + "/history_text.csv", {"dialect": "csv", "table": "history_text" });
```

- Follow post-migration instructions to drop the old tables.

MariaDB/MySQL 8.0+ without mysqlsh

This upgrade method takes more time and should be used only if an upgrade with *mysqlsh* is not possible.

Table upgrade

- Log in to MySQL console as root (recommended) or any user with FILE privileges.
- Start MySQL with local_infile variable enabled.
- Rename old tables and create new tables by running `history_pk_prepare.sql`:

```
mysql -uzabbix -p<password> zabbix < /usr/share/zabbix-sql-scripts/mysql/history_pk_prepare.sql
```

Migration with stopped server

max_execution_time must be disabled before migrating data to avoid timeout during migration.

```
SET @@max_execution_time=0;

INSERT IGNORE INTO history SELECT * FROM history_old;
INSERT IGNORE INTO history_uint SELECT * FROM history_uint_old;
INSERT IGNORE INTO history_str SELECT * FROM history_str_old;
INSERT IGNORE INTO history_log SELECT * FROM history_log_old;
INSERT IGNORE INTO history_text SELECT * FROM history_text_old;
```

Follow post-migration instructions to drop the old tables.

Migration with running server

Check for which paths import/export is enabled:

```
mysql> SELECT @@secure_file_priv;
+-----------------------+
| @@secure_file_priv    |
+-----------------------+
| /var/lib/mysql-files/ |
+-----------------------+
```

If *secure_file_priv* value is a path to a directory, export/import will be performed for files in that directory. In this case, edit paths to files in queries accordingly or set the *secure_file_priv* value to an empty string for the upgrade time.

If *secure_file_priv* value is empty, export/import can be performed from any location.

If *secure_file_priv* value is NULL, set it to the path that contains exported table data ('/var/lib/mysql-files/' in the example above).

For more information, see MySQL documentation.

max_execution_time must be disabled before exporting data to avoid timeout during export.

```
SET @@max_execution_time=0;

SELECT * INTO OUTFILE '/var/lib/mysql-files/history.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES TER
LOAD DATA INFILE '/var/lib/mysql-files/history.csv' IGNORE INTO TABLE history FIELDS TERMINATED BY ',' ESC

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_uint.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINE
LOAD DATA INFILE '/var/lib/mysql-files/history_uint.csv' IGNORE INTO TABLE history_uint FIELDS TERMINATED

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_str.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES
LOAD DATA INFILE '/var/lib/mysql-files/history_str.csv' IGNORE INTO TABLE history_str FIELDS TERMINATED BY

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_log.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINES
LOAD DATA INFILE '/var/lib/mysql-files/history_log.csv' IGNORE INTO TABLE history_log FIELDS TERMINATED BY

SELECT * INTO OUTFILE '/var/lib/mysql-files/history_text.csv' FIELDS TERMINATED BY ',' ESCAPED BY '"' LINE
LOAD DATA INFILE '/var/lib/mysql-files/history_text.csv' IGNORE INTO TABLE history_text FIELDS TERMINATED
```

Follow post-migration instructions to drop the old tables.

PostgreSQL

Export and import must be performed in tmux/screen to ensure that the session isn't dropped. For installations with TimescaleDB, skip this section and proceed to PostgreSQL + TimescaleDB.

See also: Important notes

Table upgrade

- Rename tables using `history_pk_prepare.sql`:

```
sudo -u zabbix psql zabbix < /usr/share/zabbix-sql-scripts/postgresql/history_pk_prepare.sql
```

Migration with stopped server

- Export current history, import it to the temp table, then insert the data into new tables while ignoring duplicates:

```
INSERT INTO history SELECT * FROM history_old ON CONFLICT (itemid,clock,ns) DO NOTHING;

INSERT INTO history_uint SELECT * FROM history_uint_old ON CONFLICT (itemid,clock,ns) DO NOTHING;

INSERT INTO history_str SELECT * FROM history_str_old ON CONFLICT (itemid,clock,ns) DO NOTHING;

INSERT INTO history_log SELECT * FROM history_log_old ON CONFLICT (itemid,clock,ns) DO NOTHING;

INSERT INTO history_text SELECT * FROM history_text_old ON CONFLICT (itemid,clock,ns) DO NOTHING;
```

See tips for improving INSERT performance: PostgreSQL: Bulk Loading Huge Amounts of Data, Checkpoint Distance and Amount of WAL.

- Follow post-migration instructions to drop the old tables.

Migration with running server

- Export current history, import it to the temp table, then insert the data into new tables while ignoring duplicates:

```
\copy history_old TO '/tmp/history.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history (
    itemid                   bigint                                    NOT NULL,
    clock                    integer         DEFAULT '0'               NOT NULL,
    value                    DOUBLE PRECISION DEFAULT '0.0000'          NOT NULL,
    ns                       integer         DEFAULT '0'               NOT NULL
);
\copy temp_history FROM '/tmp/history.csv' DELIMITER ',' CSV
INSERT INTO history SELECT * FROM temp_history ON CONFLICT (itemid,clock,ns) DO NOTHING;

\copy history_uint_old TO '/tmp/history_uint.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history_uint (
    itemid                   bigint                                    NOT NULL,
    clock                    integer         DEFAULT '0'               NOT NULL,
    value                    numeric(20)     DEFAULT '0'               NOT NULL,
    ns                       integer         DEFAULT '0'               NOT NULL
);
\copy temp_history_uint FROM '/tmp/history_uint.csv' DELIMITER ',' CSV
INSERT INTO history_uint SELECT * FROM temp_history_uint ON CONFLICT (itemid,clock,ns) DO NOTHING;

\copy history_str_old TO '/tmp/history_str.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history_str (
    itemid                   bigint                                    NOT NULL,
    clock                    integer         DEFAULT '0'               NOT NULL,
    value                    varchar(255)    DEFAULT ''                NOT NULL,
    ns                       integer         DEFAULT '0'               NOT NULL
);
\copy temp_history_str FROM '/tmp/history_str.csv' DELIMITER ',' CSV
INSERT INTO history_str (itemid,clock,value,ns) SELECT * FROM temp_history_str ON CONFLICT (itemid,clock,n

\copy history_log_old TO '/tmp/history_log.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history_log (
    itemid                   bigint                                    NOT NULL,
    clock                    integer         DEFAULT '0'               NOT NULL,
    timestamp                integer         DEFAULT '0'               NOT NULL,
    source                   varchar(64)     DEFAULT ''                NOT NULL,
    severity                 integer         DEFAULT '0'               NOT NULL,
    value                    text            DEFAULT ''                NOT NULL,
    logeventid               integer         DEFAULT '0'               NOT NULL,
    ns                       integer         DEFAULT '0'               NOT NULL
);
\copy temp_history_log FROM '/tmp/history_log.csv' DELIMITER ',' CSV
INSERT INTO history_log SELECT * FROM temp_history_log ON CONFLICT (itemid,clock,ns) DO NOTHING;

\copy history_text_old TO '/tmp/history_text.csv' DELIMITER ',' CSV
CREATE TEMP TABLE temp_history_text (
    itemid                   bigint                                    NOT NULL,
    clock                    integer         DEFAULT '0'               NOT NULL,
    value                    text            DEFAULT ''                NOT NULL,
    ns                       integer         DEFAULT '0'               NOT NULL
);
\copy temp_history_text FROM '/tmp/history_text.csv' DELIMITER ',' CSV
INSERT INTO history_text SELECT * FROM temp_history_text ON CONFLICT (itemid,clock,ns) DO NOTHING;
```

- Follow post-migration instructions to drop the old tables.

PostgreSQL + TimescaleDB

Export and import must be performed in tmux/screen to ensure that the session isn't dropped. Zabbix server should be down during the upgrade.

See also: Important notes

- Rename tables using `history_pk_prepare.sql`.

```
sudo -u zabbix psql zabbix < /usr/share/zabbix-sql-scripts/postgresql/history_pk_prepare.sql
```

- Run TimescaleDB hypertable migration scripts (compatible with both TSDB v2.x and v1.x version) based on compression settings:
  - If compression is enabled (on default installation), run scripts from `/usr/share/zabbix-sql-scripts/postgresql/tsdb_hi`

```
cat /usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_with_compression/history_pk.
cat /usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_with_compression/history_pk_
cat /usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_with_compression/history_pk_
cat /usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_with_compression/history_pk_
cat /usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_with_compression/history_pk_
```

  - If compression is disabled, run scripts from `/usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_`

```
cat /usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_no_compression/history_pk.sq
cat /usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_no_compression/history_pk_ui
cat /usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_no_compression/history_pk_lo
cat /usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_no_compression/history_pk_st
cat /usr/share/zabbix-sql-scripts/postgresql/tsdb_history_pk_upgrade_no_compression/history_pk_te
```

See also: Tips for improving INSERT performance.

- Follow post-migration instructions to drop the old tables.

Oracle

Export and import must be performed in tmux/screen to ensure that the session isn't dropped. Zabbix server should be down during the upgrade.

See also: Important notes

Table upgrade

- Install Oracle Data Pump (available in the Instant Client Tools package).

See Oracle Data Pump documentation for performance tips.

- Rename tables using `history_pk_prepare.sql`.

```
cd /path/to/zabbix-sources/database/oracle/option-patches
sqlplus zabbix/password@oracle_host/service
sqlplus> @history_pk_prepare.sql
```

Batch migration of history tables

- Prepare directories for Data Pump.

Data Pump must have read and write permissions to these directories.

Example:

```
mkdir -pv /export/history
chown -R oracle:oracle /export
```

- Create a directory object and grant read and write permissions to this object to the user used for Zabbix authentication ('zabbix' in the example below). Under *sysdba* role, run:

```
create directory history as '/export/history';
grant read,write on directory history to zabbix;
```

- Export tables. Replace N with the desired thread count.

```
expdp zabbix/password@oracle_host/service \
    DIRECTORY=history \
    TABLES=history_old,history_uint_old,history_str_old,history_log_old,history_text_old \
    PARALLEL=N
```

- Import tables. Replace N with the desired thread count.

```
impdp zabbix/password@oracle_host/service \
    DIRECTORY=history \
```

```
    TABLES=history_uint_old \
 REMAP_TABLE=history_old:history,history_uint_old:history_uint,history_str_old:history_str,history_log_old
    data_options=SKIP_CONSTRAINT_ERRORS table_exists_action=APPEND  PARALLEL=N CONTENT=data_only
```

- Follow post-migration instructions to drop the old tables.

Individual migration of history tables

- Prepare directories for Data Pump for each history table. Data Pump must have read and write permissions to these directories.

Example:

```
mkdir -pv /export/history /export/history_uint /export/history_str /export/history_log /export/history_tex
chown -R oracle:oracle /export
```

- Create a directory object and grant read and write permissions to this object to the user used for Zabbix authentication ('zabbix' in the example below). Under *sysdba* role, run:

```
create directory history as '/export/history';
grant read,write on directory history to zabbix;

create directory history_uint as '/export/history_uint';
grant read,write on directory history_uint to zabbix;

create directory history_str as '/export/history_str';
grant read,write on directory history_str to zabbix;

create directory history_log as '/export/history_log';
grant read,write on directory history_log to zabbix;

create directory history_text as '/export/history_text';
grant read,write on directory history_text to zabbix;
```

- Export and import each table. Replace N with the desired thread count.

```
expdp zabbix/password@oracle_host:1521/xe DIRECTORY=history TABLES=history_old PARALLEL=N

impdp zabbix/password@oracle_host:1521/xe DIRECTORY=history TABLES=history_old REMAP_TABLE=history_old:his

expdp zabbix/password@oracle_host:1521/xe DIRECTORY=history_uint TABLES=history_uint_old PARALLEL=N

impdp zabbix/password@oracle_host:1521/xe DIRECTORY=history_uint TABLES=history_uint_old REMAP_TABLE=histo

expdp zabbix/password@oracle_host:1521/xe DIRECTORY=history_str TABLES=history_str_old PARALLEL=N

impdp zabbix/password@oracle_host:1521/xe DIRECTORY=history_str TABLES=history_str_old REMAP_TABLE=history

expdp zabbix/password@oracle_host:1521/xe DIRECTORY=history_log TABLES=history_log_old PARALLEL=N

impdp zabbix/password@oracle_host:1521/xe DIRECTORY=history_log TABLES=history_log_old REMAP_TABLE=history

expdp zabbix/password@oracle_host:1521/xe DIRECTORY=history_text TABLES=history_text_old PARALLEL=N

impdp zabbix/password@oracle_host:1521/xe DIRECTORY=history_text TABLES=history_text_old REMAP_TABLE=histo
```

- Follow post-migration instructions to drop the old tables.

Post-migration

For all databases, once the migration is completed, do the following:

- Verify that everything works as expected.
- Drop old tables:

```
DROP TABLE history_old;
DROP TABLE history_uint_old;
DROP TABLE history_str_old;
```

```
DROP TABLE history_log_old;
DROP TABLE history_text_old;
```

4 Secure connection to the database

Overview

This section provides Zabbix setup steps and configuration examples for secure TLS connections between:

Database	Zabbix components
MySQL	Zabbix frontend, Zabbix server, Zabbix proxy
PostgreSQL	Zabbix frontend, Zabbix server, Zabbix proxy

To set up connection encryption within the DBMS, see official vendor documentation for details:

- MySQL: source and replica replication database servers.
- MySQL: group replication, etc. database servers.
- PostgreSQL encryption options.

All examples are based on the GA releases of MySQL CE (8.0) and PgSQL (13) available through official repositories using CentOS 8.

Requirements

The following is required to set up encryption:

- Developer-supported operating system with OpenSSL >=1.1.X or alternative.

Note:
It is recommended to avoid OS in the end-of-life status, especially in the case of new installations

- Database engine (RDBMS) installed and maintained from the official repository provided by developer. Operating systems often shipped with outdated database software versions for which encryption support is not implemented, for example RHEL 7 based systems and PostgreSQL 9.2, MariaDB 5.5 without encryption support.

Terminology

Setting this option enforces to use TLS connection to database from Zabbix server/proxy and frontend to database:

- required - connect using TLS as transport mode without identity checks;
- verify_ca - connect using TLS and verify certificate;
- verify_full - connect using TLS, verify certificate and verify that database identity (CN) specified by DBHost matches its certificate;

Zabbix configuration

Frontend to the database

A secure connection to the database can be configured during frontend installation:

- Mark the *Database TLS encryption* checkbox in the Configure DB connection step to enable transport encryption.
- Mark the *Verify database certificate* checkbox that appears when *TLS encryption* field is checked to enable encryption with certificates.

Note:
For MySQL, the *Database TLS encryption* checkbox is disabled, if *Database host* is set to localhost, because connection that uses a socket file (on Unix) or shared memory (on Windows) cannot be encrypted.
For PostgreSQL, the *TLS encryption* checkbox is disabled, if the value of the *Database host* field begins with a slash or the field is empty.

The following parameters become available in the TLS encryption in certificates mode (if both checkboxes are marked):

Parameter	Description
Database TLS CA file	Specify the full path to a valid TLS certificate authority (CA) file.
Database TLS key file	Specify the full path to a valid TLS key file.

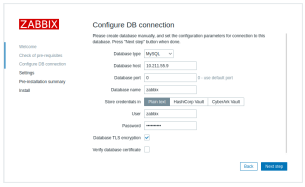
Parameter	Description
Database TLS certificate file	Specify the full path to a valid TLS certificate file.
Database host verification	Mark this checkbox to activate host verification. Disabled for MYSQL, because PHP MySQL library does not allow to skip the peer certificate validation step.
Database TLS cipher list	Specify a custom list of valid ciphers. The format of the cipher list must conform to the OpenSSL standard. Available for MySQL only.

Attention:
TLS parameters must point to valid files. If they point to non-existent or invalid files, it will lead to the authorization error. If certificate files are writable, the frontend generates a warning in the System information report that "TLS certificate files must be read-only." (displayed only if the PHP user is the owner of the certificate).

Certificates protected by passwords are not supported.

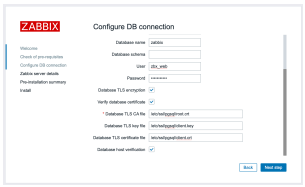
Use cases

Zabbix frontend uses GUI interface to define possible options: required, verify_ca, verify_full. Specify required options in the installation wizard step *Configure DB connections*. These options are mapped to the configuration file (zabbix.conf.php) in the following manner:

GUI settings	Configuration file	Description	Result
	... // Used for TLS connection. $DB['ENCRYPTION'] = true; $DB['KEY_FILE'] = ''; $DB['CERT_FILE'] = ''; $DB['CA_FILE'] = ''; $DB['VERIFY_HOST'] = false; $DB['CIPHER_LIST'] = ''; ...	Check *Database TLS encryption* Leave *Verify database certificate* unchecked	Enable 'required' mode.
	... $DB['ENCRYPTION'] = true;\\ $DB['KEY_FILE'] = ''; $DB['CERT_FILE'] = ''; $DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem'; $DB['VERIFY_HOST'] = false; $DB['CIPHER_LIST'] = ''; ...	1. Check *Database TLS encryption* and *Verify database certificate* 2. Specify path to *Database TLS CA file*	Enable 'verify_ca' mode.

GUI settings	Configuration file	Description	Result

	...
	// Used for TLS connection with strictly defined Cipher list.
	$DB['ENCRYPTION'] = true;
	$DB['KEY_FILE'] = '<key_file_path>';
	$DB['CERT_FILE'] = '<key_file_path>';
	$DB['CA_FILE'] = '<key_file_path>';
	$DB['VERIFY_HOST'] = true;
	$DB['CIPHER_LIST'] = '<cipher_list>';
	...
	
	Or:
	
	...
	// Used for TLS connection without Cipher list defined - selected by MySQL server
	$DB['ENCRYPTION'] = true;
	$DB['KEY_FILE'] = '<key_file_path>';
	$DB['CERT_FILE'] = '<key_file_path>';
	$DB['CA_FILE'] = '<key_file_path>';
	$DB['VERIFY_HOST'] = true;
	$DB['CIPHER_LIST'] = '';
	...
		1. Check *Database TLS encryption* and *Verify database certificate*
		2. Specify path to *Database TLS key file*
		3. Specify path to *Database TLS CA file*
		4. Specify path to *Database TLS certificate file*
		5. Specify TLS cipher list (optional)
			Enable 'verify_full' mode for MySQL.

	... $DB['ENCRYPTION'] = true; $DB['KEY_FILE'] = '<key_file_path>'; $DB['CERT_FILE'] = '<key_file_path>'; $DB['CA_FILE'] = '<key_file_path>'; $DB['VERIFY_HOST'] = true; $DB['CIPHER_LIST'] = ' '; ...	1. Check *Database TLS encryption* and *Verify database certificate* 2. Specify path to *Database TLS key file* 3. Specify path to *Database TLS CA file* 4. Specify path to *Database TLS certificate file* 5. Check *Database host verification*	Enable 'verify_full' mode for PostgreSQL.

See also: Encryption configuration examples for MySQL, Encryption configuration examples for PostgreSQL.

Zabbix server/proxy configuration

Secure connections to the database can be configured with the respective parameters in the Zabbix server and/or proxy configuration file.

Configuration	Result
None	Connection to the database without encryption.
1. Set DBTLSConnect=required	Server/proxy make a TLS connection to the database. An unencrypted connection is not allowed.
1. Set DBTLSConnect=verify_ca 2. Set DBTLSCAFile - specify the TLS certificate authority file	Server/proxy make a TLS connection to the database after verifying the database certificate.
1. Set DBTLSConnect=verify_full 2. Set DBTLSCAFile - specify TLS certificate authority file	Server/proxy make a TLS connection to the database after verifying the database certificate and the database host identity.
1. Set DBTLSCAFile - specify TLS certificate authority file 2. Set DBTLSCertFile - specify the client public key certificate file 3. Set DBTLSKeyFile - specify the client private key file	Server/proxy provide a client certificate while connecting to the database.
1. Set DBTLSCipher - the list of encryption ciphers that the client permits for connections using TLS protocols up to TLS 1.2 or DBTLSCipher13 - the list of encryption ciphers that the client permits for connections using TLS 1.3 protocol	(MySQL) TLS connection is made using a cipher from the provided list. (PostgreSQL) Setting this option will be considered as an error.

1 MySQL encryption configuration

Overview

This section provides several encryption configuration examples for CentOS 8.2 and MySQL 8.0.21 and can be used as a quickstart guide for encrypting the connection to the database.

Attention:
If MySQL host is set to localhost, encryption options will not be available. In this case a connection between Zabbix frontend and the database uses a socket file (on Unix) or shared memory (on Windows) and cannot be encrypted.

Note:
List of encryption combinations is not limited to the ones listed on this page. There are a lot more combinations available.

Pre-requisites

Install MySQL database from the official repository.

See MySQL documentation for details on how to use MySQL repo.

MySQL server is ready to accept secure connections using a self-signed certificate.

To see, which users are using an encrypted connection, run the following query (Performance Schema should be turned ON):

```
mysql> SELECT sbt.variable_value AS tls_version, t2.variable_value AS cipher, processlist_user AS user, pr
        FROM performance_schema.status_by_thread  AS sbt
        JOIN performance_schema.threads AS t ON t.thread_id = sbt.thread_id
        JOIN performance_schema.status_by_thread AS t2 ON t2.thread_id = t.thread_id
        WHERE sbt.variable_name = 'Ssl_version' and t2.variable_name = 'Ssl_cipher'
        ORDER BY tls_version;
```

Required mode

MySQL configuration

Modern versions of the database are ready out-of-the-box for 'required' encryption mode. A server-side certificate will be created after initial setup and launch.

Create users and roles for the main components:

```
mysql> CREATE USER
 'zbx_srv'@'%' IDENTIFIED WITH mysql_native_password BY '<strong_password>',
 'zbx_web'@'%' IDENTIFIED WITH mysql_native_password BY '<strong_password>'
 REQUIRE SSL
 PASSWORD HISTORY 5;
```

```
mysql> CREATE ROLE 'zbx_srv_role', 'zbx_web_role';

mysql> GRANT SELECT, UPDATE, DELETE, INSERT, CREATE, DROP, ALTER, INDEX, REFERENCES ON zabbix.* TO 'zbx_sr
mysql> GRANT SELECT, UPDATE, DELETE, INSERT ON zabbix.* TO 'zbx_web_role';

mysql> GRANT 'zbx_srv_role' TO 'zbx_srv'@'%';
mysql> GRANT 'zbx_web_role' TO 'zbx_web'@'%';

mysql> SET DEFAULT ROLE 'zbx_srv_role' TO 'zbx_srv'@'%';
mysql> SET DEFAULT ROLE 'zbx_web_role' TO 'zbx_web'@'%';
```

Note that the X.509 protocol is not used to check identity, but the user is configured to use only encrypted connections. See MySQL documentation for more details about configuring users.

Run to check connection (socket connection cannot be used to test secure connections):

```
$ mysql -u zbx_srv -p -h 10.211.55.9 --ssl-mode=REQUIRED
```

Check current status and available cipher suites:

```
mysql> status
--------------
mysql Ver 8.0.21 for Linux on x86_64 (MySQL Community Server - GPL)

Connection id: 62
Current database:
Current user: zbx_srv@bfdb.local
SSL: Cipher in use is TLS_AES_256_GCM_SHA384

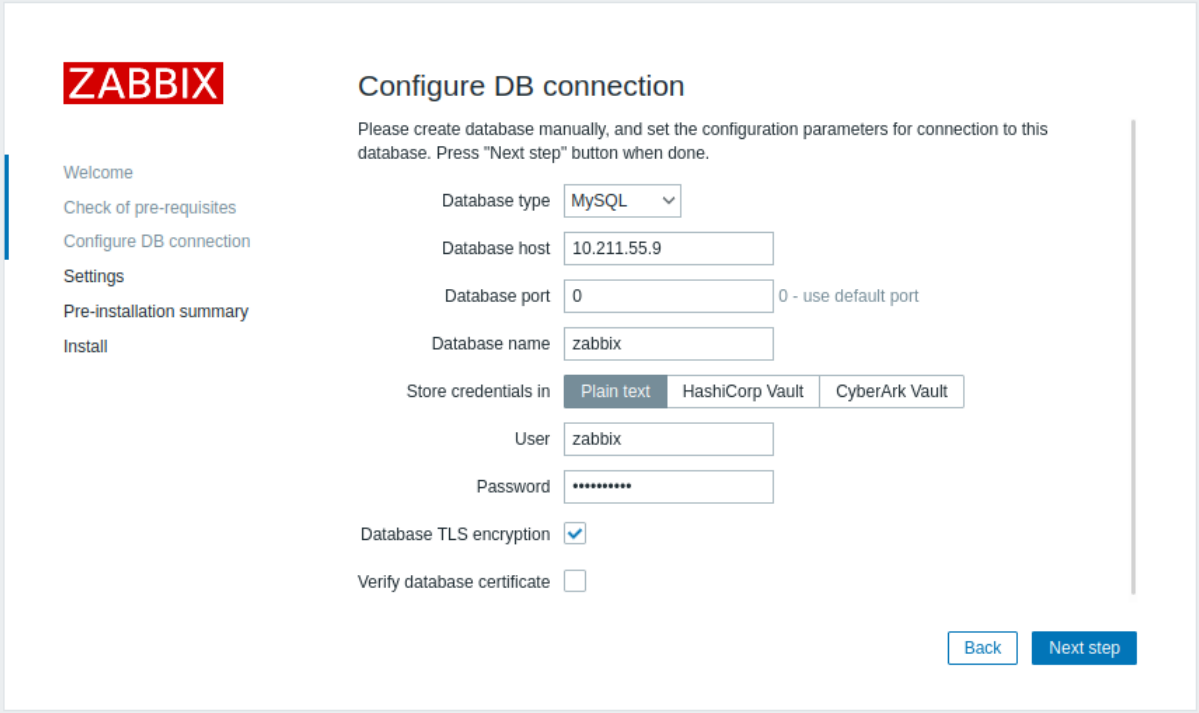

mysql> SHOW SESSION STATUS LIKE 'Ssl_cipher_list'\G;
*************************** 1. row ***************************
Variable_name: Ssl_cipher_list
Value: TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_SHA256:E
1 row in set (0.00 sec)

ERROR:
No query specified
```

Frontend

To enable transport-only encryption for connections between Zabbix frontend and the database:

- Check *Database TLS encryption*
- Leave *Verify database certificate* unchecked



Server

To enable transport-only encryption for connections between server and the database, configure */etc/zabbix/zabbix_server.conf*:

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=required
...
```

Verify CA mode

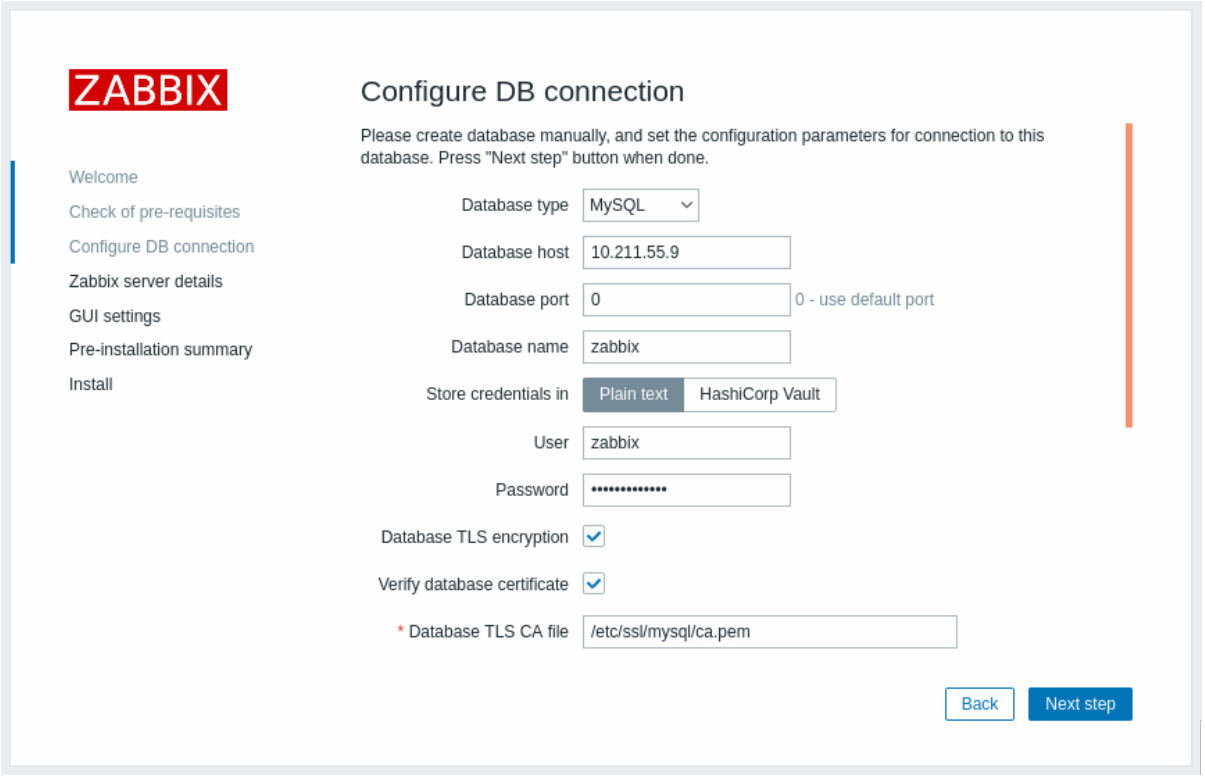
Copy required MySQL CA to the Zabbix frontend server, assign proper permissions to allow the webserver to read this file.

Note:
Verify CA mode doesn't work on SLES 12 and RHEL 7 due to older MySQL libraries.

Frontend

To enable encryption with certificate verification for connections between Zabbix frontend and the database:

- Check *Database TLS encryption* and *Verify database certificate*
- Specify path to Database TLS CA file



Alternatively, this can be set in */etc/zabbix/web/zabbix.conf.php*:

```
...
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '';
$DB['CERT_FILE'] = '';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
$DB['VERIFY_HOST'] = false;
$DB['CIPHER_LIST'] = '';
...
```

Troubleshoot user using command-line tool to check if connection is possible for required user:

```
$ mysql -u zbx_web -p -h 10.211.55.9 --ssl-mode=REQUIRED --ssl-ca=/var/lib/mysql/ca.pem
```

Server

To enable encryption with certificate verification for connections between Zabbix server and the database, configure */etc/zabbix/zabbix_server.conf*:

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_ca
DBTLSCAFile=/etc/ssl/mysql/ca.pem
...
```

Verify Full mode

MySQL configuration

Set MySQL CE server configuration option (*/etc/my.cnf.d/server-tls.cnf*) to:

```
[mysqld]
...
# in this examples keys are located in the MySQL CE datadir directory
ssl_ca=ca.pem
ssl_cert=server-cert.pem
ssl_key=server-key.pem

require_secure_transport=ON
```

```
tls_version=TLSv1.3
...
```

Keys for the MySQL CE server and client (Zabbix frontend) should be created manually according to the MySQL CE documentation: Creating SSL and RSA certificates and keys using MySQL or Creating SSL certificates and keys using openssl

Attention:
MySQL server certificate should contain the Common Name field set to the FQDN name as Zabbix frontend will use the DNS name to communicate with the database or IP address of the database host.

Create MySQL user:

```
mysql> CREATE USER
  'zbx_srv'@'%' IDENTIFIED WITH mysql_native_password BY '<strong_password>',
  'zbx_web'@'%' IDENTIFIED WITH mysql_native_password BY '<strong_password>'
  REQUIRE X509
  PASSWORD HISTORY 5;
```

Check if it is possible to log in with that user:

```
$ mysql -u zbx_web -p -h 10.211.55.9 --ssl-mode=VERIFY_IDENTITY --ssl-ca=/var/lib/mysql/ca.pem --ssl-cert=
```

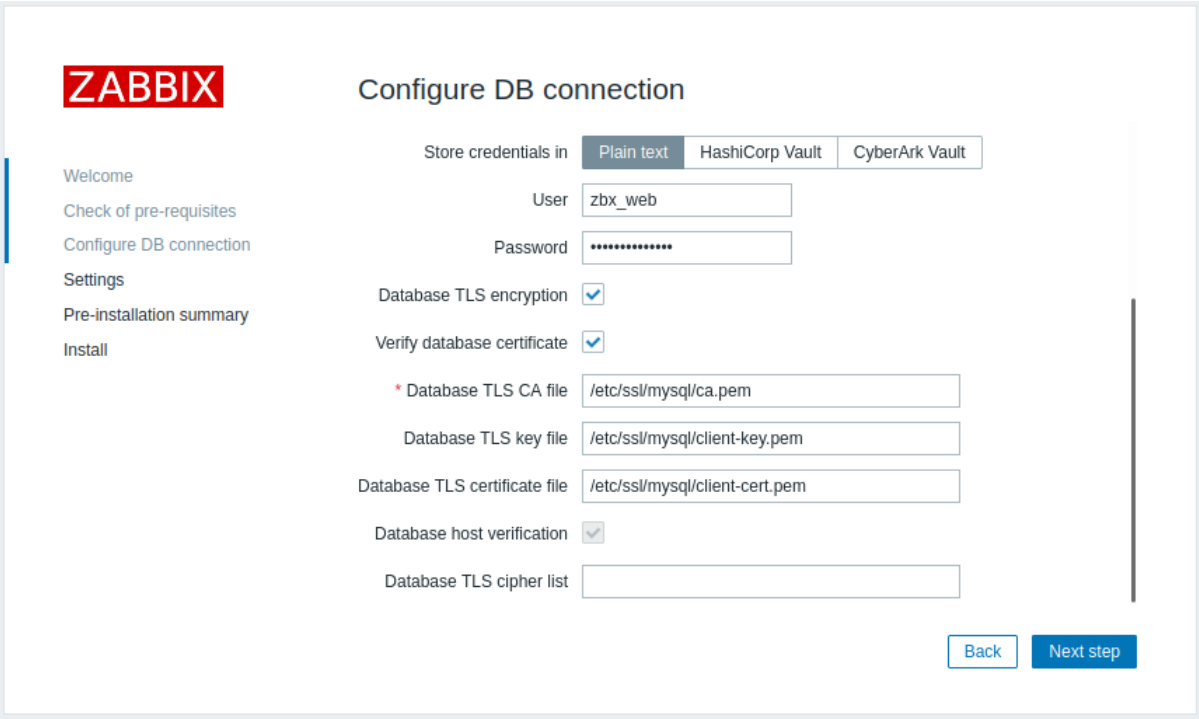
Frontend

To enable encryption with full verification for connections between Zabbix frontend and the database:

- Check Database TLS encryption and Verify database certificate
- Specify path to Database TLS key file
- Specify path to Database TLS CA file
- Specify path to Database TLS certificate file

Note that *Database host verification* is checked and grayed out - this step cannot be skipped for MySQL.

Warning:
Cipher list should be empty, so that frontend and server can negotiate required one from the supported by both ends.



Alternatively, this can be set in */etc/zabbix/web/zabbix.conf.php*:

```
...
// Used for TLS connection with strictly defined Cipher list.
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '/etc/ssl/mysql/client-key.pem';
$DB['CERT_FILE'] = '/etc/ssl/mysql/client-cert.pem';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
```

```
$DB['VERIFY_HOST'] = true;
$DB['CIPHER_LIST'] = 'TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_1
...
// or
...
// Used for TLS connection without Cipher list defined - selected by MySQL server
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '/etc/ssl/mysql/client-key.pem';
$DB['CERT_FILE'] = '/etc/ssl/mysql/client-cert.pem';
$DB['CA_FILE'] = '/etc/ssl/mysql/ca.pem';
$DB['VERIFY_HOST'] = true;
$DB['CIPHER_LIST'] = '';
...
```

Server

To enable encryption with full verification for connections between Zabbix server and the database, configure */etc/zabbix/zabbix_server.conf*:

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_full
DBTLSCAFile=/etc/ssl/mysql/ca.pem
DBTLSCertFile=/etc/ssl/mysql/client-cert.pem
DBTLSKeyFile=/etc/ssl/mysql/client-key.pem
...
```

2 PostgreSQL encryption configuration

Overview

This section provides several encryption configuration examples for CentOS 8.2 and PostgreSQL 13.

> **Note:**
> Connection between Zabbix frontend and PostgreSQL cannot be encrypted (parameters in GUI are disabled), if the value of *Database host* field begins with a slash or the field is empty.

Pre-requisites

Install the PostgreSQL database using the official repository.

PostgreSQL is not configured to accept TLS connections out-of-the-box. Please follow instructions from PostgreSQL documentation for certificate preparation with postgresql.conf and also for user access control through ph_hba.conf.

By default, the PostgreSQL socket is binded to the localhost, for the network remote connections allow to listen on the real network interface.

PostgreSQL settings for all modes can look like this:

/var/lib/pgsql/13/data/postgresql.conf:

```
...
ssl = on
ssl_ca_file = 'root.crt'
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
ssl_ciphers = 'HIGH:MEDIUM:+3DES:!aNULL'
ssl_prefer_server_ciphers = on
ssl_min_protocol_version = 'TLSv1.3'
...
```

For access control adjust */var/lib/pgsql/13/data/pg_hba.conf*:

```
...
### require
hostssl all all 0.0.0.0/0 md5
```

```
### verify CA
hostssl all all 0.0.0.0/0 md5 clientcert=verify-ca

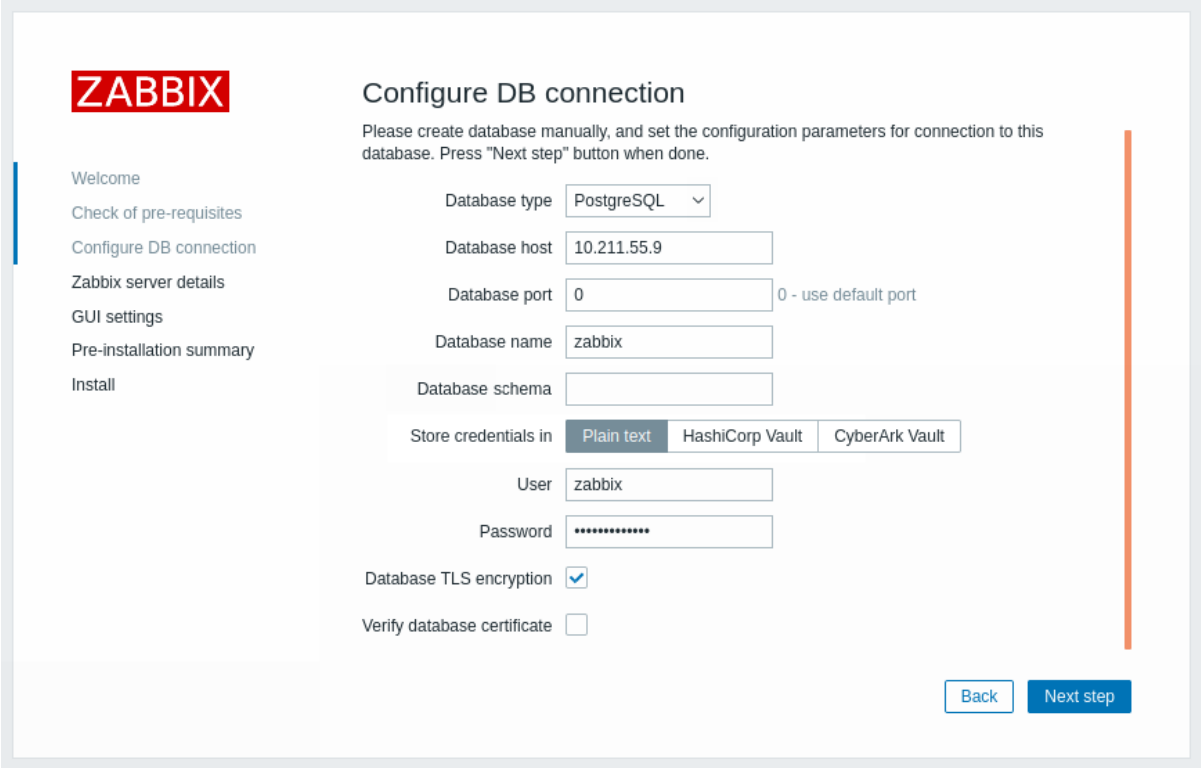
### verify full
hostssl all all 0.0.0.0/0 md5 clientcert=verify-full
...
```

Required mode

Frontend

To enable transport-only encryption for connections between Zabbix frontend and the database:

- Check *Database TLS encryption*
- Leave *Verify database certificate* unchecked



Server

To enable transport-only encryption for connections between server and the database, configure */etc/zabbix/zabbix_server.conf*:

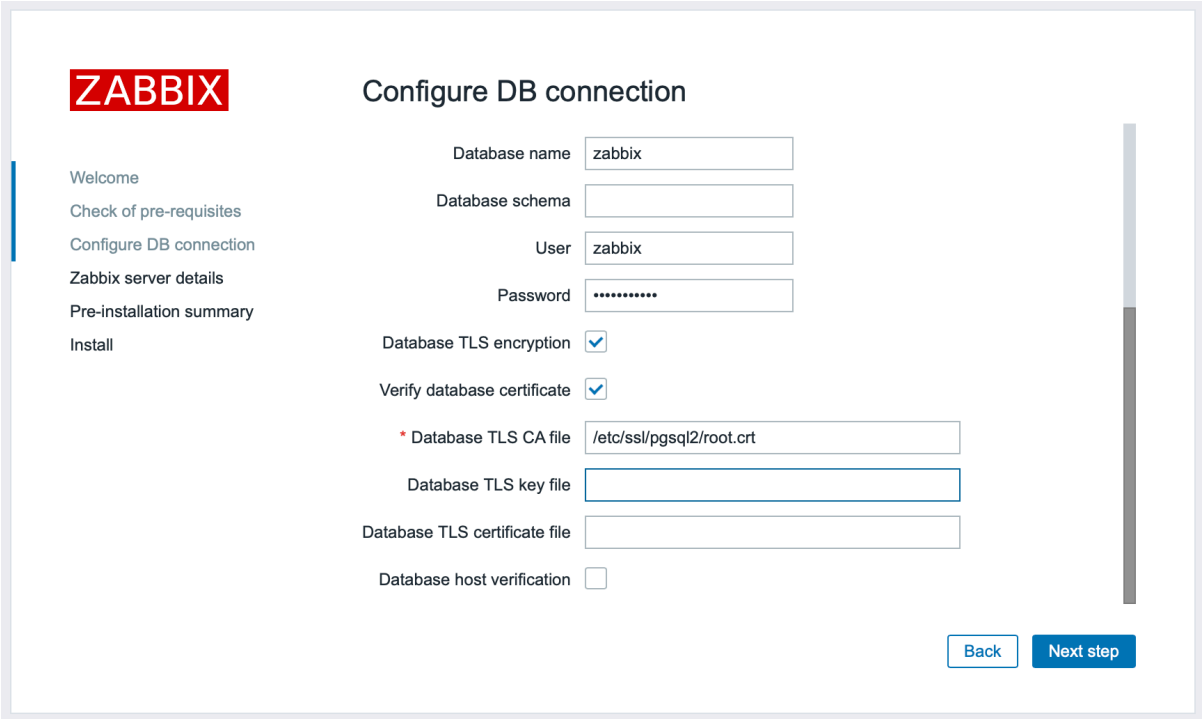
```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=required
...
```

Verify CA mode

Frontend

To enable encryption with certificate authority verification for connections between Zabbix frontend and the database:

- Check *Database TLS encryption* and *Verify database certificate*
- Specify path to *Database TLS CA file*



Alternatively, this can be set in */etc/zabbix/web/zabbix.conf.php:*

```
...
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '';
$DB['CERT_FILE'] = '';
$DB['CA_FILE'] = '/etc/ssl/pgsql/root.crt';
$DB['VERIFY_HOST'] = false;
$DB['CIPHER_LIST'] = '';
...
```

Server

To enable encryption with certificate verification for connections between Zabbix server and the database, configure */etc/zabbix/zabbix_server.conf:*

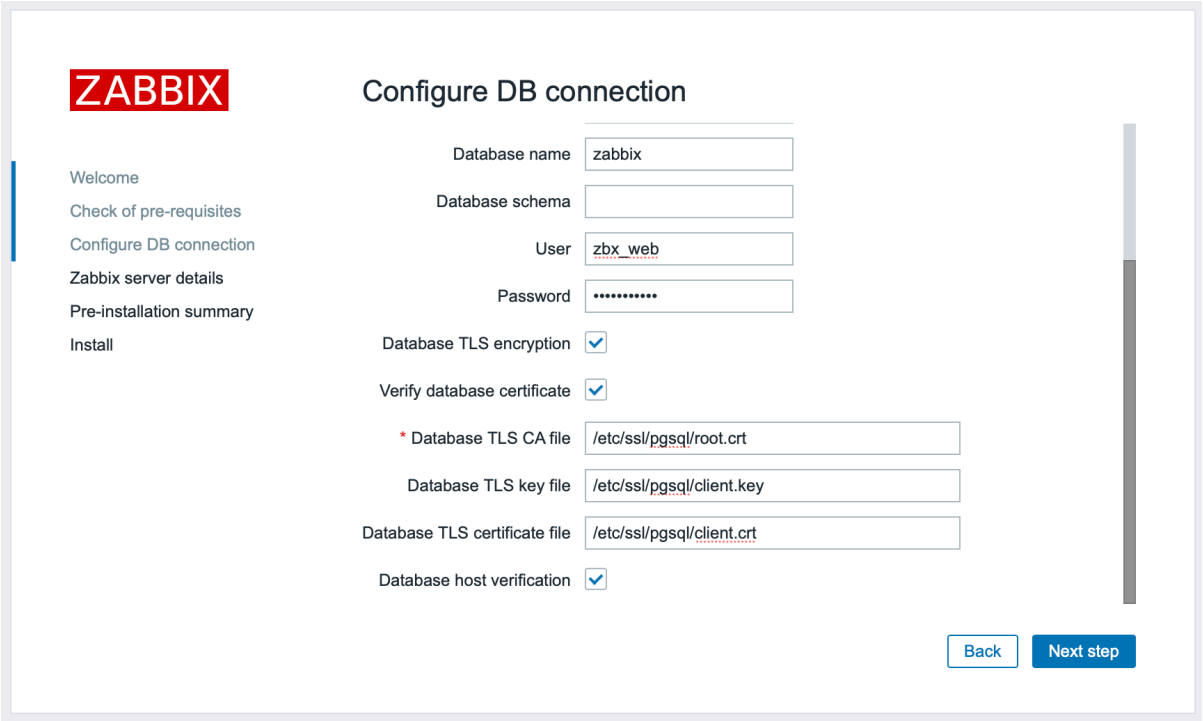
```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_ca
DBTLSCAFile=/etc/ssl/pgsql/root.crt
...
```

Verify full mode

Frontend

To enable encryption with certificate and database host identity verification for connections between Zabbix frontend and the database:

- Check *Database TLS encryption* and *Verify database certificate*
- Specify path to *Database TLS key file*
- Specify path to *Database TLS CA file*
- Specify path to *Database TLS certificate file*
- Check *Database host verification*



Alternatively, this can be set in */etc/zabbix/web/zabbix.conf.php:*

```
$DB['ENCRYPTION'] = true;
$DB['KEY_FILE'] = '';
$DB['CERT_FILE'] = '';
$DB['CA_FILE'] = '/etc/ssl/pgsql/root.crt';
$DB['VERIFY_HOST'] = true;
$DB['CIPHER_LIST'] = '';
...
```

Server

To enable encryption with certificate and database host identity verification for connections between Zabbix server and the database, configure */etc/zabbix/zabbix_server.conf*:

```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_full
DBTLSCAFile=/etc/ssl/pgsql/root.crt
DBTLSCertFile=/etc/ssl/pgsql/client.crt
DBTLSKeyFile=/etc/ssl/pgsql/client.key
...
```

5 TimescaleDB setup

Overview

Zabbix supports TimescaleDB, a PostgreSQL-based database solution of automatically partitioning data into time-based chunks to support faster performance at scale.

Warning:
Currently TimescaleDB is not supported by Zabbix proxy.

Instructions on this page can be used for creating TimescaleDB database or migrating from existing PostgreSQL tables to TimescaleDB.

Configuration

We assume that TimescaleDB extension has been already installed on the database server (see installation instructions in Timescale documentation).

TimescaleDB extension must also be enabled for the specific DB by executing:

```
echo "CREATE EXTENSION IF NOT EXISTS timescaledb CASCADE;" | sudo -u postgres psql zabbix
```

Running this command requires database administrator privileges.

Note:
If you use a database schema other than 'public' you need to add a SCHEMA clause to the command above. E.g.: `echo "CREATE EXTENSION IF NOT EXISTS timescaledb SCHEMA yourschema CASCADE;" | sudo -u postgres psql zabbix`

Then run the `postgresql/timescaledb.sql` script. For new installations the script must be run after the regular PostgreSQL database has been created with initial schema/data (see database creation):

```
cat /usr/share/zabbix-sql-scripts/postgresql/timescaledb.sql | sudo -u zabbix psql zabbix
```

Attention:
Please ignore warning messages informing that the best practices are not followed while running `timescaledb.sql` script on TimescaleDB version 2.9.0 and higher. Regardless of this warning, the configuration will be completed successfully.

The migration of existing history and trend data may take a lot of time. Zabbix server and frontend must be down for the period of migration.

The `timescaledb.sql` script sets the following housekeeping parameters:

- Override item history period
- Override item trend period

In order to use partitioned housekeeping for history and trends, both these options must be enabled. It is also possible to enable override individually either for history only or trends only.

For PostgreSQL version 10.2 or higher and TimescaleDB version 1.5 or higher, the `timescaledb.sql` script sets two additional parameters:

- Enable compression
- Compress records older than 7 days

To successfully remove compressed data by housekeeper, both *Override item history period* and *Override item trend period* options must be enabled. If override is disabled and tables have compressed chunks, the housekeeper will not remove data from these tables, and warnings about incorrect configuration will be displayed in the *Housekeeping* and *System information* sections.

All of these parameters can be changed in *Administration* → *Housekeeping* after the installation.

Note:
You may want to run the timescaledb-tune tool provided by TimescaleDB to optimize PostgreSQL configuration parameters in your `postgresql.conf`.

TimescaleDB compression

Native TimescaleDB compression is supported starting from Zabbix 5.0 for PostgreSQL version 10.2 or higher and TimescaleDB version 1.5 or higher for all Zabbix tables that are managed by TimescaleDB. During the upgrade or migration to TimescaleDB, initial compression of the large tables may take a lot of time.

Note that compression is supported under the "timescale" Timescale Community license and it is not supported under "apache" Apache 2.0 license. If Zabbix detects that compression is not supported a warning message is written into the Zabbix server log and users cannot enable compression in the frontend.

Note:
Users are encouraged to get familiar with compression in Timescale documentation before using compression.

Note that there are certain limitations imposed by compression, specifically:

- Compressed chunk modifications (inserts, deletes, updates) are not allowed
- Schema changes for compressed tables are not allowed.

Compression settings can be changed in the *History and trends compression* block in *Administration* → *Housekeeping* section of Zabbix frontend.

Parameter	Default	Comments
Enable compression	Enabled	Checking or unchecking the checkbox does not activate/deactivate compression immediately. Because compression is handled by the Housekeeper, the changes will take effect in up to 2 times `HousekeepingFrequency` hours (set in zabbix_server.conf) After disabling compression, new chunks that fall into the compression period will not be compressed. However, all previously compressed data will stay compressed. To uncompress previously compressed chunks, follow the instructions in Timescale documentation. When upgrading from older versions of Zabbix with TimescaleDB support, compression will not be enabled by default.
Compress records older than	7d	This parameter cannot be less than 7 days. Due to immutability of compressed chunks all late data (e.g. data delayed by a proxy) that is older than this value will be discarded.

6 Elasticsearch setup

Attention:
Elasticsearch support is experimental!

Zabbix supports the storage of historical data by means of Elasticsearch instead of a database. Users can choose the storage place for historical data between a compatible database and Elasticsearch. The setup procedure described in this section is applicable to Elasticsearch version 7.X. In case an earlier or later version of Elasticsearch is used, some functionality may not work as intended.

Warning:
If all history data is stored in Elasticsearch, trends are **not** calculated nor stored in the database. With no trends calculated and stored, the history storage period may need to be extended.

Configuration

To ensure proper communication between all elements involved make sure server configuration file and frontend configuration file parameters are properly configured.

Zabbix server and frontend

Zabbix server configuration file draft with parameters to be updated:

```
### Option: HistoryStorageURL
# History storage HTTP[S] URL.
#
# Mandatory: no
# Default:
# HistoryStorageURL=
### Option: HistoryStorageTypes
# Comma separated list of value types to be sent to the history storage.
#
# Mandatory: no
# Default:
# HistoryStorageTypes=uint,dbl,str,log,text
```

Example parameter values to fill the Zabbix server configuration file with:

```
HistoryStorageURL=http://test.elasticsearch.lan:9200
HistoryStorageTypes=str,log,text
```

This configuration forces Zabbix Server to store history values of numeric types in the corresponding database and textual history data in Elasticsearch.

Elasticsearch supports the following item types:

`uint,dbl,str,log,text`

Supported item type explanation:

Item value type	Database table	Elasticsearch type
Numeric (unsigned)	history_uint	uint
Numeric (float)	history	dbl
Character	history_str	str
Log	history_log	log
Text	history_text	text

Zabbix frontend configuration file (`conf/zabbix.conf.php`) draft with parameters to be updated:

```
// Elasticsearch url (can be string if same url is used for all types).
$HISTORY['url']   = [
      'uint' => 'http://localhost:9200',
      'text' => 'http://localhost:9200'
];
// Value types stored in Elasticsearch.
$HISTORY['types'] = ['uint', 'text'];
```

Example parameter values to fill the Zabbix frontend configuration file with:

```
$HISTORY['url']   = 'http://test.elasticsearch.lan:9200';
$HISTORY['types'] = ['str', 'text', 'log'];
```

This configuration forces to store `Text`, `Character` and `Log` history values in Elasticsearch.

It is also required to make `$HISTORY` global in `conf/zabbix.conf.php` to ensure everything is working properly (see `conf/zabbix.conf.php.example` for how to do it):

```
// Zabbix GUI configuration file.
global $DB, $HISTORY;
```

Installing Elasticsearch and creating mapping

Final two steps of making things work are installing Elasticsearch itself and creating mapping process.

To install Elasticsearch please refer to Elasticsearch installation guide.

Note:
Mapping is a data structure in Elasticsearch (similar to a table in a database). Mapping for all history data types is available here: `database/elasticsearch/elasticsearch.map`.

Warning:
Creating mapping is mandatory. Some functionality will be broken if mapping is not created according to the instruction.

To create mapping for `text` type send the following request to Elasticsearch:

```
curl -X PUT \
 http://your-elasticsearch.here:9200/text \
 -H 'content-type:application/json' \
 -d '{
   "settings": {
      "index": {
         "number_of_replicas": 1,
         "number_of_shards": 5
      }
   },
   "mappings": {
      "properties": {
         "itemid": {
            "type": "long"
         },
         "clock": {
            "format": "epoch_second",
```

```
            "type": "date"
         },
         "value": {
            "fields": {
               "analyzed": {
                  "index": true,
                  "type": "text",
                  "analyzer": "standard"
               }
            },
            "index": false,
            "type": "text"
         }
      }
   }
}'
```

Similar request is required to be executed for `Character` and `Log` history values mapping creation with corresponding type correction.

> **Note:**
> To work with Elasticsearch please refer to Requirement page for additional information.

> **Note:**
> Housekeeper is not deleting any data from Elasticsearch.

Storing history data in multiple date-based indices

This section describes additional steps required to work with pipelines and ingest nodes.

To begin with, you must create templates for indices.

The following example shows a request for creating uint template:

```
curl -X PUT \
 http://your-elasticsearch.here:9200/_template/uint_template \
 -H 'content-type:application/json' \
 -d '{
   "index_patterns": [
      "uint*"
   ],
   "settings": {
      "index": {
         "number_of_replicas": 1,
         "number_of_shards": 5
      }
   },
   "mappings": {
      "properties": {
         "itemid": {
            "type": "long"
         },
         "clock": {
            "format": "epoch_second",
            "type": "date"
         },
         "value": {
            "type": "long"
         }
      }
   }
}'
```

To create other templates, user should change the URL (last part is the name of template), change "`index_patterns`" field to

match index name and to set valid mapping, which can be taken from `database/elasticsearch/elasticsearch.map`.

For example, the following command can be used to create a template for text index:

```
curl -X PUT \
 http://your-elasticsearch.here:9200/_template/text_template \
 -H 'content-type:application/json' \
 -d '{
   "index_patterns": [
      "text*"
   ],
   "settings": {
      "index": {
         "number_of_replicas": 1,
         "number_of_shards": 5
      }
   },
   "mappings": {
      "properties": {
         "itemid": {
            "type": "long"
         },
         "clock": {
            "format": "epoch_second",
            "type": "date"
         },
         "value": {
            "fields": {
               "analyzed": {
                  "index": true,
                  "type": "text",
                  "analyzer": "standard"
               }
            },
            "index": false,
            "type": "text"
         }
      }
   }
}'
```

This is required to allow Elasticsearch to set valid mapping for indices created automatically. Then it is required to create the pipeline definition. Pipeline is some sort of preprocessing of data before putting data in indices. The following command can be used to create pipeline for uint index:

```
curl -X PUT \
 http://your-elasticsearch.here:9200/_ingest/pipeline/uint-pipeline \
 -H 'content-type:application/json' \
 -d '{
   "description": "daily uint index naming",
   "processors": [
      {
         "date_index_name": {
            "field": "clock",
            "date_formats": [
               "UNIX"
            ],
            "index_name_prefix": "uint-",
            "date_rounding": "d"
         }
      }
   ]
}'
```

User can change the rounding parameter (”date_rounding”) to set a specific index rotation period. To create other pipelines, user

should change the URL (last part is the name of pipeline) and change "index_name_prefix" field to match index name.

See also Elasticsearch documentation.

Additionally, storing history data in multiple date-based indices should also be enabled in the new parameter in Zabbix server configuration:

```
### Option: HistoryStorageDateIndex
# Enable preprocessing of history values in history storage to store values in different indices based on
# 0 - disable
# 1 - enable
#
# Mandatory: no
# Default:
# HistoryStorageDateIndex=0
```

Troubleshooting

The following steps may help you troubleshoot problems with Elasticsearch setup:

1. Check if the mapping is correct (GET request to required index URL like `http://localhost:9200/uint`).
2. Check if shards are not in failed state (restart of Elasticsearch should help).
3. Check the configuration of Elasticsearch. Configuration should allow access from the Zabbix frontend host and the Zabbix server host.
4. Check Elasticsearch logs.

If you are still experiencing problems with your installation then please create a bug report with all the information from this list (mapping, error logs, configuration, version, etc.)

7 Distribution-specific notes on setting up Nginx for Zabbix

RHEL

Nginx is available only in EPEL:

```
# dnf -y install epel-release
```

SLES 12

In SUSE Linux Enterprise Server 12 you need to add the Nginx repository, before installing Nginx:

```
zypper addrepo -G -t yum -c 'http://nginx.org/packages/sles/12' nginx
```

You also need to configure `php-fpm` (the path to configuration file may vary slightly depending on the service pack):

```
cp /etc/php7/fpm/php-fpm.conf{.default,}
sed -i 's/user = nobody/user = wwwrun/; s/group = nobody/group = www/' /etc/php7/fpm/php-fpm.conf
```

SLES 15

In SUSE Linux Enterprise Server 15 you need to configure `php-fpm` (the path to configuration file may vary slightly depending on the service pack):

```
cp /etc/php7/fpm/php-fpm.conf{.default,}
cp /etc/php7/fpm/php-fpm.d/www.conf{.default,}
sed -i 's/user = nobody/user = wwwrun/; s/group = nobody/group = www/' /etc/php7/fpm/php-fpm.d/www.conf
```

8 Running agent as root

Since Zabbix **5.0.0**, the systemd service file for Zabbix agent in official packages explicitly includes directives for `User` and `Group`. Both are set to `zabbix`.

It is no longer possible to configure which user Zabbix agent runs as via `zabbix_agentd.conf` file, because the agent will bypass this configuration and run as the user specified in the systemd service file. To run Zabbix agent as root you need to make the modifications described below.

Zabbix agent

To override the default user and group for Zabbix agent, run:

```
systemctl edit zabbix-agent
```

Then, add the following content:

```
[Service]
User=root
Group=root
```

Reload daemons and restart the zabbix-agent service:

```
systemctl daemon-reload
systemctl restart zabbix-agent
```

For **Zabbix agent** this re-enables the functionality of configuring user in the `zabbix_agentd.conf` file. Now you need to set `User=root` and `AllowRoot=1` configuration parameters in the agent configuration file.

Zabbix agent 2

To override the default user and group for Zabbix agent 2, run:

```
systemctl edit zabbix-agent2
```

Then, add the following content:

```
[Service]
User=root
Group=root
```

Reload daemons and restart the zabbix-agent service:

```
systemctl daemon-reload
systemctl restart zabbix-agent2
```

For **Zabbix agent2** this completely determines the user that it runs as. No additional modifications are required.

9 Zabbix agent on Microsoft Windows

Configuring agent

Both generations of Zabbix agents run as a Windows service. For Zabbix agent 2, replace *agentd* with *agent2* in the instructions below.

You can run a single instance of Zabbix agent or multiple instances of the agent on a Microsoft Windows host. A single instance can use the default configuration file `C:\zabbix_agentd.conf` or a configuration file specified in the command line. In case of multiple instances each agent instance must have its own configuration file (one of the instances can use the default configuration file).

An example configuration file is available in Zabbix source archive as `conf/zabbix_agentd.win.conf`.

See the configuration file options for details on configuring Zabbix Windows agent.

Hostname parameter

To perform active checks on a host Zabbix agent needs to have the hostname defined. Moreover, the hostname value set on the agent side should exactly match the "Host name" configured for the host in the frontend.

The hostname value on the agent side can be defined by either the **Hostname** or **HostnameItem** parameter in the agent configuration file - or the default values are used if any of these parameters are not specified.

The default value for **HostnameItem** parameter is the value returned by the "system.hostname" agent key. For Windows, it returns result of the gethostname() function, which queries namespace providers to determine the local host name. If no namespace provider responds, the NetBIOS name is returned.

The default value for **Hostname** is the value returned by the **HostnameItem** parameter. So, in effect, if both these parameters are unspecified, the actual hostname will be the host NetBIOS name; Zabbix agent will use NetBIOS host name to retrieve the list of active checks from Zabbix server and send results to it.

The "system.hostname" key supports two optional parameters - *type* and *transform*.

Type parameter determines the type of the name the item should return. Supported values:

- *netbios* (default) - returns the NetBIOS host name which is limited to 15 symbols and is in the UPPERCASE only;
- *host* - case-sensitive, returns the full, real Windows host name (without a domain);
- *shorthost* (supported since Zabbix 5.4.7) - returns part of the hostname before the first dot. It will return a full string if the name does not contain a dot.

Transform parameter is supported since Zabbix 5.4.7 and allows to specify additional transformation rule for the hostname. Supported values:

- *none* (default) - use the original letter case;
- *lower* - convert the text into lowercase.

So, to simplify the configuration of zabbix_agentd.conf file and make it unified, two different approaches could be used.

1. leave **Hostname** or **HostnameItem** parameters undefined and Zabbix agent will use NetBIOS host name as the hostname;
2. leave **Hostname** parameter undefined and define **HostnameItem** like this:
   HostnameItem=system.hostname[host] - for Zabbix agent to use the full, real (case sensitive) Windows host name as the hostname
   HostnameItem=system.hostname[shorthost,lower] - for Zabbix agent to use only part of the hostname before the first dot, converted into lowercase.

Host name is also used as part of Windows service name which is used for installing, starting, stopping and uninstalling the Windows service. For example, if Zabbix agent configuration file specifies `Hostname=Windows_db_server`, then the agent will be installed as a Windows service "`Zabbix Agent [Windows_db_server]`". Therefore, to have a different Windows service name for each Zabbix agent instance, each instance must use a different host name.

Installing agent as Windows service

To install a single instance of Zabbix agent with the default configuration file `c:\zabbix_agentd.conf`:

```
zabbix_agentd.exe --install
```

Attention:
On a 64-bit system, a 64-bit Zabbix agent version is required for all checks related to running 64-bit processes to work correctly.

If you wish to use a configuration file other than `c:\zabbix_agentd.conf`, you should use the following command for service installation:

```
zabbix_agentd.exe --config <your_configuration_file> --install
```

A full path to the configuration file should be specified.

Multiple instances of Zabbix agent can be installed as services like this:

```
zabbix_agentd.exe --config <configuration_file_for_instance_1> --install --multiple-agents
zabbix_agentd.exe --config <configuration_file_for_instance_2> --install --multiple-agents
...
zabbix_agentd.exe --config <configuration_file_for_instance_N> --install --multiple-agents
```

The installed service should now be visible in Control Panel.

Starting agent

To start the agent service, you can use Control Panel or do it from command line.

To start a single instance of Zabbix agent with the default configuration file:

```
zabbix_agentd.exe --start
```

To start a single instance of Zabbix agent with another configuration file:

```
zabbix_agentd.exe --config <your_configuration_file> --start
```

To start one of multiple instances of Zabbix agent:

```
zabbix_agentd.exe --config <configuration_file_for_this_instance> --start --multiple-agents
```

Stopping agent

To stop the agent service, you can use Control Panel or do it from command line.

To stop a single instance of Zabbix agent started with the default configuration file:

```
zabbix_agentd.exe --stop
```

To stop a single instance of Zabbix agent started with another configuration file:

```
zabbix_agentd.exe --config <your_configuration_file> --stop
```

To stop one of multiple instances of Zabbix agent:

```
zabbix_agentd.exe --config <configuration_file_for_this_instance> --stop --multiple-agents
```

Uninstalling agent Windows service

To uninstall a single instance of Zabbix agent using the default configuration file:

```
zabbix_agentd.exe --uninstall
```

To uninstall a single instance of Zabbix agent using a non-default configuration file:

```
zabbix_agentd.exe --config <your_configuration_file> --uninstall
```

To uninstall multiple instances of Zabbix agent from Windows services:

```
zabbix_agentd.exe --config <configuration_file_for_instance_1> --uninstall --multiple-agents
zabbix_agentd.exe --config <configuration_file_for_instance_2> --uninstall --multiple-agents
...
zabbix_agentd.exe --config <configuration_file_for_instance_N> --uninstall --multiple-agents
```

Limitations

Zabbix agent for Windows does not support non-standard Windows configurations where CPUs are distributed non-uniformly across NUMA nodes. If logical CPUs are distributed non-uniformly, then CPU performance metrics may not be available for some CPUs. For example, if there are 72 logical CPUs with 2 NUMA nodes, both nodes must have 36 CPUs each.

10 SAML setup with Microsoft Entra ID

Overview

This section provides guidelines for configuring single sign-on and user provisioning into Zabbix from Microsoft Entra ID (formerly Microsoft Azure Active Directory) using SAML 2.0 authentication.

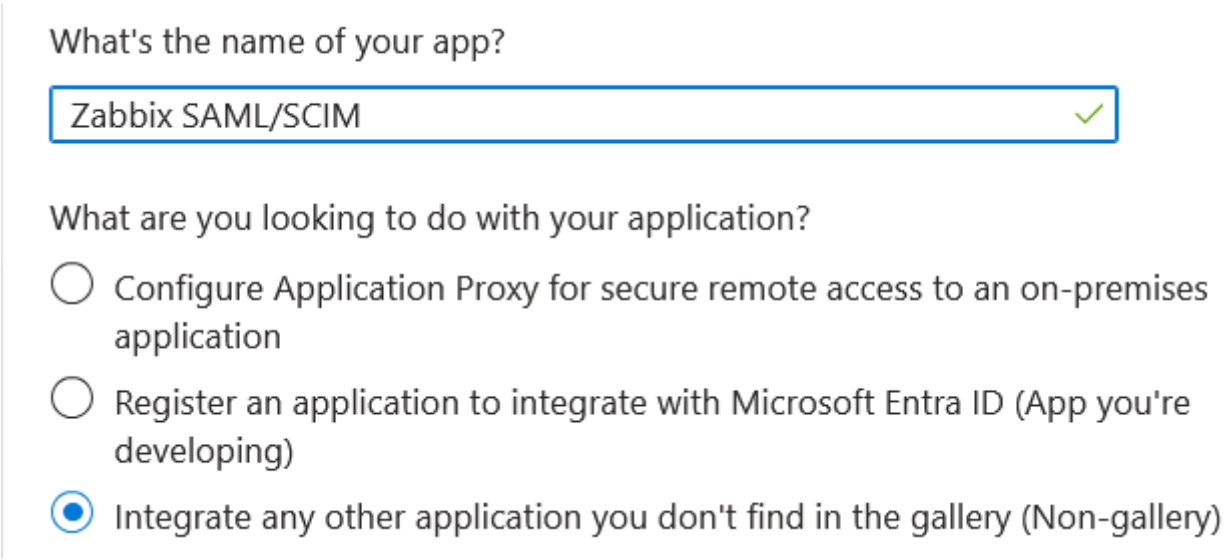
Microsoft Entra ID configuration

Creating application

1. Log into Microsoft Entra admin center at Microsoft Entra ID. For testing purposes, you may create a free trial account in Microsoft Entra ID.

2. In Microsoft Entra admin center select *Applications -> Enterprise applications -> New application -> Create your own application*.

3. Add the name of your app and select the *Integrate any other application...* option. After that, click on *Create*.

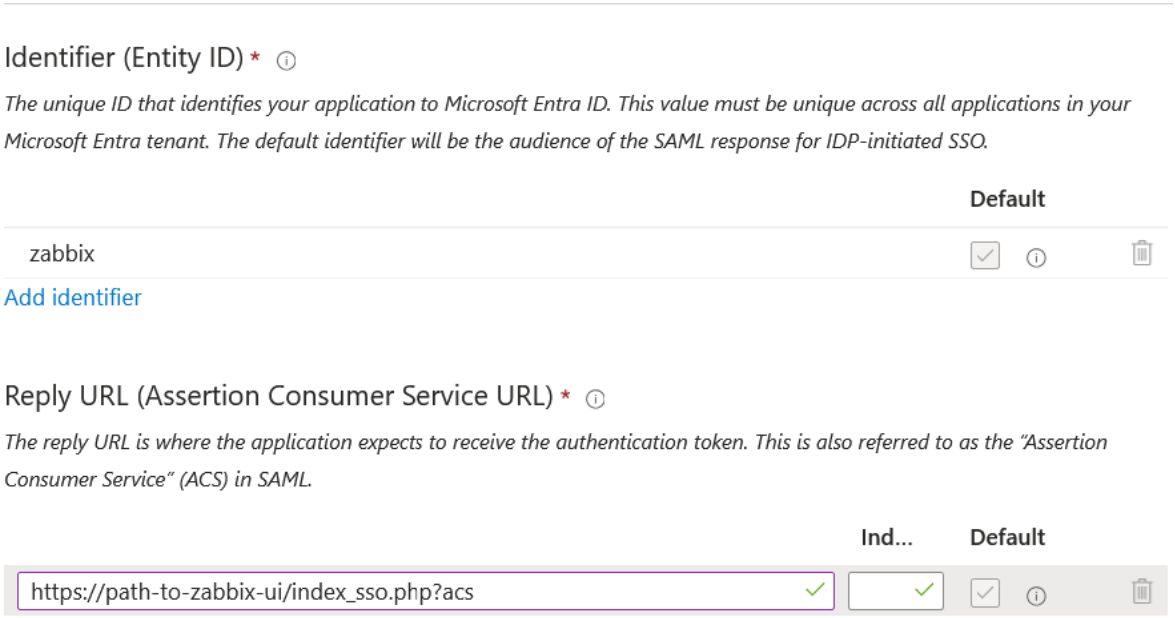


Setting up single sign-on

1. In your application page, go to *Set up single sign on* and click on *Get started*. Then select *SAML*.

2. Edit *Basic SAML Configuration*:

- In *Identifier (Entity ID)* set a unique name to identify your app to Microsoft Entra ID, for example, `zabbix`;
- In *Reply URL (Assertion Consumer Service URL)* set the Zabbix single sign-on endpoint: `https://<path-to-zabbix-ui>/index_sso`



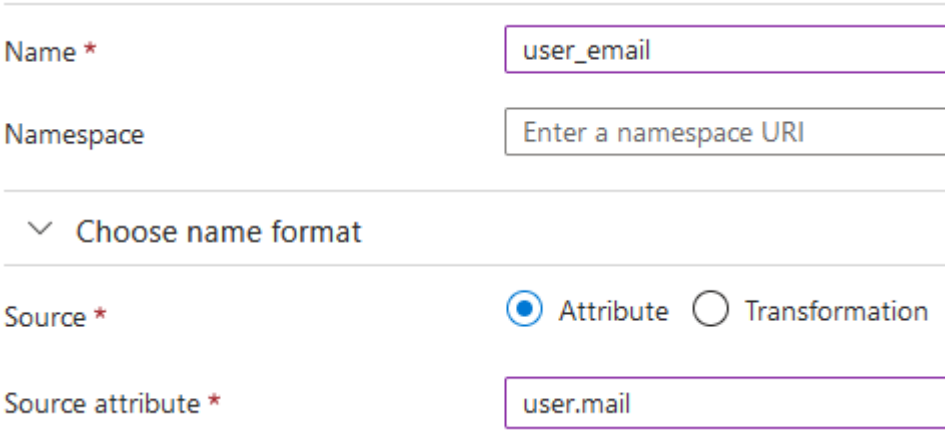
Note that "https" is required. To make that work with Zabbix, it is necessary to add to `conf/zabbix.conf.php` the following line:

```
$SSO['SETTINGS'] = ['use_proxy_headers' => true];
```

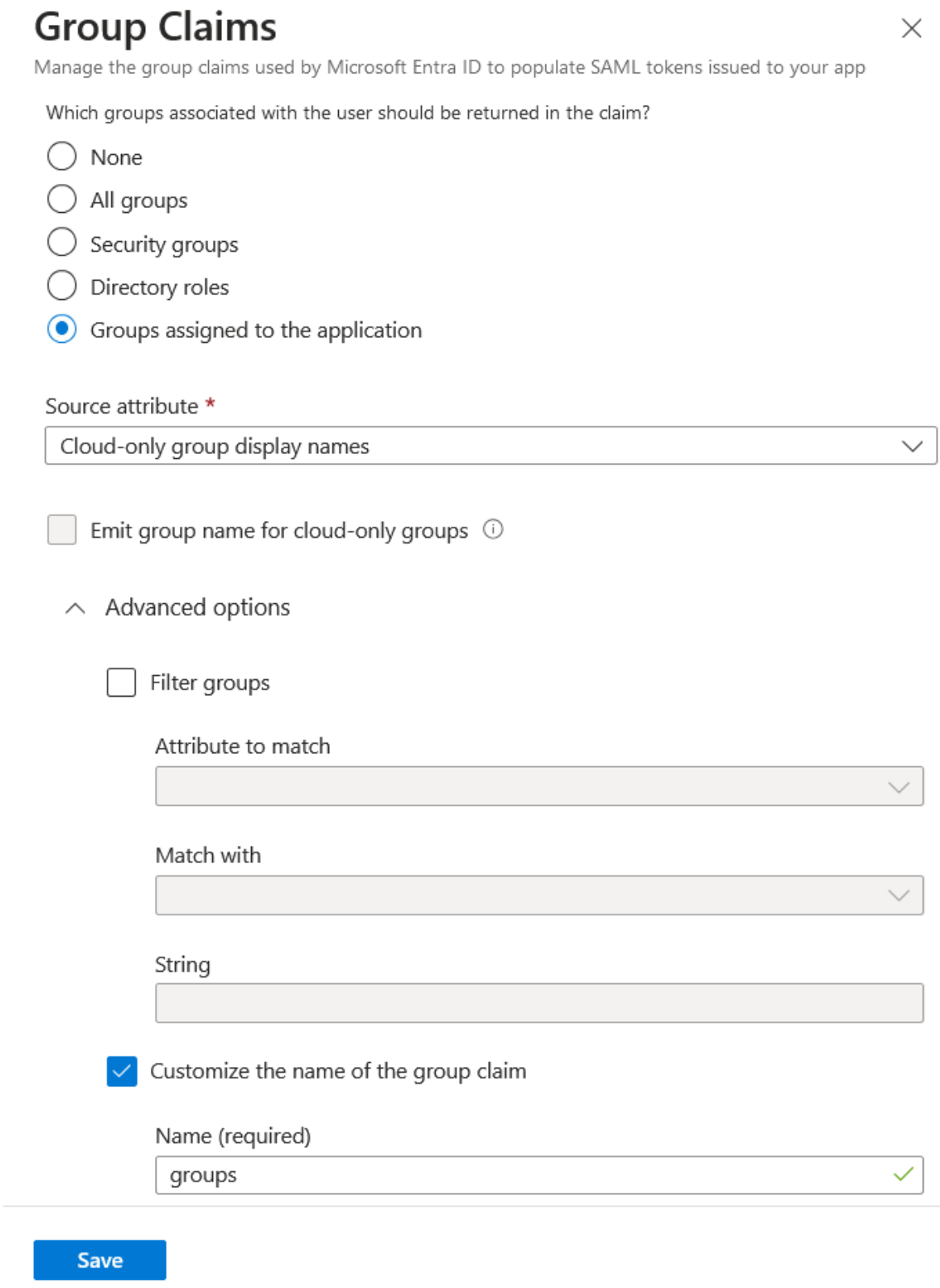
3. Edit *Attributes & Claims*. You must add all attributes that you want to pass to Zabbix (user_name, user_lastname, user_email, user_mobile, groups).

The attribute names are arbitrary. Different attribute names may be used, however, it is required that they match the respective field value in Zabbix SAML settings.

- Click on *Add new claim* to add an attribute:



- Click on *Add a group claim* to add an attribute for passing groups to Zabbix:



It is important in this claim that the group names (rather than group IDs) are passed to Zabbix by the selected *Source attribute*. Otherwise JIT user provisioning will not work properly.

4. In *SAML Certificates* download the Base64 certificate provided by Entra ID and place it into `conf/certs` of the Zabbix frontend installation.

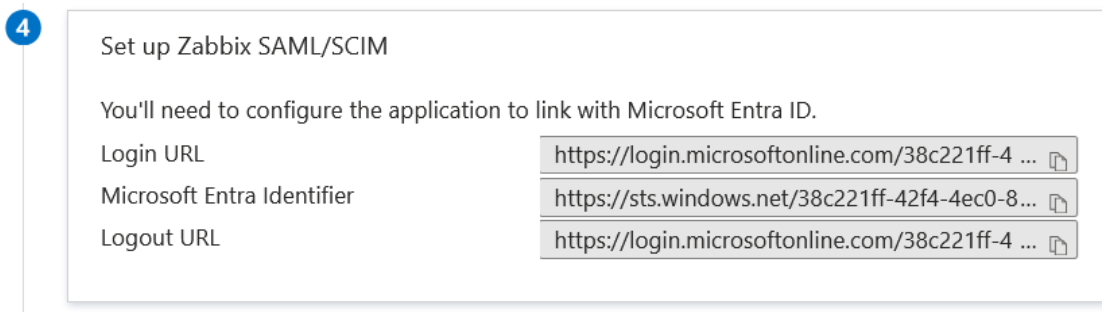
Set 644 permissions to it by running:

```
chmod 644 entra.cer
```

Make sure that `conf/zabbix.conf.php` contains the line:

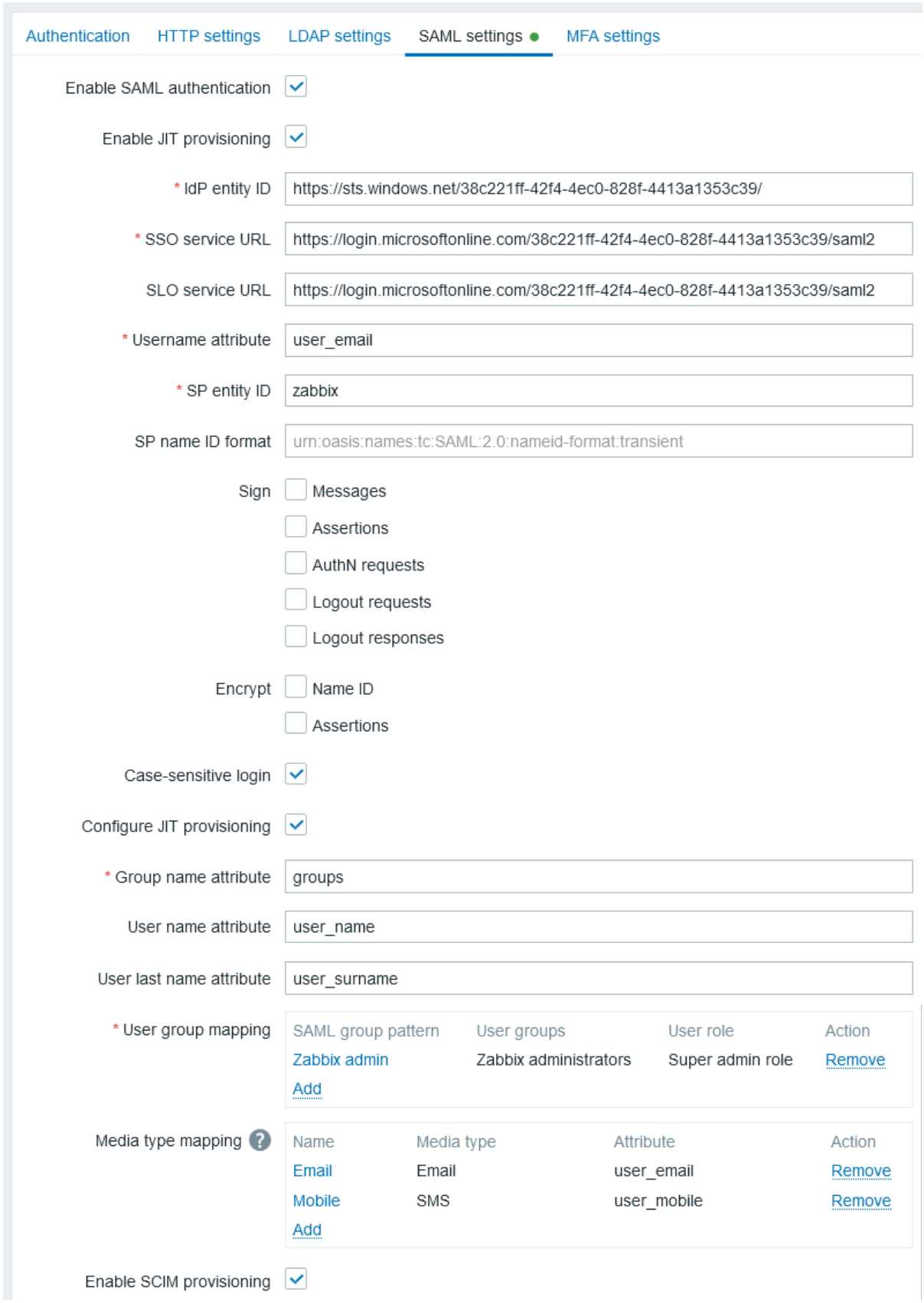
```
$SSO['IDP_CERT'] = 'conf/certs/entra.cer';
```

5. Use the values from *Set up <your app name>* in Entra ID to configure Zabbix SAML authentication (see next section):



Zabbix configuration

1. In Zabbix, go to the SAML settings and fill the configuration options based on the Entra ID configuration:



Zabbix field	Setup field in Entra ID	Sample value
IdP entity ID	Microsoft Entra identifier	
SSO service URL	Login URL	
SLO service URL	Logout URL	
SP entity ID	Identifier (Entity ID)	
Username attribute	Custom attribute (claim)	`user_email`
Group name attribute	Custom attribute (claim)	`groups`
User name attribute	Custom attribute (claim)	`user_name`

Zabbix field	Setup field in Entra ID	Sample value
User last name attribute	Custom attribute (claim)	`user_lastname`

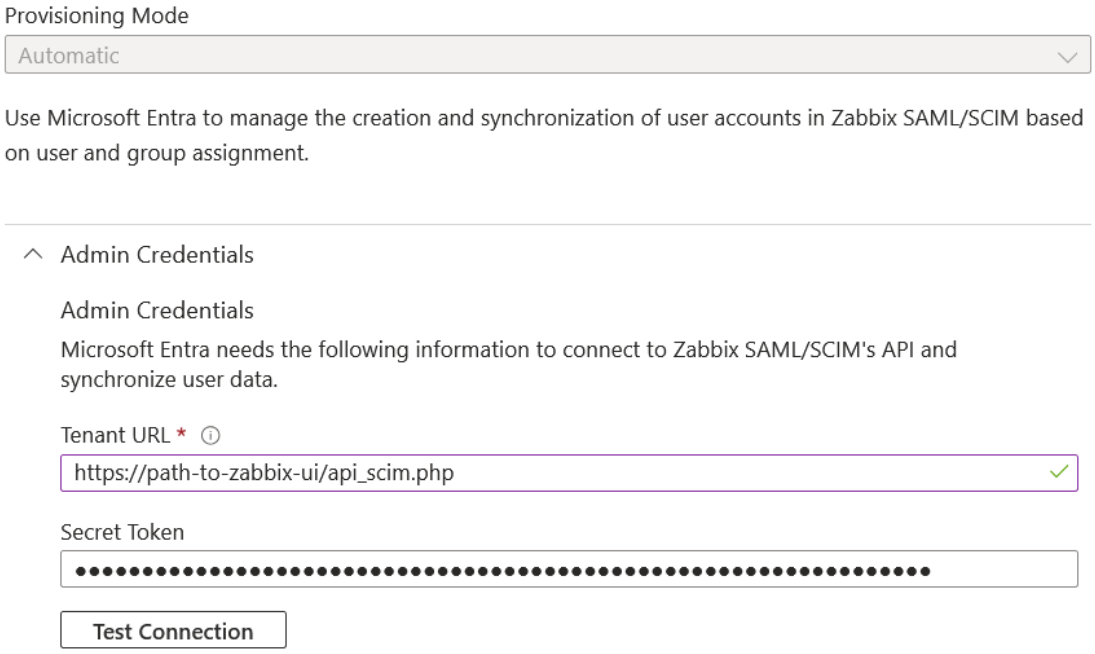
It is also required to configure user group mapping. Media mapping is optional.

Click on *Update* to save these settings.

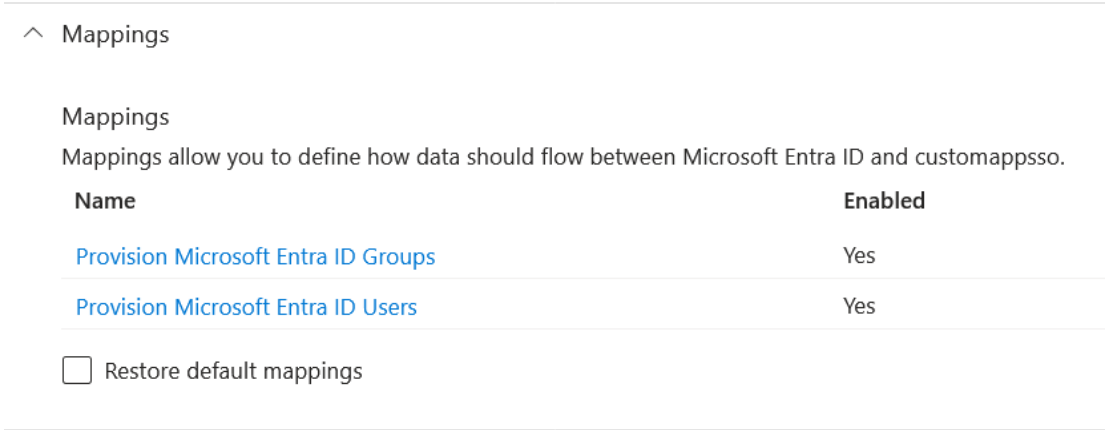
SCIM user provisioning

1. In your Entra ID application page, from the main menu open the Provisioning page. Click on *Get started* and then select Automatic provisioning mode:

- In *Tenant URL*, set the following value: `https://<path-to-zabbix-ui>/api_scim.php`
- In *Secret token*, enter a Zabbix API token with Super admin permissions.
- Click on *Test connection* to see if the connection is established.

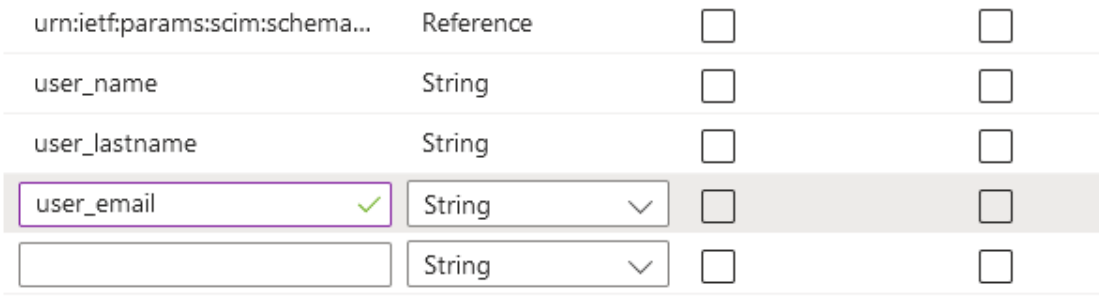


2. Now you can add all the attributes that will be passed with SCIM to Zabbix. To do that, click on *Mappings* and then on *Provision Microsoft Entra ID Users*.



At the bottom of the Attribute Mapping list, enable *Show advanced options*, and then click on *Edit attribute list for customappsso*.

At the bottom of the attribute list, add your own attributes with type 'String':



Save the list.

3. Now you can add mappings for the added attributes. At the bottom of the Attribute Mapping list, click on *Add New Mapping* and create mappings as shown below:



When all mappings are added, save the list of mappings.



4. As a prerequisite of user provisioning into Zabbix, you must have users and groups configured in Entra ID.

To do that, go to *Microsoft Entra admin center* and then add users/groups in the respective Users and Groups pages.

5. When users and groups have been created in Entra ID, you can go to the *Users and groups* menu of your application and add them to the app.

6. Go to the *Provisioning* menu of your app, and click on *Start provisioning* to have users provisioned to Zabbix.

Note that the Users PATCH request in Entra ID does not support changes in media.

11 SAML setup with Okta

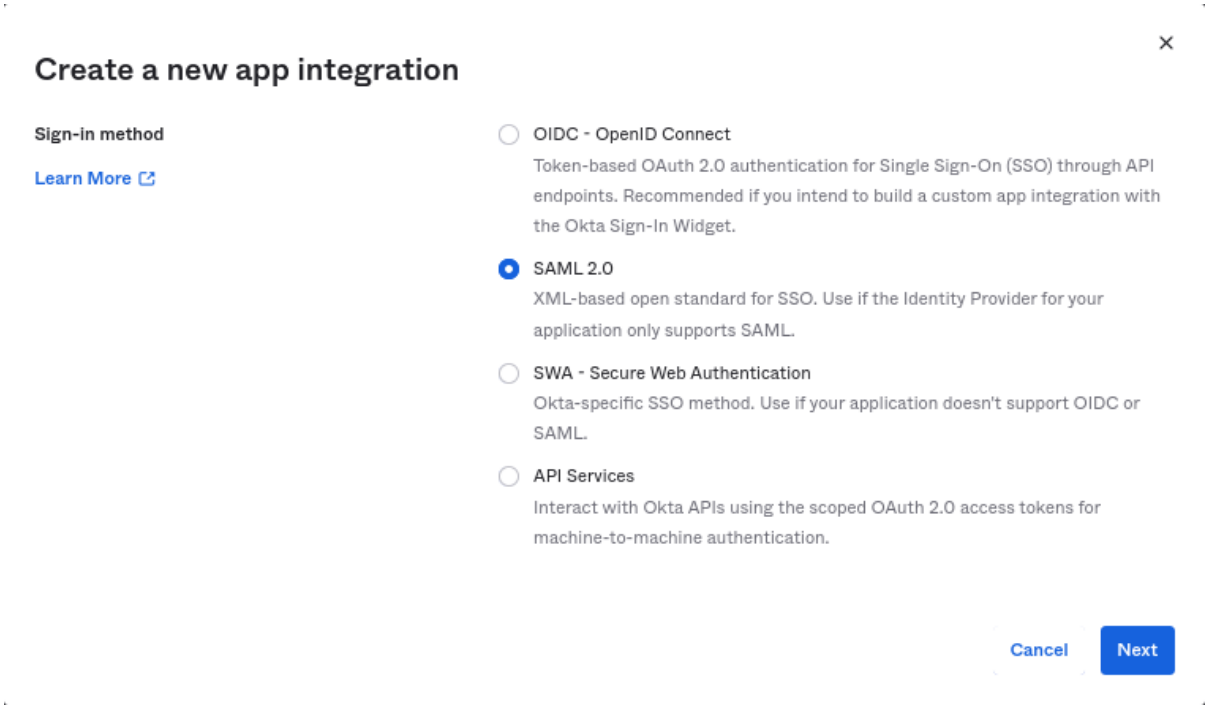
This section provides guidelines for configuring Okta to enable SAML 2.0 authentication and user provisioning for Zabbix.

Okta configuration

1. Go to https://developer.okta.com/signup/ and register/sign into your account.

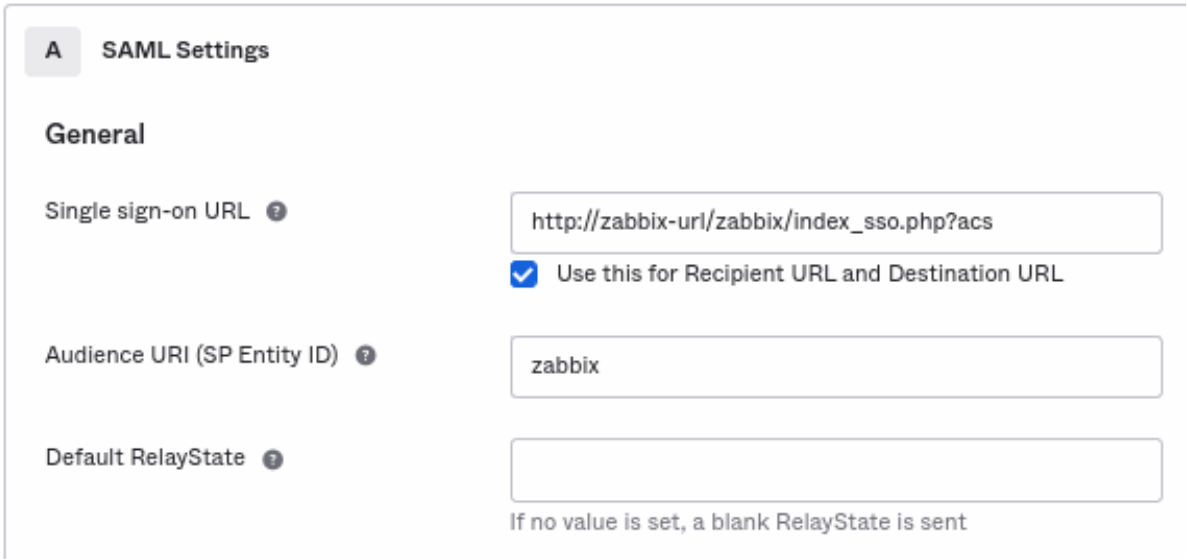
2. In the Okta web interface navigate to *Applications* → *Applications*.

3. Click on *Create App Integration*.

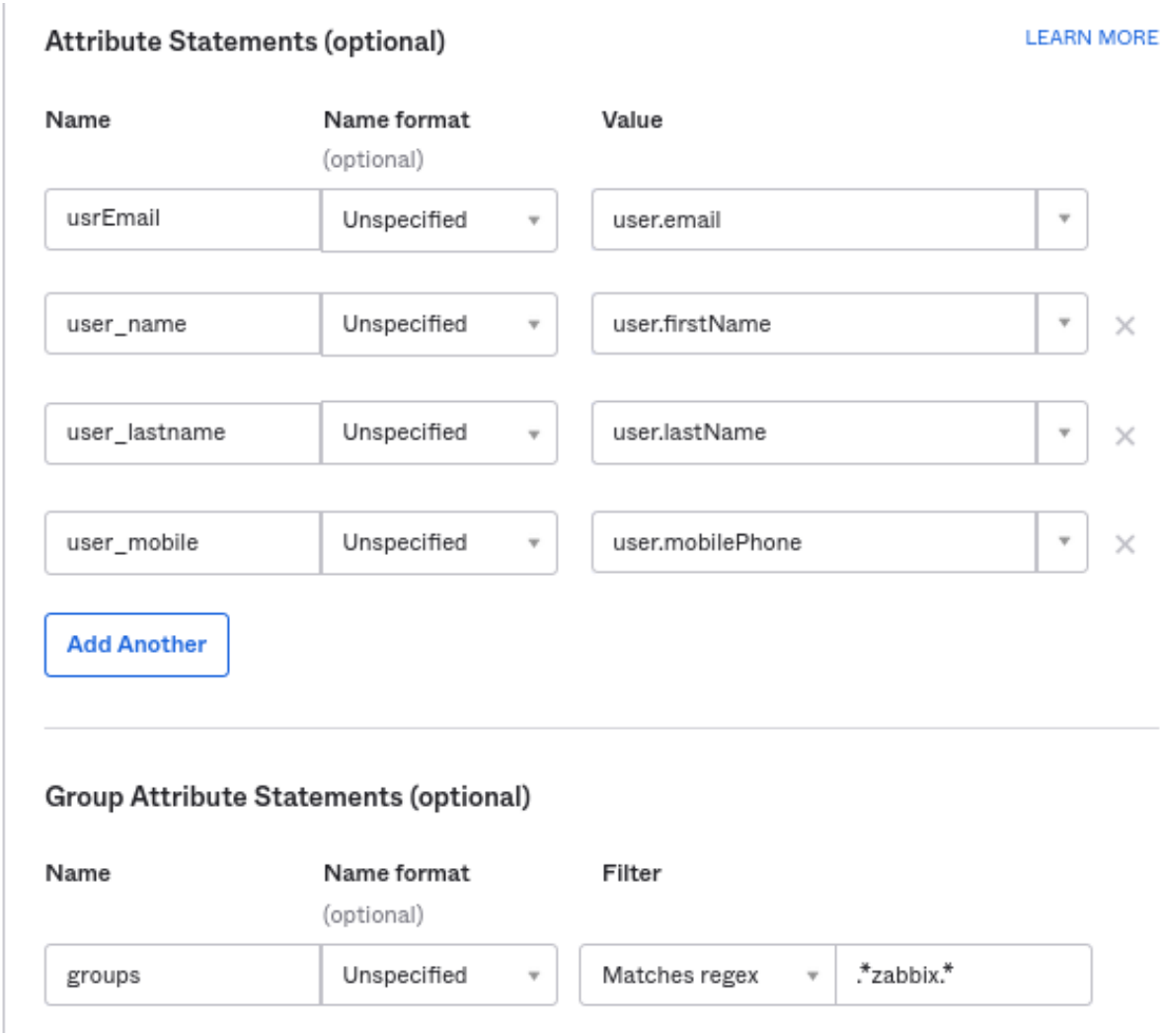


Select ”SAML 2.0” as the sign-in method and click on *Next*.

4. In general settings, fill in the app name and click on *Next*.
5. In SAML configuration, enter the values provided below, then click on *Next*.



- In **General** add:
  - *Single sign-on URL*: `http://<your-zabbix-url>/zabbix/index_sso.php?acs`
    Note the use of ”http”, and not ”https”, so that the `acs` parameter is not cut out in the request. The *Use this for Recipient URL and Destination URL* checkbox should also be marked.
  - *Audience URI (SP Entity ID)*: `zabbix`
    Note that this value will be used within the SAML assertion as a unique service provider identifier (if not matching, the operation will be rejected). It is possible to specify a URL or any string of data in this field.
  - *Default RelayState*:
    Leave this field blank; if a custom redirect is required, it can be added in Zabbix in the *Users* → *Users* settings.
  - Fill in other fields according to your preferences.
- In **Attribute Statements/Group Attribute Statements** add:



These attribute statements are inserted into the SAML assertions shared with Zabbix.

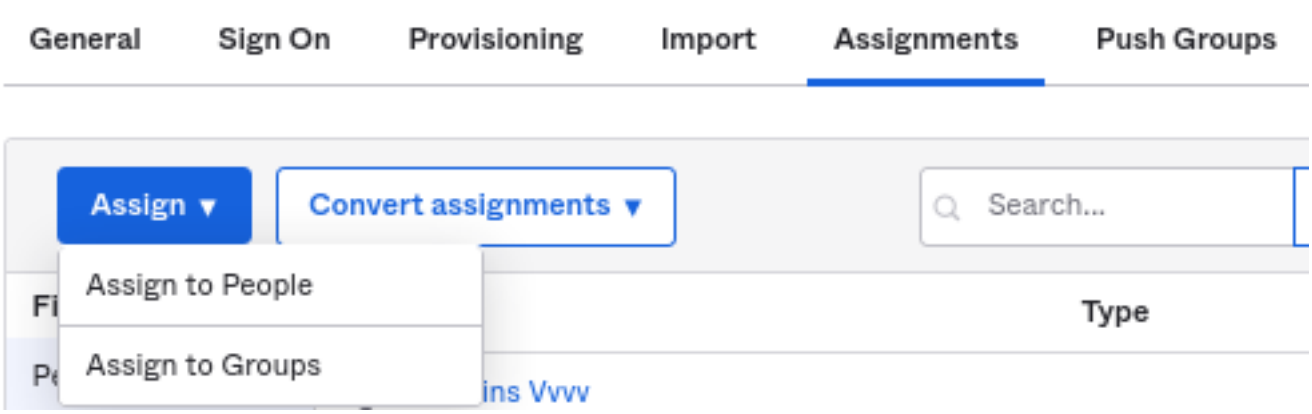
The attribute names used here are arbitrary examples. You may use different attribute names, however, it is required that they match the respective field value in Zabbix SAML settings.

If you want to configure SAML sign-in into Zabbix *without* JIT user provisioning, then only the email attribute is required.

Note:
If planning to use an encrypted connection, generate the private and public encryption certificates, then upload the public certificate to Okta. The certificate upload form appears when *Assertion Encryption* is set to "Encrypted" (click *Show Advanced Settings* to find this parameter).

6. In the next tab, select "I'm a software vendor. I'd like to integrate my app with Okta" and press "Finish".

7. Navigate to the "Assignments" tab of the newly-created application and click on the *Assign* button, then select "Assign to People" from the drop-down.



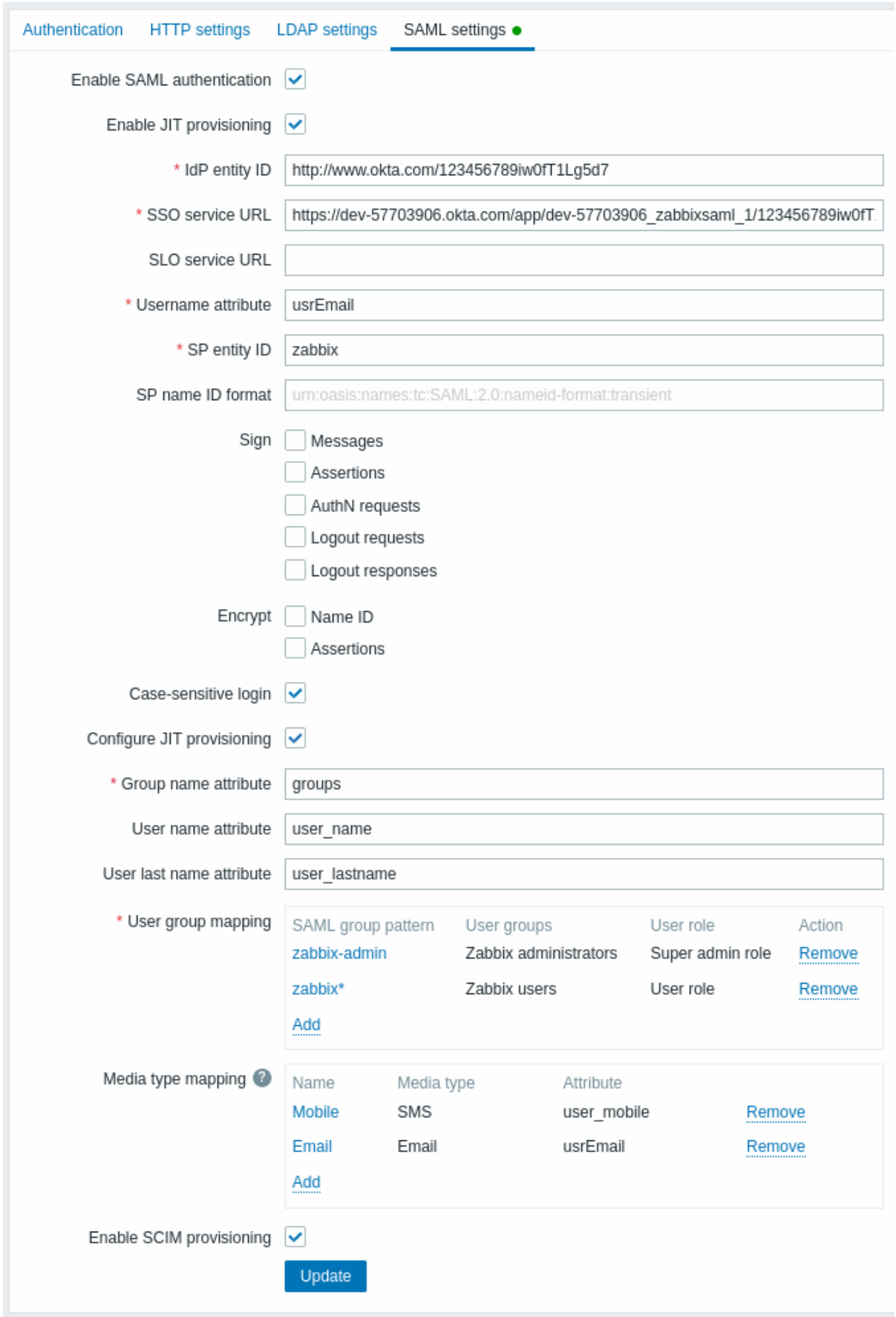
8. In a popup that appears, assign the app to people that will use SAML 2.0 to authenticate with Zabbix, then click on *Save and go back*.

9. Navigate to the ”Sign On” tab and click on the *View Setup Instructions* button.

Setup **instructions** will be opened in a new tab; keep this tab open while configuring Zabbix.

Zabbix configuration

1. In Zabbix, go to the SAML settings and fill the configuration options based on setup instructions from Okta:



Zabbix field	Setup field in Okta	Sample value
IdP entity ID	Identity Provider Issuer	
SSO service URL	Identity Provider Single Sign-On URL	
Username attribute	Attribute name	`usrEmail`
SP entity ID	Audience URI	`zabbix`
Group name attribute	Attribute name	`groups`

Zabbix field	Setup field in Okta	Sample value
User name attribute	Attribute name	`user_name`
User last name attribute	Attribute name	`user_lastname`

It is also required to configure user group and media mapping.

2. Download the certificate provided in the Okta SAML setup instructions into *ui/conf/certs* folder as idp.crt.

Set 644 permissions to it by running:

```
chmod 644 idp.crt
```

3. If *Assertion Encryption* has been set to "Encrypted" in Okta, the "Assertions" checkbox of the *Encrypt* parameter should be marked in Zabbix as well.

4. Press the "Update" button to save these settings.

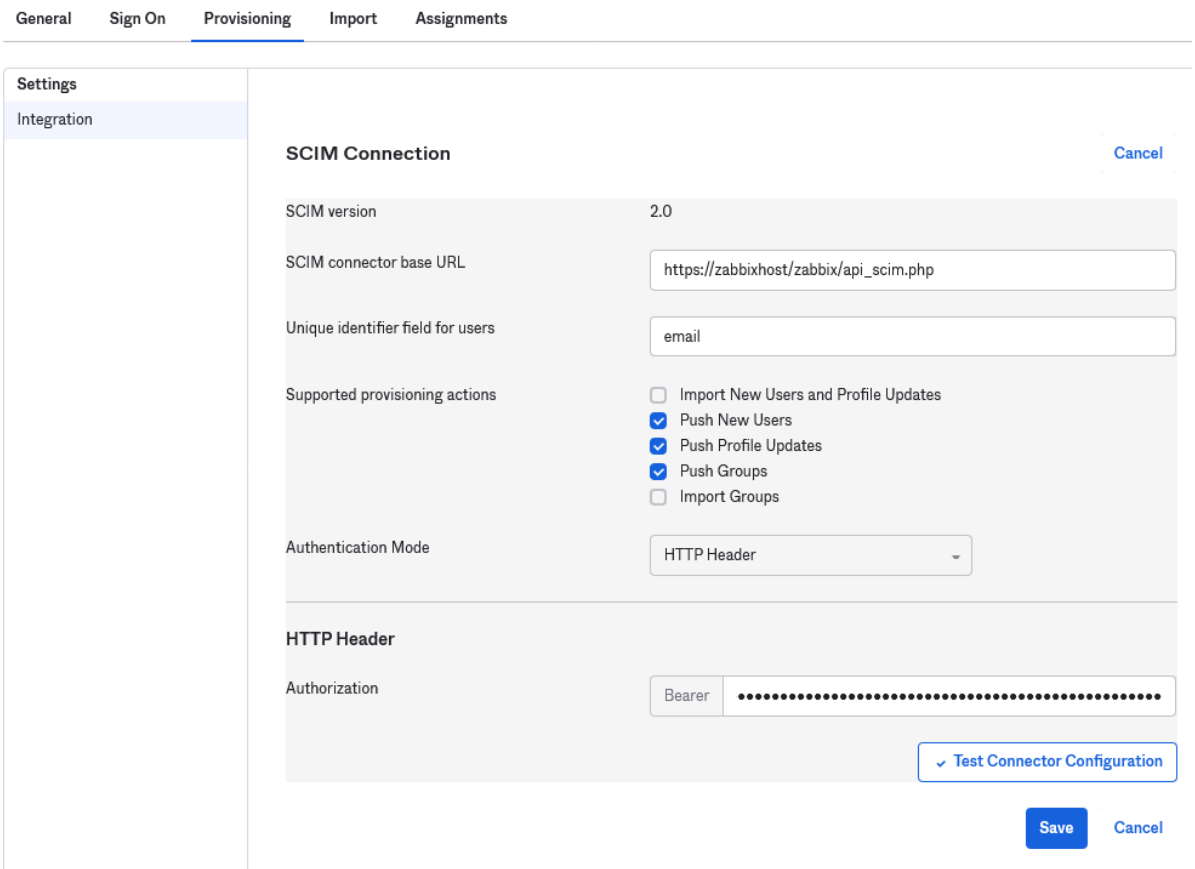
SCIM provisioning

1. To turn on SCIM provisioning, go to "General" -> "App Settings" of the application in Okta.

Mark the *Enable SCIM provisioning* checkbox. As a result, a new *Provisioning* tab appears.

2. Go to the "Provisioning" tab to set up a SCIM connection:

- In *SCIM connector base URL* specify the path to the Zabbix frontend and append `api_scim.php` to it, i.e.: `https://<your-zabbix-url>/zabbix/api_scim.php`
- *Unique identifier field for users*: `email`
- *Authentication mode*: `HTTP header`
- In *Authorization* enter a valid API token with Super admin rights



Attention:
If you are using Apache, you may need to change the default Apache configuration in `/etc/apache2/apache2.conf` by adding the following line:


```
SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
```

Otherwise, Apache might not send the Authorization header in the request.

3. Click on *Test Connector Configuration* to test the connection. If all is correct a success message will be displayed.

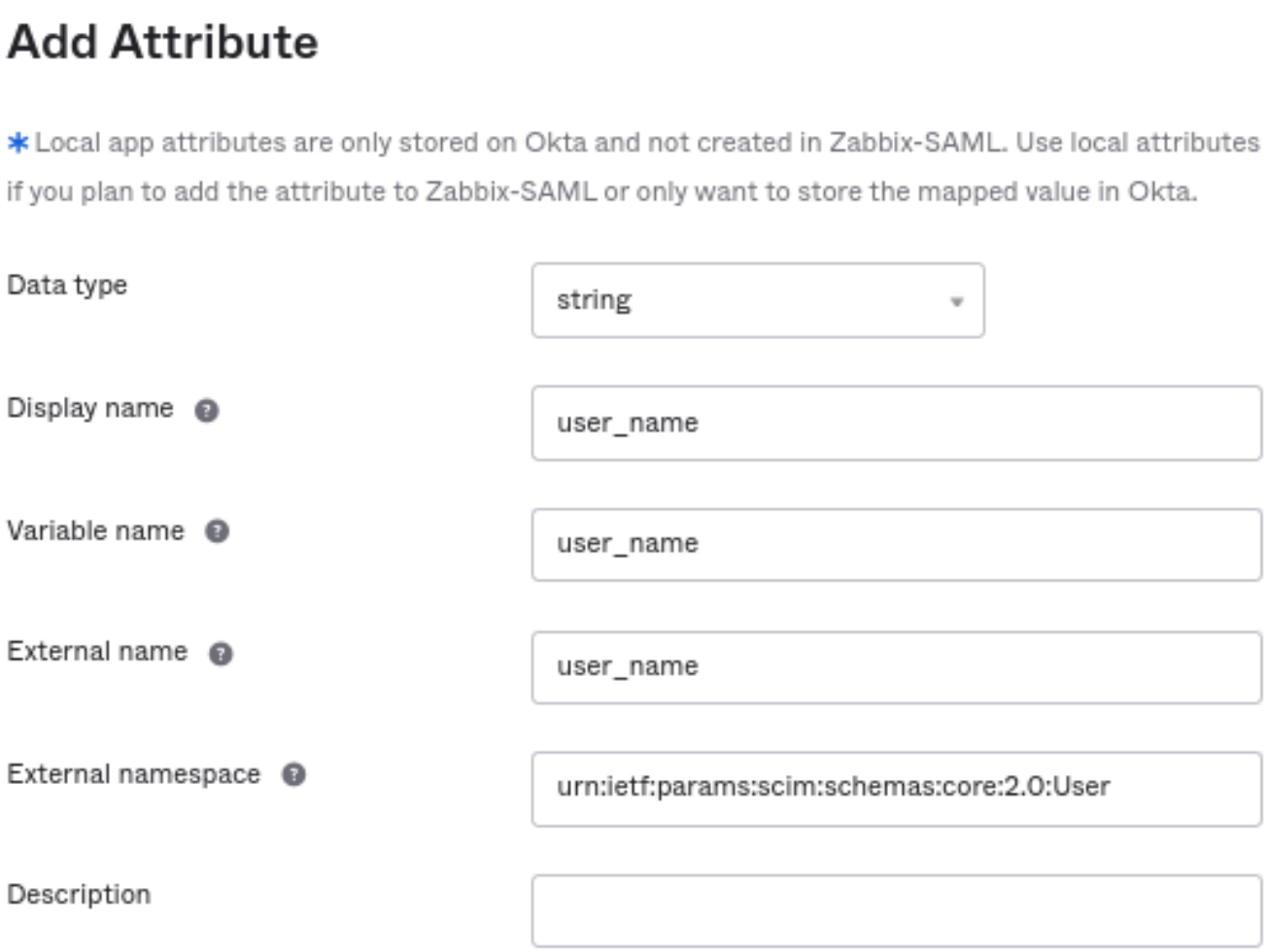
4. In ”Provisioning” -> ”To App”, make sure to mark the following checkboxes:

- Create Users
- Update User Attributes
- Deactivate Users

This will make sure that these request types will be sent to Zabbix.

5. Make sure that all attributes defined in SAML are defined in SCIM. You can access the profile editor for your app in ”Provisioning” -> ”To App”, by clicking on *Go to Profile Editor*.

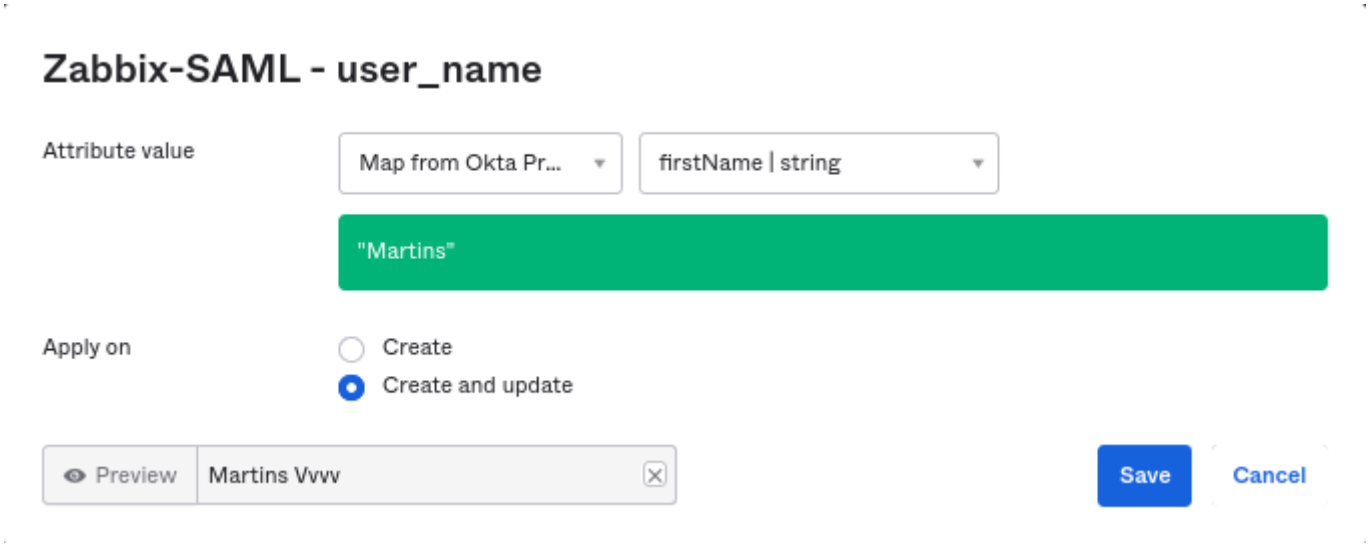
Click on *Add Attribute*. Fill the values for *Display name*, *Variable name*, *External name* with the SAML attribute name, for example, `user_name`.



External namespace should be the same as user schema: `urn:ietf:params:scim:schemas:core:2.0:User`

6. Go to ”Provisioning” -> ”To App” -> ”Attribute Mappings” of your application. Click on *Show Unmapped Attributes* at the bottom. Newly added attributes appear.

7. Map each added attribute.



8. Add users in the ”Assignments” tab. The users previously need to be added in *Directory -> People*. All these assignments will be sent as requests to Zabbix.

9. Add groups in the ”Push Groups” tab. The user group mapping pattern in Zabbix SAML settings must match a group specified here. If there is no match, the user cannot be created in Zabbix.

Information about group members is sent every time when some change is made.

12 SAML setup with OneLogin

Overview

This section provides guidelines for configuring single sign-on and user provisioning into Zabbix from OneLogin using SAML 2.0 authentication.

OneLogin configuration

Creating application

1. Log into your account at OneLogin. For testing purposes, you may create a free developer account in OneLogin.

2. In the OneLogin web interface navigate to *Applications → Applications*.

3. Click on ”Add App” and search for the appropriate app. The guidelines in this page are based on the *SCIM Provisioner with SAML (SCIM v2 Enterprise, full SAML)* app example.

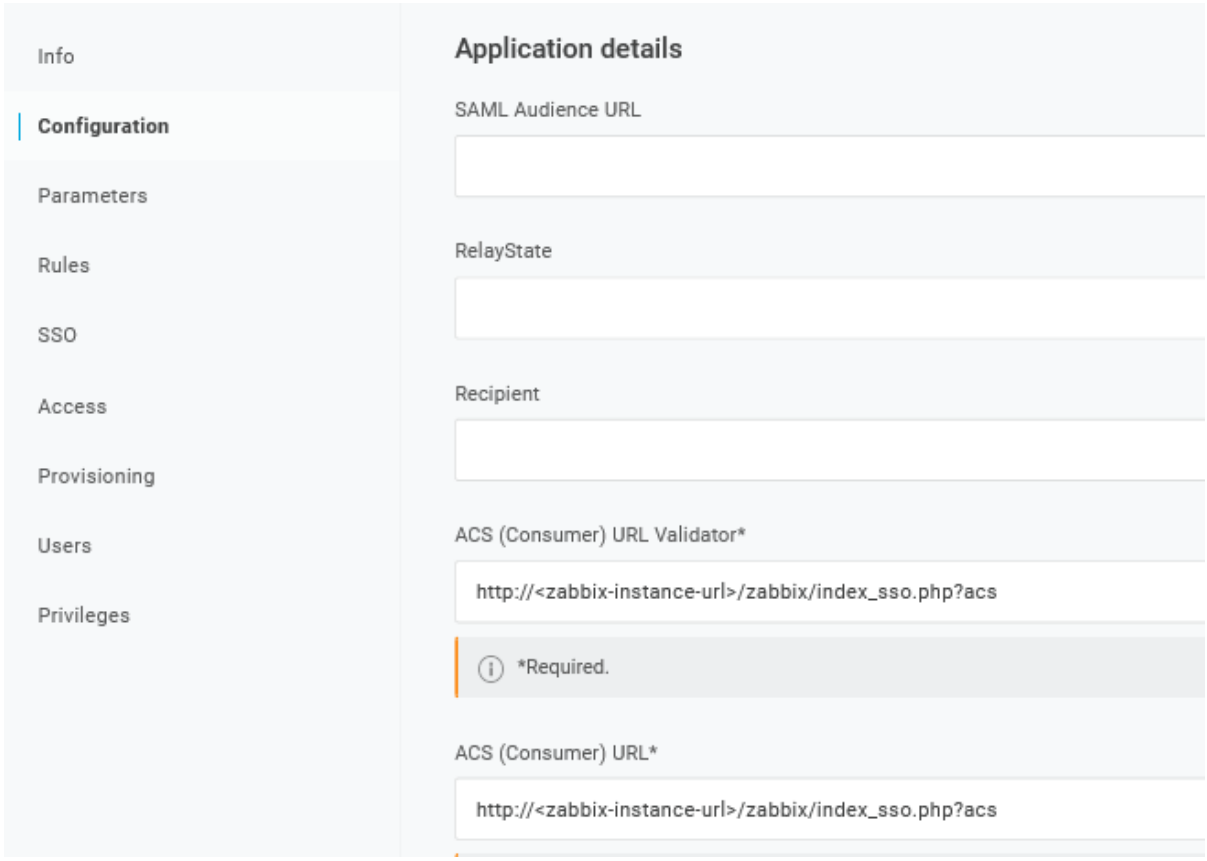
4. To begin with, you may want to customize the display name of your app. You may also want to add the icon and app details. After that, click on *Save*.

Setting up SSO/SCIM provisioning

1. In *Configuration -> Application details*, set the Zabbix single sign-on endpoint `http://<zabbix-instance-url>/zabbix/index_sso.p` as the value of these fields:

- *ACS (Consumer) URL Validator*
- *ACS (Consumer) URL*

Note the use of ”http”, and not ”https”, so that the `acs` parameter is not cut out in the request.



It is also possible to use "https". To make that work with Zabbix, it is necessary to add to `conf/zabbix.conf.php` the following line:

```
$SSO['SETTINGS'] = ['use_proxy_headers' => true];
```

Leave other options with their default values.

2. In *Configuration -> API connection*, set the following values:

- *SCIM Base URL*: `https://<zabbix-instance-url>/zabbix/api_scim.php`
- *SCIM JSON Template*: should contain all custom attributes that you would like to pass to Zabbix via SCIM such as `user_name`, `user_lastname`, `user_email`, and `user_mobile`:

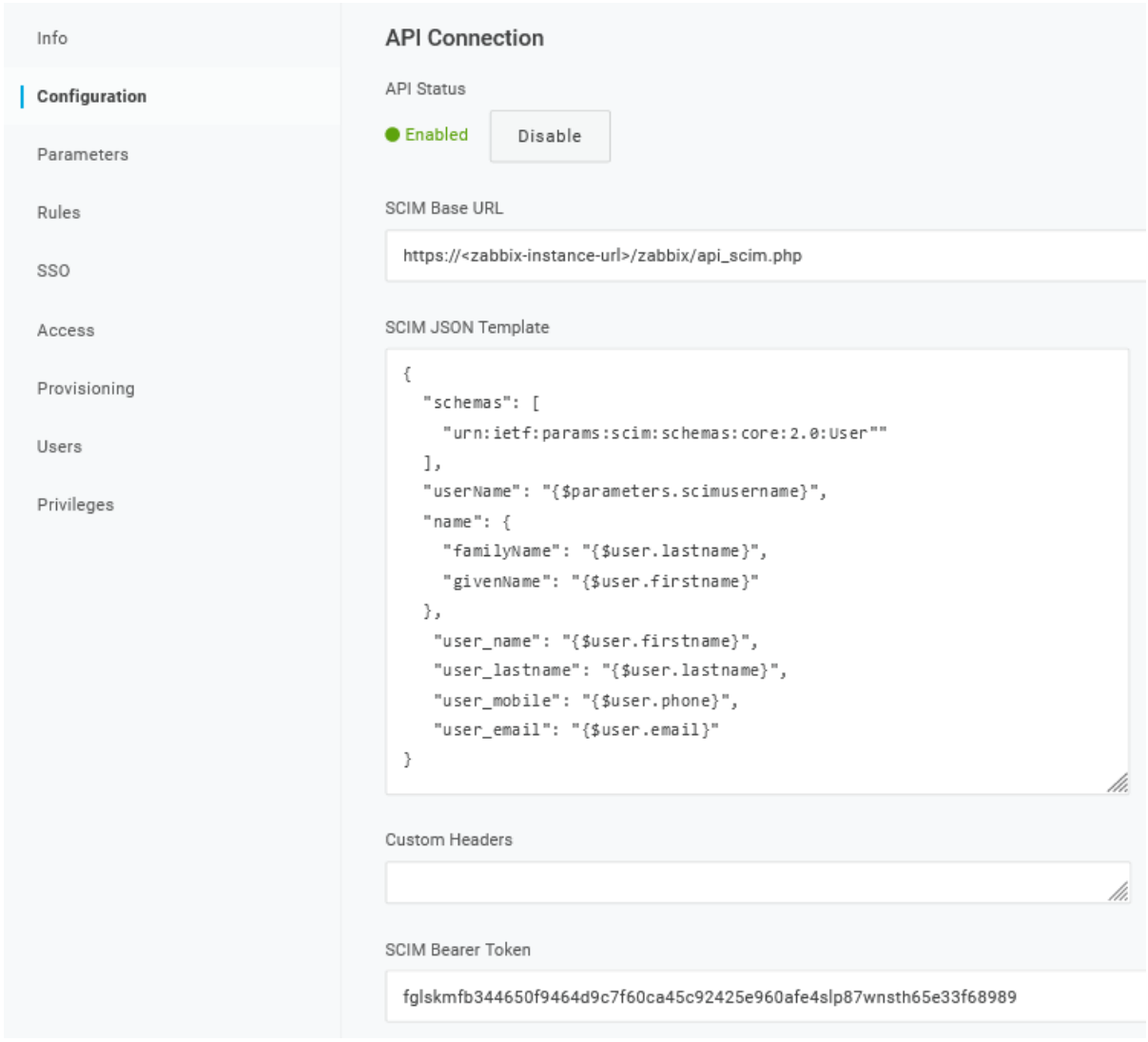
```
{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:User"
  ],
  "userName": "{$parameters.scimusername}",
  "name": {
    "familyName": "{$user.lastname}",
    "givenName": "{$user.firstname}"
  },
   "user_name": "{$user.firstname}",
   "user_lastname": "{$user.lastname}",
   "user_mobile": "{$user.phone}",
   "user_email": "{$user.email}"
}
```

The attribute names are arbitrary. Different attribute names may be used, however, it is required that they match the respective field value in Zabbix SAML settings.

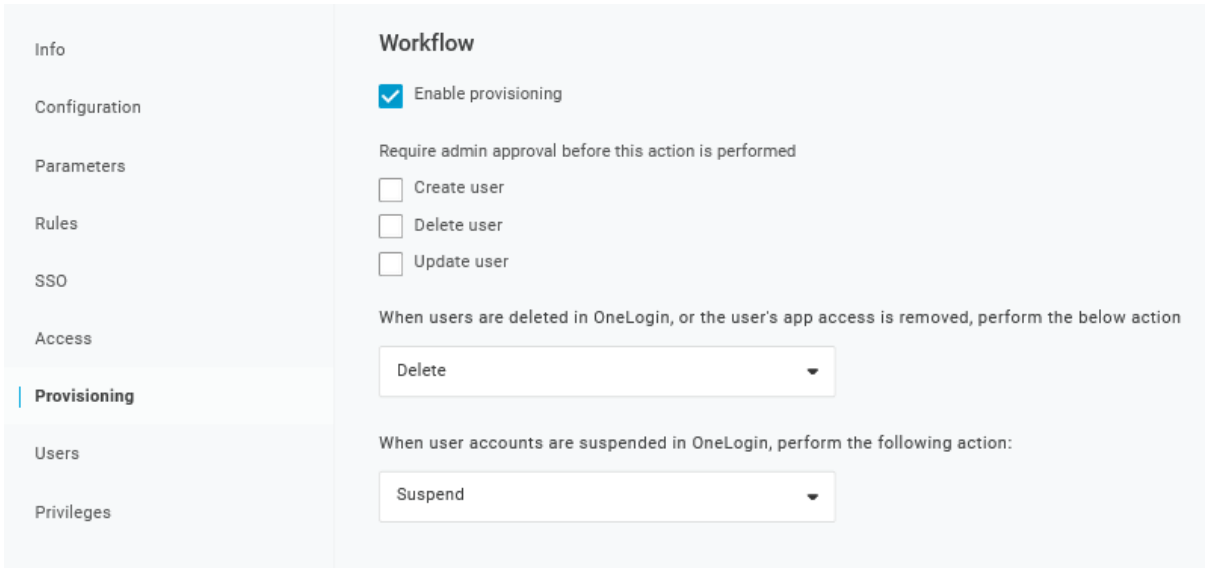
Note that for user provisioning to work, OneLogin needs to receive in response a 'name' attribute with 'givenName' and 'familyName', even if it was not required by the service provider. Thus it is necessary to specify this in the schema in the application configuration part.

- *SCIM Bearer Token*: enter a Zabbix API token with Super admin permissions.

Click on *Enable* to activate the connection.

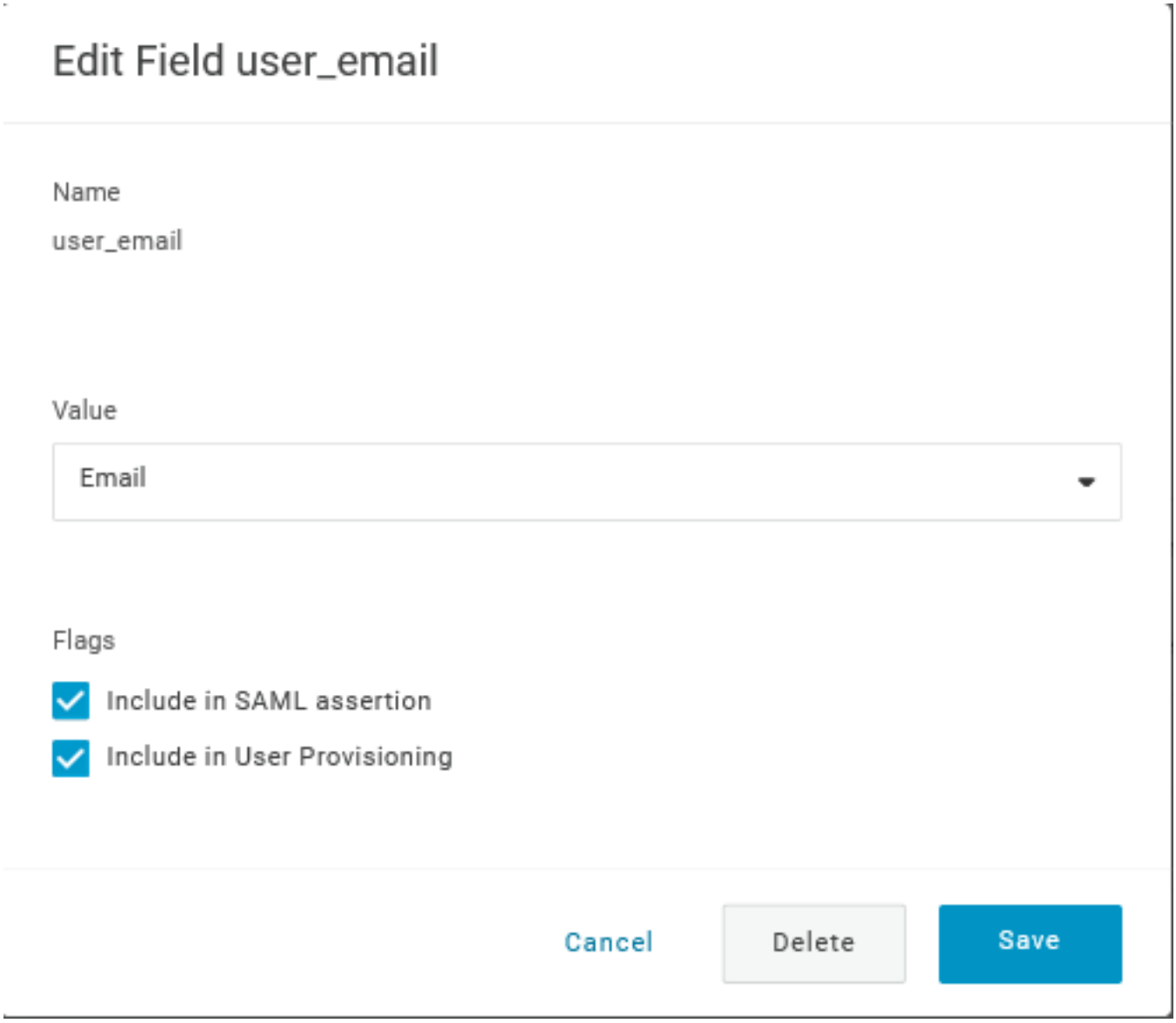


3. In the *Provisioning* page, enable the Provisioning option:



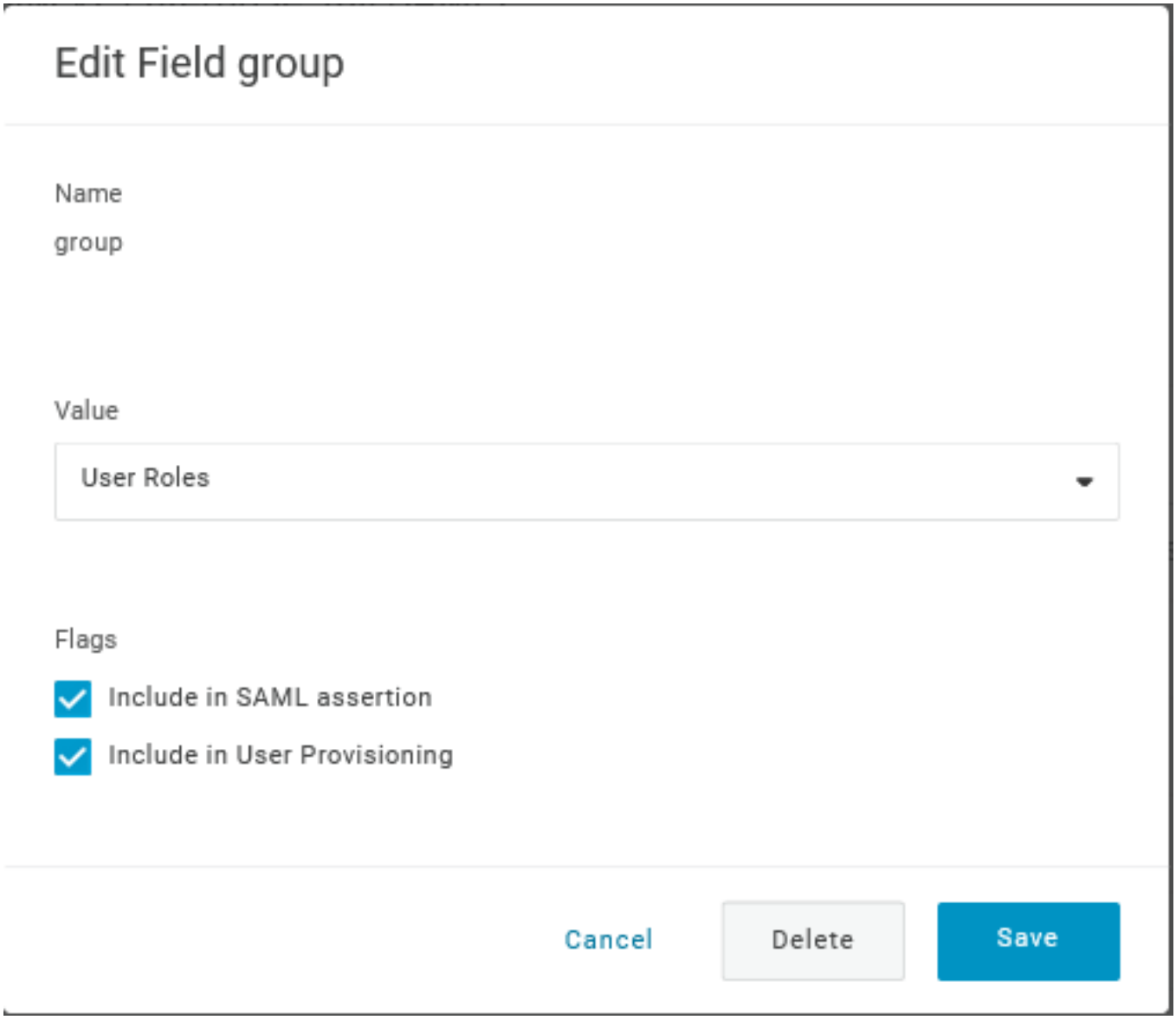
4. The *Parameters* page contains a list of default parameters:

- Make sure that the 'scimusername' matches the user login value in OneLogin (e.g. email);
- Mark the *Include in User Provisioning* option for the 'Groups' parameter;
- Click on "+" to create the custom parameters that are required for SAML assertions and user provisioning such as `user_name`, `user_lastname`, `user_email`, and `user_mobile`:

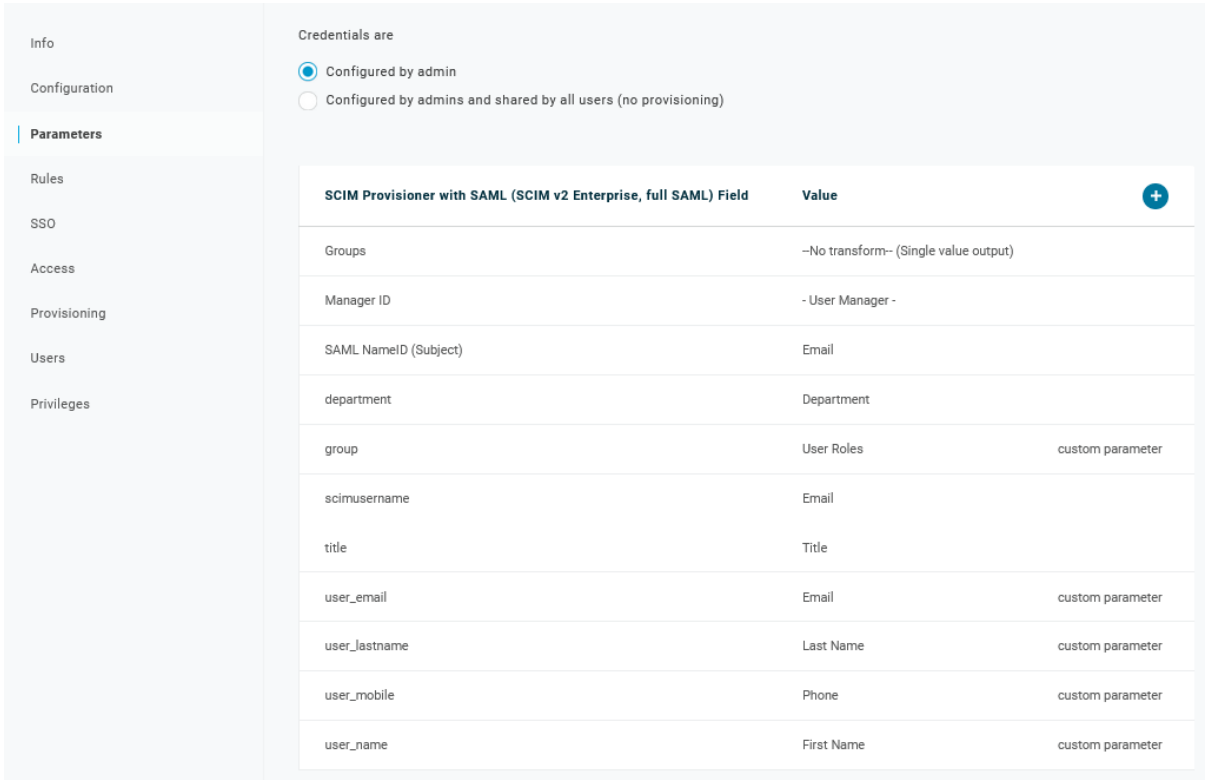


When adding a parameter, make sure to mark both the *Include in SAML assertion* and *Include in User Provisioning* options.

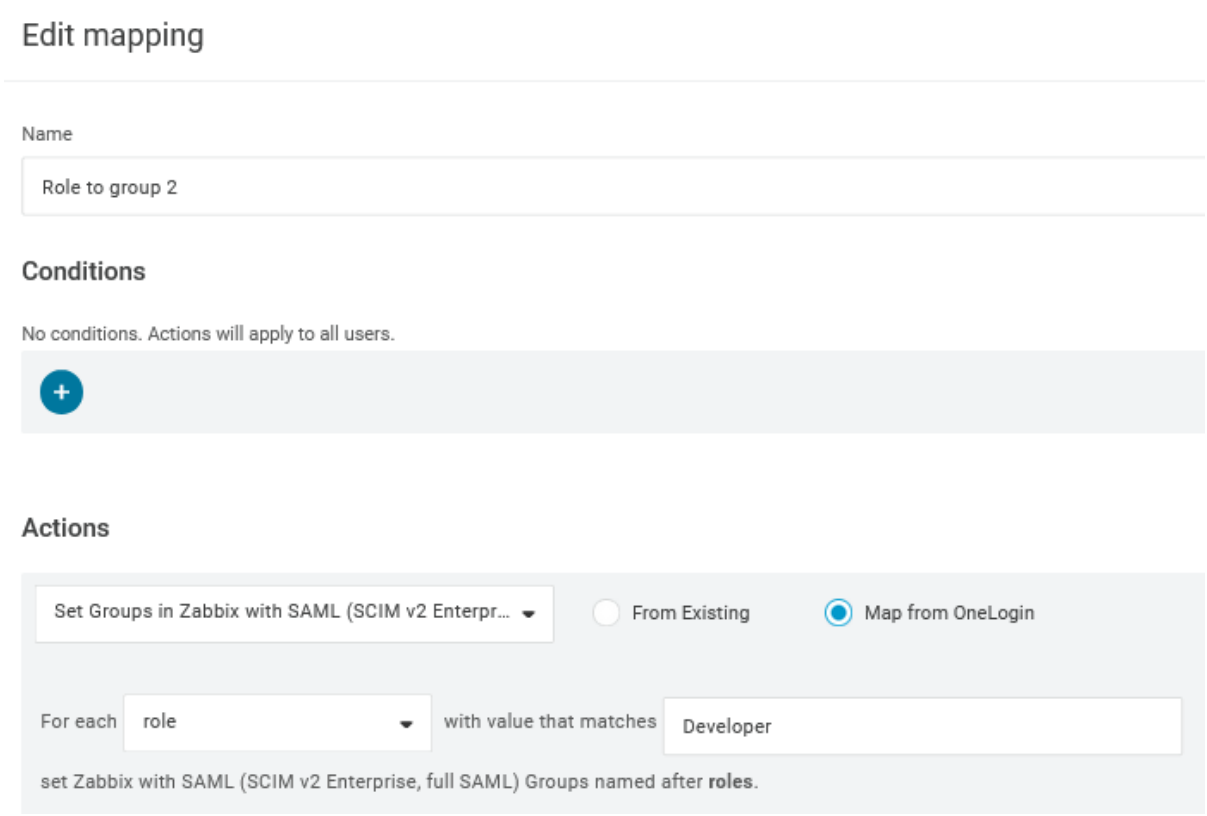
- Add a 'group' parameter that matches user roles in OneLogin. User roles will be passed as a string, separated by a semicolon `;`. The OneLogin user roles will be the used for creating user groups in Zabbix:



Verify the list of parameters:



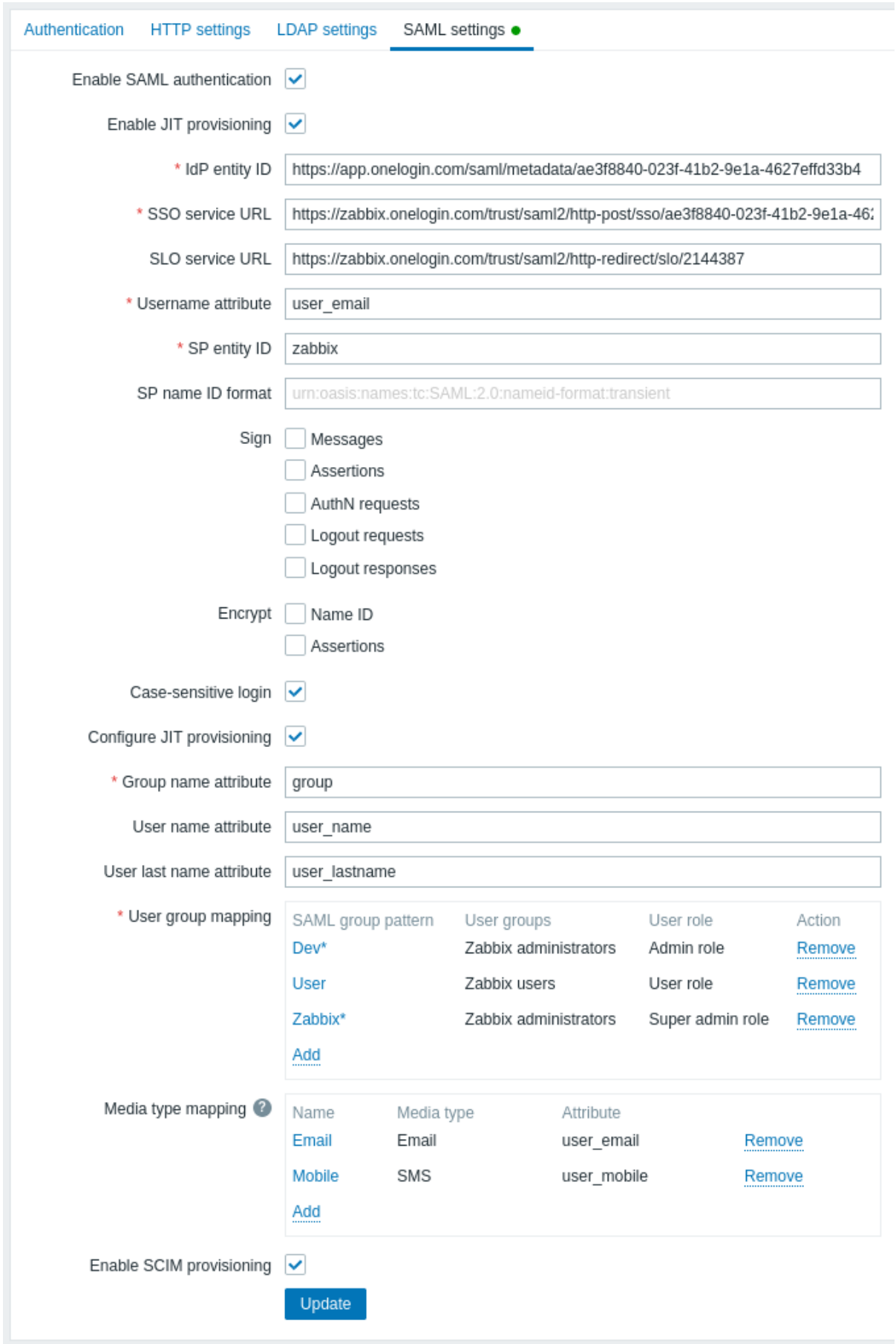
5. In the *Rules* page, create user role mappings to the default Groups parameter.



You may use a regular expression to pass specific roles as groups. The role names should not contain ; as OneLogin uses it as a separator when sending an attribute with several roles.

Zabbix configuration

1. In Zabbix, go to the SAML settings and fill the configuration options based on the OneLogin configuration:



Zabbix field	Setup field in OneLogin	Sample value
IdP entity ID	Issuer URL (see SSO tab of your application in OneLogin)	
SSO service URL	SAML 2.0 Endpoint (HTTP) (see SSO tab of your application in OneLogin)	

Zabbix field	Setup field in OneLogin	Sample value
SLO service URL	SLO Endpoint (HTTP) (see SSO tab of your application in OneLogin)	
Username attribute	Custom parameter	`user_email`
Group name attribute	Custom parameter	`group`
User name attribute	Custom parameter	`user_name`
User last name attribute	Custom parameter	`user_lastname`

It is also required to configure user group mapping. Media mapping is optional. Click on *Update* to save these settings.

2. Download the certificate provided by OneLogin and place it into `conf/certs` of the Zabbix frontend installation, as idp.crt.

Set 644 permissions to it by running:

```
chmod 644 idp.crt
```

You can access the certificate download in OneLogin in *Applications* -> *SSO* -> click on *View details* under the current certificate.

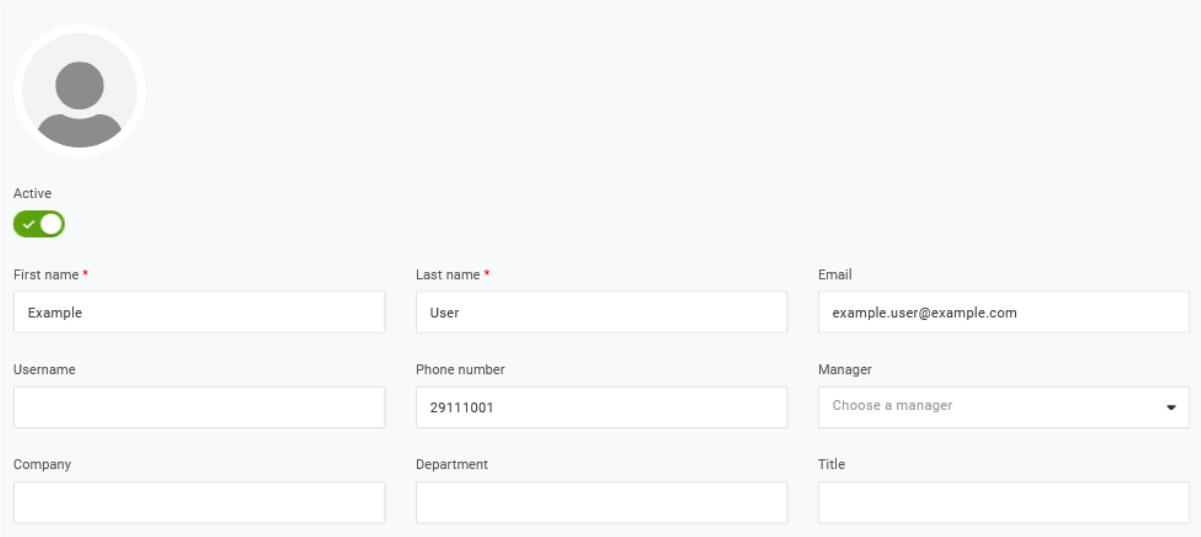
It is possible to use a different certificate name and location. In that case, make sure to add to `conf/zabbix.conf.php` the following line:

```
$SSO['IDP_CERT'] = 'path/to/certname.crt';
```

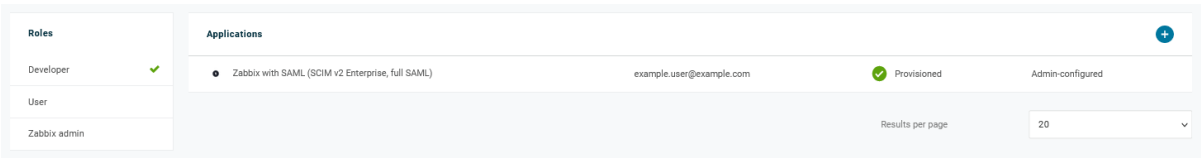
SCIM user provisioning

With user provisioning enabled, it is now possible to add/update users and their roles in OneLogin and have them immediately provisioned to Zabbix.

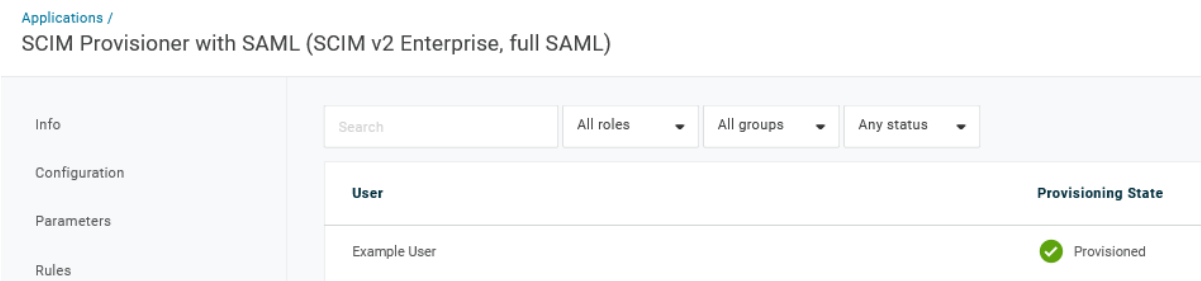
For example, you may create a new user:



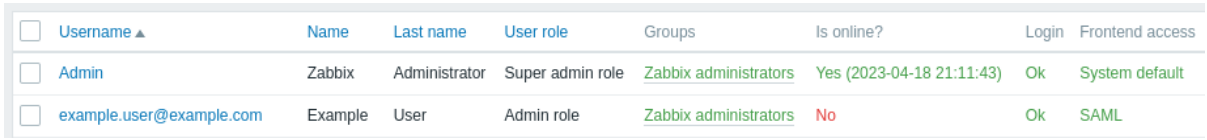
Add it to a user role and the application that will provision the user:



When saving the user, it will be provisioned to Zabbix. In Application -> Users you can check the provisioning status of current application users:



If successfully provisioned, the user can be seen in the Zabbix user list.



Username ▲	Name	Last name	User role	Groups	Is online?	Login	Frontend access
Admin	Zabbix	Administrator	Super admin role	Zabbix administrators	Yes (2023-04-18 21:11:43)	Ok	System default
example.user@example.com	Example	User	Admin role	Zabbix administrators	No	Ok	SAML

13 Oracle database setup

Overview

This section contains instructions for creating Oracle database and configuring connections between the database and Zabbix server, proxy, and frontend.

Database creation

We assume that a *zabbix* database user with *password* password exists and has permissions to create database objects in ORCL service located on the *host* Oracle database server. Zabbix requires a Unicode database character set and a UTF8 national character set. Check current settings:

```
sqlplus> select parameter,value from v$nls_parameters where parameter='NLS_CHARACTERSET' or parameter='NLS
```

Now prepare the database:

```
shell> cd /path/to/zabbix-sources/database/oracle
shell> sqlplus zabbix/password@oracle_host/ORCL
sqlplus> @schema.sql
# stop here if you are creating database for Zabbix proxy
sqlplus> @images.sql
sqlplus> @data.sql
```

Note:
Please set the initialization parameter CURSOR_SHARING=FORCE for best performance.

Connection set up

Zabbix supports two types of connect identifiers (connection methods):

- Easy Connect
- Net Service Name

Connection configuration parameters for Zabbix server and Zabbix proxy can be set in the configuration files. Important parameters for the server and proxy are *DBHost*, *DBUser*, *DBName* and *DBPassword*. The same parameters are important for the frontend: *$DB["SERVER"]*, *$DB["PORT"]*, *$DB["DATABASE"]*, *$DB["USER"]*, *$DB["PASSWORD"]*.

Zabbix uses the following connection string syntax:

```
{DBUser/DBPassword[@<connect_identifier>]}
```

<connect_identifier> can be specified either in the form of "Net Service Name" or "Easy Connect".

```
@[[//]Host[:Port]/<service_name> | <net_service_name>]
```

Easy Connect

Easy Connect uses the following parameters to connect to the database:

- *Host* - the host name or IP address of the database server computer (DBHost parameter in the configuration file).
- *Port* - the listening port on the database server (DBPort parameter in the configuration file; if not set the default 1521 port will be used).
- <service_name> - the service name of the database you want to access (DBName parameter in the configuration file).

Example:

Database parameters set in the server or proxy configuration file (zabbix_server.conf and zabbix_proxy.conf):

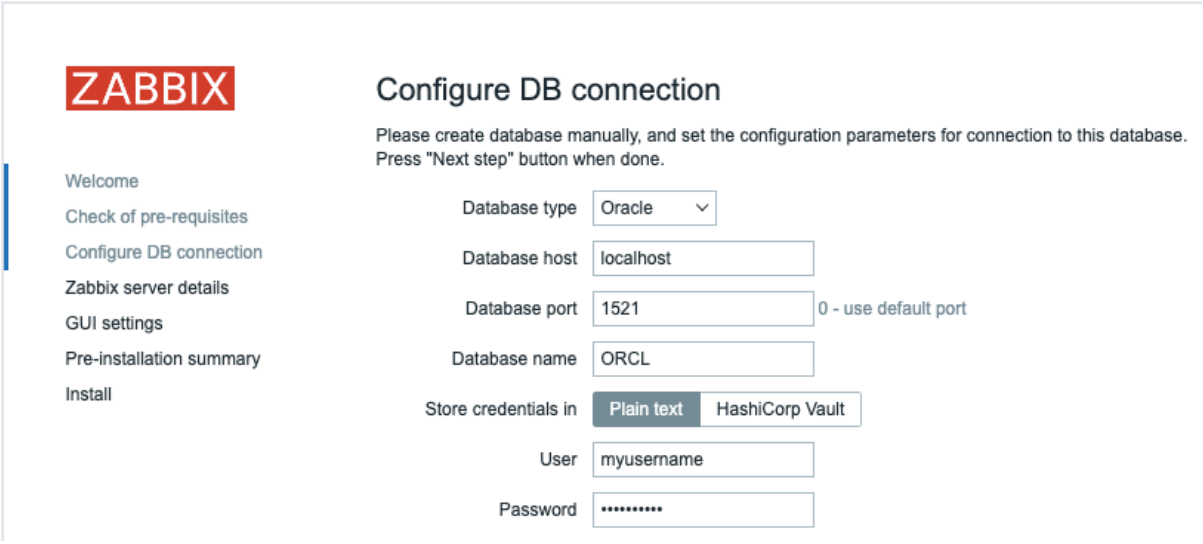
```
DBHost=localhost
DBPort=1521
DBUser=myusername
DBName=ORCL
DBPassword=mypassword
```

Connection string used by Zabbix to establish connection:

```
DBUser/DBPassword@DBHost:DBPort/DBName
```

During Zabbix frontend installation, set the corresponding parameters in the *Configure DB connection* step of the setup wizard:

- Database host: localhost
- Database port: 1521
- Database name: ORCL
- User: myusername
- Password: mypassword



Alternatively, these parameters can be set in the frontend configuration file (zabbix.conf.php):

```
$DB["TYPE"]                     = 'ORACLE';
$DB["SERVER"]                   = 'localhost';
$DB["PORT"]             = '1521';
$DB["DATABASE"]                 = 'ORCL';
$DB["USER"]                     = 'myusername';
$DB["PASSWORD"]                 = 'mypassword';
```

Net service name

Since Zabbix 5.4.0 it is possible to connect to Oracle by using net service name.

<net_service_name> is a simple name for a service that resolves to a connect descriptor.

In order to use the service name for creating a connection, this service name has to be defined in the tnsnames.ora file located on both the database server and the client systems. The easiest way to make sure that the connection will succeed is to define the location of tnsnames.ora file in the TNS_ADMIN environment variable. The default location of the tnsnames.ora file is:

```
$ORACLE_HOME/network/admin/
```

A simple tnsnames.ora file example:

```
ORCL =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = ORCL)
    )
  )
```

To set configuration parameters for the ”Net Service Name” connection method, use one of the following options:

- Set an empty parameter DBHost and set DBName as usual:

```
DBHost=
DBName=ORCL
```

- Set both parameters and leave both empty:

```
DBHost=
DBName=
```

In the second case, the TWO_TASK environment variable has to be set. It specifies the default remote Oracle service (service name). When this variable is defined, the connector connects to the specified database by using an Oracle listener that accepts connection requests. This variable is for use on Linux and UNIX only. Use the LOCAL environment variable for Microsoft Windows.

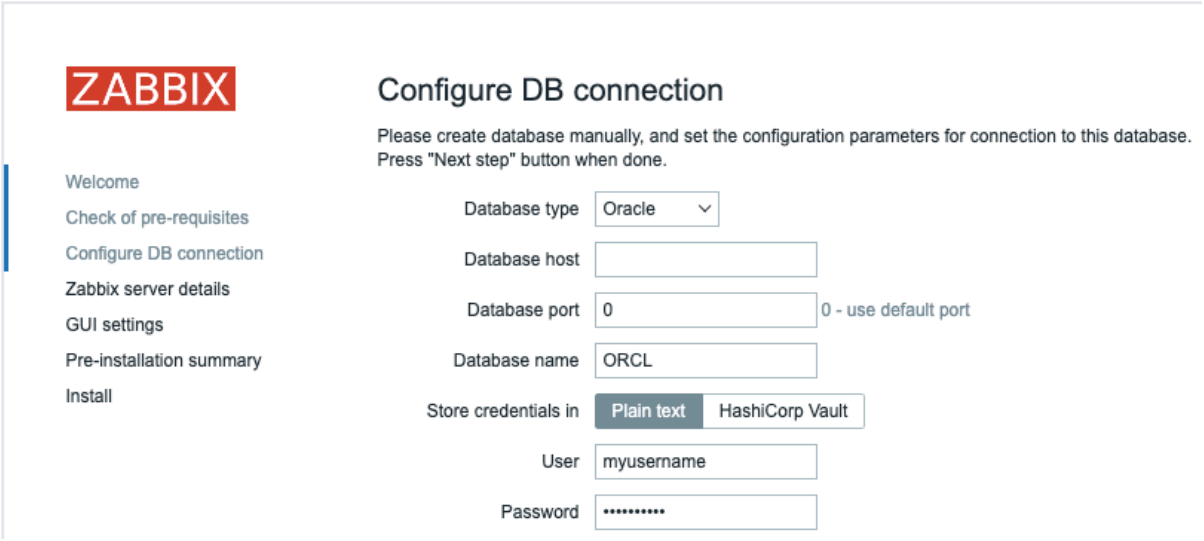
Example:

Connect to a database using Net Service Name set as ORCL and the default port. Database parameters set in the server or proxy configuration file (zabbix_server.conf and zabbix_proxy.conf):

```
DBHost=
#DBPort=
DBUser=myusername
DBName=ORCL
DBPassword=mypassword
```

During Zabbix frontend installation, set the corresponding parameters in the *Configure DB connection* step of the setup wizard:

- Database host:
- Database port: 0
- Database name: ORCL
- User: myusername
- Password: mypassword



Alternatively, these parameters can be set in the frontend configuration file (zabbix.conf.php):

```
$DB["TYPE"]                     = 'ORACLE';
$DB["SERVER"]                   = '';
$DB["PORT"]             = '0';
$DB["DATABASE"]                 = 'ORCL';
$DB["USER"]                     = 'myusername';
$DB["PASSWORD"]                 = 'mypassword';
```

Connection string used by Zabbix to establish connection:

```
DBUser/DBPassword@ORCL
```

Known issues

To improve performance, you can convert the field types from *nclob* to *nvarchar2*, see known issues.

14 Setting up scheduled reports

Overview

This section provides instructions on installing Zabbix web service and configuring Zabbix to enable generation of scheduled reports.

Attention:
Currently the support of scheduled reports is experimental.

Installation

A new Zabbix web service process and Google Chrome browser should be installed to enable generation of scheduled reports. The web service may be installed on the same machine where the Zabbix server is installed or on a different machine. Google Chrome browser should be installed on the same machine, where the web service is installed.

The official zabbix-web-service package is available in the Zabbix repository. Google Chrome browser is not included into these packages and has to be installed separately.

To compile Zabbix web service from sources, see Installing Zabbix web service.

After the installation, run zabbix_web_service on the machine, where the web service is installed:

```
shell> zabbix_web_service
```

Configuration

To ensure proper communication between all elements involved make sure server configuration file and frontend configuration parameters are properly configured.

Zabbix server

The following parameters in Zabbix server configuration file need to be updated: *WebServiceURL* and *StartReportWriters*.

WebServiceURL

This parameter is required to enable communication with the web service. The URL should be in the format `<host:port>/report`.

- By default, the web service listens on port 10053. A different port can be specified in the web service configuration file.
- Specifying the `/report` path is mandatory (the path is hardcoded and cannot be changed).

Example:

```
WebServiceURL=http://localhost:10053/report
```

Attention:
It is strongly recommended to set up encryption between Zabbix server and Zabbix web service using certificates. By default, data transmitted between Zabbix server and Zabbix web service is not encrypted, which can lead to unauthorized access.

StartReportWriters

This parameter determines how many report writer processes should be started. If it is not set or equals 0, report generation is disabled. Based on the number and frequency of reports required, it is possible to enable from 1 to 100 report writer processes.

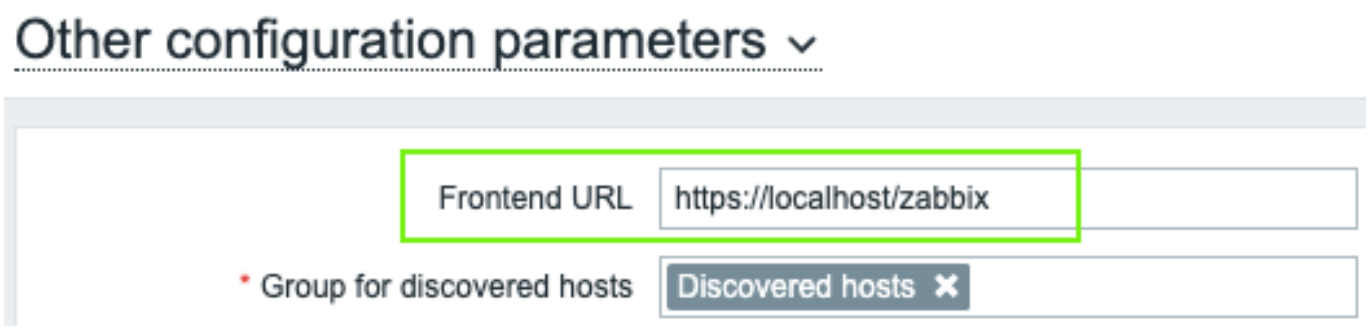
Example:

```
StartReportWriters=3
```

Zabbix frontend

A *Frontend URL* parameter should be set to enable communication between Zabbix frontend and Zabbix web service:

- Proceed to the *Administration* → *General* → *Other parameters* frontend menu section
- Specify the full URL of the Zabbix web interface in the *Frontend URL* parameter



Note:
Once the setup procedure is completed, you may want to configure and send a test report to make sure everything works correctly.

15 Upgrading to numeric values of extended range

Since Zabbix 5.0.0, numeric (float) data type supports precision of approximately 15 digits and range from approximately -1.79E+308 to 1.79E+308. This is implemented by default in new installations. However, when upgrading existing installations, created before Zabbix 5.0, a manual database upgrade patch must be applied.

If you do not apply the patch, System information in the frontend will display: "Database history tables upgraded: No. Support for the old numeric type is deprecated. Please upgrade to numeric values of extended range".

Attention:
The patch will alter data columns of history and trends tables, which usually contain lots of data, therefore it is expected to take some time to complete. Since the exact estimate depends on server performance, database management system configuration and version, and it cannot be predicted, it is recommended to first test the patch outside the production environment, even though with MySQL 8.0 and MariaDB 10.5 configured by default the patch is known to be executed instantly for large tables due to efficient algorithm and the fact that previously the same double type was used but with limited precision, meaning that data itself does not need to be modified.

Please execute the appropriate patch (SQL file) for your database; you may find these scripts in the Zabbix Git repository for:

- MySQL
- PostgreSQL
- Oracle

Warning:
Important! Run these scripts for the server database only.

To apply a patch:

- Stop Zabbix server.
- Run the script for your database.
- Start Zabbix server again.

Note that with TimescaleDB the compression support must only be turned on after applying this patch.

Note:
After upgrading database tables, please also set or update $DB['DOUBLE_IEEE754'] value to true in /ui/conf/zabbix.conf.php.

16 Additional frontend languages

Overview

In order to use any other language than English in Zabbix web interface, its locale should be installed on the web server. Additionally, the PHP gettext extension is required for the translations to work.

Installing locales

To list all installed languages, run:

```
locale -a
```

If some languages that are needed are not listed, open the */etc/locale.gen* file and uncomment the required locales. Since Zabbix uses UTF-8 encoding, you need to select locales with UTF-8 charset.

Now run:

```
locale-gen
```

Restart the web server.

The locales should now be installed. It may be required to reload Zabbix frontend page in browser using Ctrl + F5 for new languages to appear.

Installing Zabbix

If installing Zabbix directly from Zabbix git repository, translation files should be generated manually. To generate translation files, run:

```
make gettext
locale/make_mo.sh
```

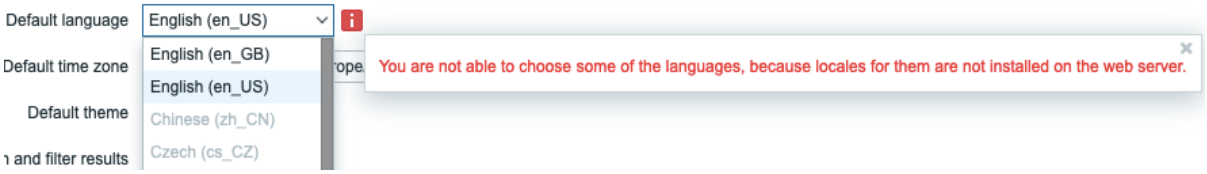
This step is not needed when installing Zabbix from packages or source tar.gz files.

Selecting a language

There are several ways to select a language in Zabbix web interface:

- When installing web interface - in the frontend installation wizard. Selected language will be set as system default.
- After the installation, system default language can be changed in the *Administration→General→GUI* menu section.
- Language for a particular user can be changed in the user profile.

If a locale for a language is not installed on the machine, this language will be greyed out in Zabbix language selector. A red icon is displayed next to the language selector if at least one locale is missing. Upon pressing on this icon the following message will be displayed: "You are not able to choose some of the languages, because locales for them are not installed on the web server."



2 Process configuration

Please use the sidebar to access content in this section.

1 Zabbix server

Overview

The parameters supported by the Zabbix server configuration file (zabbix_server.conf) are listed in this section.

The parameters are listed without additional information. Click on the parameter to see the full details.

Parameter	Description
AlertScriptsPath	The location of custom alert scripts.
AllowRoot	Allow the server to run as 'root'.
AllowUnsupportedDBVersions	Allow the server to work with unsupported database versions.
CacheSize	The size of the configuration cache.
CacheUpdateFrequency	This parameter determines how often Zabbix will perform the configuration cache update in seconds.
DBHost	The database host name.
DBName	The database name.
DBPassword	The database password.
DBPort	The database port when not using local socket.
DBSchema	The database schema name. Used for PostgreSQL.
DBSocket	The path to the MySQL socket file.
DBUser	The database user.
DBTLSConnect	Setting this option to the specified value enforces to use a TLS connection to the database.
DBTLSCAFile	The full pathname of a file containing the top-level CA(s) certificates for database certificate verification.
DBTLSCertFile	The full pathname of a file containing the Zabbix server certificate for authenticating to database.
DBTLSKeyFile	The full pathname of a file containing the private key for authenticating to database.
DBTLSCipher	The list of encryption ciphers that Zabbix server permits for TLS protocols up through TLS v1.2. Supported only for MySQL.
DBTLSCipher13	The list of encryption ciphersuites that Zabbix server permits for the TLS v1.3 protocol. Supported only for MySQL, starting from version 8.0.16.
DebugLevel	Specify the debug level.

Parameter	Description
ExportDir	The directory for real-time export of events, history and trends in newline-delimited JSON format. If set, enables the real-time export.
ExportFileSize	The maximum size per export file in bytes.
ExportType	The list of comma-delimited entity types (events, history, trends) for real-time export (all types by default).
ExternalScripts	The location of external scripts.
Fping6Location	The location of fping6.
FpingLocation	The location of fping.
HANodeName	The high availability cluster node name.
HistoryCacheSize	The size of the history cache.
HistoryIndexCacheSize	The size of the history index cache.
HistoryStorageDateIndex	Enable preprocessing of history values in history storage to store values in different indices based on date.
HistoryStorageURL	The history storage HTTP[S] URL.
HistoryStorageTypes	A comma-separated list of value types to be sent to the history storage.
HousekeepingFrequency	This parameter determines how often Zabbix will perform the housekeeping procedure in hours.
Include	You may include individual files or all files in a directory in the configuration file.
JavaGateway	The IP address (or hostname) of Zabbix Java gateway.
JavaGatewayPort	The port that Zabbix Java gateway listens on.
ListenBacklog	The maximum number of pending connections in the TCP queue.
ListenIP	A list of comma-delimited IP addresses that the trapper should listen on.
ListenPort	The listen port for trapper.
LoadModule	The module to load at server startup.
LoadModulePath	The full path to the location of server modules.
LogFile	The name of the log file.
LogFileSize	The maximum size of the log file.
LogSlowQueries	Determines how long a database query may take before being logged in milliseconds.
LogType	The type of the log output.
MaxHousekeeperDelete	No more than 'MaxHousekeeperDelete' rows (corresponding to [tablename], [field], value) will be deleted per one task in one housekeeping cycle.
NodeAddress	The IP or hostname with optional port to override how the frontend should connect to the server.
PidFile	The name of the PID file.
ProblemHousekeepingFrequency	Determines how often Zabbix will delete problems for deleted triggers.
ProxyConfigFrequency	Determines how often Zabbix server sends configuration data to a Zabbix proxy.
ProxyDataFrequency	Determines how often Zabbix server requests history data from a Zabbix proxy.
ServiceManagerSyncFrequency	Determines how often Zabbix will synchronize the configuration of a service manager.
SNMPTrapperFile	The temporary file used for passing data from the SNMP trap daemon to the server.
SocketDir	The directory to store the IPC sockets used by internal Zabbix services.
SourceIP	The source IP address.
SSHKeyLocation	The location of public and private keys for SSH checks and actions.
SSLCertLocation	The location of SSL client certificate files for client authentication.
SSLKeyLocation	The location of SSL private key files for client authentication.
SSLCALocation	Override the location of certificate authority (CA) files for SSL server certificate verification.
StartAlerters	The number of pre-forked instances of alerters.
StartConnectors	The number of pre-forked instances of connector workers.
StartDBSyncers	The number of pre-forked instances of history syncers.
StartDiscoverers	The number of pre-forked instances of discoverers.
StartEscalators	The number of pre-forked instances of escalators.
StartHistoryPollers	The number of pre-forked instances of history pollers.
StartHTTPPollers	The number of pre-forked instances of HTTP pollers.
StartIPMIPollers	The number of pre-forked instances of IPMI pollers.
StartJavaPollers	The number of pre-forked instances of Java pollers.
StartLLDProcessors	The number of pre-forked instances of low-level discovery (LLD) workers.
StartODBCPollers	The number of pre-forked instances of ODBC pollers.
StartPingers	The number of pre-forked instances of ICMP pingers.
StartPollersUnreachable	The number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java).
StartPollers	The number of pre-forked instances of pollers.
StartPreprocessors	The number of pre-started instances of preprocessing workers.
StartProxyPollers	The number of pre-forked instances of pollers for passive proxies.
StartReportWriters	The number of pre-forked instances of report writers.
StartSNMPTrapper	If set to 1, an SNMP trapper process will be started.

Parameter	Description
StartTimers	The number of pre-forked instances of timers.
StartTrappers	The number of pre-forked instances of trappers.
StartVMwareCollectors	The number of pre-forked VMware collector instances.
StatsAllowedIP	A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of external Zabbix instances. The stats request will be accepted only from the addresses listed here.
Timeout	Specifies how long we wait for agent, SNMP device or external check in seconds.
TLSCAFile	The full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	The full pathname of a file containing the server certificate or certificate chain, used for encrypted communications between Zabbix components.
TLSCipherAll	The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.
TLSCipherAll13	The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.
TLSCipherCert	The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate-based encryption.
TLSCipherCert13	The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption.
TLSCipherPSK	The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for PSK-based encryption.
TLSCipherPSK13	The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for PSK-based encryption.
TLSCRLFile	The full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.
TLSKeyFile	The full pathname of a file containing the server private key, used for encrypted communications between Zabbix components.
TmpDir	The temporary directory.
TrapperTimeout	Specifies how many seconds the trapper may spend processing new data.
TrendCacheSize	The size of the trend cache.
TrendFunctionCacheSize	The size of the trend function cache.
UnavailableDelay	Determines how often host is checked for availability during the unavailability period.
UnreachableDelay	Determines how often host is checked for availability during the unreachability period.
UnreachablePeriod	Determines after how many seconds of unreachability treats a host as unavailable.
User	Drop privileges to a specific, existing user on the system.
ValueCacheSize	The size of the history value cache.
Vault	Specifies the vault provider.
VaultDBPath	Specifies a location, from where database credentials should be retrieved by keys.
VaultTLSCertFile	The name of the SSL certificate file used for client authentication.
VaultTLSKeyFile	The name of the SSL private key file used for client authentication.
VaultToken	The HashiCorp vault authentication token.
VaultURL	The vault server HTTP[S] URL.
VMwareCacheSize	The shared memory size for storing VMware data.
VMwareFrequency	The delay in seconds between data gathering from a single VMware service.
VMwarePerfFrequency	The delay in seconds between performance counter statistics retrieval from a single VMware service.
VMwareTimeout	The maximum number of seconds a vmware collector will wait for a response from VMware service.
WebServiceURL	HTTP[S] URL to Zabbix web service in the format <host:port>/report.

All parameters are non-mandatory unless explicitly stated that the parameter is mandatory.

Note that:

- The default values reflect daemon defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

Parameter details

AlertScriptsPath

The location of custom alert scripts (depends on the *datadir* compile-time installation variable).

Default: `/usr/local/share/zabbix/alertscripts`

AllowRoot

Allow the server to run as 'root'. If disabled and the server is started by 'root', the server will try to switch to the 'zabbix' user instead. Has no effect if started under a regular user.

Default: 0
 Values: 0 - do not allow; 1 - allow

AllowUnsupportedDBVersions

Allow the server to work with unsupported database versions.

Default: 0
 Values: 0 - do not allow; 1 - allow

CacheSize

The size of the configuration cache, in bytes. The shared memory size for storing host, item and trigger data.

Default: 32M
 Range: 128K-64G

CacheUpdateFrequency

This parameter determines how often Zabbix will perform the configuration cache update in seconds. See also runtime control options.

Default: 10
 Range: 1-3600

DBHost

The database host name.
With MySQL `localhost` or empty string results in using a socket. With PostgreSQL only empty string results in attempt to use socket. With Oracle empty string results in using the Net Service Name connection method; in this case consider using the TNS_ADMIN environment variable to specify the directory of the tnsnames.ora file.

Default: `localhost`

DBName

The database name.
With Oracle, if the Net Service Name connection method is used, specify the service name from tnsnames.ora or set to empty string; set the TWO_TASK environment variable if DBName is set to empty string.

Mandatory: Yes

DBPassword

The database password. Comment this line if no password is used.

DBPort

The database port when not using local socket.
With Oracle, if the Net Service Name connection method is used, this parameter will be ignored; the port number from the tnsnames.ora file will be used instead.

Range: 1024-65535

DBSchema

The database schema name. Used for PostgreSQL.

DBSocket

The path to the MySQL socket file.

DBUser

The database user.

DBTLSConnect

Setting this option to the following values enforces to use a TLS connection to the database:
required - connect using TLS
verify_ca - connect using TLS and verify certificate
verify_full - connect using TLS, verify certificate and verify that database identity specified by DBHost matches its certificate

With MySQL, starting from 5.7.11, and PostgreSQL the following values are supported: `required`, `verify_ca`, `verify_full`.
With MariaDB, starting from version 10.2.6, the `required` and `verify_full` values are supported.
By default not set to any option and the behavior depends on database configuration.

DBTLSCAFile

The full pathname of a file containing the top-level CA(s) certificates for database certificate verification.

Mandatory: no (yes, if DBTLSConnect set to *verify_ca* or *verify_full*)

DBTLSCertFile

The full pathname of a file containing the Zabbix server certificate for authenticating to database.

DBTLSKeyFile

The full pathname of a file containing the private key for authenticating to database.

DBTLSCipher

The list of encryption ciphers that Zabbix server permits for TLS protocols up through TLS v1.2. Supported only for MySQL.

DBTLSCipher13

The list of encryption ciphersuites that Zabbix server permits for the TLS v1.3 protocol. Supported only for MySQL, starting from version 8.0.16.

DebugLevel

Specify the debug level:
0 - basic information about starting and stopping of Zabbix processes
1 - critical information;
2 - error information;
3 - warnings;
4 - for debugging (produces lots of information);
5 - extended debugging (produces even more information).
See also runtime control options.

Default: 3
 Range: 0-5

ExportDir

The directory for real-time export of events, history and trends in newline-delimited JSON format. If set, enables the real-time export.

ExportFileSize

The maximum size per export file in bytes. Used for rotation if `ExportDir` is set.

Default: 1G
 Range: 1M-1G

ExportType

The list of comma-delimited entity types (events, history, trends) for real-time export (all types by default). Valid only if ExportDir is set.
Note that if ExportType is specified, but ExportDir is not, then this is a configuration error and the server will not start.

Example for history and trends export:

```
ExportType=history,trends
```

Example for event export only:

```
ExportType=events
```

ExternalScripts

The location of external scripts (depends on the `datadir` compile-time installation variable).

Default: `/usr/local/share/zabbix/externalscripts`

Fping6Location

The location of fping6. Make sure that the fping6 binary has root ownership and the SUID flag set. Make empty (”Fping6Location=”) if your fping utility is capable to process IPv6 addresses.

Default: `/usr/sbin/fping6`

FpingLocation

The location of fping. Make sure that the fping binary has root ownership and the SUID flag set.

Default: `/usr/sbin/fping`

HANodeName

The high availability cluster node name. When empty the server is working in standalone mode and a node with empty name is created.

HistoryCacheSize

The size of the history cache, in bytes. The shared memory size for storing history data.

Default: 16M
 Range: 128K-2G

HistoryIndexCacheSize

The size of the history index cache, in bytes. The shared memory size for indexing the history data stored in history cache. The index cache size needs roughly 100 bytes to cache one item.

Default: 4M
 Range: 128K-2G

HistoryStorageDateIndex

Enable preprocessing of history values in history storage to store values in different indices based on date.

Default: 0
 Values: 0 - disable; 1 - enable

HistoryStorageURL

The history storage HTTP[S] URL. This parameter is used for Elasticsearch setup.

HistoryStorageTypes

A comma-separated list of value types to be sent to the history storage. This parameter is used for Elasticsearch setup.

Default: `uint,dbl,str,log,text`

HousekeepingFrequency

This parameter determines how often Zabbix will perform the housekeeping procedure in hours. Housekeeping is removing outdated information from the database.
Note: To prevent housekeeper from being overloaded (for example, when history and trend periods are greatly reduced), no more than 4 times HousekeepingFrequency hours of outdated information are deleted in one housekeeping cycle, for each item. Thus, if HousekeepingFrequency is 1, no more than 4 hours of outdated information (starting from the oldest entry) will be deleted per cycle.
Note: To lower load on server startup housekeeping is postponed for 30 minutes after server start. Thus, if HousekeepingFrequency is 1, the very first housekeeping procedure after server start will run after 30 minutes, and will repeat with one hour delay thereafter.
It is possible to disable automatic housekeeping by setting HousekeepingFrequency to 0. In this case the housekeeping procedure can only be started by *housekeeper_execute* runtime control option and the period of outdated information deleted in one housekeeping cycle is 4 times the period since the last housekeeping cycle, but not less than 4 hours and not greater than 4 days.
See also runtime control options.

Default: 1
 Range: 0-24

Include

You may include individual files or all files in a directory in the configuration file. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. See special notes about limitations.

Example:

```
Include=/absolute/path/to/config/files/*.conf
```

JavaGateway

The IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started.

JavaGatewayPort

The port that Zabbix Java gateway listens on.

Default: 10052
 Range: 1024-32767

ListenBacklog

The maximum number of pending connections in the TCP queue.
The default value is a hard-coded constant, which depends on the system.
The maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.

Default: SOMAXCONN
 Range: 0 - INT_MAX

ListenIP

A list of comma-delimited IP addresses that the trapper should listen on.
Trapper will listen on all network interfaces if this parameter is missing.

Default: 0.0.0.0

ListenPort

The listen port for trapper.

Default: 10051
 Range: 1024-32767

LoadModule

The module to load at server startup. Modules are used to extend the functionality of the server. The module must be located in the directory specified by LoadModulePath or the path must precede the module name. If the preceding path is absolute (starts with '/')

then LoadModulePath is ignored.
Formats:
LoadModule=<module.so>
LoadModule=<path/module.so>
LoadModule=</ab
is allowed to include multiple LoadModule parameters.

LoadModulePath

The full path to the location of server modules. The default depends on compilation options.

LogFile

The name of the log file.

Mandatory: Yes, if LogType is set to *file*; otherwise no

LogFileSize

The maximum size of the log file in MB.
0 - disable automatic log rotation.
Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

Default: 1
 Range: 0-1024
 Mandatory: Yes, if LogType is set to *file*; otherwise no

LogSlowQueries

Determines how long a database query may take before being logged in milliseconds.
0 - don't log slow queries.
This option becomes enabled starting with DebugLevel=3.

Default: 0
 Range: 0-3600000

LogType

The type of the log output:
file - write log to file specified by LogFile parameter;
system - write log to syslog;
console - write log to standard output.

Default: `file`

MaxHousekeeperDelete

No more than 'MaxHousekeeperDelete' rows (corresponding to [tablename], [field], [value]) will be deleted per one task in one housekeeping cycle.
If set to 0 then no limit is used at all. In this case you must know what you are doing, so as not to overload the database! [2]
This parameter applies only to deleting history and trends of already deleted items.

Default: 5000
 Range: 0-1000000

NodeAddress

IP or hostname with optional port to override how the frontend should connect to the server.
Format: <address>[:<port>]

If IP or hostname is not set, the value of ListenIP will be used. If ListenIP is not set, the value `localhost` will be used.
If port is not set, the value of ListenPort will be used. If ListenPort is not set, the value 10051 will be used.

This option can be overridden by the address specified in the frontend configuration.

See also: HANodeName parameter; Enabling high availability.

Default: 'localhost:10051'

PidFile

Name of the PID file.

Default: `/tmp/zabbix_server.pid`

ProblemHousekeepingFrequency

Determines how often Zabbix will delete problems for deleted triggers in seconds.

Default: 60
 Range: 1-3600

ProxyConfigFrequency

Determines how often Zabbix server sends configuration data to a Zabbix proxy in seconds. Used only for proxies in a passive mode.

Default: 10
 Range: 1-604800

ProxyDataFrequency

Determines how often Zabbix server requests history data from a Zabbix proxy in seconds. Used only for proxies in the passive mode.

Default: 1
 Range: 1-3600

ServiceManagerSyncFrequency

Determines how often Zabbix will synchronize the configuration of a service manager in seconds.

Default: 60
 Range: 1-3600

SNMPTrapperFile

Temporary file used for passing data from the SNMP trap daemon to the server.
Must be the same as in zabbix_trap_receiver.pl or SNMPTT configuration file.

Default: `/tmp/zabbix_traps.tmp`

SocketDir

Directory to store IPC sockets used by internal Zabbix services.

Default: `/tmp`

SourceIP

Source IP address for:
- outgoing connections to Zabbix proxy and Zabbix agent;
- agentless connections (VMware, SSH, JMX, SNMP, Telnet and simple checks);
- HTTP agent connections;
- script item JavaScript HTTP requests;
- preprocessing JavaScript HTTP requests;
- sending notification emails (connections to SMTP server);
- webhook notifications (JavaScript HTTP connections);
- connections to the Vault

SSHKeyLocation

Location of public and private keys for SSH checks and actions.

SSLCertLocation

Location of SSL client certificate files for client authentication.
This parameter is used in web monitoring only.

SSLKeyLocation

Location of SSL private key files for client authentication.
This parameter is used in web monitoring only.

SSLCALocation

Override the location of certificate authority (CA) files for SSL server certificate verification. If not set, system-wide directory will be used.
Note that the value of this parameter will be set as libcurl option CURLOPT_CAPATH. For libcurl versions before 7.42.0, this only has effect if libcurl was compiled to use OpenSSL. For more information see cURL web page.
This parameter is used in web monitoring and in SMTP authentication.

StartAlerters

The number of pre-forked instances of alerters.

Default: 3
 Range: 1-100

StartConnectors

The number of pre-forked instances of connector workers. The connector manager process is started automatically when a connector worker is started.

Default: 0
 Range: 0-1000

StartDBSyncers

The number of pre-forked instances of history syncers.
Note: Be careful when changing this value, increasing it may do more harm than good. Roughly, the default value should be enough to handle up to 4000 NVPS.

Default: 4
 Range: 1-100

StartDiscoverers

The number of pre-forked instances of discoverers.

Default: 1
 Range: 0-250

StartEscalators

The number of pre-forked instances of escalators.

Default: 1
 Range: 1-100

StartHistoryPollers

The number of pre-forked instances of history pollers.
Only required for calculated checks.

Default: 5
 Range: 0-1000

StartHTTPPollers

The number of pre-forked instances of HTTP pollers[1].

Default: 1
 Range: 0-1000

StartIPMIPollers

The number of pre-forked instances of IPMI pollers.

Default: 0
 Range: 0-1000

StartJavaPollers

The number of pre-forked instances of Java pollers[1].

Default: 0
 Range: 0-1000

StartLLDProcessors

The number of pre-forked instances of low-level discovery (LLD) workers[1].
The LLD manager process is automatically started when an LLD worker is started.

Default: 2
 Range: 0-100

StartODBCPollers

The number of pre-forked instances of ODBC pollers[1].

Default: 1
 Range: 0-1000

StartPingers

The number of pre-forked instances of ICMP pingers[1].

Default: 1
 Range: 0-1000

StartPollersUnreachable

The number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java)[1].
At least one poller for unreachable hosts must be running if regular, IPMI or Java pollers are started.

Default: 1
 Range: 0-1000

StartPollers

The number of pre-forked instances of pollers[1].

Default: 5
 Range: 0-1000

StartPreprocessors

The number of pre-started instances of preprocessing workers[1].

Default: 3
 Range: 1-1000

StartProxyPollers

The number of pre-forked instances of pollers for passive proxies[1].

Default: 1
 Range: 0-250

StartReportWriters

The number of pre-forked instances of report writers.
If set to 0, scheduled report generation is disabled.
The report manager process is automatically started when a report writer is started.

Default: 0
 Range: 0-100

StartSNMPTrapper

If set to 1, an SNMP trapper process will be started.

Default: 0
 Range: 0-1

StartTimers

The number of pre-forked instances of timers.
Timers process maintenance periods.

Default: 1
 Range: 1-1000

StartTrappers

The number of pre-forked instances of trappers[1].
Trappers accept incoming connections from Zabbix sender, active agents and active proxies.

Default: 5
 Range: 1-1000

StartVMwareCollectors

The number of pre-forked VMware collector instances.

Default: 0
 Range: 0-250

StatsAllowedIP

A list of comma delimited IP addresses, optionally in CIDR notation, or DNS names of external Zabbix instances. Stats request will be accepted only from the addresses listed here. If this parameter is not set no stats requests will be accepted.
If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address.

Example:

```
StatsAllowedIP=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

Timeout

Specifies how long we wait for agent, SNMP device or external check in seconds.

Default: 3
 Range: 1-30

TLSCAFile

The full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.

TLSCertFile

The full pathname of a file containing the server certificate or certificate chain, used for encrypted communications between Zabbix components.

TLSCipherAll

The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.

Example:

```
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSCipherAll13

The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.

Example for GnuTLS:

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL
```

Example for OpenSSL:

```
EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherCert

The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate-based encryption.

Example for GnuTLS:

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

Example for OpenSSL:

```
EECDH+aRSA+AES128:RSA+aRSA+AES128
```

TLSCipherCert13

The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption.

TLSCipherPSK

The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for PSK-based encryption.

Example for GnuTLS:

```
NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

Example for OpenSSL:

```
kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherPSK13

The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for PSK-based encryption.

Example:

```
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSCRLFile

The full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.

TLSKeyFile

The full pathname of a file containing the server private key, used for encrypted communications between Zabbix components.

TmpDir

The temporary directory.

Default: `/tmp`

TrapperTimeout

Specifies how many seconds the trapper may spend processing new data.

Default: 300
 Range: 1-300

TrendCacheSize

The size of the trend cache, in bytes.
The shared memory size for storing trends data.

Default: 4M
 Range: 128K-2G

TrendFunctionCacheSize

The size of the trend function cache, in bytes.
The shared memory size for caching calculated trend function data.

Default: 4M
 Range: 128K-2G

UnavailableDelay

Determines how often host is checked for availability during the unavailability period in seconds.

Default: 60
 Range: 1-3600

UnreachableDelay

Determines how often host is checked for availability during the unreachability period in seconds.

Default: 15
 Range: 1-3600

UnreachablePeriod

Determines after how many seconds of unreachability treats a host as unavailable.

Default: 45
 Range: 1-3600

User

Drop privileges to a specific, existing user on the system.
Only has effect if run as 'root' and AllowRoot is disabled.

Default: `zabbix`

ValueCacheSize

The size of the history value cache, in bytes.
The shared memory size for caching item history data requests.
Setting to 0 disables the value cache (not recommended).
When the value cache runs out of the shared memory a warning message is written to the server log every 5 minutes.

Default: 8M
 Range: 0,128K-64G

Vault

Specifies the vault provider:
HashiCorp - HashiCorp KV Secrets Engine version 2
CyberArk - CyberArk Central Credential Provider
Must match the vault provider set in the frontend.

Default: HashiCorp

VaultDBPath

Specifies a location, from where database credentials should be retrieved by keys. Depending on the Vault, can be vault path or query.
 The keys used for HashiCorp are 'password' and 'username'.

Example:

```
secret/zabbix/database
```

The keys used for CyberArk are 'Content' and 'UserName'.

Example:

```
AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_proxy_database
```

This option can only be used if DBUser and DBPassword are not specified.

VaultTLSCertFile

The name of the SSL certificate file used for client authentication
 The certificate file must be in PEM1 format.
 If the certificate file contains also the private key, leave the SSL key file field empty.
 The directory containing this file is specified by the configuration parameter SSLCertLocation.
This option can be omitted but is recommended for CyberArkCCP vault.

VaultTLSKeyFile

The name of the SSL private key file used for client authentication.
 The private key file must be in PEM1 format.
 The directory containing this file is specified by the configuration parameter SSLKeyLocation.
This option can be omitted but is recommended for CyberArkCCP vault.

VaultToken

The HashiCorp Vault authentication token that should have been generated exclusively for Zabbix server with read-only permission to the paths specified in Vault macros and read-only permission to the path specified in the optional VaultDBPath configuration parameter.
It is an error if VaultToken and VAULT_TOKEN environment variable are defined at the same time.

Mandatory: Yes, if Vault is set to *HashiCorp*; otherwise no

VaultURL

The vault server HTTP[S] URL. The system-wide CA certificates directory will be used if SSLCALocation is not specified.

Default: https://127.0.0.1:8200

VMwareCacheSize

The shared memory size for storing VMware data.
A VMware internal check zabbix[vmware,buffer,...] can be used to monitor the VMware cache usage (see Internal checks).
Note that shared memory is not allocated if there are no vmware collector instances configured to start.

Default: 8M
 Range: 256K-2G

VMwareFrequency

The delay in seconds between data gathering from a single VMware service.
This delay should be set to the least update interval of any VMware monitoring item.

Default: 60
 Range: 10-86400

VMwarePerfFrequency

The delay in seconds between performance counter statistics retrieval from a single VMware service. This delay should be set to the least update interval of any VMware monitoring item that uses VMware performance counters.

Default: 60
 Range: 10-86400

VMwareTimeout

The maximum number of seconds a vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor).

Default: 10
 Range: 1-300

WebServiceURL

The HTTP[S] URL to Zabbix web service in the format `<host:port>/report`.

Example:

```
WebServiceURL=http://localhost:10053/report
```

Footnotes

[1] Note that too many data gathering processes (pollers, unreachable pollers, ODBC pollers, HTTP pollers, Java pollers, pingers, trappers, proxypollers) together with IPMI manager, SNMP trapper and preprocessing workers can **exhaust** the per-process file descriptor limit for the preprocessing manager.

> **Warning:**
> This will cause Zabbix server to stop (usually shortly after the start, but sometimes it can take more time). The configuration file should be revised or the limit should be raised to avoid this situation.

[2] When a lot of items are deleted it increases the load to the database, because the housekeeper will need to remove all the history data that these items had. For example, if we only have to remove 1 item prototype, but this prototype is linked to 50 hosts and for every host the prototype is expanded to 100 real items, 5000 items in total have to be removed (1*50*100). If 500 is set for MaxHousekeeperDelete (MaxHousekeeperDelete=500), the housekeeper process will have to remove up to 2500000 values (5000*500) for the deleted items from history and trends tables in one cycle.

2 Zabbix proxy

Overview

The parameters supported by the Zabbix proxy configuration file (zabbix_proxy.conf) are listed in this section.

The parameters are listed without additional information. Click on the parameter to see the full details.

Parameter	Description
AllowRoot	Allow the proxy to run as 'root'.
AllowUnsupportedDBVersions	Allow the proxy to work with unsupported database versions.
CacheSize	The size of the configuration cache.
ConfigFrequency	This parameter is **deprecated** (use ProxyConfigFrequency instead). How often the proxy retrieves configuration data from Zabbix server in seconds.
DataSenderFrequency	The proxy will send collected data to the server every N seconds.
DBHost	The database host name.
DBName	The database name or path to the database file for SQLite3.
DBPassword	The database password.
DBPort	The database port when not using local socket.
DBSchema	The database schema name. Used for PostgreSQL.
DBSocket	The path to the MySQL socket file.
DBUser	The database user.
DBTLSConnect	Setting this option to the specified value enforces to use a TLS connection to the database.
DBTLSCAFile	The full pathname of a file containing the top-level CA(s) certificates for database certificate verification.
DBTLSCertFile	The full pathname of a file containing the Zabbix proxy certificate for authenticating to database.
DBTLSKeyFile	The full pathname of a file containing the private key for authenticating to database.
DBTLSCipher	The list of encryption ciphers that Zabbix proxy permits for TLS protocols up through TLS v1.2. Supported only for MySQL.
DBTLSCipher13	The list of encryption ciphersuites that Zabbix proxy permits for the TLS v1.3 protocol. Supported only for MySQL, starting from version 8.0.16.
DebugLevel	The debug level.
EnableRemoteCommands	Whether remote commands from Zabbix server are allowed.
ExternalScripts	The location of external scripts.
Fping6Location	The location of fping6.
FpingLocation	The location of fping.
HistoryCacheSize	The size of the history cache.
HistoryIndexCacheSize	The size of the history index cache.
Hostname	A unique, case sensitive proxy name.
HostnameItem	The item used for setting Hostname if it is undefined.
HousekeepingFrequency	How often Zabbix will perform the housekeeping procedure in hours.

Parameter	Description
Include	You may include individual files or all files in a directory in the configuration file.
JavaGateway	The IP address (or hostname) of Zabbix Java gateway.
JavaGatewayPort	The port that Zabbix Java gateway listens on.
ListenBacklog	The maximum number of pending connections in the TCP queue.
ListenIP	A list of comma-delimited IP addresses that the trapper should listen on.
ListenPort	The listen port for trapper.
LoadModule	The module to load at proxy startup.
LoadModulePath	The full path to the location of proxy modules.
LogFile	The name of the log file.
LogFileSize	The maximum size of the log file.
LogRemoteCommands	Enable logging of executed shell commands as warnings.
LogSlowQueries	How long a database query may take before being logged.
LogType	The type of the log output.
PidFile	The name of the PID file.
ProxyConfigFrequency	How often the proxy retrieves configuration data from Zabbix server in seconds.
ProxyLocalBuffer	The proxy will keep data locally for N hours, even if the data have already been synced with the server.
ProxyMode	The proxy operating mode (active/passive).
ProxyOfflineBuffer	The proxy will keep data for N hours in case of no connectivity with Zabbix server.
Server	If ProxyMode is set to active mode: Zabbix server IP address or DNS name (address:port) or cluster (address:port;address2:port) to get configuration data from and send data to. If ProxyMode is set to passive mode: List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix server.
SNMPTrapperFile	The temporary file used for passing data from the SNMP trap daemon to the proxy.
SocketDir	The directory to store the IPC sockets used by internal Zabbix services.
SourceIP	The source IP address.
SSHKeyLocation	The location of public and private keys for SSH checks and actions.
SSLCertLocation	The location of SSL client certificate files for client authentication.
SSLKeyLocation	The location of SSL private key files for client authentication.
SSLCALocation	Override the location of certificate authority (CA) files for SSL server certificate verification.
StartDBSyncers	The number of pre-forked instances of history syncers.
StartDiscoverers	The number of pre-forked instances of discoverers.
StartHTTPPollers	The number of pre-forked instances of HTTP pollers.
StartIPMIPollers	The number of pre-forked instances of IPMI pollers.
StartJavaPollers	The number of pre-forked instances of Java pollers.
StartODBCPollers	The number of pre-forked instances of ODBC pollers.
StartPingers	The number of pre-forked instances of ICMP pingers.
StartPollersUnreachable	The number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java).
StartPollers	The number of pre-forked instances of pollers.
StartPreprocessors	The number of pre-started instances of preprocessing workers.
StartSNMPTrapper	If set to 1, an SNMP trapper process will be started.
StartTrappers	The number of pre-forked instances of trappers.
StartVMwareCollectors	The number of pre-forked VMware collector instances.
StatsAllowedIP	A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of external Zabbix instances. The stats request will be accepted only from the addresses listed here.
Timeout	How long we wait for agent, SNMP device or external check in seconds.
TLSAccept	What incoming connections to accept from Zabbix server.
TLSCAFile	The full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	The full pathname of a file containing the server certificate or certificate chain, used for encrypted communications between Zabbix components.
TLSCipherAll	The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.
TLSCipherAll13	The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.
TLSCipherCert	The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate-based encryption.
TLSCipherCert13	The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption.
TLSCipherPSK	The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for PSK-based encryption.

Parameter	Description
TLSCipherPSK13	The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for PSK-based encryption.
TLSConnect	How the proxy should connect to Zabbix server.
TLSCRLFile	The full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.
TLSKeyFile	The full pathname of a file containing the proxy private key, used for encrypted communications between Zabbix components.
TLSPSKFile	The full pathname of a file containing the proxy pre-shared key, used for encrypted communications with Zabbix server.
TLSPSKIdentity	The pre-shared key identity string, used for encrypted communications with Zabbix server.
TLSServerCertIssuer	The allowed server certificate issuer.
TLSServerCertSubject	The allowed server certificate subject.
TmpDir	The temporary directory.
TrapperTimeout	How many seconds the trapper may spend processing new data.
UnavailableDelay	How often a host is checked for availability during the unavailability period.
UnreachableDelay	How often a host is checked for availability during the unreachability period.
UnreachablePeriod	After how many seconds of unreachability treat the host as unavailable.
User	Drop privileges to a specific, existing user on the system.
Vault	The vault provider.
VaultDBPath	The location, from where database credentials should be retrieved by keys.
VaultTLSCertFile	The name of the SSL certificate file used for client authentication.
VaultTLSKeyFile	The name of the SSL private key file used for client authentication.
VaultToken	The HashiCorp vault authentication token.
VaultURL	The vault server HTTP[S] URL.
VMwareCacheSize	The shared memory size for storing VMware data.
VMwareFrequency	The delay in seconds between data gathering from a single VMware service.
VMwarePerfFrequency	The delay in seconds between performance counter statistics retrieval from a single VMware service.
VMwareTimeout	The maximum number of seconds a vmware collector will wait for a response from VMware service.

All parameters are non-mandatory unless explicitly stated that the parameter is mandatory.

Note that:

- The default values reflect daemon defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

Parameter details

AllowRoot

Allow the proxy to run as 'root'. If disabled and the proxy is started by 'root', the proxy will try to switch to the 'zabbix' user instead. Has no effect if started under a regular user.

Default: 0
 Values: 0 - do not allow; 1 - allow

AllowUnsupportedDBVersions

Allow the proxy to work with unsupported database versions.

Default: 0
 Values: 0 - do not allow; 1 - allow

CacheSize

The size of the configuration cache, in bytes. The shared memory size for storing host and item data.

Default: 32M
 Range: 128K-64G

ConfigFrequency

This parameter is **deprecated** (use ProxyConfigFrequency instead).
How often the proxy retrieves configuration data from Zabbix server in seconds.
Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).

Default: 3600
 Range: 1-604800

DataSenderFrequency

The proxy will send collected data to the server every N seconds. Note that an active proxy will still poll Zabbix server every second for remote command tasks.
Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).

Default: 1
 Range: 1-3600

DBHost

The database host name.
With MySQL `localhost` or empty string results in using a socket. With PostgreSQL only empty string results in attempt to use socket. With Oracle empty string results in using the Net Service Name connection method; in this case consider using the TNS_ADMIN environment variable to specify the directory of the tnsnames.ora file.

Default: `localhost`

DBName

The database name or path to the database file for SQLite3 (the multi-process architecture of Zabbix does not allow to use in-memory database, e.g. `:memory:`, `file::memory:?cache=shared` or `file:memdb1?mode=memory&cache=shared`).
Warning: Do not attempt to use the same database the Zabbix server is using.
With Oracle, if the Net Service Name connection method is used, specify the service name from tnsnames.ora or set to empty string; set the TWO_TASK environment variable if DBName is set to empty string.

Mandatory: Yes

DBPassword

The database password. Comment this line if no password is used. Ignored for SQLite.

DBPort

The database port when not using local socket. Ignored for SQLite.
With Oracle, if the Net Service Name connection method is used, this parameter will be ignored; the port number from the tnsnames.ora file will be used instead.

Range: 1024-65535

DBSchema

The database schema name. Used for PostgreSQL.

DBSocket

The path to the MySQL socket file.
The database port when not using local socket. Ignored for SQLite.

Default: 3306

DBUser

The database user. Ignored for SQLite.

DBTLSConnect

Setting this option enforces to use TLS connection to the database:
required - connect using TLS
verify_ca - connect using TLS and verify certificate
verify_full - connect using TLS, verify certificate and verify that database identity specified by DBHost matches its certificate
On MySQL starting from 5.7.11 and PostgreSQL the following values are supported: ”required”, ”verify”, ”verify_full”.
On MariaDB starting from version 10.2.6 ”required” and ”verify_full” values are supported.
By default not set to any option and the behavior depends on database configuration.

DBTLSCAFile

The full pathname of a file containing the top-level CA(s) certificates for database certificate verification.

Mandatory: no (yes, if DBTLSConnect set to *verify_ca* or *verify_full*)

DBTLSCertFile

The full pathname of a file containing the Zabbix proxy certificate for authenticating to database.

DBTLSKeyFile

The full pathname of a file containing the private key for authenticating to the database.

DBTLSCipher

The list of encryption ciphers that Zabbix proxy permits for TLS protocols up through TLS v1.2. Supported only for MySQL.

DBTLSCipher13

The list of encryption ciphersuites that Zabbix proxy permits for the TLS v1.3 protocol. Supported only for MySQL, starting from version 8.0.16.

DebugLevel

Specify the debug level:
0 - basic information about starting and stopping of Zabbix processes
1 - critical information;
2 - error information;
3 - warnings;
4 - for debugging (produces lots of information);
5 - extended debugging (produces even more information).

Default: 3
 Range: 0-5

EnableRemoteCommands

Whether remote commands from Zabbix server are allowed.

Default: 0
 Values: 0 - not allowed; 1 - allowed

ExternalScripts

The location of external scripts (depends on the `datadir` compile-time installation variable).

Default: `/usr/local/share/zabbix/externalscripts`

Fping6Location

The location of fping6. Make sure that the fping6 binary has root ownership and the SUID flag set. Make empty (”Fping6Location=”) if your fping utility is capable to process IPv6 addresses.

Default: `/usr/sbin/fping6`

FpingLocation

The location of fping. Make sure that the fping binary has root ownership and the SUID flag set.

Default: `/usr/sbin/fping`

HistoryCacheSize

The size of the history cache, in bytes. The shared memory size for storing history data.

Default: `16M`
 Range: 128K-2G

HistoryIndexCacheSize

The size of the history index cache, in bytes. The shared memory size for indexing the history data stored in history cache. The index cache size needs roughly 100 bytes to cache one item.

Default: `4M`
 Range: 128K-2G

Hostname

A unique, case sensitive proxy name. Make sure the proxy name is known to the server.
Allowed characters: alphanumeric, ’.’, ’ ’, ’_’ and ’-’. Maximum length: 128

Default: Set by HostnameItem

HostnameItem

The item used for setting Hostname if it is undefined (this will be run on the proxy similarly as on an agent). Ignored if Hostname is set.
Does not support UserParameters, performance counters or aliases, but does support system.run[].

Default: system.hostname

HousekeepingFrequency

How often Zabbix will perform housekeeping procedure (in hours). Housekeeping is removing outdated information from the database.
Note: To lower load on proxy startup housekeeping is postponed for 30 minutes after proxy start. Thus, if HousekeepingFrequency is 1, the very first housekeeping procedure after proxy start will run after 30 minutes, and will repeat every hour thereafter.
Since Zabbix 3.0.0 it is possible to disable automatic housekeeping by setting HousekeepingFrequency to 0. In this case the housekeeping procedure can only be started by *housekeeper_execute* runtime control option.

Default: 1
 Range: 0-24

Include

You may include individual files or all files in a directory in the configuration file.
To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching.
See special notes about limitations.

Example:

```
Include=/absolute/path/to/config/files/*.conf
```

JavaGateway

The IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started.

JavaGatewayPort

The port that Zabbix Java gateway listens on.

Default: 10052
 Range: 1024-32767

ListenBacklog

The maximum number of pending connections in the TCP queue.
The default value is a hard-coded constant, which depends on the system.
The maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.

Default: SOMAXCONN
 Range: 0 - INT_MAX

ListenIP

A list of comma-delimited IP addresses that the trapper should listen on.
Trapper will listen on all network interfaces if this parameter is missing.

Default: 0.0.0.0

ListenPort

The listen port for trapper.

Default: 10051
 Range: 1024-32767

LoadModule

The module to load at proxy startup. Modules are used to extend the functionality of the proxy. The module must be located in the directory specified by LoadModulePath or the path must precede the module name. If the preceding path is absolute (starts with '/') then LoadModulePath is ignored.
Formats:
LoadModule=<module.so>
LoadModule=<path/module.so>
LoadModule=</ab
is allowed to include multiple LoadModule parameters.

LoadModulePath

The full path to the location of proxy modules. The default depends on compilation options.

LogFile

The name of the log file.

Mandatory: Yes, if LogType is set to *file*; otherwise no

LogFileSize

The maximum size of a log file in MB.
0 - disable automatic log rotation.
Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

Default: 1
 Range: 0-1024

LogRemoteCommands

Enable the logging of executed shell commands as warnings.

Default: 0
 Values: 0 - disabled, 1 - enabled

LogType

The type of the log output:
file - write log to the file specified by LogFile parameter;
system - write log to syslog;
console - write log to standard output.

Default: file

LogSlowQueries

How long a database query may take before being logged (in milliseconds).
0 - don't log slow queries.
This option becomes enabled starting with DebugLevel=3.

Default: 0
 Range: 0-3600000

PidFile

The name of the PID file.

Default: /tmp/zabbix_proxy.pid

ProxyConfigFrequency

How often the proxy retrieves configuration data from Zabbix server in seconds.
Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).

Default: 10
 Range: 1-604800

ProxyLocalBuffer

The proxy will keep data locally for N hours, even if the data have already been synced with the server.
This parameter may be used if local data will be used by third-party applications.

Default: 0
 Range: 0-720

ProxyMode

The proxy operating mode.
0 - proxy in the active mode
1 - proxy in the passive mode
Note that (sensitive) proxy configuration data may become available to parties having access to the Zabbix server trapper port when using an active proxy. This is possible because anyone may pretend to be an active proxy and request configuration data; authentication does not take place.

Default: 0
 Range: 0-1

ProxyOfflineBuffer

The proxy will keep data for N hours in case of no connectivity with Zabbix server.
Older data will be lost.

Default: 1
 Range: 1-720

Server

If ProxyMode is set to *active mode*:
Zabbix server IP address or DNS name (address:port) or cluster (address:port;address2:port) to get configuration data from and send data to.
If port is not specified, the default port is used.
Cluster nodes must be separated by a semicolon.

If ProxyMode is set to *passive mode*:
List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix server. Incoming connections will be accepted only from the addresses listed here. If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally.
'::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address.

Example:

```
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

Mandatory: yes

SNMPTrapperFile

A temporary file used for passing data from the SNMP trap daemon to the proxy.
Must be the same as in zabbix_trap_receiver.pl or SNMPTT configuration file.

Default: `/tmp/zabbix_traps.tmp`

SocketDir

The directory to store IPC sockets used by internal Zabbix services.

Default: `/tmp`

SourceIP

The source IP address for:
- outgoing connections to Zabbix server;
- agentless connections (VMware, SSH, JMX, SNMP, Telnet and simple checks);
- HTTP agent connections;
- script item JavaScript HTTP requests;
- preprocessing JavaScript HTTP requests;
- connections to the Vault

SSHKeyLocation

The location of public and private keys for SSH checks and actions.

SSLCertLocation

The location of SSL client certificate files for client authentication.
This parameter is used in web monitoring only.

SSLKeyLocation

The location of SSL private key files for client authentication.
This parameter is used in web monitoring only.

SSLCALocation

The location of certificate authority (CA) files for SSL server certificate verification.
Note that the value of this parameter will be set as the CURLOPT_CAPATH libcurl option. For libcurl versions before 7.42.0, this only has effect if libcurl was compiled to use OpenSSL. For more information see the cURL web page.
This parameter is used in web monitoring and in SMTP authentication.

StartDBSyncers

The number of pre-forked instances of history syncers.
Note: Be careful when changing this value, increasing it may do more harm than good.

Default: 4
 Range: 1-100

StartDiscoverers

The number of pre-forked instances of discoverers.

Default: 1
 Range: 0-250

StartHTTPPollers

The number of pre-forked instances of HTTP pollers.

Default: 1 | Range: 0-1000

StartIPMIPollers

The number of pre-forked instances of IPMI pollers.

Default: 0
 Range: 0-1000

StartJavaPollers

The number of pre-forked instances of Java pollers.

Default: 0
 Range: 0-1000

StartODBCPollers

The number of pre-forked instances of ODBC pollers.

Default: 1
 Range: 0-1000

StartPingers

The number of pre-forked instances of ICMP pingers.

Default: 1
 Range: 0-1000

StartPollersUnreachable

The number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java). At least one poller for unreachable hosts must be running if regular, IPMI or Java pollers are started.

Default: 1
 Range: 0-1000

StartPollers

The number of pre-forked instances of pollers.

Default: 5
 Range: 0-1000

StartPreprocessors

The number of pre-started instances of preprocessing workers.

Default: 3
 Range: 1-1000

StartSNMPTrapper

If set to 1, an SNMP trapper process will be started.

Default: 0
 Range: 0-1

StartTrappers

The number of pre-forked instances of trappers.
Trappers accept incoming connections from Zabbix sender and active agents.

Default: 5
 Range: 0-1000

StartVMwareCollectors

The number of pre-forked VMware collector instances.

Default: 0
 Range: 0-250

StatsAllowedIP

A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of external Zabbix instances. The stats request will be accepted only from the addresses listed here. If this parameter is not set no stats requests will be accepted.
If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address.

Example:

```
StatsAllowedIP=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

Timeout

Specifies how long we wait for agent, SNMP device or external check in seconds.

Default: 3
 Range: 1-30

TLSAccept

What incoming connections to accept from Zabbix server. Used for a passive proxy, ignored on an active proxy. Multiple values can be specified, separated by comma:
unencrypted - accept connections without encryption (default)
psk - accept connections with TLS and a pre-shared key (PSK)
cert - accept connections with TLS and a certificate

Mandatory: yes for passive proxy, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection); otherwise no

TLSCAFile

The full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.

TLSCertFile

The full pathname of a file containing the proxy certificate or certificate chain, used for encrypted communications between Zabbix components.

TLSCipherAll

The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.

Example:

```
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSCipherAll13

The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.

Example for GnuTLS:

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL
```

Example for OpenSSL:

```
EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherCert

The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate-based encryption.

Example for GnuTLS:

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

Example for OpenSSL:

```
EECDH+aRSA+AES128:RSA+aRSA+AES128
```

TLSCipherCert13

The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption.

TLSCipherPSK

The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for PSK-based encryption.

Example for GnuTLS:

```
NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

Example for OpenSSL:

```
kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherPSK13

The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for PSK-based encryption.

Example:

```
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSConnect

How the proxy should connect to Zabbix server. Used for an active proxy, ignored on a passive proxy. Only one value can be specified:
unencrypted - connect without encryption (default)
psk - connect using TLS and a pre-shared key (PSK)
cert - connect using TLS and a certificate

Mandatory: yes for active proxy, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection); otherwise no

TLSCRLFile

The full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.

TLSKeyFile

The full pathname of a file containing the proxy private key, used for encrypted communications between Zabbix components.

TLSPSKFile

The full pathname of a file containing the proxy pre-shared key, used for encrypted communications with Zabbix server.

TLSPSKIdentity

The pre-shared key identity string, used for encrypted communications with Zabbix server.

TLSServerCertIssuer

The allowed server certificate issuer.

TLSServerCertSubject

The allowed server certificate subject.

TmpDir

The temporary directory.

Default: `/tmp`

TrapperTimeout

How many seconds the trapper may spend processing new data.

Default: 300
 Range: 1-300

UnavailableDelay

How often a host is checked for availability during the unavailability period in seconds.

Default: 60
 Range: 1-3600

UnreachableDelay

How often a host is checked for availability during the unreachability period in seconds.

Default: 15
 Range: 1-3600

UnreachablePeriod

After how many seconds of unreachability treat a host as unavailable.

Default: 45
 Range: 1-3600

User

Drop privileges to a specific, existing user on the system.
Only has effect if run as 'root' and AllowRoot is disabled.

Default: `zabbix`

Vault

The vault provider:
HashiCorp - HashiCorp KV Secrets Engine version 2
CyberArk - CyberArk Central Credential Provider
Must match the vault provider set in the frontend.

Default: `HashiCorp`

VaultDBPath

The location from where database credentials should be retrieved by keys. Depending on the vault, can be vault path or query.
The keys used for HashiCorp are 'password' and 'username'.

Example:

```
secret/zabbix/database
```

The keys used for CyberArk are 'Content' and 'UserName'.

Example:

```
AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_proxy_database
```

This option can only be used if DBUser and DBPassword are not specified.

VaultTLSCertFile

The name of the SSL certificate file used for client authentication. The certificate file must be in PEM1 format.
If the certificate file contains also the private key, leave the SSL key file field empty.
The directory containing this file is specified by the SSLCertLocation configuration parameter.
This option can be omitted, but is recommended for CyberArkCCP vault.

VaultTLSKeyFile

The name of the SSL private key file used for client authentication. The private key file must be in PEM1 format.
The directory containing this file is specified by the SSLKeyLocation configuration parameter.
This option can be omitted, but is recommended for CyberArkCCP vault.

VaultToken

The HashiCorp vault authentication token that should have been generated exclusively for Zabbix proxy with read-only permission to the path specified in the optional VaultDBPath configuration parameter.
It is an error if VaultToken and the VAULT_TOKEN environment variable are defined at the same time.

Mandatory: Yes, if Vault is set to *HashiCorp*; otherwise no

VaultURL

The vault server HTTP[S] URL. The system-wide CA certificates directory will be used if SSLCALocation is not specified.

Default: `https://127.0.0.1:8200`

VMwareCacheSize

The shared memory size for storing VMware data.
A VMware internal check zabbix[vmware,buffer,...] can be used to monitor the VMware cache usage (see Internal checks).
Note that shared memory is not allocated if there are no vmware collector instances configured to start.

Default: `8M`
 Range: 256K-2G

VMwareFrequency

The delay in seconds between data gathering from a single VMware service.
This delay should be set to the least update interval of any VMware monitoring item.

Default: 60
 Range: 10-86400

VMwarePerfFrequency

The delay in seconds between performance counter statistics retrieval from a single VMware service.
This delay should be set to the least update interval of any VMware monitoring item that uses VMware performance counters.

Default: 60
 Range: 10-86400

VMwareTimeout

The maximum number of seconds a vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor).

Default: 10
 Range: 1-300

3 Zabbix agent (UNIX)

Overview

The parameters supported by the Zabbix agent configuration file (zabbix_agentd.conf) are listed in this section.

The parameters are listed without additional information. Click on the parameter to see the full details.

Parameter	Description
Alias	Sets an alias for an item key.
AllowKey	Allow the execution of those item keys that match a pattern.
AllowRoot	Allow the agent to run as 'root'.
BufferSend	Do not keep data longer than N seconds in buffer.
BufferSize	The maximum number of values in the memory buffer.
DebugLevel	The debug level.
DenyKey	Deny the execution of those item keys that match a pattern.
EnableRemoteCommands	Whether remote commands from Zabbix server are allowed.
HeartbeatFrequency	The frequency of heartbeat messages in seconds.
HostInterface	An optional parameter that defines the host interface.
HostInterfaceItem	An optional parameter that defines an item used for getting the host interface.
HostMetadata	An optional parameter that defines the host metadata.
HostMetadataItem	An optional parameter that defines a Zabbix agent item used for getting the host metadata.
Hostname	An optional parameter that defines the hostname.
HostnameItem	An optional parameter that defines a Zabbix agent item used for getting the hostname.
Include	You may include individual files or all files in a directory in the configuration file.
ListenBacklog	The maximum number of pending connections in the TCP queue.
ListenIP	A list of comma-delimited IP addresses that the agent should listen on.
ListenPort	The agent will listen on this port for connections from the server.
LoadModule	The module to load at agent startup.
LoadModulePath	The full path to the location of agent modules.
LogFile	The name of the log file.
LogFileSize	The maximum size of the log file.
LogRemoteCommands	Enable logging of executed shell commands as warnings.
LogType	The type of the log output.
MaxLinesPerSecond	The maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'logrt' active checks.
PidFile	The name of the PID file.
RefreshActiveChecks	How often the list of active checks is refreshed.
Server	A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies.
ServerActive	The Zabbix server/proxy address or cluster configuration to get active checks from.
SourceIP	The source IP address.
StartAgents	The number of pre-forked instances of zabbix_agentd that process passive checks.
Timeout	Spend no more than Timeout seconds on processing.
TLSAccept	What incoming connections to accept.
TLSCAFile	The full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	The full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix components.
TLSCipherAll	The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.
TLSCipherAll13	The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.
TLSCipherCert	The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate-based encryption.
TLSCipherCert13	The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption.
TLSCipherPSK	The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for PSK-based encryption.

Parameter	Description
TLSCipherPSK13	The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for PSK-based encryption.
TLSConnect	How the agent should connect to Zabbix server or proxy.
TLSCRLFile	The full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.
TLSKeyFile	The full pathname of a file containing the agent private key, used for encrypted communications between Zabbix components.
TLSPSKFile	The full pathname of a file containing the agent pre-shared key, used for encrypted communications with Zabbix server.
TLSPSKIdentity	The pre-shared key identity string, used for encrypted communications with Zabbix server.
TLSServerCertIssuer	The allowed server (proxy) certificate issuer.
TLSServerCertSubject	The allowed server (proxy) certificate subject.
UnsafeUserParameters	Allow all characters to be passed in arguments to user-defined parameters.
User	Drop privileges to a specific, existing user on the system.
UserParameter	A user-defined parameter to monitor.
UserParameterDir	The default search path for UserParameter commands.

All parameters are non-mandatory unless explicitly stated that the parameter is mandatory.

Note that:

- The default values reflect daemon defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

Parameter details

Alias

Sets an alias for an item key. It can be used to substitute a long and complex item key with a shorter and simpler one.
 Multiple *Alias* parameters may be present. Multiple parameters with the same *Alias* key are not allowed.
 Different *Alias* keys may reference the same item key.
 Aliases can be used in *HostMetadataItem* but not in *HostnameItem* parameter.

Example 1: Retrieving the ID of user 'zabbix'.

```
Alias=zabbix.userid:vfs.file.regexp[/etc/passwd,"^zabbix:.:([0-9]+)",,,,\1]
```

Now the **zabbix.userid** shorthand key may be used to retrieve data.

Example 2: Getting CPU utilization with default and custom parameters.

```
Alias=cpu.util:system.cpu.util
Alias=cpu.util[*]:system.cpu.util[*]
```

This allows use the **cpu.util** key to get CPU utilization percentage with default parameters as well as use **cpu.util[all, idle, avg15]** to get specific data about CPU utilization.

Example 3: Running multiple low-level discovery rules processing the same discovery items.

```
Alias=vfs.fs.discovery[*]:vfs.fs.discovery
```

Now it is possible to set up several discovery rules using **vfs.fs.discovery** with different parameters for each rule, e.g., **vfs.fs.discovery[foo]**, **vfs.fs.discovery[bar]**, etc.

AllowKey

Allow the execution of those item keys that match a pattern. The key pattern is a wildcard expression that supports the "*" character to match any number of any characters.
Multiple key matching rules may be defined in combination with DenyKey. The parameters are processed one by one according to their appearance order. See also: Restricting agent checks.

AllowRoot

Allow the agent to run as 'root'. If disabled and the agent is started by 'root', the agent will try to switch to user 'zabbix' instead. Has no effect if started under a regular user.

Default: 0
 Values: 0 - do not allow; 1 - allow

BufferSend

Do not keep data longer than N seconds in buffer.

Default: 5
 Range: 1-3600

BufferSize

The maximum number of values in the memory buffer. The agent will send all collected data to the Zabbix server or proxy if the buffer is full.

Default: 100
 Range: 2-65535

DebugLevel

Specify the debug level:
0 - basic information about starting and stopping of Zabbix processes
1 - critical information;
2 - error information;
3 - warnings;
4 - for debugging (produces lots of information);
5 - extended debugging (produces even more information).

Default: 3
 Range: 0-5

DenyKey

Deny the execution of those item keys that match a pattern. The key pattern is a wildcard expression that supports the "*" character to match any number of any characters.
Multiple key matching rules may be defined in combination with AllowKey. The parameters are processed one by one according to their appearance order. See also: Restricting agent checks.

EnableRemoteCommands

Whether remote commands from Zabbix server are allowed. This parameter is **deprecated**, use AllowKey=system.run[*] or DenyKey=system.run[*] instead.
It is an internal alias for AllowKey/DenyKey parameters depending on value:
0 - DenyKey=system.run[*]
1 - AllowKey=system.run[*]

Default: 0
 Values: 0 - do not allow, 1 - allow

HeartbeatFrequency

The frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks.
0 - heartbeat messages disabled.

Default: 60
 Range: 0-3600

HostInterface

An optional parameter that defines the host interface. The host interface is used at host autoregistration process. If not defined, the value will be acquired from HostInterfaceItem.
The agent will issue an error and not start if the value is over the limit of 255 characters.

Range: 0-255 characters

HostInterfaceItem

An optional parameter that defines an item used for getting the host interface.
Host interface is used at host autoregistration process.
During an autoregistration request the agent will log a warning message if the value returned by the specified item is over the limit of 255 characters.
The system.run[] item is supported regardless of AllowKey/DenyKey values.
This option is only used when HostInterface is not defined.

HostMetadata

An optional parameter that defines host metadata. Host metadata is used only at host autoregistration process (active agent). If not defined, the value will be acquired from HostMetadataItem.
The agent will issue an error and not start if the specified value is over the limit of 2034 bytes or a non-UTF-8 string.

Range: 0-2034 bytes

HostMetadataItem

An optional parameter that defines a Zabbix agent item used for getting host metadata. This option is only used when HostMetadata is not defined. User parameters and aliases are supported. The system.run[] item is supported regardless of AllowKey/DenyKey values.
The HostMetadataItem value is retrieved on each autoregistration attempt and is used only at host autoregistration process (active agent).
During an autoregistration request the agent will log a warning message if the value returned by the specified item is over the limit of 65535 UTF-8 code points. The value returned by the item must be a UTF-8 string otherwise it will be ignored.

Hostname

A list of comma-delimited, unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. The value is acquired from HostnameItem if undefined.
Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128 characters per hostname, 2048 characters for the entire line.

Default: Set by HostnameItem

HostnameItem

An optional parameter that defines a Zabbix agent item used for getting the host name. This option is only used when Hostname is not defined. User parameters or aliases are not supported, but the system.run[] item is supported regardless of AllowKey/DenyKey values.

Default: `system.hostname`

Include

You may include individual files or all files in a directory in the configuration file. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching.
See special notes about limitations.

Example:

`Include=/absolute/path/to/config/files/*.conf`

ListenBacklog

The maximum number of pending connections in the TCP queue.
The default value is a hard-coded constant, which depends on the system.
The maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.

Default: `SOMAXCONN`
 Range: 0 - INT_MAX

ListenIP

A list of comma-delimited IP addresses that the agent should listen on.

Default: `0.0.0.0`

ListenPort

The agent will listen on this port for connections from the server.

Default: 10050
 Range: 1024-32767

LoadModule

The module to load at agent startup. Modules are used to extend the functionality of the agent. The module must be located in the directory specified by LoadModulePath or the path must precede the module name. If the preceding path is absolute (starts with '/') then LoadModulePath is ignored.
Formats:
LoadModule=<module.so>
LoadModule=<path/module.so>
LoadModule=</ab
is allowed to include multiple LoadModule parameters.

LoadModulePath

The full path to the location of agent modules. The default depends on compilation options.

LogFile

The name of the log file.

Mandatory: Yes, if LogType is set to *file*; otherwise no

LogFileSize

The maximum size of a log file in MB.
0 - disable automatic log rotation.
Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

Default: 1
 Range: 0-1024

LogRemoteCommands

Enable logging of the executed shell commands as warnings. Commands will be logged only if executed remotely. Log entries will not be created if system.run[] is launched locally by HostMetadataItem, HostInterfaceItem or HostnameItem parameters.

Default: 0
 Values: 0 - disabled, 1 - enabled

LogType

The type of the log output:
file - write log to the file specified by LogFile parameter;
system - write log to syslog;
console - write log to standard output.

Default: `file`

MaxLinesPerSecond

The maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'logrt' active checks. The provided value will be overridden by the 'maxlines' parameter, provided in the 'log' or 'logrt' item key.
Note: Zabbix will process 10 times more new lines than set in *MaxLinesPerSecond* to seek the required string in log items.

Default: 20
 Range: 1-1000

PidFile

The name of the PID file.

Default: `/tmp/zabbix_agentd.pid`

RefreshActiveChecks

How often the list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted in 60 seconds.

Default: 5
 Range: 1-86400

Server

A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies. Incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. Note that "IPv4-compatible IPv6 addresses" (0000::/96 prefix) are supported but deprecated by RFC4291. Spaces are allowed.

Example:

```
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

Mandatory: yes, if StartAgents is not explicitly set to 0

ServerActive

The Zabbix server/proxy address or cluster configuration to get active checks from. The server/proxy address is an IP address or DNS name and optional port separated by colon.
Cluster configuration is one or more server addresses separated by semicolon. Multiple Zabbix servers/clusters and Zabbix proxies can be specified, separated by comma. More than one Zabbix proxy should not be specified from each Zabbix server/cluster. If Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified.
Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed.
If the port is not specified, default port is used.
IPv6 addresses must be enclosed in square brackets if port for that host is specified. If port is not specified, square brackets for IPv6 addresses are optional.
If this parameter is not specified, active checks are disabled.

Example for Zabbix proxy:

```
ServerActive=127.0.0.1:10051
```

Example for multiple servers:

```
ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
```

Example for high availability:

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
```

Example for high availability with two clusters and one server:

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,z
```

SourceIP

The source IP address for:
- outgoing connections to Zabbix server or Zabbix proxy;
- making connections while executing some items (web.page.get, net.tcp.port, etc.).

StartAgents

The number of pre-forked instances of zabbix_agentd that process passive checks. If set to 0, passive checks are disabled and the agent will not listen on any TCP port.

Default: 3
 Range: 0-100

Timeout

Spend no more than Timeout seconds on processing.

Default: 3
 Range: 1-30

TLSAccept

What incoming connections to accept. Used for a passive checks. Multiple values can be specified, separated by comma:
unencrypted - accept connections without encryption (default)
psk - accept connections with TLS and a pre-shared key (PSK)
cert - accept connections with TLS and a certificate

Mandatory: yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection); otherwise no

TLSCAFile

The full pathname of the file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.

TLSCertFile

The full pathname of the file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.

TLSCipherAll

The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.

Example:

```
TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSCipherAll13

The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate- and PSK-based encryption.

Example for GnuTLS:

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL
```

Example for OpenSSL:

```
EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherCert

The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for certificate-based encryption.

Example for GnuTLS:

```
NONE:+VERS-TLS1.2:+ECDHE-RSA:+RSA:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

Example for OpenSSL:

```
EECDH+aRSA+AES128:RSA+aRSA+AES128
```

TLSCipherCert13

The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption.

TLSCipherPSK

The GnuTLS priority string or OpenSSL (TLS 1.2) cipher string. Override the default ciphersuite selection criteria for PSK-based encryption.

Example for GnuTLS:

```
NONE:+VERS-TLS1.2:+ECDHE-PSK:+PSK:+AES-128-GCM:+AES-128-CBC:+AEAD:+SHA256:+SHA1:+CURVE-ALL:+COMP-NULL:+SIG
```

Example for OpenSSL:

```
kECDHEPSK+AES128:kPSK+AES128
```

TLSCipherPSK13

The cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for PSK-based encryption.

Example:

```
TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

TLSConnect

How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified:
unencrypted - connect without encryption (default)
psk - connect using TLS and a pre-shared key (PSK)
cert - connect using TLS and a certificate

Mandatory: yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection); otherwise no

TLSCRLFile

The full pathname of the file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.

TLSKeyFile

The full pathname of the file containing the agent private key, used for encrypted communications between Zabbix components.

TLSPSKFile

The full pathname of the file containing the agent pre-shared key, used for encrypted communications with Zabbix server.

TLSPSKIdentity

The pre-shared key identity string, used for encrypted communications with Zabbix server.

TLSServerCertIssuer

The allowed server (proxy) certificate issuer.

TLSServerCertSubject

The allowed server (proxy) certificate subject.

UnsafeUserParameters

Allow all characters to be passed in arguments to user-defined parameters. The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @
Additionally, newline characters are not allowed.

Default: 0
 Values: 0 - do not allow, 1 - allow

User

Drop privileges to a specific, existing user on the system.
Only has effect if run as 'root' and AllowRoot is disabled.

Default: `zabbix`

UserParameter

A user-defined parameter to monitor. There can be several user-defined parameters.
Format: UserParameter=<key>,<shell command>
Note that the shell command must not return empty string or EOL only. Shell commands may have relative paths, if the UserParameterDir parameter is specified.

Example:

```
UserParameter=system.test,who|wc -l
UserParameter=check_cpu,./custom_script.sh
```

UserParameterDir

The default search path for UserParameter commands. If used, the agent will change its working directory to the one specified here before executing a command. Thereby, UserParameter commands can have a relative `./` prefix instead of a full path.
Only one entry is allowed.

Example:

```
UserParameterDir=/opt/myscripts
```

See also

1. Differences in the Zabbix agent configuration for active and passive checks starting from version 2.0.0

4 Zabbix agent 2 (UNIX)

Overview

Zabbix agent 2 is a new generation of Zabbix agent and may be used in place of Zabbix agent.

The parameters supported by the Zabbix agent 2 configuration file (zabbix_agent2.conf) are listed in this section.

The parameters are listed without additional information. Click on the parameter to see the full details.

Parameter	Description
Alias	Sets an alias for an item key.
AllowKey	Allow the execution of those item keys that match a pattern.
BufferSend	Do not keep data longer than N seconds in buffer.
BufferSize	The maximum number of values in the memory buffer.
ControlSocket	The control socket, used to send runtime commands with the '-R' option.
DebugLevel	The debug level.
DenyKey	Deny the execution of those item keys that match a pattern.
EnablePersistentBuffer	Enable the usage of local persistent storage for active items.
ForceActiveChecksOnStart	Perform active checks immediately after the restart for the first received configuration.
HeartbeatFrequency	The frequency of heartbeat messages in seconds.
HostInterface	An optional parameter that defines the host interface.
HostInterfaceItem	An optional parameter that defines an item used for getting the host interface.
HostMetadata	An optional parameter that defines the host metadata.
HostMetadataItem	An optional parameter that defines a Zabbix agent item used for getting the host metadata.
Hostname	An optional parameter that defines the hostname.
HostnameItem	An optional parameter that defines a Zabbix agent item used for getting the hostname.
Include	You may include individual files or all files in a directory in the configuration file.
ListenIP	A list of comma-delimited IP addresses that the agent should listen on.
ListenPort	The agent will listen on this port for connections from the server.
LogFile	The name of the log file.
LogFileSize	The maximum size of the log file.
LogType	The type of the log output.
PersistentBufferFile	The file where Zabbix agent 2 should keep the SQLite database.
PersistentBufferPeriod	The time period for which data should be stored when there is no connection to the server or proxy.
PidFile	The name of the PID file.
Plugins.<PluginName>.System.Capacity	The capacity of checks per plugin that can be executed at the same time.
Plugins.Log.MaxLinesPerSecond	The maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'logrt' active checks.
Plugins.SystemRun.LogRemoteCommands	Enable the logging of the executed shell commands as warnings.
PluginSocket	The path to the UNIX socket for loadable plugin communications.
PluginTimeout	The timeout for connections with loadable plugins, in seconds.
RefreshActiveChecks	How often the list of active checks is refreshed.
Server	A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies.
ServerActive	The Zabbix server/proxy address or cluster configuration to get active checks from.
SourceIP	The source IP address.
StatusPort	If set, the agent will listen on this port for HTTP status requests (http://localhost:<port>/status).
Timeout	Spend no more than Timeout seconds on processing.
TLSAccept	What incoming connections to accept.
TLSCAFile	The full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	The full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix components.
TLSConnect	How the agent should connect to Zabbix server or proxy.
TLSCRLFile	The full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.
TLSKeyFile	The full pathname of a file containing the agent private key, used for encrypted communications between Zabbix components.
TLSPSKFile	The full pathname of a file containing the agent pre-shared key, used for encrypted communications with Zabbix server.
TLSPSKIdentity	The pre-shared key identity string, used for encrypted communications with Zabbix server.
TLSServerCertIssuer	The allowed server (proxy) certificate issuer.
TLSServerCertSubject	The allowed server (proxy) certificate subject.
UnsafeUserParameters	Allow all characters to be passed in arguments to user-defined parameters.
UserParameter	A user-defined parameter to monitor.
UserParameterDir	The default search path for UserParameter commands.

All parameters are non-mandatory unless explicitly stated that the parameter is mandatory.

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

Parameter details

Alias

Sets an alias for an item key. It can be used to substitute a long and complex item key with a shorter and simpler one.
 Multiple *Alias* parameters may be present. Multiple parameters with the same *Alias* key are not allowed.
 Different *Alias* keys may reference the same item key.
 Aliases can be used in *HostMetadataItem* but not in *HostnameItem* parameter.

Example 1: Retrieving the ID of user 'zabbix'.

```
Alias=zabbix.userid:vfs.file.regexp[/etc/passwd,"^zabbix:.:([0-9]+)",,,,\1]
```

Now the **zabbix.userid** shorthand key may be used to retrieve data.

Example 2: Getting CPU utilization with default and custom parameters.

```
Alias=cpu.util:system.cpu.util
Alias=cpu.util[*]:system.cpu.util[*]
```

This allows use the **cpu.util** key to get CPU utilization percentage with default parameters as well as use **cpu.util[all, idle, avg15]** to get specific data about CPU utilization.

Example 3: Running multiple low-level discovery rules processing the same discovery items.

```
Alias=vfs.fs.discovery[*]:vfs.fs.discovery
```

Now it is possible to set up several discovery rules using **vfs.fs.discovery** with different parameters for each rule, e.g., **vfs.fs.discovery[foo]**, **vfs.fs.discovery[bar]**, etc.

AllowKey

Allow the execution of those item keys that match a pattern. The key pattern is a wildcard expression that supports the "*" character to match any number of any characters.
Multiple key matching rules may be defined in combination with DenyKey. The parameters are processed one by one according to their appearance order. See also: Restricting agent checks.

BufferSend

The time interval in seconds which determines how often values are sent from the buffer to Zabbix server. Note that if the buffer is full, the data will be sent sooner.

Default: 5
 Range: 1-3600

BufferSize

The maximum number of values in the memory buffer. The agent will send all collected data to the Zabbix server or proxy if the buffer is full. This parameter should only be used if persistent buffer is disabled (*EnablePersistentBuffer=0*).

Default: 100
 Range: 2-65535

ControlSocket

The control socket, used to send runtime commands with the '-R' option.

Default: `/tmp/agent.sock`

DebugLevel

Specify the debug level:
0 - basic information about starting and stopping of Zabbix processes
1 - critical information;
2 - error information;
3 - warnings;
4 - for debugging (produces lots of information);
5 - extended debugging (produces even more information).

Default: 3
 Range: 0-5

DenyKey

Deny the execution of those item keys that match a pattern. The key pattern is a wildcard expression that supports the "*" character to match any number of any characters.
Multiple key matching rules may be defined in combination with AllowKey. The parameters are processed one by one according to their appearance order. See also: Restricting agent checks.

EnablePersistentBuffer

Enable the usage of local persistent storage for active items. If persistent storage is disabled, the memory buffer will be used.

Default: 0
 Values: 0 - disabled, 1 - enabled

ForceActiveChecksOnStart

Perform active checks immediately after the restart for the first received configuration. Also available as a per-plugin configuration parameter, for example: `Plugins.Uptime.System.ForceActiveChecksOnStart=1`

Default: 0
 Values: 0 - disabled, 1 - enabled

HeartbeatFrequency

The frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks.
0 - heartbeat messages disabled.

Default: 60
 Range: 0-3600

HostInterface

An optional parameter that defines the host interface. The host interface is used at host autoregistration process. If not defined, the value will be acquired from HostInterfaceItem.
The agent will issue an error and not start if the value is over the limit of 255 characters.

Range: 0-255 characters

HostInterfaceItem

An optional parameter that defines an item used for getting the host interface.
Host interface is used at host autoregistration process. This option is only used when HostInterface is not defined.
The system.run[] item is supported regardless of AllowKey/DenyKey values.
During an autoregistration request the agent will log a warning message if the value returned by the specified item is over the limit of 255 characters.

HostMetadata

An optional parameter that defines host metadata. Host metadata is used only at host autoregistration process. If not defined, the value will be acquired from HostMetadataItem.
The agent will issue an error and not start if the specified value is over the limit of 2034 bytes or a non-UTF-8 string.

Range: 0-2034 bytes

HostMetadataItem

An optional parameter that defines an item used for getting host metadata. This option is only used when HostMetadata is not defined. User parameters and aliases are supported. The system.run[] item is supported regardless of AllowKey/DenyKey values.
The HostMetadataItem value is retrieved on each autoregistration attempt and is used only at host autoregistration process.
During an autoregistration request the agent will log a warning message if the value returned by the specified item is over the limit of 65535 UTF-8 code points. The value returned by the item must be a UTF-8 string otherwise it will be ignored.

Hostname

A list of comma-delimited, unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. The value is acquired from HostnameItem if undefined.
Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128 characters per hostname, 2048 characters for the entire line.

Default: Set by HostnameItem

HostnameItem

An optional parameter that defines an item used for getting the host name. This option is only used when Hostname is not defined. User parameters or aliases are not supported, but the system.run[] item is supported regardless of AllowKey/DenyKey values.

Default: `system.hostname`

Include

You may include individual files or all files in a directory in the configuration file. During the installation Zabbix will create the include directory in /usr/local/etc, unless modified during the compile time. The path can be relative to the *zabbix_agent2.conf* file location.
To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching.
See special notes about limitations.

Example:

```
Include=/absolute/path/to/config/files/*.conf
```

ListenIP

A list of comma-delimited IP addresses that the agent should listen on. The first IP address is sent to the Zabbix server, if connecting to it, to retrieve the list of active checks.

Default: `0.0.0.0`

ListenPort

The agent will listen on this port for connections from the server.

Default: 10050
 Range: 1024-32767

LogFile

The name of the log file.

Default: `/tmp/zabbix_agent2.log`
 Mandatory: Yes, if LogType is set to *file*; otherwise no

LogFileSize

The maximum size of a log file in MB.
0 - disable automatic log rotation.
Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

Default: 1
 Range: 0-1024

LogType

The type of the log output:
file - write log to the file specified by LogFile parameter;
system - write log to syslog;
console - write log to standard output

Default: `file`

PersistentBufferFile

The file where Zabbix agent 2 should keep the SQLite database. Must be a full filename. This parameter is only used if persistent buffer is enabled (*EnablePersistentBuffer=1*).

PersistentBufferPeriod

The time period for which data should be stored when there is no connection to the server or proxy. Older data will be lost. Log data will be preserved. This parameter is only used if persistent buffer is enabled (*EnablePersistentBuffer=1*).

Default: `1h`
 Range: 1m-365d

PidFile

The name of the PID file.

Default: `/tmp/zabbix_agent2.pid`

Plugins.<PluginName>.System.Capacity

The limit of checks per `<PluginName>` plugin that can be executed at the same time.

Default: 100 Range: 1-1000

Plugins.Log.MaxLinesPerSecond

The maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'logrt' active checks. The provided value will be overridden by the 'maxlines' parameter, provided in the 'log' and 'logrt' item key.
Note: Zabbix will process 10 times more new lines than set in *MaxLinesPerSecond* to seek the required string in log items.

Default: 20
 Range: 1-1000

Plugins.SystemRun.LogRemoteCommands

Enable the logging of the executed shell commands as warnings. The commands will be logged only if executed remotely. Log entries will not be created if system.run[] is launched locally by the HostMetadataItem, HostInterfaceItem or HostnameItem parameters.

Default: 0
 Values: 0 - disabled, 1 - enabled

PluginSocket

The path to the UNIX socket for loadable plugin communications.

Default: `/tmp/agent.plugin.sock`

PluginTimeout

The timeout for connections with loadable plugins, in seconds.

Default: `Timeout`
 Range: 1-30

RefreshActiveChecks

How often the list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted in 60 seconds.

Default: 5
 Range: 1-86400

Server

A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers or Zabbix proxies. Incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. Spaces are allowed.

Example:

```
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

Mandatory: yes

ServerActive

The Zabbix server/proxy address or cluster configuration to get active checks from. The server/proxy address is an IP address or DNS name and optional port separated by colon.
The cluster configuration is one or more server addresses separated by semicolon. Multiple Zabbix servers/clusters and Zabbix proxies can be specified, separated by comma. More than one Zabbix proxy should not be specified from each Zabbix server/cluster. If a Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified.
Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed.
If the port is not specified, default port is used.
IPv6 addresses must be enclosed in square brackets if port for that host is specified. If port is not specified, square brackets for IPv6 addresses are optional.
If this parameter is not specified, active checks are disabled.

Example for Zabbix proxy:

```
ServerActive=127.0.0.1:10051
```

Example for multiple servers:

```
ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
```

Example for high availability:

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
```

Example for high availability with two clusters and one server:

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,z
```

SourceIP

The source IP address for:
- outgoing connections to Zabbix server or Zabbix proxy;
- making connections while executing some items (web.page.get, net.tcp.port, etc.).

StatusPort

If set, the agent will listen on this port for HTTP status requests (http://localhost:<port>/status).

Range: 1024-32767

Timeout

Spend no more than Timeout seconds on processing.

Default: 3
 Range: 1-30

TLSAccept

The incoming connections to accept. Used for passive checks. Multiple values can be specified, separated by comma:
unencrypted - accept connections without encryption (default)
psk - accept connections with TLS and a pre-shared key (PSK)
cert - accept connections with TLS and a certificate

Mandatory: yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection); otherwise no

TLSCAFile

The full pathname of the file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.

TLSCertFile

The full pathname of the file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.

TLSConnect

How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified:
unencrypted - connect without encryption (default)
psk - connect using TLS and a pre-shared key (PSK)
cert - connect using TLS and a certificate

Mandatory: yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection); otherwise no

TLSCRLFile

The full pathname of the file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.

TLSKeyFile

The full pathname of the file containing the agent private key, used for encrypted communications between Zabbix components.

TLSPSKFile

The full pathname of the file containing the agent pre-shared key, used for encrypted communications with Zabbix server.

TLSPSKIdentity

The pre-shared key identity string, used for encrypted communications with Zabbix server.

TLSServerCertIssuer

The allowed server (proxy) certificate issuer.

TLSServerCertSubject

The allowed server (proxy) certificate subject.

UnsafeUserParameters

Allow all characters to be passed in arguments to user-defined parameters. The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @
Additionally, newline characters are not allowed.

Default: 0
 Values: 0 - do not allow, 1 - allow

UserParameter

A user-defined parameter to monitor. There can be several user-defined parameters.
Format: UserParameter=<key>,<shell command>
Note that the shell command must not return empty string or EOL only. Shell commands may have relative paths, if the UserParameterDir parameter is specified.

Example:

```
UserParameter=system.test,who|wc -l
UserParameter=check_cpu,./custom_script.sh
```

UserParameterDir

The default search path for UserParameter commands. If used, the agent will change its working directory to the one specified here before executing a command. Thereby, UserParameter commands can have a relative `./` prefix instead of a full path.
Only one entry is allowed.

Example:

```
UserParameterDir=/opt/myscripts
```

5 Zabbix agent (Windows)

Overview

The parameters supported by the Windows Zabbix agent configuration file (zabbix_agentd.conf) are listed in this section.

The parameters are listed without additional information. Click on the parameter to see the full details.

Parameter	Description
Alias	Sets an alias for an item key.
AllowKey	Allow the execution of those item keys that match a pattern.
BufferSend	Do not keep data longer than N seconds in buffer.

Parameter	Description
BufferSize	The maximum number of values in the memory buffer.
DebugLevel	The debug level.
DenyKey	Deny the execution of those item keys that match a pattern.
EnableRemoteCommands	Whether remote commands from Zabbix server are allowed.
HeartbeatFrequency	The frequency of heartbeat messages in seconds.
HostInterface	An optional parameter that defines the host interface.
HostInterfaceItem	An optional parameter that defines an item used for getting the host interface.
HostMetadata	An optional parameter that defines the host metadata.
HostMetadataItem	An optional parameter that defines a Zabbix agent item used for getting the host metadata.
Hostname	An optional parameter that defines the hostname.
HostnameItem	An optional parameter that defines a Zabbix agent item used for getting the hostname.
Include	You may include individual files or all files in a directory in the configuration file.
ListenBacklog	The maximum number of pending connections in the TCP queue.
ListenIP	A list of comma-delimited IP addresses that the agent should listen on.
ListenPort	The agent will listen on this port for connections from the server.
LogFile	The name of the log file.
LogFileSize	The maximum size of the log file.
LogRemoteCommands	Enable logging of executed shell commands as warnings.
LogType	The type of the log output.
MaxLinesPerSecond	The maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'logrt' active checks.
PerfCounter	Defines a new parameter <parameter_name> which is the average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds).
PerfCounterEn	Defines a new parameter <parameter_name> which is the average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds). Compared to PerfCounter, the perfcounter paths must be in English.
RefreshActiveChecks	How often the list of active checks is refreshed.
Server	A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies.
ServerActive	The Zabbix server/proxy address or cluster configuration to get active checks from.
SourceIP	The source IP address.
StartAgents	The number of pre-forked instances of zabbix_agentd that process passive checks.
Timeout	Spend no more than Timeout seconds on processing.
TLSAccept	What incoming connections to accept.
TLSCAFile	The full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	The full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix components.
TLSConnect	How the agent should connect to Zabbix server or proxy.
TLSCRLFile	The full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.
TLSKeyFile	The full pathname of a file containing the agent private key, used for encrypted communications between Zabbix components.
TLSPSKFile	The full pathname of a file containing the agent pre-shared key, used for encrypted communications with Zabbix server.
TLSPSKIdentity	The pre-shared key identity string, used for encrypted communications with Zabbix server.
TLSServerCertIssuer	The allowed server (proxy) certificate issuer.
TLSServerCertSubject	The allowed server (proxy) certificate subject.
UnsafeUserParameters	Allow all characters to be passed in arguments to user-defined parameters.
UserParameter	A user-defined parameter to monitor.
UserParameterDir	The default search path for UserParameter commands.

All parameters are non-mandatory unless explicitly stated that the parameter is mandatory.

Note that:

- The default values reflect daemon defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

Parameter details

Alias

Sets an alias for an item key. It can be used to substitute a long and complex item key with a shorter and simpler one.
 Multiple *Alias* parameters may be present. Multiple parameters with the same *Alias* key are not allowed.
 Different *Alias* keys may reference the same item key.
 Aliases can be used in *HostMetadataItem* but not in *HostnameItem* or *PerfCounter* parameter.

Example 1: Retrieving the paging file usage in percentage from the server.

```
Alias=pg_usage:perf_counter[\Paging File(_Total)\% Usage]
```

Now the shorthand key **pg_usage** may be used to retrieve data.

Example 2: Getting the CPU load with default and custom parameters.

```
Alias=cpu.load:system.cpu.load
Alias=cpu.load[*]:system.cpu.load[*]
```

This allows use **cpu.load** key to get the CPU load with default parameters as well as use **cpu.load[percpu,avg15]** to get specific data about the CPU load.

Example 3: Running multiple low-level discovery rules processing the same discovery items.

```
Alias=vfs.fs.discovery[*]:vfs.fs.discovery
```

Now it is possible to set up several discovery rules using **vfs.fs.discovery** with different parameters for each rule, e.g., **vfs.fs.discovery[foo]**, **vfs.fs.discovery[bar]**, etc.

AllowKey

Allow the execution of those item keys that match a pattern. The key pattern is a wildcard expression that supports the "*" character to match any number of any characters.
Multiple key matching rules may be defined in combination with DenyKey. The parameters are processed one by one according to their appearance order. See also: Restricting agent checks.

BufferSend

Do not keep data longer than N seconds in buffer.

Default: 5
 Range: 1-3600

BufferSize

The maximum number of values in the memory buffer. The agent will send all collected data to the Zabbix server or proxy if the buffer is full.

Default: 100
 Range: 2-65535

DebugLevel

Specify the debug level:
0 - basic information about starting and stopping of Zabbix processes
1 - critical information;
2 - error information;
3 - warnings;
4 - for debugging (produces lots of information);
5 - extended debugging (produces even more information).

Default: 3
 Range: 0-5

DenyKey

Deny the execution of those item keys that match a pattern. The key pattern is a wildcard expression that supports the "*" character to match any number of any characters.
Multiple key matching rules may be defined in combination with AllowKey. The parameters are processed one by one according to their appearance order. See also: Restricting agent checks.

EnableRemoteCommands

Whether remote commands from Zabbix server are allowed. This parameter is **deprecated**, use AllowKey=system.run[*] or DenyKey=system.run[*] instead.
It is an internal alias for AllowKey/DenyKey parameters depending on value:
0 - DenyKey=system.run[*]
1 - AllowKey=system.run[*]

Default: 0
 Values: 0 - do not allow, 1 - allow

HeartbeatFrequency

The frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks.
0 - heartbeat messages disabled.

Default: 60
 Range: 0-3600

HostInterface

An optional parameter that defines the host interface. The host interface is used at host autoregistration process. If not defined, the value will be acquired from HostInterfaceItem.
The agent will issue an error and not start if the value is over the limit of 255 characters.

Range: 0-255 characters

HostInterfaceItem

An optional parameter that defines an item used for getting the host interface.
Host interface is used at host autoregistration process.
During an autoregistration request the agent will log a warning message if the value returned by the specified item is over the limit of 255 characters.
The system.run[] item is supported regardless of AllowKey/DenyKey values.
This option is only used when HostInterface is not defined.

HostMetadata

An optional parameter that defines host metadata. Host metadata is used only at host autoregistration process (active agent). If not defined, the value will be acquired from HostMetadataItem.
The agent will issue an error and not start if the specified value is over the limit of 2034 bytes or a non-UTF-8 string.

Range: 0-2034 bytes

HostMetadataItem

An optional parameter that defines a Zabbix agent item used for getting host metadata. This option is only used when HostMetadata is not defined. User parameters, performance counters and aliases are supported. The system.run[] item is supported regardless of AllowKey/DenyKey values.
The HostMetadataItem value is retrieved on each autoregistration attempt and is used only at host autoregistration process (active agent).
During an autoregistration request the agent will log a warning message if the value returned by the specified item is over the limit of 65535 UTF-8 code points. The value returned by the item must be a UTF-8 string otherwise it will be ignored.

Hostname

A list of comma-delimited, unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. The value is acquired from HostnameItem if undefined.
Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128 characters per hostname, 2048 characters for the entire line.

Default: Set by HostnameItem

HostnameItem

An optional parameter that defines a Zabbix agent item used for getting the host name. This option is only used when Hostname is not defined. User parameters, performance counters or aliases are not supported, but the system.run[] item is supported regardless of AllowKey/DenyKey values.
See also a more detailed description.

Default: `system.hostname`

Include

You may include individual files or all files in a directory in the configuration file (located in `C:\Program Files\Zabbix Agent` by default if Zabbix agent is installed using Windows MSI installer packages; located in the folder specified during installation if Zabbix agent is installed as a zip archive). All included files must have correct syntax, otherwise agent will not start.
To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching.
See special notes about limitations.

Example:

```
Include=C:\Program Files\Zabbix Agent\zabbix_agentd.d\*.conf
```

ListenBacklog

The maximum number of pending connections in the TCP queue.
The default value is a hard-coded constant, which depends on the system.
The maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.

Default: `SOMAXCONN`
 Range: 0 - INT_MAX

ListenIP

A list of comma-delimited IP addresses that the agent should listen on.

Default: `0.0.0.0`

ListenPort

The agent will listen on this port for connections from the server.

Default: 10050
 Range: 1024-32767

LogFile

The name of the agent log file.

Default: `C:\\zabbix_agentd.log`
 Mandatory: Yes, if LogType is set to *file*; otherwise no

LogFileSize

The maximum size of a log file in MB.
0 - disable automatic log rotation.
Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

Default: 1
 Range: 0-1024

LogRemoteCommands

Enable the logging of the executed shell commands as warnings. Commands will be logged only if executed remotely. Log entries will not be created if system.run[] is launched locally by HostMetadataItem, HostInterfaceItem or HostnameItem parameters.

Default: 0
 Values: 0 - disabled, 1 - enabled

LogType

The type of the log output:
file - write log to the file specified by LogFile parameter;
system - write log to Windows Event Log;
console - write log to standard output.

Default: `file`

MaxLinesPerSecond

The maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log', 'logrt', and 'eventlog' active checks. The provided value will be overridden by the 'maxlines' parameter, provided in the 'log', 'logrt', or 'eventlog' item key.
Note: Zabbix will process 10 times more new lines than set in *MaxLinesPerSecond* to seek the required string in log items.

Default: 20
 Range: 1-1000

PerfCounter

Defines a new parameter <parameter_name> which is the average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds).
Syntax: <parameter_name>,"<perf_counter_path>",<period>

For example, if you wish to receive the average number of processor interrupts per second for the last minute, you can define a new parameter "interrupts" as the following:

```
PerfCounter = interrupts,"\Processor(0)\Interrupts/sec",60
```

Please note the double quotes around the performance counter path. The parameter name (interrupts) is to be used as the item key when creating an item. Samples for calculating the average value will be taken every second.
You may run "typeperf -qx" to get the list of all performance counters available in Windows.

PerfCounterEn

Defines a new parameter <parameter_name> which is the average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds). Compared to PerfCounter, the perfcounter paths must be in English. Supported only on **Windows Server 2008/Vista** and later.
Syntax: <parameter_name>,"<perf_counter_path>",<period>

For example, if you wish to receive the average number of processor interrupts per second for the last minute, you can define a new parameter "interrupts" as the following:

```
PerfCounterEn = interrupts,"\Processor(0)\Interrupts/sec",60
```

Please note the double quotes around the performance counter path. The parameter name (interrupts) is to be used as the item key when creating an item. Samples for calculating the average value will be taken every second.
You can find the list of English strings by viewing the following registry key: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\009`.

RefreshActiveChecks

How often the list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted in 60 seconds.

Default: 5
 Range: 1-86400

Server

A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers or Zabbix proxies. Incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. Note that "IPv4-compatible IPv6 addresses" (0000::/96 prefix) are supported but deprecated by RFC4291. Spaces are allowed.

Example:

```
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

Mandatory: yes, if StartAgents is not explicitly set to 0

ServerActive

The Zabbix server/proxy address or cluster configuration to get active checks from. The server/proxy address is an IP address or DNS name and optional port separated by colon.
The cluster configuration is one or more server addresses separated by semicolon. Multiple Zabbix servers/clusters and Zabbix proxies can be specified, separated by comma. More than one Zabbix proxy should not be specified from each Zabbix server/cluster. If Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified.
Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed.
If the port is not specified, default port is used.
IPv6 addresses must be enclosed in square brackets if port for that host is specified. If port is not specified, square brackets for IPv6 addresses are optional.
If this parameter is not specified, active checks are disabled.

Example for Zabbix proxy:

```
ServerActive=127.0.0.1:10051
```

Example for multiple servers:

```
ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
```

Example for high availability:

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
```

Example for high availability with two clusters and one server:

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,z
```

Range: (*)

SourceIP

The source IP address for:
- outgoing connections to Zabbix server or Zabbix proxy;
- making connections while executing some items (web.page.get, net.tcp.port, etc.).

StartAgents

The number of pre-forked instances of zabbix_agentd that process passive checks. If set to 0, passive checks are disabled and the agent will not listen on any TCP port.

Default: 3
 Range: 0-63 (*)

Timeout

Spend no more than Timeout seconds on processing.

Default: 3
 Range: 1-30

TLSAccept

The incoming connections to accept. Used for passive checks. Multiple values can be specified, separated by comma:
unencrypted - accept connections without encryption (default)
psk - accept connections with TLS and a pre-shared key (PSK)
cert - accept connections with TLS and a certificate

Mandatory: yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection); otherwise no

TLSCAFile

The full pathname of the file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.

TLSCertFile

The full pathname of the file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.

TLSConnect

How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified:
unencrypted - connect without encryption (default)
psk - connect using TLS and a pre-shared key (PSK)
cert - connect using TLS and a certificate

Mandatory: yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection); otherwise no

TLSCRLFile

The full pathname of the file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.

TLSKeyFile

The full pathname of the file containing the agent private key, used for encrypted communications between Zabbix components.

TLSPSKFile

The full pathname of the file containing the agent pre-shared key, used for encrypted communications with Zabbix server.

TLSPSKIdentity

The pre-shared key identity string, used for encrypted communications with Zabbix server.

TLSServerCertIssuer

The allowed server (proxy) certificate issuer.

TLSServerCertSubject

The allowed server (proxy) certificate subject.

UnsafeUserParameters

Allow all characters to be passed in arguments to user-defined parameters. The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @
Additionally, newline characters are not allowed.

Default: 0
 Values: 0 - do not allow, 1 - allow

UserParameter

A user-defined parameter to monitor. There can be several user-defined parameters.
Format: UserParameter=<key>,<shell command>
Note that the shell command must not return empty string or EOL only. Shell commands may have relative paths, if the UserParameterDir parameter is specified.

Example:

```
UserParameter=system.test,who|wc -l
UserParameter=check_cpu,./custom_script.sh
```

UserParameterDir

The default search path for UserParameter commands. If used, the agent will change its working directory to the one specified here before executing a command. Thereby, UserParameter commands can have a relative `./` prefix instead of a full path. Only one entry is allowed.

Example:

```
UserParameterDir=/opt/myscripts
```

Note:
(*) The number of active servers listed in ServerActive plus the number of pre-forked instances for passive checks specified in StartAgents must be less than 64.

See also

1. Differences in the Zabbix agent configuration for active and passive checks starting from version 2.0.0.

6 Zabbix agent 2 (Windows)

Overview

Zabbix agent 2 is a new generation of Zabbix agent and may be used in place of Zabbix agent.

The parameters supported by the Windows Zabbix agent 2 configuration file (zabbix_agent2.conf) are listed in this section.

The parameters are listed without additional information. Click on the parameter to see the full details.

Parameter	Description
Alias	Sets an alias for an item key.
AllowKey	Allow the execution of those item keys that match a pattern.
BufferSend	Do not keep data longer than N seconds in buffer.
BufferSize	The maximum number of values in the memory buffer.
ControlSocket	The control socket, used to send runtime commands with the '-R' option.
DebugLevel	The debug level.
DenyKey	Deny the execution of those item keys that match a pattern.
EnablePersistentBuffer	Enable the usage of local persistent storage for active items.
ForceActiveChecksOnStart	Perform active checks immediately after the restart for the first received configuration.
HeartbeatFrequency	The frequency of heartbeat messages in seconds.
HostInterface	An optional parameter that defines the host interface.
HostInterfaceItem	An optional parameter that defines an item used for getting the host interface.
HostMetadata	An optional parameter that defines the host metadata.
HostMetadataItem	An optional parameter that defines a Zabbix agent item used for getting the host metadata.
Hostname	An optional parameter that defines the hostname.
HostnameItem	An optional parameter that defines a Zabbix agent item used for getting the hostname.
Include	You may include individual files or all files in a directory in the configuration file.
ListenIP	A list of comma-delimited IP addresses that the agent should listen on.
ListenPort	The agent will listen on this port for connections from the server.
LogFile	The name of the log file.
LogFileSize	The maximum size of the log file.
LogType	The type of the log output.
PersistentBufferFile	The file where Zabbix agent 2 should keep the SQLite database.
PersistentBufferPeriod	The time period for which data should be stored when there is no connection to the server or proxy.
Plugins.<PluginName>.System.Capacity	The limit of checks per plugin that can be executed at the same time.
Plugins.Log.MaxLinesPerSecond	The maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'logrt' active checks.
Plugins.SystemRun.LogRemoteCommands	Enable the logging of the executed shell commands as warnings.
PluginSocket	The path to the UNIX socket for loadable plugin communications.
PluginTimeout	The timeout for connections with loadable plugins, in seconds.
RefreshActiveChecks	How often the list of active checks is refreshed.
Server	A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies.
ServerActive	The Zabbix server/proxy address or cluster configuration to get active checks from.
SourceIP	The source IP address.
StatusPort	If set, the agent will listen on this port for HTTP status requests (http://localhost:<port>/status).
Timeout	Spend no more than Timeout seconds on processing.
TLSAccept	What incoming connections to accept.
TLSCAFile	The full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	The full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix components.
TLSConnect	How the agent should connect to Zabbix server or proxy.
TLSCRLFile	The full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.
TLSKeyFile	The full pathname of a file containing the agent private key, used for encrypted communications between Zabbix components.
TLSPSKFile	The full pathname of a file containing the agent pre-shared key, used for encrypted communications with Zabbix server.
TLSPSKIdentity	The pre-shared key identity string, used for encrypted communications with Zabbix server.
TLSServerCertIssuer	The allowed server (proxy) certificate issuer.
TLSServerCertSubject	The allowed server (proxy) certificate subject.
UnsafeUserParameters	Allow all characters to be passed in arguments to user-defined parameters.
UserParameter	A user-defined parameter to monitor.
UserParameterDir	The default search path for UserParameter commands.

All parameters are non-mandatory unless explicitly stated that the parameter is mandatory.

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;

- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported in the beginning of the line.

Parameter details

Alias

Sets an alias for an item key. It can be used to substitute a long and complex item key with a shorter and simpler one.
 Multiple *Alias* parameters may be present. Multiple parameters with the same *Alias* key are not allowed.
 Different *Alias* keys may reference the same item key.
 Aliases can be used in *HostMetadataItem* but not in the *HostnameItem* parameter.

Example 1: Retrieving the paging file usage in percentage from the server.

```
Alias=pg_usage:perf_counter[\Paging File(_Total)\% Usage]
```

Now the shorthand key **pg_usage** may be used to retrieve data.

Example 2: Getting the CPU load with default and custom parameters.

```
Alias=cpu.load:system.cpu.load
Alias=cpu.load[*]:system.cpu.load[*]
```

This allows use **cpu.load** key to get the CPU load with default parameters as well as use **cpu.load[percpu,avg15]** to get specific data about the CPU load.

Example 3: Running multiple low-level discovery rules processing the same discovery items.

```
Alias=vfs.fs.discovery[*]:vfs.fs.discovery
```

Now it is possible to set up several discovery rules using **vfs.fs.discovery** with different parameters for each rule, e.g., **vfs.fs.discovery[foo]**, **vfs.fs.discovery[bar]**, etc.

AllowKey

Allow the execution of those item keys that match a pattern. The key pattern is a wildcard expression that supports the "*" character to match any number of any characters.
Multiple key matching rules may be defined in combination with DenyKey. The parameters are processed one by one according to their appearance order. See also: Restricting agent checks.

BufferSend

The time interval in seconds which determines how often values are sent from the buffer to Zabbix server.
Note that if the buffer is full, the data will be sent sooner.

Default: 5
 Range: 1-3600

BufferSize

The maximum number of values in the memory buffer. The agent will send all collected data to the Zabbix server or proxy if the buffer is full.
This parameter should only be used if persistent buffer is disabled (*EnablePersistentBuffer=0*).

Default: 100
 Range: 2-65535

ControlSocket

The control socket, used to send runtime commands with the '-R' option.

Default: `\\.\pipe\agent.sock`

DebugLevel

Specify the debug level:
0 - basic information about starting and stopping of Zabbix processes
1 - critical information;
2 - error information;
3 - warnings;
4 - for debugging (produces lots of information);
5 - extended debugging (produces even more information).

Default: 3
 Range: 0-5

DenyKey

Deny the execution of those item keys that match a pattern. The key pattern is a wildcard expression that supports the "*" character to match any number of any characters.
Multiple key matching rules may be defined in combination with AllowKey. The parameters are processed one by one according to their appearance order. See also: Restricting agent checks.

EnablePersistentBuffer

Enable the usage of local persistent storage for active items. If persistent storage is disabled, the memory buffer will be used.

Default: 0
 Values: 0 - disabled, 1 - enabled

ForceActiveChecksOnStart

Perform active checks immediately after the restart for the first received configuration. Also available as a per-plugin configuration parameter, for example: `Plugins.Uptime.System.ForceActiveChecksOnStart=1`

Default: 0
 Values: 0 - disabled, 1 - enabled

HeartbeatFrequency

The frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks.
0 - heartbeat messages disabled.

Default: 60
 Range: 0-3600

HostInterface

An optional parameter that defines the host interface. The host interface is used at host autoregistration process. If not defined, the value will be acquired from HostInterfaceItem.
The agent will issue an error and not start if the value is over the limit of 255 characters.

Range: 0-255 characters

HostInterfaceItem

An optional parameter that defines an item used for getting the host interface.
Host interface is used at host autoregistration process. This option is only used when HostInterface is not defined.
The system.run[] item is supported regardless of AllowKey/DenyKey values.
During an autoregistration request the agent will log a warning message if the value returned by the specified item is over the limit of 255 characters.

HostMetadata

An optional parameter that defines host metadata. Host metadata is used only at host autoregistration process (active agent). If not defined, the value will be acquired from HostMetadataItem.
The agent will issue an error and not start if the specified value is over the limit of 2034 bytes or a non-UTF-8 string.

Range: 0-2034 bytes

HostMetadataItem

An optional parameter that defines an item used for getting host metadata. This option is only used when HostMetadata is not defined. User parameters and aliases are supported. The system.run[] item is supported regardless of AllowKey/DenyKey values.
The HostMetadataItem value is retrieved on each autoregistration attempt and is used only at host autoregistration process.
During an autoregistration request the agent will log a warning message if the value returned by the specified item is over the limit of 65535 UTF-8 code points. The value returned by the item must be a UTF-8 string otherwise it will be ignored.

Hostname

A list of comma-delimited, unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. The value is acquired from HostnameItem if undefined.
Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128 characters per hostname, 2048 characters for the entire line.

Default: Set by HostnameItem

HostnameItem

An optional parameter that defines an item used for getting the host name. This option is only used when Hostname is not defined. User parameters or aliases are not supported, but the system.run[] item is supported regardless of AllowKey/DenyKey values.

Default: `system.hostname`

Include

You may include individual files or all files in a directory in the configuration file (located in `C:\Program Files\Zabbix Agent 2` by default if Zabbix agent is installed using Windows MSI installer packages; located in the folder specified during installation if Zabbix agent is installed as a zip archive). All included files must have correct syntax, otherwise agent will not start. The path can be relative to the *zabbix_agent2.conf* file location (e.g., `Include=.\zabbix_agent2.d\plugins.d\*.conf`).
To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching.
See special notes about limitations.

Example:

```
Include=C:\Program Files\Zabbix Agent2\zabbix_agent2.d\*.conf
```

ListenIP

A list of comma-delimited IP addresses that the agent should listen on. The first IP address is sent to the Zabbix server, if connecting to it, to retrieve the list of active checks.

Default: `0.0.0.0`

ListenPort

The agent will listen on this port for connections from the server.

Default: 10050
 Range: 1024-32767

LogFile

The name of the agent log file.

Default: `c:\\zabbix_agent2.log`
 Mandatory: Yes, if LogType is set to *file*; otherwise no

LogFileSize

The maximum size of a log file in MB.
0 - disable automatic log rotation.
Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

Default: 1
 Range: 0-1024

LogType

The type of the log output:
file - write log to the file specified by LogFile parameter;
console - write log to standard output.

Default: `file`

PersistentBufferFile

The file where Zabbix agent 2 should keep the SQLite database. Must be a full filename. This parameter is only used if persistent buffer is enabled (*EnablePersistentBuffer=1*).

PersistentBufferPeriod

The time period for which data should be stored when there is no connection to the server or proxy. Older data will be lost. Log data will be preserved. This parameter is only used if persistent buffer is enabled (*EnablePersistentBuffer=1*).

Default: `1h`
 Range: 1m-365d

Plugins.<PluginName>.System.Capacity

The limit of checks per `<PluginName>` plugin that can be executed at the same time.

Default: 100 Range: 1-1000

Plugins.Log.MaxLinesPerSecond

The maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log', 'logrt' and 'eventlog' active checks. The provided value will be overridden by the 'maxlines' parameter, provided in the 'log', 'logrt' or 'eventlog' item key.
Note: Zabbix will process 10 times more new lines than set in *MaxLinesPerSecond* to seek the required string in log items.

Default: 20
 Range: 1-1000

Plugins.SystemRun.LogRemoteCommands

Enable the logging of the executed shell commands as warnings. The commands will be logged only if executed remotely. Log entries will not be created if system.run[] is launched locally by the HostMetadataItem, HostInterfaceItem or HostnameItem parameters.

Default: 0
 Values: 0 - disabled, 1 - enabled

PluginSocket

The path to the UNIX socket for loadable plugin communications.

Default: `\\.\pipe\agent.plugin.sock`

PluginTimeout

The timeout for connections with loadable plugins, in seconds.

Default: `Timeout`
 Range: 1-30

RefreshActiveChecks

How often the list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted in 60 seconds.

Default: 5
 Range: 1-86400

Server

A list of comma-delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers or Zabbix proxies. Incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. Spaces are allowed.

Example:

```
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
```

Mandatory: yes

ServerActive

The Zabbix server/proxy address or cluster configuration to get active checks from. The server/proxy address is an IP address or DNS name and optional port separated by colon.
The cluster configuration is one or more server addresses separated by semicolon. Multiple Zabbix servers/clusters and Zabbix proxies can be specified, separated by comma. More than one Zabbix proxy should not be specified from each Zabbix server/cluster. If a Zabbix proxy is specified then Zabbix server/cluster for that proxy should not be specified.
Multiple comma-delimited addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed.
If the port is not specified, default port is used.
IPv6 addresses must be enclosed in square brackets if port for that host is specified. If port is not specified, square brackets for IPv6 addresses are optional.
If this parameter is not specified, active checks are disabled.

Example for Zabbix proxy:

```
ServerActive=127.0.0.1:10051
```

Example for multiple servers:

```
ServerActive=127.0.0.1:20051,zabbix.domain,\[::1\]:30051,::1,\[12fc::1\]
```

Example for high availability:

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051;zabbix.cluster.node3
```

Example for high availability with two clusters and one server:

```
ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.cluster2.node1;zabbix.cluster2.node2,z
```

SourceIP

The source IP address for:
- outgoing connections to Zabbix server or Zabbix proxy;
- making connections while executing some items (web.page.get, net.tcp.port, etc.).

StatusPort

If set, the agent will listen on this port for HTTP status requests (http://localhost:<port>/status).

Range: 1024-32767

Timeout

Spend no more than Timeout seconds on processing.

Default: 3
 Range: 1-30

TLSAccept

The incoming connections to accept. Used for passive checks. Multiple values can be specified, separated by comma:
unencrypted - accept connections without encryption (default)
psk - accept connections with TLS and a pre-shared key (PSK)
cert - accept connections with TLS and a certificate

Mandatory: yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection); otherwise no

TLSCAFile

The full pathname of the file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.

TLSCertFile

The full pathname of the file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.

TLSConnect

How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified:
unencrypted - connect without encryption (default)
psk - connect using TLS and a pre-shared key (PSK)
cert - connect using TLS and a certificate

Mandatory: yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection); otherwise no

TLSCRLFile

The full pathname of the file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components.

TLSKeyFile

The full pathname of the file containing the agent private key, used for encrypted communications between Zabbix components.

TLSPSKFile

The full pathname of the file containing the agent pre-shared key, used for encrypted communications with Zabbix server.

TLSPSKIdentity

The pre-shared key identity string, used for encrypted communications with Zabbix server.

TLSServerCertIssuer

The allowed server (proxy) certificate issuer.

TLSServerCertSubject

The allowed server (proxy) certificate subject.

UnsafeUserParameters

Allow all characters to be passed in arguments to user-defined parameters. The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) < > | # @
Additionally, newline characters are not allowed.

Default: 0
 Values: 0 - do not allow, 1 - allow

UserParameter

A user-defined parameter to monitor. There can be several user-defined parameters.
Format: UserParameter=<key>,<shell command>
Note that the shell command must not return empty string or EOL only. Shell commands may have relative paths, if the UserParameterDir parameter is specified.

Example:

```
UserParameter=system.test,who|wc -l
UserParameter=check_cpu,./custom_script.sh
```

UserParameterDir

The default search path for UserParameter commands. If used, the agent will change its working directory to the one specified here before executing a command. Thereby, UserParameter commands can have a relative `./` prefix instead of a full path.
Only one entry is allowed.

Example:

```
UserParameterDir=/opt/myscripts
```

7 Zabbix agent 2 plugins

Overview

This section contains descriptions of configuration file parameters for Zabbix agent 2 plugins. Please use the sidebar to access information about the specific plugin.

1 Ceph plugin

Overview

This section lists parameters supported in the Ceph Zabbix agent 2 plugin configuration file (ceph.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;

- Comments starting with "#" are only supported at the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.Ceph.Default.ApiKey				Default API key for connecting to Ceph; used if no value is specified in an item key or named session. Supported since version 6.4.3
Plugins.Ceph.Default.User				Default username for connecting to Ceph; used if no value is specified in an item key or named session. Supported since version 6.4.3
Plugins.Ceph.Default.Uri			https://localhost:8003	Default URI for connecting to Ceph; used if no value is specified in an item key or named session. Should not include embedded credentials (they will be ignored). Must match the URI format. Only `https` scheme is supported; a scheme can be omitted. A port can be omitted (default=8003). Examples: `https://127.0.0.1:8003` `localhost` Supported since version 6.4.3
Plugins.Ceph.InsecureSkipVerify		false / true	false	Determines whether an http client should verify the server's certificate chain and host name. If *true*, TLS accepts any certificate presented by the server and any host name in that certificate. In this mode, TLS is susceptible to man-in-the-middle attacks (should be used only for testing).
Plugins.Ceph.KeepAlive		60-900	300	The maximum time of waiting (in seconds) before unused plugin connections are closed.
Plugins.Ceph.Sessions.<SessionName>.ApiKey				Named session API key. **<SessionName>** - define name of a session for using in item keys.
Plugins.Ceph.Sessions.<SessionName>.User				Named session username. **<SessionName>** - define name of a session for using in item keys.
Plugins.Ceph.Sessions.<SessionName>.Uri				Connection string of a named session. **<SessionName>** - define name of a session for using in item keys. Should not include embedded credentials (they will be ignored). Must match the URI format. Only `https` scheme is supported; a scheme can be omitted. A port can be omitted (default=8003). Examples: `https://127.0.0.1:8003` `localhost`
Plugins.Ceph.Timeout		1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down).

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

2 Docker plugin

Overview

This section lists parameters supported in the Docker Zabbix agent 2 plugin configuration file (docker.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;

- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with ”#” are only supported at the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.Docker.Endpoint			unix:///var/run/docker.sock	Docker daemon unix-socket location. Must contain a scheme (only `unix://` is supported).
Plugins.Docker.Timeout		1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down).

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

3 Memcached plugin

Overview

This section lists parameters supported in the Memcached Zabbix agent 2 plugin configuration file (memcached.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with ”#” are only supported at the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.Memcached.Default.Password				Default password for connecting to Memcached; used if no value is specified in an item key or named session. Supported since version 6.4.3

Plugins.Memcached.Default.Uri			tcp://localhost:11211	Default URI for connecting to Memcached; used if no value is specified in an item key or named session.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`; a scheme can be omitted.
				A port can be omitted (default=11211).
				Examples: `tcp://localhost:11211`
				`localhost`
				`unix:/var/run/memcached.sock`
				Supported since version 6.4.3

Plugins.Memcached.Default.User				Default username for connecting to Memcached; used if no value is specified in an item key or named session. Supported since version 6.4.3
Plugins.Memcached.KeepAlive		60-900	300	The maximum time of waiting (in seconds) before unused plugin connections are closed.
Plugins.Memcached.Sessions.<SessionName>.Password				Named session password. **<SessionName>** - define name of a session for using in item keys.

Parameter	Mandatory	Range	Default	Description

Plugins.Memcached.Sessions.<SessionName>.Uri				Connection string of a named session.
				<SessionName> - define name of a session for using in item keys.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`; a scheme can be omitted.
				A port can be omitted (default=11211).
				Examples: `tcp://localhost:11211`
				`localhost`
				`unix:/var/run/memcached.sock`

Plugins.Memcached.Sessions.<SessionName>.User				Named session username. **<SessionName>** - define name of a session for using in item keys.
Plugins.Memcached.Timeout		1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down).

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

4 Modbus plugin

Overview

This section lists parameters supported in the Modbus Zabbix agent 2 plugin configuration file (modbus.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.Modbus.Sessions.<SessionName>.Endpoint				Endpoint is a connection string consisting of a protocol scheme, a host address and a port or serial port name and attributes. **<SessionName>** - define name of a session for using in item keys.
Plugins.Modbus.Sessions.<SessionName>.SlaveID				Slave ID of a named session. **<SessionName>** - define name of a session for using in item keys. Example: `Plugins.Modbus.Sessions.MB1.SlaveID=20` *Note* that this named session parameter is checked only if the value provided in the item key slave ID parameter is empty.
Plugins.Modbus.Sessions.<SessionName>.Timeout				Timeout of a named session. **<SessionName>** - define name of a session for using in item keys. Example: `Plugins.Modbus.Sessions.MB1.Timeout=2`
Plugins.Modbus.Timeout		1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down).

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

5 MongoDB plugin

Overview

This section lists parameters supported in the MongoDB Zabbix agent 2 plugin configuration file (mongo.conf).

The MongoDB plugin is a loadable plugin and is available and fully described in the MongoDB plugin repository.

Pre-compiled plugin binaries for Windows are available since Zabbix 6.4.4 and are compatible with previous 6.4 versions.

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Options

Parameter	Description
-V --version	Print the plugin version and license information.
-h --help	Print help information (shorthand).

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.MongoDB.Default.Password				Default password for connecting to MongoDB; used if no value is specified in an item key or named session. Supported since version 6.4.3
Plugins.MongoDB.Default.Uri				Default URI for connecting to MongoDB; used if no value is specified in an item key or named session. Should not include embedded credentials (they will be ignored). Must match the URI format. Only `tcp` scheme is supported; a scheme can be omitted. A port can be omitted (default=27017). Examples: `tcp://127.0.0.1:27017`, `tcp:localhost`, `localhost` Supported since version 6.4.3
Plugins.MongoDB.Default.User				Default username for connecting to MongoDB; used if no value is specified in an item key or named session. Supported since version 6.4.3
Plugins.MongoDB.KeepAlive		60-900	300	The maximum time of waiting (in seconds) before unused plugin connections are closed.
Plugins.MongoDB.Sessions.<SessionName>.Password				Named session password. **<SessionName>** - define name of a session for using in item keys.
Plugins.MongoDB.Sessions.<SessionName>.TLSCAFile	(yes, if Plugins.MongoDB.Sessions.<SessionName>.TLSConnect is set to one of: verify_ca, verify_full)			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix agent 2 and monitored databases. **<SessionName>** - define name of a session for using in item keys.

Parameter	Mandatory	Range	Default	Description
Plugins.MongoDB.Sessions.<SessionName>.TLSCertFile	Yes, if Plugins.MongoDB.Sessions.<SessionName>.TLSKeyFile is specified Before version 6.4.9 always mandatory if Plugins.MongoDB.Sessions.<SessionName>.TLSConnect is set to one of: verify_ca, verify_full)			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix agent 2 and monitored databases. **<SessionName>** - define name of a session for using in item keys.
Plugins.MongoDB.Sessions.<SessionName>.TLSConnect	No			Encryption type for communications between Zabbix agent 2 and monitored databases. **<SessionName>** - define name of a session for using in item keys. Supported values: *required* - require TLS connection; *verify_ca* - verify certificates; *verify_full* - verify certificates and IP address. Supported since plugin version 1.2.1
Plugins.MongoDB.Sessions.<SessionName>.TLSKeyFile	Yes, if Plugins.MongoDB.Sessions.<SessionName>.TLSCertFile is specified Before version 6.4.9 always mandatory if Plugins.MongoDB.Sessions.<SessionName>.TLSConnect is set to one of: verify_ca, verify_full)			Full pathname of a file containing the database private key used for encrypted communications between Zabbix agent 2 and monitored databases. **<SessionName>** - define name of a session for using in item keys.
Plugins.MongoDB.Sessions.<SessionName>.Uri	No			Connection string of a named session. **<SessionName>** - define name of a session for using in item keys. Should not include embedded credentials (they will be ignored). Must match the URI format. Only `tcp` scheme is supported; a scheme can be omitted. A port can be omitted (default=27017). Examples: `tcp://127.0.0.1:27017`, `tcp:localhost`, `localhost`
Plugins.MongoDB.Sessions.<SessionName>.User	No			Named session username. **<SessionName>** - define name of a session for using in item keys.
Plugins.MongoDB.System.Path	No			Path to plugin executable.
Plugins.MongoDB.Timeout	No	1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down).

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

6 MQTT plugin

Overview

This section lists parameters supported in the MQTT Zabbix agent 2 plugin configuration file (mqtt.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.MQTT.Default.Password				Default password for connecting to MQTT; used if no value is specified in an item key or named session. Supported since version 6.4.4
Plugins.MQTT.Default.TLSCAFile				Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification for encrypted communications between Zabbix agent 2 and MQTT broker; used if no value is specified in a named session. Supported since version 6.4.4
Plugins.MQTT.Default.TLSCertFile				Full pathname of a file containing the agent certificate or certificate chain for encrypted communications between Zabbix agent 2 and MQTT broker; used if no value is specified in a named session. Supported since version 6.4.4
Plugins.MQTT.Default.TLSKeyFile				Full pathname of a file containing the MQTT private key for encrypted communications between Zabbix agent 2 and MQTT broker; used if no value is specified in a named session. Supported since version 6.4.4

Plugins.MQTT.Default.Topic				Default topic for MQTT subscription; used if no value is specified in an item key or named session.
				
				The topic may contain wildcards ("+","#")
				Examples: `path/to/file`
				`path/to/#`
				`path/+/topic`
				Supported since version 6.4.4

Plugins.MQTT.Default.Url			tcp://localhost:1883	Default MQTT broker connection string; used if no value is specified in an item key or named session.
				
				Should not include query parameters.
				Must match the URL format.
				Supported schemes: `tcp` (default), `ws`, `tls`; a scheme can be omitted.
				A port can be omitted (default=1883).
				Examples: `tcp://host:1883`
				`localhost`
				`ws://host:8080`
				Supported since version 6.4.4

Plugins.MQTT.Default.User				Default username for connecting to MQTT; used if no value is specified in an item key or named session. Supported since version 6.4.4
Plugins.MQTT.Sessions.<SessionName>.Password				Named session password. **<SessionName>** - define name of a session for using in item keys. Supported since version 6.4.4

Parameter	Mandatory	Range	Default	Description
Plugins.MQTT.Sessions.<SessionName>.TLSCAFile				Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix agent 2 and MQTT broker. **<SessionName>** - define name of a session for using in item keys. Supported since version 6.4.4
Plugins.MQTT.Sessions.<SessionName>.TLSCertFile				Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix agent 2 and MQTT broker. **<SessionName>** - define name of a session for using in item keys. Supported since version 6.4.4
Plugins.MQTT.Sessions.<SessionName>.TLSKeyFile				Full pathname of a file containing the MQTT private key used for encrypted communications between Zabbix agent 2 and MQTT broker. **<SessionName>** - define name of a session for using in item keys. Supported since version 6.4.4

Plugins.MQTT.Sessions.<SessionName>.Topic				Named session topic for MQTT subscription.
				<SessionName> - define name of a session for using in item keys.
				
				The topic may contain wildcards (”+”,”#”)
				Examples: `path/to/file`
				`path/to/#`
				`path/+/topic`
				Supported since version 6.4.4

Plugins.MQTT.Sessions.<SessionName>.Url				Connection string of a named session.
				<SessionName> - define name of a session for using in item keys.
				
				Should not include query parameters.
				Must match the URL format.
				Supported schemes: `tcp` (default), `ws`, `tls`; a scheme can be omitted.
				A port can be omitted (default=1883).
				Examples: `tcp://host:1883`
				`localhost`
				`ws://host:8080`
				Supported since version 6.4.4

Plugins.MQTT.Sessions.<SessionName>.User				Named session username. **<SessionName>** - define name of a session for using in item keys. Supported since version 6.4.4
Plugins.MQTT.Timeout		1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down).

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

7 MSSQL plugin

Overview

This section lists parameters supported in the MSSQL Zabbix agent 2 plugin configuration file (mssql.conf).

The MSSQL plugin is a loadable plugin and is available and fully described in the MSSQL plugin repository.

This plugin is supported since Zabbix 6.4.12. Pre-compiled plugin binaries for Windows are available since Zabbix 6.4.13 and are compatible with 6.4.12 version.

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.MSSQL.CustomQueriesDir			empty	Specifies the file path to a directory containing user-defined .sql files with custom queries that the plugin can execute. The plugin loads all available .sql files in the configured directory at startup. This means that any changes to the custom query files will not be reflected until the plugin is restarted. The plugin is started and stopped together with Zabbix agent 2.
Plugins.MSSQL.Default.CACertPath				The default file path to the public key certificate of the certificate authority (CA) that issued the certificate of the MSSQL server. The certificate must be in PEM format.
Plugins.MSSQL.Default.Database				The default database name to connect to.
Plugins.MSSQL.Default.Encrypt				Specifies the default connection encryption type. Possible values are: *true* - data sending between plugin and server is encrypted; *false* - data sending between plugin and server is not encrypted beyond the login packet; *strict* - data sending between plugin and server is encrypted E2E using TDS8; *disable* - data sending between plugin and server is not encrypted.
Plugins.MSSQL.Default.HostNameInCertificate				The common name (CN) of the certificate of the MSSQL server by default.
Plugins.MSSQL.Default.Password				The password to be sent to a protected MSSQL server by default.
Plugins.MSSQL.Default.TLSMinVersion				The minimum TLS version to use by default. Possible values are: `1.0`, `1.1`, `1.2`, `1.3`.
Plugins.MSSQL.Default.TrustServerCertificate				Whether the plugin should trust the server certificate without validating it by default. Possible values: `true`, `false`.
Plugins.MSSQL.Default.Uri			sqlserver://localhost:1433	The default URI to connect. The only supported schema is `sqlserver://`. A schema can be omitted. Embedded credentials will be ignored.
Plugins.MSSQL.Default.User				The default username to be sent to a protected MSSQL server.
Plugins.MSSQL.KeepAlive		60-900	300	The maximum time of waiting (in seconds) before unused plugin connections are closed.
Plugins.MSSQL.Sessions.<SessionName>.CACertPath				The file path to the public key certificate of the certificate authority (CA) that issued the certificate of the MSSQL server for the named session. The certificate must be in PEM format. **<SessionName>** - define name of a session for using in item keys.
Plugins.MSSQL.Sessions.<SessionName>.Database				The database name to connect to for the named session. **<SessionName>** - define name of a session for using in item keys.
Plugins.MSSQL.Sessions.<SessionName>.Encrypt				Specifies the connection encryption type for the named session. Possible values are: *true* - data sending between plugin and server is encrypted; *false* - data sending between plugin and server is not encrypted beyond the login packet; *strict* - data sending between plugin and server is encrypted E2E using TDS8; *disable* - data sending between plugin and server is not encrypted. **<SessionName>** - define name of a session for using in item keys.

Parameter	Mandatory	Range	Default	Description
Plugins.MSSQL.Sessions.<SessionName>.HostNameInCertificate				The common name (CN) of the certificate of the MSSQL server for the named session. **<SessionName>** - define name of a session for using in item keys.
Plugins.MSSQL.Sessions.<SessionName>.Password				The password to be sent to a protected MSSQL server for the named session. **<SessionName>** - define name of a session for using in item keys.
Plugins.MSSQL.Sessions.<SessionName>.TLSMinVersion				The minimum TLS version to use for the named session. Possible values are: `1.0`, `1.1`, `1.2`, `1.3`. **<SessionName>** - define name of a session for using in item keys.
Plugins.MSSQL.Sessions.<SessionName>.TrustServerCertificate				Whether the plugin should trust the server certificate without validating it for the named session. Possible values: `true`, `false`. **<SessionName>** - define name of a session for using in item keys.
Plugins.MSSQL.Sessions.<SessionName>.Uri			sqlserver://localhost:1433	The URI to connect, for the named session. The only supported schema is `sqlserver://`. A schema can be omitted. Embedded credentials will be ignored. **<SessionName>** - define name of a session for using in item keys.
Plugins.MSSQL.Sessions.<SessionName>.User				The username to be sent to a protected MSSQL server for the named session. **<SessionName>** - define name of a session for using in item keys.
Plugins.MSSQL.System.Path				Path to the MSSQL plugin executable. Global setting for the MSSQL plugin. Applied to all connections. Example usage: `Plugins.MSSQL.System.Path=/usr/sbin/zabbix-agent2-plugin/`
Plugins.MSSQL.Timeout		1-30	global timeout	The amount of time to wait for a server to respond when first connecting and on follow-up operations in the session.

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

8 MySQL plugin

Overview

This section lists parameters supported in the MySQL Zabbix agent 2 plugin configuration file (mysql.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.Mysql.CallTimeout		1-30	global timeout	The maximum amount of time in seconds to wait for a request to be done.
Plugins.Mysql.CustomQueriesPath			empty	Full path to the directory used for storing custom queries. Supported since version 6.4.6.
Plugins.Mysql.Default.Password				Default password for connecting to MySQL; used if no value is specified in an item key or named session. Supported since version 6.4.3

Parameter	Mandatory	Range	Default	Description
Plugins.Mysql.Default.TLSCAFile	no (yes, if Plugins.Mysql.Default.TLSConnect is set to *verify_ca* or *verify_full*)			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification for encrypted communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported since version 6.4.3
Plugins.Mysql.Default.TLSCertFile	no (yes, Plugins.Mysql.Default.TLSConnect is set to *verify_ca* or *verify_full*)			Full pathname of a file containing the agent certificate or certificate chain for encrypted communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported since version 6.4.3
Plugins.Mysql.Default.TLSConnect	no			Encryption type for communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported values: *required* - require TLS connection; *verify_ca* - verify certificates; *verify_full* - verify certificates and IP address. Supported since version 6.4.3
Plugins.Mysql.Default.TLSKeyFile	no (yes, Plugins.Mysql.Default.TLSConnect is set to *verify_ca* or *verify_full*)			Full pathname of a file containing the database private key for encrypted communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported since version 6.4.3

Plugins.Mysql.Default.Uri	no		tcp://localhost:3306	Default URI for connecting to MySQL; used if no value is specified in an item key or named session.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`; a scheme can be omitted.
				A port can be omitted (default=3306).
				Examples: `tcp://localhost:3306`
				`localhost`
				`unix:/var/run/mysql.sock`
				Supported since version 6.4.3

Plugins.Mysql.Default.User	no			Default username for connecting to MySQL; used if no value is specified in an item key or named session. Supported since version 6.4.3
Plugins.Mysql.KeepAlive	no	60-900	300	The maximum time of waiting (in seconds) before unused plugin connections are closed.
Plugins.Mysql.Sessions.<SessionName>.Password	no			Named session password. **<SessionName>** - define name of a session for using in item keys.
Plugins.Mysql.Sessions.<SessionName>.TLSCAFile	no (yes, if Plugins.Mysql.Sessions.<SessionName>.TLSConnect is set to *verify_ca* or *verify_full*)			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix agent 2 and monitored databases. **<SessionName>** - define name of a session for using in item keys.

Parameter	Mandatory	Range	Default	Description
Plugins.Mysql.Sessions.<SessionName>.TLSCertFile	yes, if Plugins.Mysql.Sessions.<SessionName>.TLSKeyFile is specified Before version 6.4.8 always mandatory if Plugins.Mysql.Sessions.<SessionName>.TLSConnect is set to *verify_ca* or *verify_full*			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications between Zabbix agent 2 and monitored databases. **<SessionName>** - define name of a session for using in item keys.
Plugins.Mysql.Sessions.<SessionName>.TLSConnect				Encryption type for communications between Zabbix agent 2 and monitored databases. **<SessionName>** - define name of a session for using in item keys. Supported values: *required* - require TLS connection; *verify_ca* - verify certificates; *verify_full* - verify certificates and IP address.
Plugins.Mysql.Sessions.<SessionName>.TLSKeyFile	yes, if Plugins.Mysql.Sessions.<SessionName>.TLSCertFile is specified Before version 6.4.8 always mandatory if Plugins.Mysql.Sessions.<SessionName>.TLSConnect is set to *verify_ca* or *verify_full*			Full pathname of a file containing the database private key used for encrypted communications between Zabbix agent 2 and monitored databases. **<SessionName>** - define name of a session for using in item keys.

Plugins.Mysql.Sessions.<SessionName>.Uri				Connection string of a named session.
				<SessionName> - define name of a session for using in item keys.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`; a scheme can be omitted.
				A port can be omitted (default=3306).
				Examples: `tcp://localhost:3306`
				`localhost`
				`unix:/var/run/mysql.sock`

Plugins.Mysql.Sessions.<SessionName>.User				Named session username. **<SessionName>** - define name of a session for using in item keys.
Plugins.Mysql.Timeout		1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down).

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

9 Oracle plugin

Overview

This section lists parameters supported in the Oracle Zabbix agent 2 plugin configuration file (oracle.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.Oracle.CallTimeout	no	1-30	global timeout	The maximum wait time in seconds for a request to be completed.
Plugins.Oracle.ConnectTimeout	no	1-30	global timeout	The maximum wait time in seconds for a connection to be established.
Plugins.Oracle.CustomQueriesPath	no			Full pathname of a directory containing .sql files with custom queries. Disabled by default. Example: `/etc/zabbix/oracle/sql`
Plugins.Oracle.Default.Password	no			Default password for connecting to Oracle; used if no value is specified in an item key or named session. Supported since version 6.4.3
Plugins.Oracle.Default.Service	no			Default service name for connecting to Oracle (SID is not supported); used if no value is specified in an item key or named session. Supported since version 6.4.3
Plugins.Oracle.Default.Uri	no		tcp://localhost:1521	Default URI for connecting to Oracle; used if no value is specified in an item key or named session. Should not include embedded credentials (they will be ignored). Must match the URI format. Only `tcp` scheme is supported; a scheme can be omitted. A port can be omitted (default=1521). Examples: `tcp://127.0.0.1:1521` `localhost` Supported since version 6.4.3
Plugins.Oracle.Default.User	no			Default username for connecting to Oracle; used if no value is specified in an item key or named session. Supported since version 6.4.3
Plugins.Oracle.KeepAlive	no	60-900	300	The maximum time of waiting (in seconds) before unused plugin connections are closed.
Plugins.Oracle.Sessions.<SessionName>.Password	no			Named session password. **<SessionName>** - define name of a session for using in item keys.
Plugins.Oracle.Sessions.<SessionName>.Service	no			Named session service name to be used for connection (SID is not supported). **<SessionName>** - define name of a session for using in item keys.
Plugins.Oracle.Sessions.<SessionName>.Uri	no			Named session connection string for Oracle. **<SessionName>** - define name of a session for using in item keys. Should not include embedded credentials (they will be ignored). Must match the URI format. Only `tcp` scheme is supported; a scheme can be omitted. A port can be omitted (default=1521). Examples: `tcp://127.0.0.1:1521` `localhost`
Plugins.Oracle.Sessions.<SessionName>.User	no			Named session username. **<SessionName>** - define name of a session for using in item keys.

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

10 PostgreSQL plugin

Overview

This section lists parameters supported in the PostgreSQL Zabbix agent 2 plugin configuration file (postgresql.conf).

The PostgreSQL plugin is a loadable plugin and is available and fully described in the PostgreSQL plugin repository.

Pre-compiled plugin binaries for Windows are available since Zabbix 6.4.4 and are compatible with previous 6.4 versions.

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Options

Parameter	Description
-V --version	Print the plugin version and license information.
-h --help	Print help information (shorthand).

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.PostgreSQL.Default.CacheMode			prepare	Cache mode for the PostgreSQL connection. Supported values: *prepare* (default) - will create prepared statements on the PostgreSQL server; *describe* - will use the anonymous prepared statement to describe a statement without creating a statement on the server. Note that "describe" is primarily useful when the environment does not allow prepared statements such as when running a connection pooler like PgBouncer. Supported since version 6.4.10.
Plugins.PostgreSQL.CallTimeout		1-30	global timeout	Maximum wait time (in seconds) for a request to be completed.
Plugins.PostgreSQL.CustomQueriesPath			disabled	Full pathname of the directory containing *.sql* files with custom queries.
Plugins.PostgreSQL.Default.Database				Default database for connecting to PostgreSQL; used if no value is specified in an item key or named session. Supported since version 6.4.3.
Plugins.PostgreSQL.Default.Password				Default password for connecting to PostgreSQL; used if no value is specified in an item key or named session. Supported since version 6.4.3.
Plugins.PostgreSQL.Default.TLSCAFile	no (yes, if Plugins.PostgreSQL.Default.TLSConnect is set to *verify_ca* or *verify_full*)			Full pathname of a file containing the top-level CA(s) certificate for peer certificate verification for encrypted communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported since version 6.4.3.
Plugins.PostgreSQL.Default.TLSCertFile	no (yes, if Plugins.PostgreSQL.Default.TLSConnect is set to *verify_ca* or *verify_full*)			Full pathname of a file containing the PostgreSQL certificate or certificate chain for encrypted communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported since version 6.4.3.

Parameter	Mandatory	Range	Default	Description
Plugins.PostgreSQL.Default.TLSConnect				Encryption type for communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported values: *required* - connect using TLS as transport mode without identity checks; *verify_ca* - connect using TLS and verify certificate; *verify_full* - connect using TLS, verify certificate and verify that database identity (CN) specified by DBHost matches its certificate. Undefined encryption type means unencrypted connection. Supported since version 6.4.3.
Plugins.PostgreSQL.Default.TLSKeyFile	(yes, if Plugins.PostgreSQL.Default.TLSConnect is set to *verify_ca* or *verify_full*)			Full pathname of a file containing the PostgreSQL private key for encrypted communications between Zabbix agent 2 and monitored databases; used if no value is specified in a named session. Supported since version 6.4.3.

Plugins.PostgreSQL.Default.Uri				Default URI for connecting to PostgreSQL; used if no value is specified in an item key or named session.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`.
				Examples: `tcp://127.0.0.1:5432`
				`tcp://localhost`
				`unix:/var/run/postgresql/.s.PGSQL.5432`
				Supported since version 6.4.3.

Plugins.PostgreSQL.Default.User				Default username for connecting to PostgreSQL; used if no value is specified in an item key or named session. Supported since version 6.4.3.
Plugins.PostgreSQL.KeepAlive		60-900	300	Maximum time of waiting (in seconds) before unused plugin connections are closed.
Plugins.PostgreSQL.Sessions.<SessionName>.CachePolicy			prepare	Cache mode for the PostgreSQL connection. **<SessionName>** - define name of a session for using in item keys. Supported values: *prepare* (default) - will create prepared statements on the PostgreSQL server; *describe* - will use the anonymous prepared statement to describe a statement without creating a statement on the server. Note that ”describe” is primarily useful when the environment does not allow prepared statements such as when running a connection pooler like PgBouncer. Supported since version 6.4.10.
Plugins.PostgreSQL.Sessions.<SessionName>.Database				Database for session connection. **<SessionName>** - define name of a session for using in item keys.
Plugins.PostgreSQL.Sessions.<SessionName>.Password		Must match the password format.		Password for session connection. **<SessionName>** - define name of a session for using in item keys.
Plugins.PostgreSQL.Sessions.<SessionName>.TLSCAFile	(yes, if Plugins.PostgreSQL.Sessions.<SessionName>.TLSConnect is set to *verify_ca* or *verify_full*)			Full pathname of a file containing the top-level CA(s) certificate peer certificate verification. **<SessionName>** - define name of a session for using in item keys.

Parameter	Mandatory	Range	Default	Description
Plugins.PostgreSQL.Sessions.<SessionName>.TLSCertFile	Yes, if Plugins.PostgreSQL.Sessions.<SessionName>.TLSKeyFile is specified Before version 6.4.7 always mandatory if Plugins.PostgreSQL.Sessions.<SessionName>.TLSConnect is set to *verify_ca* or *verify_full*)			Full pathname of a file containing the PostgreSQL certificate or certificate chain. **<SessionName>** - define name of a session for using in item keys.
Plugins.PostgreSQL.Sessions.<SessionName>.TLSConnect				Encryption type for PostgreSQL connection. **<SessionName>** - define name of a session for using in item keys. Supported values: *required* - connect using TLS as transport mode without identity checks; *verify_ca* - connect using TLS and verify certificate; *verify_full* - connect using TLS, verify certificate and verify that database identity (CN) specified by DBHost matches its certificate. Undefined encryption type means unencrypted connection.
Plugins.PostgreSQL.Sessions.<SessionName>.TLSKeyFile	Yes, if Plugins.PostgreSQL.Sessions.<SessionName>.TLSCertFile is specified Before version 6.4.7 always mandatory if Plugins.PostgreSQL.Sessions.<SessionName>.TLSConnect is set to *verify_ca* or *verify_full*)			Full pathname of a file containing the PostgreSQL private key. **<SessionName>** - define name of a session for using in item keys.

Plugins.PostgreSQL.Sessions.<SessionName>.Uri				Connection string of a named session.
				<SessionName> - define name of a session for using in item keys.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`.
				Examples: `tcp://127.0.0.1:5432`
				`tcp://localhost`
				`unix:/var/run/postgresql/.s.PGSQL.5432`

Plugins.PostgreSQL.Sessions.<SessionName>.User				Named session username. **<SessionName>** - define name of a session for using in item keys.
Plugins.PostgreSQL.System.Path	Yes			Path to external plugin executable.
Plugins.PostgreSQL.Timeout		1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down).

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

11 Redis plugin

Overview

This section lists parameters supported in the Redis Zabbix agent 2 plugin configuration file (redis.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.Redis.Default.Password	no			Default password for connecting to Redis; used if no value is specified in an item key or named session. Supported since version 6.4.3

Plugins.Redis.Default.Uri	no		tcp://localhost:6379	Default URI for connecting to Redis; used if no value is specified in an item key or named session.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`; a scheme can be omitted (since version 5.2.3).
				A port can be omitted (default=6379).
				Examples: `tcp://localhost:6379`
				`localhost`
				`unix:/var/run/redis.sock`
				Supported since version 6.4.3

Plugins.Redis.KeepAlive	no	60-900	300	The maximum time of waiting (in seconds) before unused plugin connections are closed.
Plugins.Redis.Sessions.<SessionName>.Password				Named session password. **<SessionName>** - define name of a session for using in item keys.

Plugins.Redis.Sessions.<SessionName>.Uri				Connection string of a named session.
				<SessionName> - define name of a session for using in item keys.
				
				Should not include embedded credentials (they will be ignored).
				Must match the URI format.
				Supported schemes: `tcp`, `unix`; a scheme can be omitted.
				A port can be omitted (default=6379).
				Examples: `tcp://localhost:6379`
				`localhost`
				`unix:/var/run/redis.sock`

Plugins.Redis.Timeout	no	1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down).

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

12 Smart plugin

Overview

This section lists parameters supported in the Smart Zabbix agent 2 plugin configuration file (smart.conf).

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;

- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Plugins.Smart.Path	no		smartctl	Path to the smartctl executable.
Plugins.Smart.Timeout	no	1-30	global timeout	Request execution timeout (how long to wait for a request to complete before shutting it down).

See also:

- Description of general Zabbix agent 2 configuration parameters: Zabbix agent 2 (UNIX) / Zabbix agent 2 (Windows)
- Instructions for configuring plugins

8 Zabbix Java gateway

If you use `startup.sh` and `shutdown.sh` scripts for starting Zabbix Java gateway, then you can specify the necessary configuration parameters in the `settings.sh` file. The startup and shutdown scripts source the settings file and take care of converting shell variables (listed in the first column) to Java properties (listed in the second column).

If you start Zabbix Java gateway manually by running `java` directly, then you specify the corresponding Java properties on the command line.

Variable	Property	Mandatory	Range	Default	Description
LISTEN_IP	zabbix.listenIP	no		0.0.0.0	IP address to listen on.
LISTEN_PORT	zabbix.listenPort	no	1024-32767	10052	Port to listen on.
PID_FILE	zabbix.pidFile	no		/tmp/zabbix_java.pid	Name of PID file. If omitted, Zabbix Java Gateway is started as a console application.
PROPERTIES_FILE	zabbix.propertiesFile	no			Name of properties file. Can be used to set additional properties using a key-value format in such a way that they are not visible on a command line or to overwrite existing ones. For example: "javax.net.ssl.trustStorePassword=<p
START_POLLERS	zabbix.startPollers	no	1-1000	5	Number of worker threads to start.
TIMEOUT	zabbix.timeout	no	1-30	3	How long to wait for network operations.

Warning:
Port 10052 is not IANA registered.

9 Zabbix web service

Overview

The Zabbix web service is a process that is used for communication with external web services.

The parameters supported by the Zabbix web service configuration file (zabbix_web_service.conf) are listed in this section.

The parameters are listed without additional information. Click on the parameter to see the full details.

Parameter	Description
AllowedIP	A list of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies.
DebugLevel	The debug level.
ListenPort	The agent will listen on this port for connections from the server.
LogFile	The name of the log file.
LogFileSize	The maximum size of the log file.
LogType	The type of the log output.
Timeout	Spend no more than Timeout seconds on processing.
TLSAccept	What incoming connections to accept.
TLSCAFile	The full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	The full pathname of a file containing the service certificate or certificate chain, used for encrypted communications between Zabbix components.
TLSKeyFile	The full pathname of a file containing the service private key, used for encrypted communications between Zabbix components.

All parameters are non-mandatory unless explicitly stated that the parameter is mandatory.

Note that:

- The default values reflect process defaults, not the values in the shipped configuration files;
- Zabbix supports configuration files only in UTF-8 encoding without BOM;
- Comments starting with "#" are only supported at the beginning of the line.

Parameter details

AllowedIP

A list of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and Zabbix proxies. Incoming connections will be accepted only from the hosts listed here.
If IPv6 support is enabled then `127.0.0.1`, `::127.0.0.1`, `::ffff:127.0.0.1` are treated equally and `::/0` will allow any IPv4 or IPv6 address. `0.0.0.0/0` can be used to allow any IPv4 address.

Example:

`127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com`

Mandatory: yes

DebugLevel

Specify the debug level:
0 - basic information about starting and stopping of Zabbix processes
1 - critical information;
2 - error information;
3 - warnings;
4 - for debugging (produces lots of information);
5 - extended debugging (produces even more information).

Default: 3
 Range: 0-5

ListenPort

The service will listen on this port for connections from the server.

Default: 10053
 Range: 1024-32767

LogFile

The name of the log file.

Example:

`/tmp/zabbix_web_service.log`

Mandatory: Yes, if LogType is set to *file*; otherwise no

LogFileSize

The maximum size of a log file in MB.
0 - disable automatic log rotation.
Note: If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

Default: 1
 Range: 0-1024

LogType

The type of the log output:
file - write log to the file specified by LogFile parameter;
system - write log to syslog;
console - write log to standard output.

Default: `file`

Timeout

Spend no more than Timeout seconds on processing.

Default: 3
 Range: 1-30

TLSAccept

What incoming connections to accept:
unencrypted - accept connections without encryption (default)
cert - accept connections with TLS and a certificate

Default: `unencrypted`

TLSCAFile

The full pathname of the file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.

TLSCertFile

The full pathname of the file containing the service certificate or certificate chain, used for encrypted communications with Zabbix components.

TLSKeyFile

The full pathname of the file containing the service private key, used for encrypted communications between Zabbix components.

10 Inclusion

Overview

Additional files or directories can be included into server/proxy/agent configuration using the `Include` parameter.

Notes on inclusion

If the `Include` parameter is used for including a file, the file must be readable.

If the `Include` parameter is used for including a directory:

- All files in the directory must be readable.
- No particular order of inclusion should be assumed (e.g. files are not included in alphabetical order). Therefore do not define one parameter in several ”Include” files (e.g. to override a general setting with a specific one).
- All files in the directory are included into configuration.
- Beware of file backup copies automatically created by some text editors. For example, if editing the ”include/my_specific.conf” file produces a backup copy ”include/my_specific_conf.BAK” then both files will be included. Move ”include/my_specific.conf.BAK” out of the ”Include” directory. On Linux, contents of the ”Include” directory can be checked with a ”ls -al” command for unnecessary files.

If the `Include` parameter is used for including files using a pattern:

- All files matching the pattern must be readable.
- No particular order of inclusion should be assumed (e.g. files are not included in alphabetical order). Therefore do not define one parameter in several ”Include” files (e.g. to override a general setting with a specific one).

3 Protocols

Please use the sidebar to access content in this section.

1 Server-proxy data exchange protocol

Overview

Server - proxy data exchange is based on JSON format.

Request and response messages must begin with header and data length.

Passive proxy

Configuration request

The server will first send an empty `proxy config` request. This request is sent every `ProxyConfigFrequency` (server configuration parameter) seconds.

The proxy responds with the current proxy version, session token and configuration revision. The server responds with the configuration data that need to be updated.

name	value type	description
server→proxy:		
request	*string*	'proxy config'
proxy→server:		
version	*string*	Proxy version (<major>.<minor>.<build>).
session	*string*	Proxy configuration session token.
config_revision	*number*	Proxy configuration revision.
server→proxy:		
full_sync	*number*	1 - if full configuration data is sent; absent - otherwise (optional).
data	*array*	Object of table data. Absent if configuration has not been changed (optional).
<table>	*object*	One or more objects with <table> data (optional, depending on changes).
fields	*array*	Array of field names.
-	*string*	Field name.
data	*array*	Array of rows.
-	*array*	Array of columns.
-	*string*,*number*	Column value with type depending on column type in database schema.
macro.secrets	*object*	Secret macro information, absent if there are no changes in vault macros (optional).
config_revision	*number*	Configuration cache revision - sent with configuration data (optional).
del_hostids	*array*	Array of removed hostids (optional).
-	*number*	Host identifier.
del_macro_hostids	*array*	Array of hostids with all macros removed (optional).
-	*number*	Host identifier.
proxy→server:		
response	*string*	Request success information ('success' or 'failed').
version	*string*	Proxy version (<major>.<minor>.<build>).

Example:

server→proxy:

server→proxy:

```
{
  "request":"proxy config"
}
```

proxy→server:

```
{
  "version": "6.4.0",
  "session": "0033124949800811e5686dbfd9bcea98",
  "config_revision": 0
}
```

server→proxy:

```
{
"full_sync": 1,
"data": {
"hosts": {
"fields": ["hostid", "host", "status", "ipmi_authtype", "ipmi_privilege", "ipmi_username", "ipmi_password"
"data": [
[10084, "Zabbix server", 0, -1, 2, "", "", "Zabbix server", 1, 1, "", "", "", ""]
]
},
"interface": {
"fields": ["interfaceid", "hostid", "main", "type", "useip", "ip", "dns", "port", "available"],
"data": [
[1, 10084, 1, 1, 1, "127.0.0.1", "", "10053", 1]
]
},
"interface_snmp": {
"fields": ["interfaceid", "version", "bulk", "community", "securityname", "securitylevel", "authpassphrase
"data": []
},
"host_inventory": {
"fields": ["hostid", "type", "type_full", "name", "alias", "os", "os_full", "os_short", "serialno_a", "ser
"data": [
[10084, "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "5
]
},
"items": {
"fields": ["itemid", "type", "snmp_oid", "hostid", "key_", "delay", "history", "status", "value_type", "tr
"data": [
[44161, 7, "", 10084, "agent.hostmetadata", "10s", "90d", 0, 1, "", "", "", "", 0, "", "", "", "", 0, null
[44162, 0, "", 10084, "agent.ping", "10s", "90d", 0, 3, "", "", "", "", 0, "", "", "", "", 0, 1, 0, "", nu
]
},
"item_rtdata": {
"fields": ["itemid", "lastlogsize", "mtime"],
"data": [
[44161, 0, 0],
[44162, 0, 0]
]
},
"item_preproc": {
"fields": ["item_preprocid", "itemid", "step", "type", "params", "error_handler", "error_handler_params"],
"data": []
},
"item_parameter": {
"fields": ["item_parameterid", "itemid", "name", "value"],
"data": []
},
"globalmacro": {
"fields": ["globalmacroid", "macro", "value", "type"],
"data": [
[2, "{$SNMP_COMMUNITY}", "public", 0]
]
},
"hosts_templates": {
"fields": ["hosttemplateid", "hostid", "templateid", "link_type"],
"data": []
},
"hostmacro": {
"fields": ["hostmacroid", "hostid", "macro", "value", "type", "automatic"],
"data": [
[5676, 10084, "{$M}", "AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix:Content", 2, 0]
]
```

```
},
"drules": {
"fields": ["druleid", "name", "iprange", "delay"],
"data": [
[2, "Local network", "127.0.0.1", "10s"]
]
},
"dchecks": {
"fields": ["dcheckid", "druleid", "type", "key_", "snmp_community", "ports", "snmpv3_securityname", "snmpv
"data": [
[2, 2, 9, "system.uname", "", "10052", "", 0, "", "", 0, 0, 0, "", 1, 0]
]
},
"regexps": {
"fields": ["regexpid", "name"],
"data": [
[1, "File systems for discovery"],
[2, "Network interfaces for discovery"],
[3, "Storage devices for SNMP discovery"],
[4, "Windows service names for discovery"],
[5, "Windows service startup states for discovery"]
]
},
"expressions": {
"fields": ["expressionid", "regexpid", "expression", "expression_type", "exp_delimiter", "case_sensitive"]
"data": [
[1, 1, "^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|zfs)$", 3, ",", 0
[3, 3, "^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$", 4, ",", 1],
[5, 4, "^(MMCSS|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_optimization_v4.0.30319_32)$", 4, ","
[6, 5, "^(automatic|automatic delayed)$", 3, ",", 1],
[7, 2, "^Software Loopback Interface", 4, ",", 1],
[8, 2, "^(In)?[Ll]oop[Bb]ack[0-9._]*$", 4, ",", 1],
[9, 2, "^NULL[0-9.]*$", 4, ",", 1],
[10, 2, "^[Ll]o[0-9.]*$", 4, ",", 1],
[11, 2, "^[Ss]ystem$", 4, ",", 1],
[12, 2, "^Nu[0-9.]*$", 4, ",", 1]
]
},
"config": {
"fields": ["configid", "snmptrap_logging", "hk_history_global", "hk_history", "autoreg_tls_accept"],
"data": [
[1, 1, 0, "90d", 1]
]
},
"httptest": {
"fields": ["httptestid", "name", "delay", "agent", "authentication", "http_user", "http_password", "hostid
"data": []
},
"httptestitem": {
"fields": ["httptestitemid", "httptestid", "itemid", "type"],
"data": []
},
"httptest_field": {
"fields": ["httptest_fieldid", "httptestid", "type", "name", "value"],
"data": []
},
"httpstep": {
"fields": ["httpstepid", "httptestid", "name", "no", "url", "timeout", "posts", "required", "status_codes"
"data": []
},
"httpstepitem": {
"fields": ["httpstepitemid", "httpstepid", "itemid", "type"],
```

```
"data": []
},
"httpstep_field": {
"fields": ["httpstep_fieldid", "httpstepid", "type", "name", "value"],
"data": []
},
"config_autoreg_tls": {
"fields": ["autoreg_tlsid", "tls_psk_identity", "tls_psk"],
"data": [
[1, "", ""]
]
}
},
"macro.secrets": {
"AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix": {
"Content": "738"
}
},
"config_revision": 2
}
```


proxy→server:


```
{
  "response": "success",
  "version": "6.4.0"
}
```


Data request

The `proxy data` request is used to obtain host interface availability, historical, discovery and autoregistration data from proxy. This request is sent every `ProxyDataFrequency` (server configuration parameter) seconds.

name	value type	description
server→proxy:		
request	*string*	'proxy data'
proxy→server:		
session	*string*	Data session token.
interface availability	*array*	*(optional)* Array of interface availability data objects.
interfaceid	*number*	Interface identifier.
available	*number*	Interface availability: **0**, *INTERFACE_AVAILABLE_UNKNOWN* - unknown **1**, *INTERFACE_AVAILABLE_TRUE* - available **2**, *INTERFACE_AVAILABLE_FALSE* - unavailable
error	*string*	Interface error message or empty string.
history data	*array*	*(optional)* Array of history data objects.
itemid	*number*	Item identifier.
clock	*number*	Item value timestamp (seconds).
ns	*number*	Item value timestamp (nanoseconds).
value	*string*	*(optional)* Item value.
id	*number*	Value identifier (ascending counter, unique within one data session).
timestamp	*number*	*(optional)* Timestamp of log type items.
source	*string*	*(optional)* Eventlog item source value.
severity	*number*	*(optional)* Eventlog item severity value.
eventid	*number*	*(optional)* Eventlog item eventid value.
state	*string*	*(optional)* Item state: **0**, *ITEM_STATE_NORMAL* **1**, *ITEM_STATE_NOTSUPPORTED*
lastlogsize	*number*	*(optional)* Last log size of log type items.

name	value type	description
mtime	*number*	*(optional)* Modification time of log type items.
discovery data	*array*	*(optional)* Array of discovery data objects.
clock	*number*	Discovery data timestamp.
druleid	*number*	Discovery rule identifier.
dcheckid	*number*	Discovery check identifier or null for discovery rule data.
type	*number*	Discovery check type: **-1** discovery rule data **0**, *SVC_SSH* - SSH service check **1**, *SVC_LDAP* - LDAP service check **2**, *SVC_SMTP* - SMTP service check **3**, *SVC_FTP* - FTP service check **4**, *SVC_HTTP* - HTTP service check **5**, *SVC_POP* - POP service check **6**, *SVC_NNTP* - NNTP service check **7**, *SVC_IMAP* - IMAP service check **8**, *SVC_TCP* - TCP port availability check **9**, *SVC_AGENT* - Zabbix agent **10**, *SVC_SNMPv1* - SNMPv1 agent **11**, *SVC_SNMPv2* - SNMPv2 agent **12**, *SVC_ICMPPING* - ICMP ping **13**, *SVC_SNMPv3* - SNMPv3 agent **14**, *SVC_HTTPS* - HTTPS service check **15**, *SVC_TELNET* - Telnet availability check
ip	*string*	Host IP address.
dns	*string*	Host DNS name.
port	*number*	*(optional)* Service port number.
key_	*string*	*(optional)* Item key for discovery check of type **9** *SVC_AGENT*
value	*string*	*(optional)* Value received from the service, can be empty for most of services.
status	*number*	*(optional)* Service status: **0**, *DOBJECT_STATUS_UP* - Service UP **1**, *DOBJECT_STATUS_DOWN* - Service DOWN
auto registration	*array*	*(optional)* Array of autoregistration data objects.
clock	*number*	Autoregistration data timestamp.
host	*string*	Host name.
ip	*string*	*(optional)* Host IP address.
dns	*string*	*(optional)* Resolved DNS name from IP address.
port	*string*	*(optional)* Host port.
host_metadata	*string*	*(optional)* Host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter).
tasks	*array*	*(optional)* Array of tasks.
type	*number*	Task type: **0**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT* - remote command result
status	*number*	Remote-command execution status: **0**, *ZBX_TM_REMOTE_COMMAND_COMPLETED* - remote command completed successfully **1**, *ZBX_TM_REMOTE_COMMAND_FAILED* - remote command failed
error	*string*	*(optional)* Error message.
parent_taskid	*number*	Parent task ID.
more	*number*	*(optional)* 1 - there are more history data to send.
clock	*number*	*(optional)* Data transfer timestamp (seconds).

name	value type	description
ns	*number*	*(optional)* Data transfer timestamp (nanoseconds).
version	*string*	Proxy version (<major>.<minor>.<build>).
server→proxy:		
response	*string*	Request success information ('success' or 'failed').
tasks	*array*	*(optional)* Array of tasks.
type	*number*	Task type: **1**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND* - remote command
clock	*number*	Task creation time.
ttl	*number*	Time in seconds after which the task expires.
commandtype	*number*	Remote-command type: **0**, *ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT* - use custom script **1**, *ZBX_SCRIPT_TYPE_IPMI* - use IPMI **2**, *ZBX_SCRIPT_TYPE_SSH* - use SSH **3**, *ZBX_SCRIPT_TYPE_TELNET* - use Telnet **4**, *ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT* - use global script (currently functionally equivalent to custom script)
command	*string*	Remote command to execute.
execute_on	*number*	Execution target for custom scripts: **0**, *ZBX_SCRIPT_EXECUTE_ON_AGENT* - execute script on agent **1**, *ZBX_SCRIPT_EXECUTE_ON_SERVER* - execute script on server **2**, *ZBX_SCRIPT_EXECUTE_ON_PROXY* - execute script on proxy
port	*number*	*(optional)* Port for Telnet and SSH commands.
authtype	*number*	*(optional)* Authentication type for SSH commands.
username	*string*	*(optional)* User name for Telnet and SSH commands.
password	*string*	*(optional)* Password for Telnet and SSH commands.
publickey	*string*	*(optional)* Public key for SSH commands.
privatekey	*string*	*(optional)* Private key for SSH commands.
parent_taskid	*number*	Parent task ID.
hostid	*number*	Target host ID.

Example:

server→proxy:

```
{
  "request": "proxy data"
}
```

proxy→server:

```
{
    "session": "12345678901234567890123456789012"
    "interface availability": [
        {
            "interfaceid": 1,
            "available": 1,
            "error": ""
    },
        {
            "interfaceid": 2,
            "available": 2,
            "error": "Get value from agent failed: cannot connect to [[127.0.0.1]:10049]: [111] Connection
    },
        {
            "interfaceid": 3,
            "available": 1,
            "error": ""
    },
        {
```

```
            "interfaceid": 4,
            "available": 1,
            "error": ""
    }
    ],
    "history data":[
        {
            "itemid":"12345",
            "clock":1478609647,
            "ns":332510044,
            "value":"52956612",
            "id": 1
        },
        {
            "itemid":"12346",
            "clock":1478609647,
            "ns":330690279,
            "state":1,
            "value":"Cannot find information for this network interface in /proc/net/dev.",
            "id": 2
        }
    ],
    "discovery data":[
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":3,
            "type":12,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        },
        {
            "clock":1478608764,
            "drule":2,
            "dcheck":null,
            "type":-1,
            "ip":"10.3.0.10",
            "dns":"vdebian",
            "status":1
        }
    ],
    "auto registration":[
        {
            "clock":1478608371,
            "host":"Logger1",
            "ip":"10.3.0.1",
            "dns":"localhost",
            "port":"10050"
        },
        {
            "clock":1478608381,
            "host":"Logger2",
            "ip":"10.3.0.2",
            "dns":"localhost",
            "port":"10050"
        }
    ],
    "tasks":[
        {
            "type": 0,
            "status": 0,
```

```
            "parent_taskid": 10
        },
        {
            "type": 0,
            "status": 1,
            "error": "No permissions to execute task.",
            "parent_taskid": 20
        }
    ],
    "version":"6.4.0"
}
```

server→proxy:

```
{
  "response": "success",
  "tasks":[
      {
         "type": 1,
         "clock": 1478608371,
         "ttl": 600,
         "commandtype": 2,
         "command": "restart_service1.sh",
         "execute_on": 2,
         "port": 80,
         "authtype": 0,
         "username": "userA",
         "password": "password1",
         "publickey": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjTkVxwTCpvKe",
         "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u09lHNsj6tQ5QCqGKukO1De7zhd",
         "parent_taskid": 10,
         "hostid": 10070
      },
      {
         "type": 1,
         "clock": 1478608381,
         "ttl": 600,
         "commandtype": 1,
         "command": "restart_service2.sh",
         "execute_on": 0,
         "authtype": 0,
         "username": "",
         "password": "",
         "publickey": "",
         "privatekey": "",
         "parent_taskid": 20,
         "hostid": 10084
      }
  ]
}
```

Active proxy

Configuration request

The `proxy config` request is sent by active proxy to obtain proxy configuration data. This request is sent every `ProxyConfigFrequency` (proxy configuration parameter) seconds.

name	value type	description
proxy→server:		
request	*string*	'proxy config'
host	*string *	Proxy name.
version	*string*	Proxy version (<major>.<minor>.<build>).
session	*string*	Proxy configuration session token.

name	value type	description
config_revision	*number*	Proxy configuration revision.
server→proxy:		
fullsync	*number*	1 - if full configuration data is sent, absent otherwise (optional).
data	*array*	Object of table data. Absent if configuration has not been changed (optional).
<table>	*object*	One or more objects with <table> data (optional, depending on changes).
fields	*array*	Array of field names.
-	*string*	Field name.
data	*array*	Array of rows.
-	*array*	Array of columns.
-	*string*,*number*	Column value with type depending on column type in database schema.
macro.secrets	*object*	Secret macro information, absent if there are no changes in vault macros (optional).
config_revision	*number*	Configuration cache revision - sent with configuration data (optional).
del_hostids	*array*	Array of removed hostids (optional).
-	*number*	Host identifier.
del_macro_hostids	*array*	Array of hostids with all macros removed (optional).
-	*number*	Host identifier.

Example:

proxy→server:

```
{
  "request": "proxy config",
  "host": "Zabbix proxy",
  "version":"6.4.0",
  "session": "fd59a09ff4e9d1fb447de1f04599bcf6",
  "config_revision": 0
}
```

server→proxy:

```
{
"full_sync": 1,
"data": {
"hosts": {
"fields": ["hostid", "host", "status", "ipmi_authtype", "ipmi_privilege", "ipmi_username", "ipmi_password"
"data": [
[10084, "Zabbix server", 0, -1, 2, "", "", "Zabbix server", 1, 1, "", "", "", ""]
]
},
"interface": {
"fields": ["interfaceid", "hostid", "main", "type", "useip", "ip", "dns", "port", "available"],
"data": [
[1, 10084, 1, 1, 1, "127.0.0.1", "", "10053", 1]
]
},
"interface_snmp": {
"fields": ["interfaceid", "version", "bulk", "community", "securityname", "securitylevel", "authpassphrase
"data": []
},
"host_inventory": {
"fields": ["hostid", "type", "type_full", "name", "alias", "os", "os_full", "os_short", "serialno_a", "ser
"data": [
[10084, "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "5
]
```

```
},
"items": {
"fields": ["itemid", "type", "snmp_oid", "hostid", "key_", "delay", "history", "status", "value_type", "tr
"data": [
[44161, 7, "", 10084, "agent.hostmetadata", "10s", "90d", 0, 1, "", "", "", "", 0, "", "", "", "", 0, null
[44162, 0, "", 10084, "agent.ping", "10s", "90d", 0, 3, "", "", "", "", 0, "", "", "", "", 0, 1, 0, "", nu
]
},
"item_rtdata": {
"fields": ["itemid", "lastlogsize", "mtime"],
"data": [
[44161, 0, 0],
[44162, 0, 0]
]
},
"item_preproc": {
"fields": ["item_preprocid", "itemid", "step", "type", "params", "error_handler", "error_handler_params"],
"data": []
},
"item_parameter": {
"fields": ["item_parameterid", "itemid", "name", "value"],
"data": []
},
"globalmacro": {
"fields": ["globalmacroid", "macro", "value", "type"],
"data": [
[2, "{$SNMP_COMMUNITY}", "public", 0]
]
},
"hosts_templates": {
"fields": ["hosttemplateid", "hostid", "templateid", "link_type"],
"data": []
},
"hostmacro": {
"fields": ["hostmacroid", "hostid", "macro", "value", "type", "automatic"],
"data": [
[5676, 10084, "{$M}", "AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix:Content", 2, 0]
]
},
"drules": {
"fields": ["druleid", "name", "iprange", "delay"],
"data": [
[2, "Local network", "127.0.0.1", "10s"]
]
},
"dchecks": {
"fields": ["dcheckid", "druleid", "type", "key_", "snmp_community", "ports", "snmpv3_securityname", "snmpv
"data": [
[2, 2, 9, "system.uname", "", "10052", "", 0, "", "", 0, 0, 0, "", 1, 0]
]
},
"regexps": {
"fields": ["regexpid", "name"],
"data": [
[1, "File systems for discovery"],
[2, "Network interfaces for discovery"],
[3, "Storage devices for SNMP discovery"],
[4, "Windows service names for discovery"],
[5, "Windows service startup states for discovery"]
]
},
"expressions": {
```

```
"fields": ["expressionid", "regexpid", "expression", "expression_type", "exp_delimiter", "case_sensitive"]
"data": [
[1, 1, "^(btrfs|ext2|ext3|ext4|reiser|xfs|ffs|ufs|jfs|jfs2|vxfs|hfs|apfs|refs|ntfs|fat32|zfs)$", 3, ",", C
[3, 3, "^(Physical memory|Virtual memory|Memory buffers|Cached memory|Swap space)$", 4, ",", 1],
[5, 4, "^(MMCSS|gupdate|SysmonLog|clr_optimization_v2.0.50727_32|clr_optimization_v4.0.30319_32)$", 4, ","
[6, 5, "^(automatic|automatic delayed)$", 3, ",", 1],
[7, 2, "^Software Loopback Interface", 4, ",", 1],
[8, 2, "^(In)?[Ll]oop[Bb]ack[0-9._]*$", 4, ",", 1],
[9, 2, "^NULL[0-9.]*$", 4, ",", 1],
[10, 2, "^[Ll]o[0-9.]*$", 4, ",", 1],
[11, 2, "^[Ss]ystem$", 4, ",", 1],
[12, 2, "^Nu[0-9.]*$", 4, ",", 1]
]
},
"config": {
"fields": ["configid", "snmptrap_logging", "hk_history_global", "hk_history", "autoreg_tls_accept"],
"data": [
[1, 1, 0, "90d", 1]
]
},
"httptest": {
"fields": ["httptestid", "name", "delay", "agent", "authentication", "http_user", "http_password", "hostid
"data": []
},
"httptestitem": {
"fields": ["httptestitemid", "httptestid", "itemid", "type"],
"data": []
},
"httptest_field": {
"fields": ["httptest_fieldid", "httptestid", "type", "name", "value"],
"data": []
},
"httpstep": {
"fields": ["httpstepid", "httptestid", "name", "no", "url", "timeout", "posts", "required", "status_codes"
"data": []
},
"httpstepitem": {
"fields": ["httpstepitemid", "httpstepid", "itemid", "type"],
"data": []
},
"httpstep_field": {
"fields": ["httpstep_fieldid", "httpstepid", "type", "name", "value"],
"data": []
},
"config_autoreg_tls": {
"fields": ["autoreg_tlsid", "tls_psk_identity", "tls_psk"],
"data": [
[1, "", ""]
]
}
},
"macro.secrets": {
"AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix": {
"Content": "738"
}
},
"config_revision": 2
}
```


Data request

The `proxy data` request is sent by proxy to provide host interface availability, history, discovery and autoregistration data. This request is sent every `DataSenderFrequency` (proxy configuration parameter) seconds. Note that active proxy will still poll Zabbix

server every second for remote command tasks (with an empty `proxy data` request).

name	value type	description
proxy→server:		
request	*string*	'proxy data'
host	*string*	Proxy name.
session	*string*	Data session token.
interface availability	*array*	*(optional)* Array of interface availability data objects.
interfaceid	*number*	Interface identifier.
available	*number*	Interface availability: **0**, *INTERFACE_AVAILABLE_UNKNOWN* - unknown **1**, *INTERFACE_AVAILABLE_TRUE* - available **2**, *INTERFACE_AVAILABLE_FALSE* - unavailable
error	*string*	Interface error message or empty string.
history data	*array*	*(optional)* Array of history data objects.
itemid	*number*	Item identifier.
clock	*number*	Item value timestamp (seconds).
ns	*number*	Item value timestamp (nanoseconds).
value	*string*	*(optional)* Item value.
id	*number*	Value identifier (ascending counter, unique within one data session).
timestamp	*number*	*(optional)* Timestamp of log type items.
source	*string*	*(optional)* Eventlog item source value.
severity	*number*	*(optional)* Eventlog item severity value.
eventid	*number*	*(optional)* Eventlog item eventid value.
state	*string*	*(optional)* Item state: **0**, *ITEM_STATE_NORMAL* **1**, *ITEM_STATE_NOTSUPPORTED*
lastlogsize	*number*	*(optional)* Last log size of log type items.
mtime	*number*	*(optional)* Modification time of log type items.
discovery data	*array*	*(optional)* Array of discovery data objects.
clock	*number*	Discovery data timestamp.
druleid	*number*	Discovery rule identifier.
dcheckid	*number*	Discovery check identifier or null for discovery rule data.
type	*number*	Discovery check type: **-1** discovery rule data **0**, *SVC_SSH* - SSH service check **1**, *SVC_LDAP* - LDAP service check **2**, *SVC_SMTP* - SMTP service check **3**, *SVC_FTP* - FTP service check **4**, *SVC_HTTP* - HTTP service check **5**, *SVC_POP* - POP service check **6**, *SVC_NNTP* - NNTP service check **7**, *SVC_IMAP* - IMAP service check **8**, *SVC_TCP* - TCP port availability check **9**, *SVC_AGENT* - Zabbix agent **10**, *SVC_SNMPv1* - SNMPv1 agent **11**, *SVC_SNMPv2* - SNMPv2 agent **12**, *SVC_ICMPPING* - ICMP ping **13**, *SVC_SNMPv3* - SNMPv3 agent **14**, *SVC_HTTPS* - HTTPS service check **15**, *SVC_TELNET* - Telnet availability check
ip	*string*	Host IP address.
dns	*string*	Host DNS name.
port	*number*	*(optional)* Service port number.
key_	*string*	*(optional)* Item key for discovery check of type **9** *SVC_AGENT*

name	value type	description
value	*string*	*(optional)* Value received from the service, can be empty for most services.
status	*number*	*(optional)* Service status: **0**, *DOBJECT_STATUS_UP* - Service UP **1**, *DOBJECT_STATUS_DOWN* - Service DOWN
autoregistration	*array*	*(optional)* Array of autoregistration data objects.
clock	*number*	Autoregistration data timestamp.
host	*string*	Host name.
ip	*string*	*(optional)* Host IP address.
dns	*string*	*(optional)* Resolved DNS name from IP address.
port	*string*	*(optional)* Host port.
host_metadata	*string*	*(optional)* Host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter).
tasks	*array*	*(optional)* Array of tasks.
type	*number*	Task type: **0**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT* - remote command result
status	*number*	Remote-command execution status: **0**, *ZBX_TM_REMOTE_COMMAND_COMPLETED* - remote command completed successfully **1**, *ZBX_TM_REMOTE_COMMAND_FAILED* - remote command failed
error	*string*	*(optional)* Error message.
parent_taskid	*number*	Parent task ID.
more	*number*	*(optional)* 1 - there are more history data to send
clock	*number*	*(optional)* Data transfer timestamp (seconds).
ns	*number*	*(optional)* Data transfer timestamp (nanoseconds).
version	*string*	Proxy version (<major>.<minor>.<build>).
server→proxy:		
response	*string*	Request success information (’success’ or ’failed’).
upload	*string*	Upload control for historical data (history, autoregistration, host availability, network discovery). Possible values: **enabled** - normal operation **disabled** - server is not accepting data (possibly due to internal cache over limit)
tasks	*array*	*(optional)* Array of tasks.
type	*number*	Task type: **1**, *ZBX_TM_TASK_PROCESS_REMOTE_COMMAND* - remote command
clock	*number*	Task creation time.
ttl	*number*	Time in seconds after which the task expires.
commandtype	*number*	Remote-command type: **0**, *ZBX_SCRIPT_TYPE_CUSTOM_SCRIPT* - use custom script **1**, *ZBX_SCRIPT_TYPE_IPMI* - use IPMI **2**, *ZBX_SCRIPT_TYPE_SSH* - use SSH **3**, *ZBX_SCRIPT_TYPE_TELNET* - use Telnet **4**, *ZBX_SCRIPT_TYPE_GLOBAL_SCRIPT* - use global script (currently functionally equivalent to custom script)
command	*string*	Remote command to execute.
execute_on	*number*	Execution target for custom scripts: **0**, *ZBX_SCRIPT_EXECUTE_ON_AGENT* - execute script on agent **1**, *ZBX_SCRIPT_EXECUTE_ON_SERVER* - execute script on server **2**, *ZBX_SCRIPT_EXECUTE_ON_PROXY* - execute script on proxy
port	*number*	*(optional)* Port for Telnet and SSH commands.
authtype	*number*	*(optional)* Authentication type for SSH commands.
username	*string*	*(optional)* User name for Telnet and SSH commands.

name	value type	description
password	*string*	*(optional)* Password for Telnet and SSH commands.
publickey	*string*	*(optional)* Public key for SSH commands.
privatekey	*string*	*(optional)* Private key for SSH commands.
parent_taskid	*number*	Parent task ID.
hostid	*number*	Target host ID.

Example:

proxy→server:


```
{
"request": "proxy data",
"host": "Zabbix proxy",
"session": "818cdd1b537bdc5e50c09ed4969235b6",
"interface availability": [{
"interfaceid": 1,
"available": 1,
"error": ""
}],
"history data": [{
"id": 1114,
"itemid": 44162,
"clock": 1665730632,
"ns": 798953105,
"value": "1"
}, {
"id": 1115,
"itemid": 44161,
"clock": 1665730633,
"ns": 811684663,
"value": "58"
}],
"auto registration": [{
"clock": 1665730633,
"host": "Zabbix server",
"ip": "127.0.0.1",
"dns": "localhost",
"port": "10053",
"host_metadata": "58",
"tls_accepted": 1
}],
"discovery data": [{
"clock": 1665732232,
"drule": 2,
"dcheck": 2,
"ip": "127.0.0.1",
"dns": "localhost",
"port": 10052,
"status": 1
}, {
"clock": 1665732232,
"drule": 2,
"dcheck": null,
"ip": "127.0.0.1",
"dns": "localhost",
"status": 1
}],
"host data": [{
"hostid": 10084,
"active_status": 1
}],
"tasks": [{
```

```
"type": 3,
"clock": 1665730985,
"ttl": 0,
"status": -1,
"info": "Remote commands are not enabled",
"parent_taskid": 3
}],
"version": "6.4.0",
"clock": 1665730643,
"ns": 65389964
}
```


server→proxy:


```
{
"upload": "enabled",
"response": "success",
"tasks": [{
"type": 2,
"clock": 1665730986,
"ttl": 600,
"commandtype": 0,
"command": "ping -c 3 127.0.0.1; case $? in [01]) true;; *) false;; esac",
"execute_on": 2,
"port": 0,
"authtype": 0,
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
"alertid": 0,
"parent_taskid": 4,
"hostid": 10084
}]
}
```


2 Zabbix agent protocol

Please refer to Passive and active agent checks page for more information.

3 Zabbix agent 2 protocol

Overview

This section provides information on:

- Agent2 -> Server : active checks request
- Server -> Agent2 : active checks response
- Agent2 -> Server : agent data request
- Server -> Agent2 : agent data response
- Agent2 -> Server : heartbeat message

Active checks request

The active checks request is used to obtain the active checks to be processed by agent. This request is sent by the agent upon start and then with RefreshActiveChecks intervals.

Field	Type	Mandatory	Value
request	*string*	yes	`active checks`
host	*string*	yes	Host name.
version	*string*	yes	The agent version: <major>.<minor>.
host_metadata	*string*	no	The configuration parameter HostMetadata or HostMetadataItem metric value.
interface	*string*	no	The configuration parameter HostInterface or HostInterfaceItem metric value.
ip	*string*	no	The configuration parameter ListenIP first IP if set.

Field	Type	Mandatory	Value
port	*number*	no	The configuration parameter ListenPort value if set and not default agent listening port.
config_revision	*number*	no	Configuration identifier for incremental configuration sync.
session	*string*	no	Session identifier for incremental configuration sync.

Example:

```
{
  "request": "active checks",
  "host": "Zabbix server",
  "version": "6.0",
  "host_metadata": "mysql,nginx",
  "hostinterface": "zabbix.server.lan",
  "ip": "159.168.1.1",
  "port": 12050,
  "config_revision": 1,
  "session": "e3dcbd9ace2c9694e1d7bbd030eeef6e"
}
```

Active checks response

The active checks response is sent by the server back to agent after processing active checks request.

Field	Type	Mandatory	Value
response	*string*	yes	`success` | `failed`
info	*string*	no	Error information in the case of failure.
data	*array of objects*	no	Active check items. Omitted if host configuration is unchanged.
key	*string*	no	Item key with expanded macros.
itemid	*number*	no	Item identifier.
delay	*string*	no	Item update interval.
lastlogsize	*number*	no	Item lastlogsize.
mtime	*number*	no	Item mtime.
regexp	*array of objects*	no	Global regular expressions.
name	*string*	no	Global regular expression name.
expression	*string*	no	Global regular expression.
expression_type	*number*	no	Global regular expression type.
exp_delimiter	*string*	no	Global regular expression delimiter.
case_sensitive	*number*	no	Global regular expression case sensitivity setting.
config_revision	*number*	no	Configuration identifier for incremental configuration sync. Omitted if host configuration is unchanged. Incremented if host configuration is changed.

Example:

```
{
  "response": "success",
  "data": [
    {
      "key": "log[/home/zabbix/logs/zabbix_agentd.log]",
      "itemid": 1234,
      "delay": "30s",
      "lastlogsize": 0,
      "mtime": 0
    },
    {
      "key": "agent.version",
      "itemid": 5678,
      "delay": "10m",
```

```
      "lastlogsize": 0,
      "mtime": 0
    }
  ],
  "config_revision": 2
}
```

Agent data request

The agent data request contains the gathered item values.

Field	Type	Mandatory	Value
request	*string*	yes	`agent data`
host	*string*	yes	Host name.
version	*string*	yes	The agent version: <major>.<minor>.
session	*string*	yes	Unique session identifier generated each time when agent is started.
data	*array of objects*	yes	Item values.
id	*number*	yes	The value identifier (incremental counter used for checking duplicated values in the case of network problems).
itemid	*number*	yes	Item identifier.
value	*string*	no	The item value.
lastlogsize	*number*	no	The item lastlogsize.
mtime	*number*	no	The item mtime.
state	*number*	no	The item state.
source	*string*	no	The value event log source.
eventid	*number*	no	The value event log eventid.
severity	*number*	no	The value event log severity.
timestamp	*number*	no	The value event log timestamp.
clock	*number*	yes	The value timestamp (seconds since Epoch).
ns	*number*	yes	The value timestamp nanoseconds.

Example:

```
{
  "request": "agent data",
  "data": [
    {
      "id": 1,
      "itemid": 5678,
      "value": "2.4.0",
      "clock": 1400675595,
      "ns": 76808644
    },
    {
      "id": 2,
      "itemid": 1234,
      "lastlogsize": 112,
      "value": " 19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision 5000
      "clock": 1400675595,
      "ns": 77053975
    }
  ],
  "host": "Zabbix server",
  "version": "6.0",
  "session": "1234456akdsjhfoui"
}
```

Agent data response

The agent data response is sent by the server back to agent after processing the agent data request.

Field	Type	Mandatory	Value
response	*string*	yes	`success` | `failed`
info	*string*	yes	Item processing results.

Example:

```
{
  "response": "success",
  "info": "processed: 2; failed: 0; total: 2; seconds spent: 0.003534"
}
```

Heartbeat message

The heartbeat message is sent by an active agent to Zabbix server/proxy every HeartbeatFrequency seconds (configured in the Zabbix agent 2 configuration file).

It is used to monitor the availability of active checks.

```
{
  "request": "active check heartbeat",
  "host": "Zabbix server",
  "heartbeat_freq": 60
}
```

Field	Type	Mandatory	Value
request	*string*	yes	`active check heartbeat`
host	*string*	yes	The host name.
heartbeat_freq	*number*	yes	The agent heartbeat frequency (HeartbeatFrequency configuration parameter).

4 Zabbix agent 2 plugin protocol

Zabbix agent 2 protocol is based on code, size and data model.

Code

Type	Size	Comments
Byte	4	Payload type, currently only JSON is supported.

Size

Type	Size	Comments
Byte	4	Size of the current payload in bytes.

Payload data

Type	Size	Comments
Byte	Defined by the *Size* field	JSON formatted data.

Payload data definition

Common data

These parameters are present in all requests/responses:

Name	Type	Comments
id	uint32	For requests - the incrementing identifier used to link requests with responses. Unique within a request direction (i.e. from agent to plugin or from plugin to agent). For responses - ID of the corresponding request.
type	uint32	The request type.

Log request

A request sent by a plugin to write a log message into the agent log file.

direction	plugin → agent
response	no

Parameters specific to log requests:

Name	Type	Comments
severity	uint32	The message severity (log level).
message	string	The message to log.

Example:

```
{"id":0,"type":1,"severity":3,"message":"message"}
```

Register request

A request sent by the agent during the agent startup phase to obtain provided metrics to register a plugin.

direction	agent → plugin
response	yes

Parameters specific to register requests:

Name	Type	Comments
version	string	The protocol version <major>.<minor>

Example:

```
{"id":1,"type":2,"version":"1.0"}
```

Register response

Plugin’s response to the register request.

direction	plugin → agent
response	n/a

Parameters specific to register responses:

Name	Type	Comments
name	string	The plugin name.
metrics	array of strings (optional)	The metrics with descriptions as used in the plugin. Returns RegisterMetrics(). Absent if error is returned.
interfaces	uint32 (optional)	The bit mask of plugin’s supported interfaces. Absent if error is returned.

Name	Type	Comments
error	string (optional)	An error message returned if a plugin cannot be started. Absent, if metrics are returned.

Examples:

```
{"id":2,"type":3,"metrics":["external.test", "External exporter Test."], "interfaces": 4}
```

or

```
{"id":2,"type":3,"error":"error message"}
```

Start request

A request to execute the Start function of the Runner interface.

direction	agent → plugin
response	no

The request doesn't have specific parameters, it only contains common data parameters.

Example:

```
{"id":3,"type":4}
```

Terminate request

A request sent by the agent to shutdown a plugin.

direction	agent → plugin
response	no

The request doesn't have specific parameters, it only contains common data parameters.

Example:

```
{"id":3,"type":5}
```

Export request

A request to execute the Export function of the Exporter interface.

direction	agent → plugin
response	no

Parameters specific to export requests:

Name	Type	Comments
key	string	The plugin key.
parameters	array of strings (optional)	The parameters for Export function.

Example:

```
{"id":4,"type":6,"key":"test.key","parameters":["foo","bar"]}
```

Export response

Response from the Export function of the Exporter interface.

direction	plugin → agent
response	n/a

Parameters specific to export responses:

Name	Type	Comments
value	string (optional)	Response value from the Export function. Absent, if error is returned.
error	string (optional)	Error message if the Export function has not been executed successfully. Absent, if value is returned.

Examples:

```
{"id":5,"type":7,"value":"response"}
```

or

```
{"id":5,"type":7,"error":"error message"}
```

Configure request

A request to execute the *Configure* function of the *Configurator* interface.

direction	agent → plugin
response	n/a

Parameters specific to *Configure* requests:

Name	Type	Comments
global_options	JSON object	JSON object containing global agent configuration options.
private_options	JSON object (optional)	JSON object containing private plugin configuration options, if provided.

Example:

```
{"id":6,"type":8,"global_options":{...},"private_options":{...}}
```

Validate request

A request to execute *Validate* function of the *Configurator* interface.

direction	agent → plugin
response	yes

Parameters specific to *Validate* requests:

Name	Type	Comments
private_options	JSON object (optional)	JSON object containing private plugin configuration options, if provided.

Example:

```
{"id":7,"type":9,"private_options":{...}}
```

Validate response

Response from *Validate* function of *Configurator* interface.

direction	plugin → agent
response	n/a

Parameters specific to *Validate* responses:

Name	Type	Comments
error	string (optional)	An error message returned if the Validate function is not executed successfully. Absent if executed successfully.

Example:

```
{"id":8,"type":10}
```

or

```
{"id":8,"type":10,"error":"error message"}
```

5 Zabbix sender protocol

Please refer to the trapper item page for more information.

6 Header

Overview

The header is present in all request and response messages between Zabbix components. It is required to determine the message length, if it is compressed or not, if it is a large packet or not.

Zabbix communications protocol has 1GB packet size limit per connection. The limit of 1GB is applied to both the received packet data length and the uncompressed data length.

When sending configuration to Zabbix proxy, the packet size limit is increased to 4GB to allow syncing large configurations. When data length before compression exceeds 4GB, Zabbix server automatically starts using the large packet format (0x04 flag) which increases the packet size limit to 16GB.

Note that while a large packet format can be used for sending any data, currently only the Zabbix proxy configuration syncer can handle packets that are larger than 1GB.

Structure

The header consists of four fields. All numbers in the header are formatted as little-endian.

Field	Size	Size (large packet)	Description
<PROTOCOL>	4	4	"ZBXD" or 5A 42 58 44
<FLAGS>	1	1	Protocol flags: 0x01 - Zabbix communications protocol 0x02 - compression 0x04 - large packet
<DATALEN>	4	8	Data length.
<RESERVED>	4	8	When compression is used (0x02 flag) - the length of uncompressed data When compression is not used - 00 00 00 00

Examples

Here are some code snippets showing how to add Zabbix protocol header to the data you want to send in order to obtain the packet you should send to Zabbix so that it is interpreted correctly. These code snippets assume that the data is not larger than 1GB, thus the large packet format is not used.

Python

```
packet = b"ZBXD\1" + struct.pack("<II", len(data), 0) + data
```

or

```
def zbx_create_header(plain_data_size, compressed_data_size=None):
    protocol = b"ZBXD"
    flags = 0x01
    if compressed_data_size is None:
        datalen = plain_data_size
        reserved = 0
    else:
        flags |= 0x02
        datalen = compressed_data_size
        reserved = plain_data_size
    return protocol + struct.pack("<BII", flags, datalen, reserved)

packet = zbx_create_header(len(data)) + data
```

Perl

```
my $packet = "ZBXD\1" . pack("(II)<", length($data), 0) . $data;
```

or

```
sub zbx_create_header($;$)
{
    my $plain_data_size = shift;
    my $compressed_data_size = shift;

    my $protocol = "ZBXD";
    my $flags = 0x01;
    my $datalen;
    my $reserved;

    if (!defined($compressed_data_size))
    {
        $datalen = $plain_data_size;
        $reserved = 0;
    }
    else
    {
        $flags |= 0x02;
        $datalen = $compressed_data_size;
        $reserved = $plain_data_size;
    }

    return $protocol . chr($flags) . pack("(II)<", $datalen, $reserved);
}

my $packet = zbx_create_header(length($data)) . $data;
```

PHP

```
$packet = "ZBXD\1" . pack("VV", strlen($data), 0) . $data;
```

or

```
function zbx_create_header($plain_data_size, $compressed_data_size = null)
{
    $protocol = "ZBXD";
    $flags = 0x01;
    if (is_null($compressed_data_size))
    {
        $datalen = $plain_data_size;
        $reserved = 0;
    }
    else
    {
        $flags |= 0x02;
```

```
        $datalen = $compressed_data_size;
        $reserved = $plain_data_size;
    }
    return $protocol . chr($flags) . pack("VV", $datalen, $reserved);
}

$packet = zbx_create_header(strlen($data)) . $data;
```

Bash

```
datalen=$(printf "%08x" ${#data})
datalen="\\x${datalen:6:2}\\x${datalen:4:2}\\x${datalen:2:2}\\x${datalen:0:2}"
printf "ZBXD\1${datalen}\0\0\0\0%s" "$data"
```

7 Newline-delimited JSON export protocol

This section presents details of the export protocol in a newline-delimited JSON format, used in:

- data export to files
- streaming to external systems

The following can be exported:

- trigger events
- item values
- trends (export to files only)

All files have a .ndjson extension. Each line of the export file is a JSON object.

Trigger events

The following information is exported for a problem event:

Field		Type	Description
clock		number	Number of seconds since Epoch to the moment when problem was detected (integer part).
ns		number	Number of nanoseconds to be added to `clock` to get a precise problem detection time.
value		number	1 (always).
eventid		number	Problem event ID.
name		string	Problem event name.
severity		number	Problem event severity (0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster).
hosts		array	List of hosts involved in the trigger expression; there should be at least one element in array.
-		object	
	host	string	Host name.
	name	string	Visible host name.
groups		array	List of host groups of all hosts involved in the trigger expression; there should be at least one element in array.
-		string	Host group name.
tags		array	List of problem tags (can be empty).
-		object	
	tag	string	Tag name.
	value	string	Tag value (can be empty).

The following information is exported for a recovery event:

Field	Type	Description
clock	number	Number of seconds since Epoch to the moment when problem was resolved (integer part).
ns	number	Number of nanoseconds to be added to `clock` to get a precise problem resolution time.

Field	Type	Description
value	number	0 (always).
eventid	number	Recovery event ID.
p_eventid	number	Problem event ID.

Examples

Problem:

```
{"clock":1519304285,"ns":123456789,"value":1,"name":"Either Zabbix agent is unreachable on Host B or polle
```

Recovery:

```
{"clock":1519304345,"ns":987654321,"value":0,"eventid":43,"p_eventid":42}
```

Problem (multiple problem event generation):

```
{"clock":1519304286,"ns":123456789,"value":1,"eventid":43,"name":"Either Zabbix agent is unreachable on Ho
```

```
{"clock":1519304286,"ns":123456789,"value":1,"eventid":43,"name":"Either Zabbix agent is unreachable on Ho
```

Recovery:

```
{"clock":1519304346,"ns":987654321,"value":0,"eventid":44,"p_eventid":43}
```

```
{"clock":1519304346,"ns":987654321,"value":0,"eventid":44,"p_eventid":42}
```

Item values

The following information is exported for a collected item value:

Field	Type	Description
host	object	Host name of the item host.
host	string	Host name.
name	string	Visible host name.
groups	array	List of host groups of the item host; there should be at least one element in array.
-	string	Host group name.
item_tags	array	List of item tags (can be empty).
-	object	
tag	string	Tag name.
value	string	Tag value (can be empty).
itemid	number	Item ID.
name	string	Visible item name.
clock	number	Number of seconds since Epoch to the moment when value was collected (integer part).
ns	number	Number of nanoseconds to be added to `clock` to get a precise value collection time.
timestamp (*Log* only)	number	0 if not available.
source (*Log* only)	string	Empty string if not available.
severity (*Log* only)	number	0 if not available.
eventid (*Log* only)	number	0 if not available.
value	number (for numeric items) or string (for text items)	Collected item value.

Field	Type	Description
type	number	Collected value type: 0 - numeric float, 1 - character, 2 - log, 3 - numeric unsigned, 4 - text

Examples

Numeric (unsigned) value:

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"item_tags":[{"
```

Numeric (float) value:

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"item_tags":[{"
```

Character, text value:

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"item_tags":[{"
```

Log value:

```
{"host":{"host":"Host A","name":"Host A visible"},"groups":["Group X","Group Y","Group Z"],"item_tags":[{"
```

Trends

The following information is exported for a calculated trend value:

Field	Type	Description
host	object	Host name of the item host.
host	string	Host name.
name	string	Visible host name.
groups	array	List of host groups of the item host; there should be at least one element in array.
-	string	Host group name.
item_tags	array	List of item tags (can be empty).
-	object	
tag	string	Tag name.
value	string	Tag value (can be empty).
itemid	number	Item ID.
name	string	Visible item name.
clock	number	Number of seconds since Epoch to the moment when value was collected (integer part).
count	number	Number of values collected for a given hour.
min	number	Minimum item value for a given hour.
avg	number	Average item value for a given hour.
max	number	Maximum item value for a given hour.
type	number	Value type: 0 - numeric float, 3 - numeric unsigned

Examples

Numeric (unsigned) value:

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"item_tags":[{"
```

Numeric (float) value:

```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"item_tags":[{"
```

4 Items

Please use the sidebar to access content in this section.

1 vm.memory.size parameters

Overview

This section provides some parameter details for the vm.memory.size[<mode>] agent item.

Parameters

The following parameters are available for this item:

- **active** - memory currently in use or very recently used, and so it is in RAM
- **anon** - memory not associated with a file (cannot be re-read from it)
- **available** - available memory, calculated differently depending on the platform (see the table below)
- **buffers** - cache for things like file system metadata
- **cached** - cache for various things
- **exec** - executable code, typically from a (program) file
- **file** - cache for contents of recently accessed files
- **free** - memory that is readily available to any entity requesting memory
- **inactive** - memory that is marked as not used
- **pavailable** - 'available' memory as percentage of 'total' (calculated as `available/total*100`)
- **pinned** - same as 'wired'
- **pused** - 'used' memory as percentage of 'total' (calculated as `used/total*100`)
- **shared** - memory that may be simultaneously accessed by multiple processes
- **slab** - total amount of memory used by the kernel to cache data structures for its own use
- **total** - total physical memory available
- **used** - used memory, calculated differently depending on the platform (see the table below)
- **wired** - memory that is marked to always stay in RAM. It is never moved to disk.

Warning:
Some of these parameters are platform-specific and might not be available on your platform. See Zabbix agent items for details.

Platform-specific calculation of **available** and **used**:

Platform	**"available"**	**"used"**
AIX	free + cached	real memory in use
FreeBSD	inactive + cached + free	active + wired + cached
HP UX	free	total - free
Linux<3.14	free + buffers + cached	total - free
Linux 3.14+ (also backported to 3.10 on RHEL 7)	/proc/meminfo, see "MemAvailable" in Linux kernel documentation for details. Note that free + buffers + cached is no longer equal to 'available' due to not all the page cache can be freed and low watermark being used in calculation.	total - free
NetBSD	inactive + execpages + file + free	total - free
OpenBSD	inactive + free + cached	active + wired
OSX	inactive + free	active + wired
Solaris	free	total - free
Win32	free	total - free

Attention:
The sum of *vm.memory.size[used]* and *vm.memory.size[available]* does not necessarily equal total. For instance, on FreeBSD:
* Active, inactive, wired, cached memories are considered used, because they store some useful information.
* At the same time inactive, cached, free memories are considered available, because these kinds of memories can be given instantly to processes that request more memory.

So inactive memory is both used and available simultaneously. Because of this, the *vm.memory.size[used]* item is designed for informational purposes only, while *vm.memory.size[available]* is designed to be used in triggers.

See also

1. Additional details about memory calculation in different OS

2 Passive and active agent checks

Overview

This section provides details on passive and active checks performed by Zabbix agent.

Zabbix uses a JSON based communication protocol for communicating with Zabbix agent.

See also: Zabbix agent 2 protocol details.

Passive checks

A passive check is a simple data request. Zabbix server or proxy asks for some data (for example, CPU load) and Zabbix agent sends back the result to the server.

Server request

For definition of header and data length please refer to protocol details.

```
<item key>
```

Agent response

```
<DATA>[\0<ERROR>]
```

Above, the part in square brackets is optional and is only sent for not supported items.

For example, for supported items:

1. Server opens a TCP connection
2. Server sends **<HEADER><DATALEN>agent.ping**
3. Agent reads the request and responds with **<HEADER><DATALEN>1**
4. Server processes data to get the value, '1' in our case
5. TCP connection is closed

For not supported items:

1. Server opens a TCP connection
2. Server sends **<HEADER><DATALEN>vfs.fs.size[/nono]**
3. Agent reads the request and responds with **<HEADER><DATALEN>ZBX_NOTSUPPORTED\0Cannot obtain filesystem information: [2] No such file or directory**
4. Server processes data, changes item state to not supported with the specified error message
5. TCP connection is closed

Active checks

Active checks require more complex processing. The agent must first retrieve from the server(s) a list of items for independent processing.

The servers to get the active checks from are listed in the 'ServerActive' parameter of the agent configuration file. The frequency of asking for these checks is set by the 'RefreshActiveChecks' parameter in the same configuration file. However, if refreshing active checks fails, it is retried after hardcoded 60 seconds.

> **Note:**
> In order to decrease network traffic and resources usage Zabbix server or Zabbix proxy will provide configuration only if Zabbix agent still hasn't received configuration or if something has changed in host configuration, global macros or global regular expressions.

The agent then periodically sends the new values to the server(s).

> **Note:**
> If an agent is behind the firewall you might consider using only Active checks because in this case you wouldn't need to modify the firewall to allow initial incoming connections.

Getting the list of items

Agent request

The active checks request is used to obtain the active checks to be processed by agent. This request is sent by the agent upon start and then with RefreshActiveChecks intervals.

```
{
  "request": "active checks",
  "host": "Zabbix server",
  "host_metadata": "mysql,nginx",
  "hostinterface": "zabbix.server.lan",
  "ip": "159.168.1.1",
  "port": 12050,
  "config_revision": 1,
  "session": "e3dcbd9ace2c9694e1d7bbd030eeef6e"
}
```

Field	Type	Mandatory	Value
request	*string*	yes	`active checks`
host	*string*	yes	Host name.
host_metadata	*string*	no	The configuration parameter HostMetadata or HostMetadataItem metric value.
hostinterface	*string*	no	The configuration parameter HostInterface or HostInterfaceItem metric value.
ip	*string*	no	The configuration parameter ListenIP first IP if set.
port	*number*	no	The configuration parameter ListenPort value if set and not default agent listening port.
config_revision	*number*	no	Configuration identifier for incremental configuration sync.
session	*string*	no	Session identifier for incremental configuration sync.

Server response

The active checks response is sent by the server back to agent after processing the active checks request.

```
{
  "response": "success",
  "data": [
    {
      "key": "log[/home/zabbix/logs/zabbix_agentd.log]",
      "key_orig": "log[/home/zabbix/logs/zabbix_agentd.log]",
      "itemid": 1234,
      "delay": "30s",
      "lastlogsize": 0,
      "mtime": 0
    },
    {
      "key": "agent.version",
      "key_orig": "agent.version",
      "itemid": 5678,
      "delay": "10m",
      "lastlogsize": 0,
      "mtime": 0
    }
  ],
  "config_revision": 2
}
```

Field	Type	Mandatory	Value
response	*string*	yes	`success` | `failed`
info	*string*	no	Error information in the case of failure.
data	*array of objects*	no	Active check items. Omitted if host configuration is unchanged.
key	*string*	no	Item key with expanded macros.
key_orig	*string*	no	Item key without expanded macros.
itemid	*number*	no	Item identifier.
delay	*string*	no	Item update interval.
lastlogsize	*number*	no	Item lastlogsize.
mtime	*number*	no	Item mtime.

Field	Type	Mandatory	Value
refresh_unsupported	*number*	no	Unsupported item refresh interval.
regexp	*array of objects*	no	Global regular expressions.
name	*string*	no	Global regular expression name.
expression	*string*	no	Global regular expression.
expression_type	*number*	no	Global regular expression type.
exp_delimiter	*string*	no	Global regular expression delimiter.
case_sensitive	*number*	no	Global regular expression case sensitivity setting.
config_revision	*number*	no	Configuration identifier for incremental configuration sync. Omitted if host configuration is unchanged. Incremented if host configuration is changed.

The server must respond with success.

For example:

1. Agent opens a TCP connection
2. Agent asks for the list of checks
3. Server responds with a list of items (item key, delay)
4. Agent parses the response
5. TCP connection is closed
6. Agent starts periodical collection of data

Attention:
Note that (sensitive) configuration data may become available to parties having access to the Zabbix server trapper port when using an active check. This is possible because anyone may pretend to be an active agent and request item configuration data; authentication does not take place unless you use encryption options.

Sending in collected data

Agent sends

The agent data request contains the gathered item values.

```
{
  "request": "agent data",
  "data": [
    {
      "host": "Zabbix server",
      "key": "agent.version",
      "value": "2.4.0",
      "clock": 1400675595,
      "ns": 76808644
    },
    {
      "host": "Zabbix server",
      "key": "log[/home/zabbix/logs/zabbix_agentd.log]",
      "lastlogsize": 112,
      "value": " 19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision 5000
      "clock": 1400675595,
      "ns": 77053975
    }
  ],
  "session": "1234456akdsjhfoui"
}
```

Field	Type	Mandatory	Value
request	*string*	yes	`agent data`
session	*string*	yes	Unique session identifier generated each time when agent is started.
data	*array of objects*	yes	Item values.

Field	Type	Mandatory	Value
id	*number*	yes	The value identifier (incremental counter used for checking duplicated values in the case of network problems).
host	*string*	yes	Host name.
key	*string*	yes	The item key.
value	*string*	no	The item value.
lastlogsize	*number*	no	The item lastlogsize.
mtime	*number*	no	The item mtime.
state	*number*	no	The item state.
source	*string*	no	The value event log source.
eventid	*number*	no	The value event log eventid.
severity	*number*	no	The value event log severity.
timestamp	*number*	no	The value event log timestamp.
clock	*number*	yes	The value timestamp (seconds since Epoch).
ns	*number*	yes	The value timestamp nanoseconds.

A virtual ID is assigned to each value. Value ID is a simple ascending counter, unique within one data session (identified by the session token). This ID is used to discard duplicate values that might be sent in poor connectivity environments.

Server response

The agent data response is sent by the server back to agent after processing the agent data request.

```
{
  "response": "success",
  "info": "processed: 2; failed: 0; total: 2; seconds spent: 0.003534"
}
```

Field	Type	Mandatory	Value
response	*string*	yes	`success` | `failed`
info	*string*	yes	Item processing results.

Attention:
If sending of some values fails on the server (for example, because host or item has been disabled or deleted), agent will not retry sending of those values.

For example:

1. Agent opens a TCP connection
2. Agent sends a list of values
3. Server processes the data and sends the status back
4. TCP connection is closed

Note how in the example above the not supported status for vfs.fs.size[/nono] is indicated by the ”state” value of 1 and the error message in ”value” property.

Attention:
Error message will be trimmed to 2048 symbols on server side.

Heartbeat message

The heartbeat message is sent by an active agent to Zabbix server/proxy every HeartbeatFrequency seconds (configured in the Zabbix agent configuration file).

It is used to monitor the availability of active checks.

```
{
  "request": "active check heartbeat",
  "host": "Zabbix server",
  "heartbeat_freq": 60
}
```

Field	Type	Mandatory	Value
request	*string*	yes	`active check heartbeat`
host	*string*	yes	The host name.
heartbeat_freq	*number*	yes	The agent heartbeat frequency (HeartbeatFrequency configuration parameter).

Older XML protocol

Note:
Zabbix will take up to 16 MB of XML Base64-encoded data, but a single decoded value should be no longer than 64 KB otherwise it will be truncated to 64 KB while decoding.

3 Trapper items

Overview

Zabbix server uses a JSON- based communication protocol for receiving data from Zabbix sender with the help of trapper item.

Request and response messages must begin with header and data length.

Zabbix sender request

```
{
    "request":"sender data",
    "data":[
        {
            "host":"<hostname>",
            "key":"trap",
            "value":"test value"
        }
    ]
}
```

Zabbix server response

```
{
    "response":"success",
    "info":"processed: 1; failed: 0; total: 1; seconds spent: 0.060753"
}
```

Zabbix sender request with a timestamp

Alternatively Zabbix sender can send a request with a timestamp and nanoseconds.

```
{
    "request":"sender data",
    "data":[
        {
            "host":"<hostname>",
            "key":"trap",
            "value":"test value",
            "clock":1516710794,
            "ns":592397170
        },
        {
            "host":"<hostname>",
            "key":"trap",
            "value":"test value",
            "clock":1516710795,
            "ns":192399456
        }
    ],
    "clock":1516712029,
    "ns":873386094
}
```

Zabbix server response

```
{
    "response":"success",
    "info":"processed: 2; failed: 0; total: 2; seconds spent: 0.060904"
}
```

4 Minimum permission level for Windows agent items

Overview

When monitoring systems using an agent, a good practice is to obtain metrics from the host on which the agent is installed. To use the principle of least privilege, it is necessary to determine what metrics are obtained from the agent.

The table in this document allows you to select the minimum rights for guaranteed correct operation of Zabbix agent.

If a different user is selected for the agent to work, rather than 'LocalSystem', then for the operation of agent as a Windows service, the new user must have the rights "Log on as a service" from "Local Policy→User Rights Assignment" and the right to create, write and delete the Zabbix agent log file. An Active Directory user must be added to the *Performance Monitor Users* group.

Note:
When working with the rights of an agent based on the "minimum technically acceptable" group, prior provision of rights to objects for monitoring is required.

Common agent items supported on Windows

Item key	User group	
	Recommended	Minimum technically acceptable (functionality is limited)
agent.hostname	Guests	Guests
agent.ping	Guests	Guests
agent.variant	Guests	Guests
agent.version	Guests	Guests
log	Administrators	Guests
log.count	Administrators	Guests
logrt	Administrators	Guests
logrt.count	Administrators	Guests
net.dns	Guests	Guests
net.dns.record	Guests	Guests
net.if.discovery	Guests	Guests
net.if.in	Guests	Guests
net.if.out	Guests	Guests
net.if.total	Guests	Guests
net.tcp.listen	Guests	Guests
net.tcp.port	Guests	Guests
net.tcp.service	Guests	Guests
net.tcp.service.perf	Guests	Guests
net.udp.service	Guests	Guests
net.udp.service.perf	Guests	Guests
proc.num	Administrators	Guests
system.cpu.discovery	Performance Monitor Users	Performance Monitor Users
system.cpu.load	Performance Monitor Users	Performance Monitor Users
system.cpu.num	Guests	Guests
system.cpu.util	Performance Monitor Users	Performance Monitor Users
system.hostname	Guests	Guests
system.localtime	Guests	Guests
system.run	Administrators	Guests
system.sw.arch	Guests	Guests
system.swap.size	Guests	Guests
system.uname	Guests	Guests
system.uptime	Performance Monitor Users	Performance Monitor Users
vfs.dir.count	Administrators	Guests
vfs.dir.get	Administrators	Guests
vfs.dir.size	Administrators	Guests

Item key	User group	
vfs.file.cksum	Administrators	Guests
vfs.file.contents	Administrators	Guests
vfs.file.exists	Administrators	Guests
vfs.file.md5sum	Administrators	Guests
vfs.file.regexp	Administrators	Guests
vfs.file.regmatch	Administrators	Guests
vfs.file.size	Administrators	Guests
vfs.file.time	Administrators	Guests
vfs.fs.discovery	Administrators	Guests
vfs.fs.size	Administrators	Guests
vm.memory.size	Guests	Guests
web.page.get	Guests	Guests
web.page.perf	Guests	Guests
web.page.regexp	Guests	Guests
zabbix.stats	Guests	Guests

Windows-specific item keys

Item key	User group	
	Recommended	Minimum technically acceptable (functionality is limited)
eventlog	Event Log Readers	Guests
net.if.list	Guests	Guests
perf_counter	Performance Monitor Users	Performance Monitor Users
proc_info	Administrators	Guests
service.discovery	Guests	Guests
service.info	Guests	Guests
services	Guests	Guests
wmi.get	Administrators	Guests
vm.vmemory.size	Guests	Guests

5 Encoding of returned values

Zabbix server expects every returned text value in the UTF8 encoding. This is related to any type of checks: Zabbix agent, SSH, Telnet, etc.

Different monitored systems/devices and checks can return non-ASCII characters in the value. For such cases, almost all possible **zabbix** keys contain an additional item key parameter - **<encoding>**. This key parameter is optional but it should be specified if the returned value is not in the UTF8 encoding and it contains non-ASCII characters. Otherwise the result can be unexpected and unpredictable.

A description of behavior with different database backends in such cases follows.

MySQL

If a value contains a non-ASCII character in non UTF8 encoding - this character and the following will be discarded when the database stores this value. No warning messages will be written to the *zabbix_server.log*.
Relevant for at least MySQL version 5.1.61

PostgreSQL

If a value contains a non-ASCII character in non UTF8 encoding - this will lead to a failed SQL query (PGRES_FATAL_ERROR:ERROR invalid byte sequence for encoding) and data will not be stored. An appropriate warning message will be written to the *zabbix_server.log*.
Relevant for at least PostgreSQL version 9.1.3

6 Large file support

Large file support, often abbreviated to LFS, is the term applied to the ability to work with files larger than 2 GB on 32-bit operating systems. Since Zabbix 2.0 support for large files has been added. This change affects at least log file monitoring and all vfs.file.* items. Large file support depends on the capabilities of a system at Zabbix compilation time, but is completely disabled on a 32-bit Solaris due to its incompatibility with procfs and swapctl.

7 Sensor

Each sensor chip gets its own directory in the sysfs /sys/devices tree. To find all sensor chips, it is easier to follow the device symlinks from /sys/class/hwmon/hwmon*, where * is a real number (0,1,2,...).

The sensor readings are located either in /sys/class/hwmon/hwmon*/ directory for virtual devices, or in /sys/class/hwmon/hwmon*/device directory for non-virtual devices. A file, called name, located inside hwmon* or hwmon*/device directories contains the name of the chip, which corresponds to the name of the kernel driver used by the sensor chip.

There is only one sensor reading value per file. The common scheme for naming the files that contain sensor readings inside any of the directories mentioned above is: <type><number>_<item>, where

- **type** - for sensor chips is ”in” (voltage), ”temp” (temperature), ”fan” (fan), etc.,
- **item** - ”input” (measured value), ”max” (high threshold), ”min” (low threshold), etc.,
- **number** - always used for elements that can be present more than once (usually starts from 1, except for voltages which start from 0). If files do not refer to a specific element they have a simple name with no number.

The information regarding sensors available on the host can be acquired using **sensor-detect** and **sensors** tools (lm-sensors package: http://lm-sensors.org/). **Sensors-detect** helps to determine which modules are necessary for available sensors. When modules are loaded the **sensors** program can be used to show the readings of all sensor chips. The labeling of sensor readings, used by this program, can be different from the common naming scheme (<type><number>_<item>):

- if there is a file called <type><number>_label, then the label inside this file will be used instead of <type><number><item> name;
- if there is no <type><number>_label file, then the program searches inside the /etc/sensors.conf (could be also /etc/sensors3.conf, or different) for the name substitution.

This labeling allows user to determine what kind of hardware is used. If there is neither <type><number>_label file nor label inside the configuration file the type of hardware can be determined by the name attribute (hwmon*/device/name). The actual names of sensors, which zabbix_agent accepts, can be obtained by running **sensors** program with -u parameter (**sensors -u**).

In **sensor** program the available sensors are separated by the bus type (ISA adapter, PCI adapter, SPI adapter, Virtual device, ACPI interface, HID adapter).

On Linux 2.4:

(Sensor readings are obtained from /proc/sys/dev/sensors directory)

- **device** - device name (if <mode> is used, it is a regular expression);
- **sensor** - sensor name (if <mode> is used, it is a regular expression);
- **mode** - possible values: avg, max, min (if this parameter is omitted, device and sensor are treated verbatim).

Example key: sensor[w83781d-i2c-0-2d,temp1]

Prior to Zabbix 1.8.4, the sensor[temp1] format was used.

On Linux 2.6+:

(Sensor readings are obtained from /sys/class/hwmon directory)

- **device** - device name (non regular expression). The device name could be the actual name of the device (e.g 0000:00:18.3) or the name acquired using sensors program (e.g. k8temp-pci-00c3). It is up to the user to choose which name to use;
- **sensor** - sensor name (non regular expression);
- **mode** - possible values: avg, max, min (if this parameter is omitted, device and sensor are treated verbatim).

Example key:

sensor[k8temp-pci-00c3,temp,max] or sensor[0000:00:18.3,temp1]

sensor[smsc47b397-isa-0880,in,avg] or sensor[smsc47b397.2176,in1]

Obtaining sensor names

Sensor labels, as printed by the *sensors* command, cannot always be used directly because the naming of labels may be different for each sensor chip vendor. For example, *sensors* output might contain the following lines:

```
$ sensors
in0:         +2.24 V  (min =  +0.00 V, max =  +3.32 V)
Vcore:       +1.15 V  (min =  +0.00 V, max =  +2.99 V)
+3.3V:       +3.30 V  (min =  +2.97 V, max =  +3.63 V)
+12V:       +13.00 V  (min =  +0.00 V, max = +15.94 V)
M/B Temp:    +30.0°C  (low  = -127.0°C, high = +127.0°C)
```

Out of these, only one label may be used directly:

```
$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,in0]
2.240000
```

Attempting to use other labels (like *Vcore* or *+12V*) will not work.

```
$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,Vcore]
ZBX_NOTSUPPORTED
```

To find out the actual sensor name, which can be used by Zabbix to retrieve the sensor readings, run *sensors -u*. In the output, the following may be observed:

```
$ sensors -u
...
Vcore:
  in1_input: 1.15
  in1_min: 0.00
  in1_max: 2.99
  in1_alarm: 0.00
...
+12V:
  in4_input: 13.00
  in4_min: 0.00
  in4_max: 15.94
  in4_alarm: 0.00
...
```

So *Vcore* should be queried as *in1*, and *+12V* should be queried as *in4*. According to specification, these are voltages on chip pins and generally speaking may need scaling.

```
$ zabbix_get -s 127.0.0.1 -k sensor[lm85-i2c-0-2e,in1]
1.301000
```

Not only voltage (in), but also current (curr), temperature (temp) and fan speed (fan) readings can be retrieved by Zabbix.

8 Notes on memtype parameter in proc.mem items

Overview

The **memtype** parameter is supported on Linux, AIX, FreeBSD, and Solaris platforms.

Three common values of 'memtype' are supported on all of these platforms: `pmem`, `rss` and `vsize`. Additionally, platform-specific 'memtype' values are supported on some platforms.

AIX

See values supported for 'memtype' parameter on AIX in the table.

Supported value	Description	Source in procentry64 structure	Tries to be compatible with
vsize [1]	Virtual memory size	pi_size	
pmem	Percentage of real memory	pi_prm	ps -o pmem
rss	Resident set size	pi_trss + pi_drss	ps -o rssize
size	Size of process (code + data)	pi_dvm	"ps gvw" SIZE column
dsize	Data size	pi_dsize	
tsize	Text (code) size	pi_tsize	"ps gvw" TSIZ column
sdsize	Data size from shared library	pi_sdsize	
drss	Data resident set size	pi_drss	
trss	Text resident set size	pi_trss	

Notes for AIX:

1. When choosing parameters for proc.mem[] item key on AIX, try to specify narrow process selection criteria. Otherwise there is a risk of getting unwanted processes counted into proc.mem[] result.

Example:

```
$ zabbix_agentd -t proc.mem[,,,NonExistingProcess,rss]
proc.mem[,,,NonExistingProcess,rss]           [u|2879488]
```

This example shows how specifying only command line (regular expression to match) parameter results in Zabbix agent self-accounting - probably not what you want.

2. Do not use "ps -ef" to browse processes - it shows only non-kernel processes. Use "ps -Af" to see all processes which will be seen by Zabbix agent.
3. Let's go through example of 'topasrec' how Zabbix agent proc.mem[] selects processes.

```
$ ps -Af | grep topasrec
root 10747984        1   0   Mar 16      -  0:00 /usr/bin/topasrec  -L -s 300 -R 1 -r 6 -o /var/perf daily
```

proc.mem[] has arguments:

```
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]
```

The 1st criterion is a process name (argument <name>). In our example Zabbix agent will see it as 'topasrec'. In order to match, you need to either specify 'topasrec' or to leave it empty. The 2nd criterion is a user name (argument <user>). To match, you need to either specify 'root' or to leave it empty. The 3rd criterion used in process selection is an argument <cmdline>. Zabbix agent will see its value as '/usr/bin/topasrec -L -s 300 -R 1 -r 6 -o /var/perf/daily/ -ypersistent=1 -O type=bin -ystart_time=04:08:54,Mar16,2023'. To match, you need to either specify a regular expression which matches this string or to leave it empty.

Arguments <mode> and <memtype> are applied after using the three criteria mentioned above.

FreeBSD

See values supported for 'memtype' parameter on FreeBSD in the table.

Supported value	Description	Source in kinfo_proc structure	Tries to be compatible with
vsize	Virtual memory size	kp_eproc.e_vm.vm_map_size or ki_size	ps -o vsz
pmem	Percentage of real memory	calculated from rss	ps -o pmem
rss	Resident set size	kp_eproc.e_vm.vm_rssize or ki_rssize	ps -o rss
size [1]	Size of process (code + data + stack)	tsize + dsize + ssize	
tsize	Text (code) size	kp_eproc.e_vm.vm_tsize or ki_tsize	ps -o tsiz
dsize	Data size	kp_eproc.e_vm.vm_dsize or ki_dsize	ps -o dsiz
ssize	Stack size	kp_eproc.e_vm.vm_ssize or ki_ssize	ps -o ssiz

Linux

See values supported for 'memtype' parameter on Linux in the table.

Supported value	Description	Source in /proc/<pid>/status file
vsize [1]	Virtual memory size	VmSize
pmem	Percentage of real memory	(VmRSS/total_memory) * 100
rss	Resident set size	VmRSS
data	Size of data segment	VmData
exe	Size of code segment	VmExe
hwm	Peak resident set size	VmHWM
lck	Size of locked memory	VmLck
lib	Size of shared libraries	VmLib
peak	Peak virtual memory size	VmPeak
pin	Size of pinned pages	VmPin
pte	Size of page table entries	VmPTE
size	Size of process code + data + stack segments	VmExe + VmData + VmStk

Supported value	Description	Source in /proc/<pid>/status file
stk	Size of stack segment	VmStk
swap	Size of swap space used	VmSwap

Notes for Linux:

1. Not all 'memtype' values are supported by older Linux kernels. For example, Linux 2.4 kernels do not support `hwm`, `pin`, `peak`, `pte` and `swap` values.
2. We have noticed that self-monitoring of the Zabbix agent active check process with `proc.mem[...,...,...,...,data]` shows a value that is 4 kB larger than reported by `VmData` line in the agent's /proc/<pid>/status file. At the time of self-measurement the agent's data segment increases by 4 kB and then returns to the previous size.

Solaris

See values supported for 'memtype' parameter on Solaris in the table.

Supported value	Description	Source in psinfo structure	Tries to be compatible with
vsize [1]	Size of process image	pr_size	ps -o vsz
pmem	Percentage of real memory	pr_pctmem	ps -o pmem
rss	Resident set size It may be underestimated - see rss description in "man ps".	pr_rssize	ps -o rss

Footnotes

[1] Default value.

9 Notes on selecting processes in proc.mem and proc.num items

Processes modifying their commandline

Some programs use modifying their commandline as a method for displaying their current activity. A user can see the activity by running `ps` and `top` commands. Examples of such programs include *PostgreSQL*, *Sendmail*, *Zabbix*.

Let's see an example from Linux. Let's assume we want to monitor a number of Zabbix agent processes.

`ps` command shows processes of interest as

```
$ ps -fu zabbix
UID        PID  PPID  C STIME TTY          TIME CMD
...
zabbix    6318     1  0 12:01 ?        00:00:00 sbin/zabbix_agentd -c /home/zabbix/ZBXNEXT-1078/zabbix_age
zabbix    6319  6318  0 12:01 ?        00:00:01 sbin/zabbix_agentd: collector [idle 1 sec]
zabbix    6320  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #1 [waiting for connection]
zabbix    6321  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #2 [waiting for connection]
zabbix    6322  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: listener #3 [waiting for connection]
zabbix    6323  6318  0 12:01 ?        00:00:00 sbin/zabbix_agentd: active checks #1 [idle 1 sec]
...
```

Selecting processes by name and user does the job:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd,zabbix]'
6
```

Now let's rename `zabbix_agentd` executable to `zabbix_agentd_30` and restart it.

`ps` now shows

```
$ ps -fu zabbix
UID        PID  PPID  C STIME TTY          TIME CMD
...
zabbix    6715     1  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30 -c /home/zabbix/ZBXNEXT-1078/zabbix_
zabbix    6716  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: collector [idle 1 sec]
zabbix    6717  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #1 [waiting for connection
zabbix    6718  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #2 [waiting for connection
```

```
zabbix    6719  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: listener #3 [waiting for connection
zabbix    6720  6715  0 12:53 ?        00:00:00 sbin/zabbix_agentd_30: active checks #1 [idle 1 sec]
...
```

Now selecting processes by name and user produces an incorrect result:

```
$ zabbix_get -s localhost -k 'proc.num[zabbix_agentd_30,zabbix]'
1
```

Why a simple renaming of executable to a longer name lead to quite different result ?

Zabbix agent starts with checking the process name. /proc/<pid>/status file is opened and the line Name is checked. In our case the Name lines are:

```
$ grep Name /proc/{6715,6716,6717,6718,6719,6720}/status
/proc/6715/status:Name:   zabbix_agentd_3
/proc/6716/status:Name:   zabbix_agentd_3
/proc/6717/status:Name:   zabbix_agentd_3
/proc/6718/status:Name:   zabbix_agentd_3
/proc/6719/status:Name:   zabbix_agentd_3
/proc/6720/status:Name:   zabbix_agentd_3
```

The process name in status file is truncated to 15 characters.

A similar result can be seen with ps command:

```
$ ps -u zabbix
  PID TTY          TIME CMD
...
 6715 ?        00:00:00 zabbix_agentd_3
 6716 ?        00:00:01 zabbix_agentd_3
 6717 ?        00:00:00 zabbix_agentd_3
 6718 ?        00:00:00 zabbix_agentd_3
 6719 ?        00:00:00 zabbix_agentd_3
 6720 ?        00:00:00 zabbix_agentd_3
 ...
```

Obviously, that is not equal to our proc.num[] name parameter value zabbix_agentd_30. Having failed to match the process name from status file the Zabbix agent turns to /proc/<pid>/cmdline file.

How the agent sees the "cmdline" file can be illustrated with running a command

```
$ for i in 6715 6716 6717 6718 6719 6720; do cat /proc/$i/cmdline | awk '{gsub(/\x0/,"<NUL>"); print};'; d
sbin/zabbix_agentd_30<NUL>-c<NUL>/home/zabbix/ZBXNEXT-1078/zabbix_agentd.conf<NUL>
sbin/zabbix_agentd_30: collector [idle 1 sec]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><
sbin/zabbix_agentd_30: listener #1 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: listener #2 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: listener #3 [waiting for connection]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><N
sbin/zabbix_agentd_30: active checks #1 [idle 1 sec]<NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL><NUL
```

/proc/<pid>/cmdline files in our case contain invisible, non-printable null bytes, used to terminate strings in *C* language. The null bytes are shown as "<NUL>" in this example.

Zabbix agent checks "cmdline" for the main process and takes a zabbix_agentd_30, which matches our name parameter value zabbix_agentd_30. So, the main process is counted by item proc.num[zabbix_agentd_30,zabbix].

When checking the next process, the agent takes zabbix_agentd_30: collector [idle 1 sec] from the cmdline file and it does not meet our name parameter zabbix_agentd_30. So, only the main process which does not modify its commandline, gets counted. Other agent processes modify their command line and are ignored.

This example shows that the name parameter cannot be used in proc.mem[] and proc.num[] for selecting processes in this case.

Note:
For proc.get[] item, when Zabbix agent checks "cmdline" for the process name, it will only use part of the name starting from the last slash and until the first space or colon sign. Process name received from cmdline file will only be used if its beginning completely matches the shortened process name in the status file. The algorithm is the same for both process name in the filter and in the JSON output.

Using cmdline parameter with a proper regular expression produces a correct result:

```
$ zabbix_get -s localhost -k 'proc.num[,zabbix,,zabbix_agentd_30[ :]]'
6
```

Be careful when using proc.get[], proc.mem[] and proc.num[] items for monitoring programs which modify their command lines.

Before putting name and cmdline parameters into proc.get[], proc.mem[] and proc.num[] items, you may want to test the parameters using proc.num[] item and ps command.

Linux kernel threads

Threads cannot be selected with cmdline parameter in proc.get[], proc.mem[] and proc.num[] items

Let's take as an example one of kernel threads:

```
$ ps -ef| grep kthreadd
root         2     0  0 09:33 ?        00:00:00 [kthreadd]
```

It can be selected with process name parameter:

```
$ zabbix_get -s localhost -k 'proc.num[kthreadd,root]'
1
```

But selection by process cmdline parameter does not work:

```
$ zabbix_get -s localhost -k 'proc.num[,root,,kthreadd]'
0
```

The reason is that Zabbix agent takes the regular expression specified in cmdline parameter and applies it to contents of process /proc/<pid>/cmdline. For kernel threads their /proc/<pid>/cmdline files are empty. So, cmdline parameter never matches.

Counting of threads in proc.mem[] and proc.num[] items

Linux kernel threads are counted by proc.num[] item but do not report memory in proc.mem[] item. For example:

```
$ ps -ef | grep kthreadd
root         2     0  0 09:51 ?        00:00:00 [kthreadd]

$ zabbix_get -s localhost -k 'proc.num[kthreadd]'
1

$ zabbix_get -s localhost -k 'proc.mem[kthreadd]'
ZBX_NOTSUPPORTED: Cannot get amount of "VmSize" memory.
```

But what happens if there is a user process with the same name as a kernel thread ? Then it could look like this:

```
$ ps -ef | grep kthreadd
root         2     0  0 09:51 ?        00:00:00 [kthreadd]
zabbix    9611  6133  0 17:58 pts/1    00:00:00 ./kthreadd

$ zabbix_get -s localhost -k 'proc.num[kthreadd]'
2

$ zabbix_get -s localhost -k 'proc.mem[kthreadd]'
4157440
```

proc.num[] counted both the kernel thread and the user process. proc.mem[] reports memory for the user process only and counts the kernel thread memory as if it was 0. This is different from the case above when ZBX_NOTSUPPORTED was reported.

Be careful when using proc.mem[] and proc.num[] items if the program name happens to match one of the thread.

Before putting parameters into proc.mem[] and proc.num[] items, you may want to test the parameters using proc.num[] item and ps command.

10 Implementation details of net.tcp.service and net.udp.service checks

Implementation of net.tcp.service and net.udp.service checks is detailed on this page for various services specified in the service parameter.

Item net.tcp.service parameters

ftp

Creates a TCP connection and expects the first 4 characters of the response to be ”220 ”, then sends ”QUIT\r\n”. Default port 21 is used if not specified.

http

Creates a TCP connection without expecting and sending anything. Default port 80 is used if not specified.

https

Uses (and only works with) libcurl, does not verify the authenticity of the certificate, does not verify the host name in the SSL certificate, only fetches the response header (HEAD request). Default port 443 is used if not specified.

imap

Creates a TCP connection and expects the first 4 characters of the response to be ”* OK”, then sends ”a1 LOGOUT\r\n”. Default port 143 is used if not specified.

ldap

Opens a connection to an LDAP server and performs an LDAP search operation with filter set to (objectClass=*). Expects successful retrieval of the first attribute of the first entry. Default port 389 is used if not specified.

nntp

Creates a TCP connection and expects the first 3 characters of the response to be ”200” or ”201”, then sends ”QUIT\r\n”. Default port 119 is used if not specified.

pop

Creates a TCP connection and expects the first 3 characters of the response to be ”+OK”, then sends ”QUIT\r\n”. Default port 110 is used if not specified.

smtp

Creates a TCP connection and expects the first 3 characters of the response to be ”220”, followed by a space, the line ending or a dash. The lines containing a dash belong to a multiline response and the response will be re-read until a line without the dash is received. Then sends ”QUIT\r\n”. Default port 25 is used if not specified.

ssh

Creates a TCP connection. If the connection has been established, both sides exchange an identification string (SSH-major.minor-XXXX), where major and minor are protocol versions and XXXX is a string. Zabbix checks if the string matching the specification is found and then sends back the string ”SSH-major.minor-zabbix_agent\r\n” or ”0\n” on mismatch. Default port 22 is used if not specified.

tcp

Creates a TCP connection without expecting and sending anything. Unlike the other checks requires the port parameter to be specified.

telnet

Creates a TCP connection and expects a login prompt (’:’ at the end). Default port 23 is used if not specified.

Item net.udp.service parameters

ntp

Sends an SNTP packet over UDP and validates the response according to RFC 4330, section 5. Default port 123 is used if not specified.

11 proc.get parameters

Overview

The item **proc.get**[<name>,<user>,<cmdline>,<mode>] is supported on Linux, Windows, FreeBSD, OpenBSD, and NetBSD.

List of process parameters returned by the item varies depending on the operating system and ’mode’ argument value.

Linux

The following process parameters are returned on Linux for each mode:

mode=process	mode=thread	mode=summary
pid: PID	pid: PID	name: process name

mode=process	mode=thread	mode=summary
ppid: parent PID	ppid: parent PID	processes: number of processes
name: process name	name: process name	vsize: virtual memory size
cmdline: command with arguments	user: user (real) the process runs under	pmem: percentage of real memory
user: user (real) the process runs under	group: group (real) the process runs under	rss: resident set size
group: group (real) the process runs under	uid: user ID	data: size of data segment
uid: user ID	gid: ID of the group the process runs under	exe: size of code segment
gid: ID of the group the process runs under	tid: thread ID	lib: size of shared libraries
vsize: virtual memory size	tname: thread name	lck: size of locked memory
pmem: percentage of real memory	cputime_user: total CPU seconds (user)	pin: size of pinned pages
rss: resident set size	cputime_system: total CPU seconds (system)	pte: size of page table entries
data: size of data segment	state: thread state	size: size of process code + data + stack segments
exe: size of code segment	ctx_switches: number of context switches	stk: size of stack segment
hwm: peak resident set size	page_faults: number of page faults	swap: size of swap space used
lck: size of locked memory		cputime_user: total CPU seconds (user)
lib: size of shared libraries		cputime_system: total CPU seconds (system)
peak: peak virtual memory size		ctx_switches: number of context switches
pin: size of pinned pages		threads: number of threads
pte: size of page table entries		page_faults: number of page faults
size: size of process code + data + stack segments		
stk: size of stack segment		
swap: size of swap space used		
cputime_user: total CPU seconds (user)		
cputime_system: total CPU seconds (system)		
state: process state (transparently retrieved from procfs, long form)		
ctx_switches: number of context switches		
threads: number of threads		
page_faults: number of page faults		

BSD-based OS

The following process parameters are returned on FreeBSD, OpenBSD, and NetBSD for each mode:

mode=process	mode=thread	mode=summary
pid: PID	pid: PID	name: process name
ppid: parent PID	ppid: parent PID	processes: number of processes
jid: ID of jail (FreeBSD only)	jid: ID of jail (FreeBSD only)	vsize: virtual memory size
jname: name of jail (FreeBSD only)	jname: name of jail (FreeBSD only)	pmem: percentage of real memory (FreeBSD only)
name: process name	name: process name	rss: resident set size
cmdline: command with arguments	user: user (real) the process runs under	size: size of process (code + data + stack)
user: user (real) the process runs under	group: group (real) the process runs under	tsize: text (code) size

mode=process	mode=thread	mode=summary
group: group (real) the process runs under	uid: user ID	dsize: data size
uid: user ID	gid: ID of the group the process runs under	ssize: stack size
gid: ID of the group the process runs under	tid: thread ID	cputime_user: total CPU seconds (user)
vsize: virtual memory size	tname: thread name	cputime_system: total CPU seconds (system)
pmem: percentage of real memory (FreeBSD only)	cputime_user: total CPU seconds (user)	ctx_switches: number of context switches
rss: resident set size	cputime_system: total CPU seconds (system)	threads: number of threads (not supported for NetBSD)
size: size of process (code + data + stack)	state: thread state	stk: size of stack segment
tsize: text (code) size	ctx_switches: number of context switches	page_faults: number of page faults
dsize: data size	io_read_op: number of times the system had to perform input	fds: number of file descriptors (OpenBSD only)
ssize: stack size	io_write_op: number of times the system had to perform output	swap: size of swap space used
cputime_user: total CPU seconds (user)		io_read_op: number of times the system had to perform input
cputime_system: total CPU seconds (system)		io_write_op: number of times the system had to perform output
state: process state (disk sleep/running/sleeping/tracing stop/zombie/other)		
ctx_switches: number of context switches		
threads: number of threads (not supported for NetBSD)		
page_faults: number of page faults		
fds: number of file descriptors (OpenBSD only)		
swap: size of swap space used		
io_read_op: number of times the system had to perform input		
io_write_op: number of times the system had to perform output		

Windows

The following process parameters are returned on Windows for each mode:

mode=process	mode=thread	mode=summary
pid: PID	pid: PID	name: process name
ppid: parent PID	ppid: parent PID	processes: number of processes
name: process name	name: process name	vmsize: virtual memory size
user: user the process runs under	user: user the process runs under	wkset: size of process working set
sid: user SID	sid: user SID	cputime_user: total CPU seconds (user)
vmsize: virtual memory size	tid: thread ID	cputime_system: total CPU seconds (system)
wkset: size of process working set		threads: number of threads
cputime_user: total CPU seconds (user)		page_faults: number of page faults
cputime_system: total CPU seconds (system)		handles: number of handles
threads: number of threads		io_read_b: IO bytes read
page_faults: number of page faults		io_write_b: IO bytes written

mode=process	mode=thread	mode=summary
handles: number of handles io_read_b: IO bytes read io_write_b: IO bytes written io_read_op: IO read operations io_write_op: IO write operations io_other_b: IO bytes transferred, other than read and write operations io_other_op: IO operations, other than read and write operations		io_read_op: IO read operations io_write_op: IO write operations io_other_b: IO bytes transferred, other than read and write operations io_other_op: IO operations, other than read and write operations

12 Unreachable/unavailable host interface settings

Overview

Several configuration parameters define how Zabbix server should behave when an agent check (Zabbix, SNMP, IPMI, JMX) fails and a host interface becomes unreachable.

Unreachable interface

A host interface is treated as unreachable after a failed check (network error, timeout) by Zabbix, SNMP, IPMI or JMX agents. Note that Zabbix agent active checks do not influence interface availability in any way.

From that moment **UnreachableDelay** defines how often an interface is rechecked using one of the items (including LLD rules) in this unreachability situation and such rechecks will be performed already by unreachable pollers (or IPMI pollers for IPMI checks). By default it is 15 seconds before the next check.

In the Zabbix server log unreachability is indicated by messages like these:

```
Zabbix agent item "system.cpu.load[percpu,avg1]" on host "New host" failed: first network error, wait for
Zabbix agent item "system.cpu.load[percpu,avg15]" on host "New host" failed: another network error, wait f
```

Note that the exact item that failed is indicated and the item type (Zabbix agent).

Note:
The *Timeout* parameter will also affect how early an interface is rechecked during unreachability. If the Timeout is 20 seconds and UnreachableDelay 30 seconds, the next check will be in 50 seconds after the first attempt.

The **UnreachablePeriod** parameter defines how long the unreachability period is in total. By default UnreachablePeriod is 45 seconds. UnreachablePeriod should be several times bigger than UnreachableDelay, so that an interface is rechecked more than once before an interface becomes unavailable.

Switching interface back to available

When the unreachability period is over, the interface is polled again, decreasing priority for item that turned the interface into unreachable state. If the unreachable interface reappears, the monitoring returns to normal automatically:

```
resuming Zabbix agent checks on host "New host": connection restored
```

Note:
Once interface becomes available, the host does not poll all its items immediately for two reasons:
- It might overload the host.
- The interface restore time is not always matching planned item polling schedule time.

So, after the interface becomes available, items are not polled immediately, but they are getting rescheduled to their next polling round.

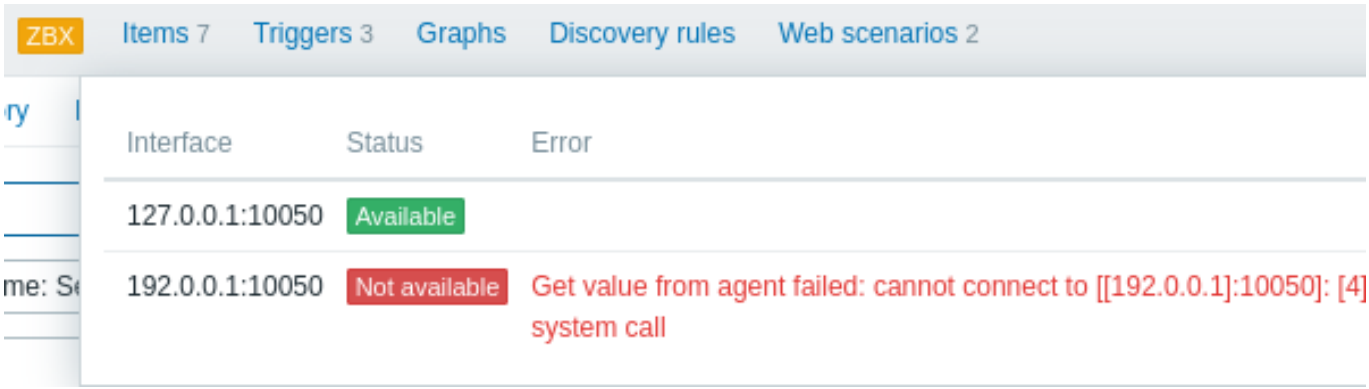
Unavailable interface

After the UnreachablePeriod ends and the interface has not reappeared, the interface is treated as unavailable.

In the server log it is indicated by messages like these:

```
temporarily disabling Zabbix agent checks on host "New host": interface unavailable
```

and in the frontend the host availability icon goes from green/gray to yellow/red (the unreachable interface details can be seen in the hint box that is displayed when a mouse is positioned on the host availability icon):



The **UnavailableDelay** parameter defines how often an interface is checked during interface unavailability.

By default it is 60 seconds (so in this case "temporarily disabling", from the log message above, will mean disabling checks for one minute).

When the connection to the interface is restored, the monitoring returns to normal automatically, too:

```
enabling Zabbix agent checks on host "New host": interface became available
```

13 Remote monitoring of Zabbix stats

Overview

It is possible to make some internal metrics of Zabbix server and proxy accessible remotely by another Zabbix instance or a third-party tool. This can be useful so that supporters/service providers can monitor their client Zabbix servers/proxies remotely or, in organizations where Zabbix is not the main monitoring tool, that Zabbix internal metrics can be monitored by a third-party system in an umbrella-monitoring setup.

Zabbix internal stats are exposed to a configurable set of addresses listed in the new 'StatsAllowedIP' server/proxy parameter. Requests will be accepted only from these addresses.

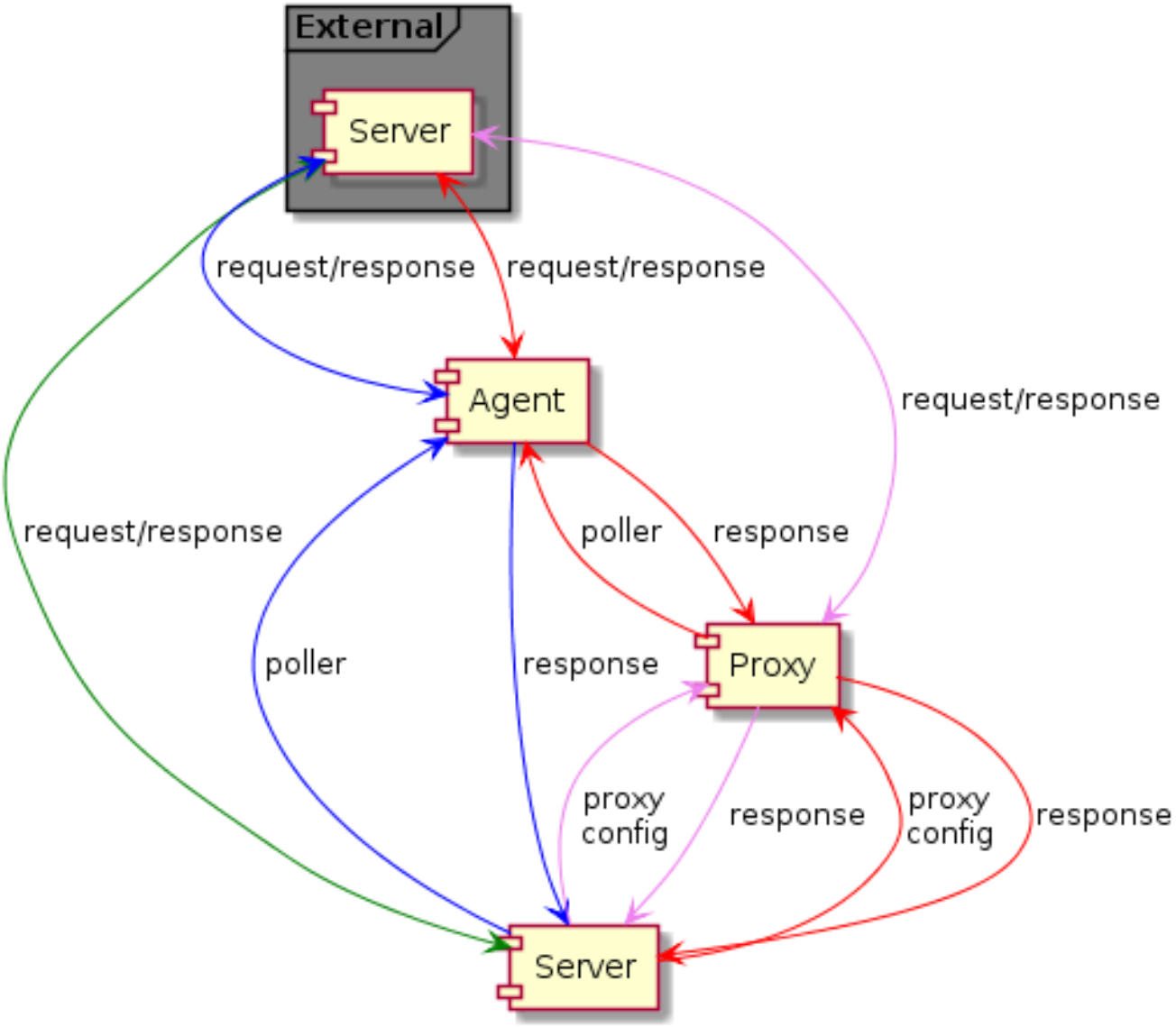
Items

To configure querying of internal stats on another Zabbix instance, you may use two items:

- `zabbix[stats,<ip>,<port>]` internal item - for direct remote queries of Zabbix server/proxy. <ip> and <port> are used to identify the target instance.
- `zabbix.stats[<ip>,<port>]` agent item - for agent-based remote queries of Zabbix server/proxy. <ip> and <port> are used to identify the target instance.

See also: Internal items, Zabbix agent items

The following diagram illustrates the use of either item depending on the context.



- - Server → external Zabbix instance (`zabbix[stats,<ip>,<port>]`)
- - Server → proxy → external Zabbix instance (`zabbix[stats,<ip>,<port>]`)
- - Server → agent → external Zabbix instance (`zabbix.stats[<ip>,<port>]`)
- - Server → proxy → agent → external Zabbix instance (`zabbix.stats[<ip>,<port>]`)

To make sure that the target instance allows querying it by the external instance, list the address of the external instance in the 'StatsAllowedIP' parameter on the target instance.

Exposed metrics

The stats items gather the statistics in bulk and return a JSON, which is the basis for dependent items to get their data from. The following internal metrics are returned by either of the two items:

- `zabbix[boottime]`
- `zabbix[hosts]`
- `zabbix[items]`
- `zabbix[items_unsupported]`
- `zabbix[preprocessing_queue]` (server only)
- `zabbix[process,<type>,<mode>,<state>]` (only process type based statistics)
- `zabbix[rcache,<cache>,<mode>]`
- `zabbix[requiredperformance]`
- `zabbix[triggers]` (server only)
- `zabbix[uptime]`
- `zabbix[vcache,buffer,<mode>]` (server only)
- `zabbix[vcache,cache,<parameter>]`

- zabbix[version]
- zabbix[vmware,buffer,<mode>]
- zabbix[wcache,<cache>,<mode>] ('trends' cache type server only)

Templates

Templates are available for remote monitoring of Zabbix server or proxy internal metrics from an external instance:

- Remote Zabbix server health
- Remote Zabbix proxy health

Note that in order to use a template for remote monitoring of multiple external instances, a separate host is required for each external instance monitoring.

Trapper process

Receiving internal metric requests from an external Zabbix instance is handled by the trapper process that validates the request, gathers the metrics, creates the JSON data buffer and sends the prepared JSON back, for example, from server:

```
{
  "response": "success",
  "data": {
    "boottime": N,
    "uptime": N,
    "hosts": N,
    "items": N,
    "items_unsupported": N,
    "preprocessing_queue": N,
    "process": {
      "alert manager": {
        "busy": {
          "avg": N,
          "max": N,
          "min": N
        },
        "idle": {
          "avg": N,
          "max": N,
          "min": N
        },
        "count": N
      },
...
    },
    "queue": N,
    "rcache": {
      "total": N,
      "free": N,
      "pfree": N,
      "used": N,
      "pused": N
    },
    "requiredperformance": N,
    "triggers": N,
    "uptime": N,
    "vcache": {
      "buffer": {
        "total": N,
        "free": N,
        "pfree": N,
        "used": N,
        "pused": N
      },
      "cache": {
        "requests": N,
        "hits": N,
```

```
        "misses": N,
        "mode": N
      }
    },
    "vmware": {
      "total": N,
      "free": N,
      "pfree": N,
      "used": N,
      "pused": N
    },
    "version": "N",
    "wcache": {
      "values": {
        "all": N,
        "float": N,
        "uint": N,
        "str": N,
        "log": N,
        "text": N,
        "not supported": N
      },
      "history": {
        "pfree": N,
        "free": N,
        "total": N,
        "used": N,
        "pused": N
      },
      "index": {
        "pfree": N,
        "free": N,
        "total": N,
        "used": N,
        "pused": N
      },
      "trend": {
        "pfree": N,
        "free": N,
        "total": N,
        "used": N,
        "pused": N
      }
    }
  }
}
```

Internal queue items

There are also another two items specifically allowing to remote query internal queue stats on another Zabbix instance:

- `zabbix[stats,<ip>,<port>,queue,<from>,<to>]` internal item - for direct internal queue queries to remote Zabbix server/proxy
- `zabbix.stats[<ip>,<port>,queue,<from>,<to>]` agent item - for agent-based internal queue queries to remote Zabbix server/proxy

See also: Internal items, Zabbix agent items

14 Configuring Kerberos with Zabbix

Overview

Kerberos authentication can be used in web monitoring and HTTP items in Zabbix since version 4.4.0.

This section describes an example of configuring Kerberos with Zabbix server to perform web monitoring of `www.example.com` with user 'zabbix'.

Steps

Step 1

Install Kerberos package.

For Debian/Ubuntu:

```
apt install krb5-user
```

For RHEL:

```
dnf install krb5-workstation
```

Step 2

Configure Kerberos configuration file (see MIT documentation for details)

```
cat /etc/krb5.conf
[libdefaults]
    default_realm = EXAMPLE.COM
#### The following krb5.conf variables are only for MIT Kerberos.
    kdc_timesync = 1
    ccache_type = 4
    forwardable = true
    proxiable = true

[realms]
    EXAMPLE.COM = {
    }

[domain_realm]
    .example.com=EXAMPLE.COM
    example.com=EXAMPLE.COM
```

Step 3

Create a Kerberos ticket for user *zabbix*. Run the following command as user *zabbix*:

```
kinit zabbix
```

Attention:
It is important to run the above command as user *zabbix*. If you run it as *root* the authentication will not work.

Step 4

Create a web scenario or HTTP agent item with Kerberos authentication type.

Optionally can be tested with the following curl command:

```
curl -v --negotiate -u : http://example.com
```

Note that for lengthy web monitoring it is necessary to take care of renewing the Kerberos ticket. Default time of ticket expiration is 10h.

15 modbus.get parameters

Overview

The table below presents details of the modbus.get[] item parameters.

Parameters

Parameter	Description	Defaults	Example
endpoint	Protocol and address of the endpoint, defined as `protocol://connection_string` Possible protocol values: *rtu*, *ascii* (Agent 2 only), *tcp* Connection string format: with *tcp* - `address:port` with serial line: *rtu*, *ascii* - `port_name:speed:params` where 'speed' - 1200, 9600 etc 'params' - data bits (5,6,7 or 8), parity (n,e or o for none/even/odd), stop bits (1 or 2)	protocol: none *rtu/ascii* protocol: port_name: none speed: 115200 params: 8n1 *tcp* protocol: address: none port: 502	tcp://192.168.6.1:511 tcp://192.168.6.2 tcp://[::1]:511 tcp://::1 tcp://localhost:511 tcp://localhost rtu://COM1:9600:8n ascii://COM2:1200:7o2 rtu://ttyS0:9600 ascii://ttyS1
slave id	Modbus address of the device it is intended for (1 to 247), see MODBUS Messaging Implementation Guide (page 23) tcp device (not GW) will ignore the field	serial: 1 tcp: 255 (0xFF)	2
function	Empty or value of a supported function: 1 - Read Coil, 2 - Read Discrete Input, 3 - Read Holding Registers, 4 - Read Input Registers	empty	3
address	Address of the first registry, coil or input. If 'function' is empty, then 'address' should be in range for: Coil - 00001 - 09999 Discrete input - 10001 - 19999 Input register - 30001 - 39999 Holding register - 40001 - 49999 If 'function' is not empty, the 'address' field will be from 0 till 65535 and used without modification (PDU)	empty function: 00001 non-empty function: 0	9999
count	Count of sequenced 'type' which will be read from device, where: for Coil or Discrete input the 'type' = 1 bit for other cases: (count*type)/2 = real count of registers for reading If 'offset' is not 0, the value will be added to 'real count' Acceptable range for 'real count' is 1:65535	1	2
type	Data type: for Read Coil and Read Discrete Input - *bit* for Read Holding Registers and Read Input Registers: *int8* - 8bit *uint8* - 8bit (unsigned) *int16* - 16bit *uint16* - 16bit (unsigned) *int32* - 32bit *uint32* - 32bit (unsigned) *float* - 32bit *uint64* - 64bit (unsigned) *double* - 64bit	bit uint16	uint64

Parameter	Description	Defaults	Example
endianness	Endianness type: *be* - Big Endian *le* - Little Endian *mbe* - Mid-Big Endian *mle* - Mid-Little Endian Limitations: for 1 bit - be for 8 bits - be,le for 16 bits - be,le	be	le
offset	Number of registers, starting from 'address', the result of which will be discarded. The size of each register is 16bit (needed to support equipment that does not support random read access).	0	4

16 Creating custom performance counter names for VMware

Overview

The VMware performance counter path has the `group/counter[rollup]` format where:

- `group` - the performance counter group, for example *cpu*
- `counter` - the performance counter name, for example *usagemhz*
- `rollup` - the performance counter rollup type, for example *average*

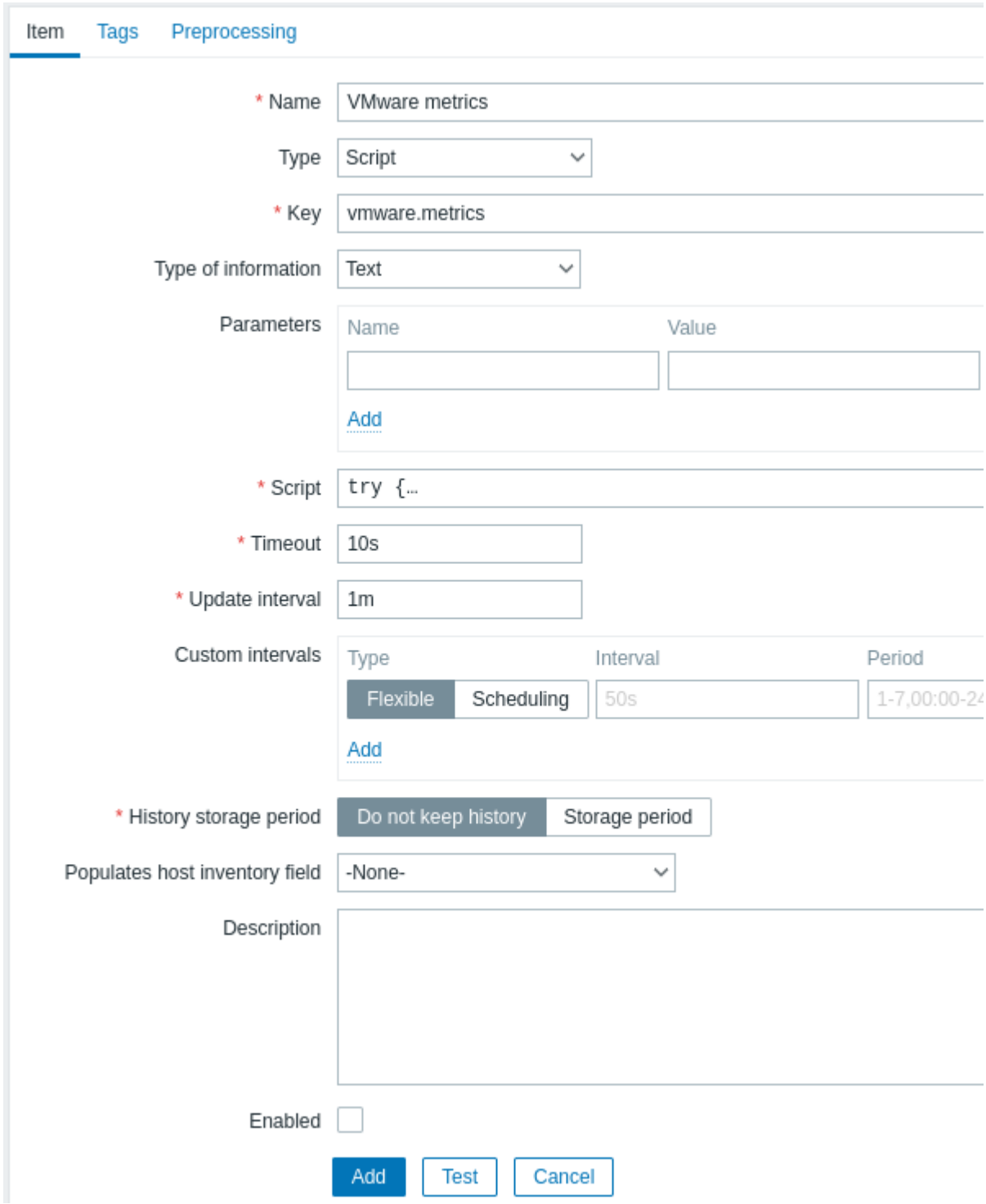
So the above example would give the following counter path: `cpu/usagemhz[average]`

The performance counter group descriptions, counter names and rollup types can be found in VMware documentation.

It is possible to obtain internal names and create custom performance counter names by using script item in Zabbix.

Configuration

1. Create disabled Script item on the main VMware host (where the **eventlog[]** item is present) with the following parameters:



- *Name*: VMware metrics
- *Type*: Script
- *Key*: vmware.metrics
- *Type of information*: Text
- *Script*: copy and paste the script provided below
- *Timeout*: 10
- *History storage period*: Do not keep history
- *Enabled*: unmarked

Script

```
try {
    Zabbix.log(4, 'vmware metrics script');

    var result, resp,
    req = new HttpRequest();
    req.addHeader('Content-Type: application/xml');
```

```
    req.addHeader('SOAPAction: "urn:vim25/6.0"');

    login = '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:vi
    <soapenv:Header/>\
    <soapenv:Body>\
        <urn:Login>\
            <urn:_this type="SessionManager">SessionManager</urn:_this>\
            <urn:userName>{$VMWARE.USERNAME}</urn:userName>\
            <urn:password>{$VMWARE.PASSWORD}</urn:password>\
        </urn:Login>\
    </soapenv:Body>\
</soapenv:Envelope>'
    resp = req.post("{$VMWARE.URL}", login);
    if (req.getStatus() != 200) {
        throw 'Response code: '+req.getStatus();
    }

    query = '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:vi
<soapenv:Header/>\
    <soapenv:Body>\
        <urn:RetrieveProperties>\
            <urn:_this type="PropertyCollector">propertyCollector</urn:_this>\
            <urn:specSet>\
                <urn:propSet>\
                   <urn:type>PerformanceManager</urn:type>\
                   <urn:pathSet>perfCounter</urn:pathSet>\
                </urn:propSet>\
                <urn:objectSet>\
                   <urn:obj type="PerformanceManager">PerfMgr</urn:obj>\
                </urn:objectSet>\
            </urn:specSet>\
        </urn:RetrieveProperties>\
    </soapenv:Body>\
</soapenv:Envelope>'
    resp = req.post("{$VMWARE.URL}", query);
    if (req.getStatus() != 200) {
        throw 'Response code: '+req.getStatus();
    }
    Zabbix.log(4, 'vmware metrics=' + resp);
    result = resp;

    logout = '<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:v
    <soapenv:Header/>\
    <soapenv:Body>\
        <urn:Logout>\
            <urn:_this type="SessionManager">SessionManager</urn:_this>\
        </urn:Logout>\
    </soapenv:Body>\
</soapenv:Envelope>'

    resp = req.post("{$VMWARE.URL}",logout);
    if (req.getStatus() != 200) {
        throw 'Response code: '+req.getStatus();
    }

} catch (error) {
    Zabbix.log(4, 'vmware call failed : '+error);
    result = {};
}

return result;
```

Once the item is configured, press *Test* button, then press *Get value*.



Copy received XML to any XML formatter and find the desired metric.

An example of XML for one metric:

```
<PerfCounterInfo xsi:type="PerfCounterInfo">
    <key>6</key>
    <nameInfo>
        <label>Usage in MHz</label>
        <summary>CPU usage in megahertz during the interval</summary>
        <key>usagemhz</key>
    </nameInfo>
    <groupInfo>
        <label>CPU</label>
        <summary>CPU</summary>
        <key>cpu</key>
    </groupInfo>
    <unitInfo>
        <label>MHz</label>
        <summary>Megahertz</summary>
        <key>megaHertz</key>
    </unitInfo>
    <rollupType>average</rollupType>
    <statsType>rate</statsType>
    <level>1</level>
    <perDeviceLevel>3</perDeviceLevel>
</PerfCounterInfo>
```

Use XPath to extract the counter path from received XML. For the example above, the XPath will be:

field	xPath	value
group	//groupInfo[../key=6]/key	cpu
counter	//nameInfo[../key=6]/key	usagemhz
rollup	//rollupType[../key=6]	average

Resulting performance counter path in this case is: `cpu/usagemhz[average]`

17 Return values

Overview

This section provides return value details for some Zabbix agent items.

system.sw.packages.get

The output of this item is an array of objects each containing the following keys:

- **name** - package name
- **manager** - package manager that reported this data (`rpm`, `dpkg`, `pacman`, or `pkgtool`)
- **version** - package version
- **size** - uncompressed package size in bytes (if not available, set to 0 (for Zabbix agent 2 - since 6.4.11))
- **arch** - package architecture
- **buildtime** - an object with 2 entries:
  - **timestamp** - UNIX timestamp when the package was built (if not available, set to 0)
  - **value** - human readable date and time when the package was built (if not available, set to empty string)
- **installtime** - an object with 2 entries:

- **timestamp** - UNIX timestamp when the package was installed (if not available, set to 0)
- **value** - human readable date and time when the package was installed (if not available, set to empty string)

For example:

```
[
    {
        "name": "util-linux-core",
        "manager": "rpm",
        "version": "2.37.4-3.el9",
        "size": 1296335,
        "arch": "x86_64",
        "buildtime": {
            "timestamp" : 1653552239,
            "value" : "Sep 20 01:39:40 2021 UTC"
        },
        "installtime": {
            "timestamp" : 1660780885,
            "value" : "Aug 18 00:01:25 2022 UTC"
        }
    },
    {
        "name": "xfonts-base",
        "manager": "dpkg",
        "version": "1:1.0.5",
        "size": 7337984,
        "arch": "all",
        "buildtime": {
            "timestamp": 0,
            "value": ""
        },
        "installtime": {
            "timestamp": 0,
            "value": ""
        }
    }
]
```

5 Supported functions

Click on the respective function group to see more details.

Function group	Functions
Aggregate functions	avg, bucket_percentile, count, histogram_quantile, item_count, kurtosis, mad, max, min, skewness, stddevpop, stddevsamp, sum, sumofsquares, varpop, varsamp
Foreach functions	avg_foreach, bucket_rate_foreach, count_foreach, exists_foreach, last_foreach, max_foreach, min_foreach, sum_foreach
Bitwise functions	bitand, bitlshift, bitnot, bitor, bitrshift, bitxor
Date and time functions	date, dayofmonth, dayofweek, now, time
History functions	change, changecount, count, countunique, find, first, fuzzytime, last, logeventid, logseverity, logsource, monodec, monoinc, nodata, percentile, rate

Function group	Functions
Trend functions	baselinedev, baselinewma, trendavg, trendcount, trendmax, trendmin, trendstl, trendsum
Mathematical functions	abs, acos, asin, atan, atan2, avg, cbrt, ceil, cos, cosh, cot, degrees, e, exp, expm1, floor, log, log10, max, min, mod, pi, power, radians, rand, round, signum, sin, sinh, sqrt, sum, tan, truncate
Operator functions	between, in
Prediction functions	forecast, timeleft
String functions	ascii, bitlength, bytelength, char, concat, insert, left, length, ltrim, mid, repeat, replace, right, rtrim, trim

These functions are supported in trigger expressions and calculated items.

Foreach functions are supported only for aggregate calculations.

1 Aggregate functions

Except where stated otherwise, all functions listed here are supported in:

- Trigger expressions
- Calculated items

Aggregate functions can work with either:

- history of items, for example, `min(/host/key,1h)`
- foreach functions as the only parameter, for example, `min(last_foreach(/*/key))` (only in calculated items; cannot be used in triggers)

The functions are listed without additional information. Click on the function to see the full details.

Function	Description
avg	The average value of an item within the defined evaluation period.
bucket_percentile	Calculates the percentile from the buckets of a histogram.
count	The count of values in an array returned by a foreach function.
histogram_quantile	Calculates the φ-quantile from the buckets of a histogram.
item_count	The count of existing items in configuration that match the filter criteria.
kurtosis	The "tailedness" of the probability distribution in collected values within the defined evaluation period.
mad	The median absolute deviation in collected values within the defined evaluation period.
max	The highest value of an item within the defined evaluation period.
min	The lowest value of an item within the defined evaluation period.
skewness	The asymmetry of the probability distribution in collected values within the defined evaluation period.
stddevpop	The population standard deviation in collected values within the defined evaluation period.
stddevsamp	The sample standard deviation in collected values within the defined evaluation period.
sum	The sum of collected values within the defined evaluation period.
sumofsquares	The sum of squares in collected values within the defined evaluation period.
varpop	The population variance of collected values within the defined evaluation period.
varsamp	The sample variance of collected values within the defined evaluation period.

Common parameters

- `/host/key` is a common mandatory first parameter for the functions referencing the host item history
- `(sec|#num)<:time shift>` is a common second parameter for the functions referencing the host item history, where:
  - **sec** - maximum evaluation period in seconds (time suffixes can be used), or
  - **#num** - maximum evaluation range in latest collected values (if preceded by a hash mark)

- **time shift** (optional) allows to move the evaluation point back in time. See more details on specifying time shift.

Function details

Some general notes on function parameters:

- Function parameters are separated by a comma
- Optional function parameters (or parameter parts) are indicated by < >
- Function-specific parameters are described with each function
- `/host/key` and `(sec|#num)<:time shift>` parameters must never be quoted

avg(/host/key,(sec|#num)<:time shift>)

The average value of an item within the defined evaluation period.
 Supported value types: *Float*, *Integer*.
 Supported foreach functions: *avg_foreach*, *count_foreach*, *exists_foreach*, *last_foreach*, *max_foreach*, *min_foreach*, *sum_foreach*.

Parameters: see common parameters.

Time shift is useful when there is a need to compare the current average value with the average value some time ago.

Examples:

```
avg(/host/key,1h) #the average value for the last hour until now
avg(/host/key,1h:now-1d) #the average value for an hour from 25 hours ago to 24 hours ago from now
avg(/host/key,#5) #the average value of the five latest values
avg(/host/key,#5:now-1d) #the average value of the five latest values excluding the values received in the
```

bucket_percentile(item filter,time period,percentage)

Calculates the percentile from the buckets of a histogram.

Parameters:

- **item filter** - see item filter;
- **time period** - see time period;
- **percentage** - percentage (0-100).

Comments:

- Supported only in calculated items;
- This function is an alias for `histogram_quantile(percentage/100, bucket_rate_foreach(item filter, time period, 1))`.

count(func_foreach(item filter,<time period>))

The count of values in an array returned by a foreach function.
 Supported value type: *Integer*.
 Supported foreach functions: *avg_foreach*, *count_foreach*, *exists_foreach*, *last_foreach*, *max_foreach*, *min_foreach*, *sum_foreach*.

Parameters:

- **func_foreach** - foreach function for which the number of returned values should be counted (with supported arguments). See foreach functions for details.
- **item filter** - see item filter;
- **time period** - see time period.

Using **count()** with a history-related foreach function (max_foreach, avg_foreach, etc.) may lead to performance implications, whereas using **exists_foreach()**, which works only with configuration data, will not have such effect.

Examples:

```
count(max_foreach(/*/net.if.in[*],1h)) #the number of net.if.in items that received data in the last hour
```

histogram_quantile(quantile,bucket1,value1,bucket2,value2,...)

Calculates the φ-quantile from the buckets of a histogram.
 Supported foreach function: *bucket_rate_foreach*.

Parameters:

- **quantile** - $0 \leq \varphi \leq 1$;
- **bucketN, valueN** - manually entered pairs (>=2) of parameters or the response of bucket_rate_foreach.

Comments:

- Supported only in calculated items;
- Functionally corresponds to 'histogram_quantile' of PromQL;
- Returns -1 if values of the last 'Infinity' bucket ("+*inf*") are equal to 0.

Examples:

```
histogram_quantile(0.75,1.0,last(/host/rate_bucket[1.0]),"+Inf",last(/host/rate_bucket[Inf]))
histogram_quantile(0.5,bucket_rate_foreach(//item_key,30s))
```

item_count(item filter)

The count of existing items in configuration that match the filter criteria.
 Supported value type: *Integer*.

Parameter:

- **item filter** - criteria for item selection, allows referencing by host group, host, item key, and tags. Wildcards are supported. See item filter for more details.

Comments:

- Supported only in calculated items;
- Works as an alias for the *count(exists_foreach(item_filter))* function.

Examples:

```
item_count(/*/agent.ping?[group="Host group 1"]) #the number of hosts with the *agent.ping* item in the "H
```

kurtosis(/host/key,(sec|#num)<:time shift>)

The "tailedness" of the probability distribution in collected values within the defined evaluation period. See also: Kurtosis.
 Supported value types: *Float*, *Integer*.
 Supported foreach function: *last_foreach*.

Parameters: see common parameters.

Example:

```
kurtosis(/host/key,1h) #kurtosis for the last hour until now
```

mad(/host/key,(sec|#num)<:time shift>)

The median absolute deviation in collected values within the defined evaluation period. See also: Median absolute deviation.
 Supported value types: *Float*, *Integer*.
 Supported foreach function: *last_foreach*.

Parameters: see common parameters.

Example:

```
mad(/host/key,1h) #median absolute deviation for the last hour until now
```

max(/host/key,(sec|#num)<:time shift>)

The highest value of an item within the defined evaluation period.
 Supported value types: *Float*, *Integer*.
 Supported foreach functions: *avg_foreach*, *count_foreach*, *exists_foreach*, *last_foreach*, *max_foreach*, *min_foreach*, *sum_foreach*.

Parameters: see common parameters.

Example:

```
max(/host/key,1h) - min(/host/key,1h) #calculate the difference between the maximum and minimum values wit
```

min(/host/key,(sec|#num)<:time shift>)

The lowest value of an item within the defined evaluation period.
 Supported value types: *Float*, *Integer*.
 Supported foreach functions: *avg_foreach*, *count_foreach*, *exists_foreach*, *last_foreach*, *max_foreach*, *min_foreach*, *sum_foreach*.

Parameters: see common parameters.

Example:

```
max(/host/key,1h) - min(/host/key,1h) #calculate the difference between the maximum and minimum values wit
```

skewness(/host/key,(sec|#num)<:time shift>)

The asymmetry of the probability distribution in collected values within the defined evaluation period. See also: Skewness.
 Supported value types: *Float*, *Integer*.
 Supported foreach function: *last_foreach*.

Parameters: see common parameters.

Example:

```
skewness(/host/key,1h) #the skewness for the last hour until now
```

stddevpop(/host/key,(sec|#num)<:time shift>)

The population standard deviation in collected values within the defined evaluation period. See also: Standard deviation.
 Supported value types: *Float*, *Integer*.
 Supported foreach function: *last_foreach*.

Parameters: see common parameters.

Example:

```
stddevpop(/host/key,1h) #the population standard deviation for the last hour until now
```

stddevsamp(/host/key,(sec|#num)<:time shift>)

The sample standard deviation in collected values within the defined evaluation period. See also: Standard deviation.
 Supported value types: *Float*, *Integer*.
 Supported foreach function: *last_foreach*.

Parameters: see common parameters.

At least two data values are required for this function to work.

Example:

```
stddevsamp(/host/key,1h) #the sample standard deviation for the last hour until now
```

sum(/host/key,(sec|#num)<:time shift>)

The sum of collected values within the defined evaluation period.
 Supported value types: *Float*, *Integer*.
 Supported foreach functions: *avg_foreach*, *count_foreach*, *exists_foreach*, *last_foreach*, *max_foreach*, *min_foreach*, *sum_foreach*.

Parameters: see common parameters.

Example:

```
sum(/host/key,1h) #the sum of values for the last hour until now
```

sumofsquares(/host/key,(sec|#num)<:time shift>)

The sum of squares in collected values within the defined evaluation period.
 Supported value types: *Float*, *Integer*.
 Supported foreach function: *last_foreach*.

Parameters: see common parameters.

Example:

```
sumofsquares(/host/key,1h) #the sum of squares for the last hour until now
```

varpop(/host/key,(sec|#num)<:time shift>)

The population variance of collected values within the defined evaluation period. See also: Variance.
 Supported value types: *Float*, *Integer*.
 Supported foreach function: *last_foreach*.

Parameters: see common parameters.

Example:

```
varpop(/host/key,1h) #the population variance for the last hour until now
```

varsamp(/host/key,(sec|#num)<:time shift>)

The sample variance of collected values within the defined evaluation period. See also: Variance.
 Supported value types: *Float*, *Integer*.
 Supported foreach function: *last_foreach*.

Parameters: see common parameters.

At least two data values are required for this function to work.

Example:

```
varsamp(/host/key,1h) #the sample variance for the last hour until now
```

See all supported functions.

1 Foreach functions

Overview

Foreach functions are used in aggregate calculations to return one aggregate value for each item that is selected by the used **item filter**. An array of values is returned.

For example, the *avg_foreach* function will return an array of values, where each value is the *average* history value of the selected item, during the time interval that is specified.

The item filter is part of the syntax used by foreach functions. The use of wildcards is supported in the item filter, thus the required items can be selected quite flexibly.

Supported functions

Function	Description
avg_foreach	Returns the average value for each item.
bucket_rate_foreach	Returns pairs (bucket upper bound, rate value) suitable for use in the histogram_quantile() function, where "bucket upper bound" is the value of item key parameter defined by the <parameter number> parameter.
count_foreach	Returns the number of values for each item.
exists_foreach	Returns '1' for each enabled item.
last_foreach	Returns the last value for each item.
max_foreach	Returns the maximum value for each item.
min_foreach	Returns the minimum value for each item.
sum_foreach	Returns the sum of values for each item.

Function syntax

Foreach functions support two common parameters: `item filter` (see details below) and `time period`:

```
foreach_function(item filter,time period)
```

For example:

```
avg_foreach(/*/mysql.qps?[group="MySQL Servers"],5m)
```

will return the five-minute average of each 'mysql.qps' item in the MySQL server group.

Note that some functions support additional parameters.

Item filter syntax

The item filter:

```
/host/key[parameters]?[conditions]
```

consists of four parts, where:

- host - host name
- key - item key (without parameters)
- parameters - item key parameters
- conditions - host group and/or item tag based conditions (as expression)

Spaces are allowed only inside the conditions expression.

Wildcard usage

- Wildcard can be used to replace the host name, item key or an individual item key parameter.
- Either the host or item key must be specified without wildcard. So `/host/*` and `/*/key` are valid filters, but `/*/*` is invalid.
- Wildcard cannot be used for a *part* of host name, item key, item key parameter.
- Wildcard does not match more than a single item key parameter. So a wildcard must be specified for each parameter in separation (i.e. `key[abc,*,*]`).

Conditions expression

The conditions expression supports:

- operands:
  - group - host group
  - tag - item tag
  - "<text>" - string constant, with the \ escape character to escape " and \
- case-sensitive string comparison operators: =, <>
- logical operators: `and`, `or`, `not`
- grouping with parentheses: `( )`

Quotation of string constants is mandatory. Only case-sensitive full string comparison is supported.

Warning:
When specifying tags in the filter (i.e. `tag="tagname:value"`), the colon ":" is used as a delimiter. Everything after it is considered the tag value. Thus it is currently not supported to specify a tag name containing ":" in it.

Examples

A complex filter may be used, referencing the item key, host group and tags, as illustrated by the examples:

Syntax example	Description
`/host/key[abc,*]`	Matches similar items on this host.
`/*/key`	Matches the same item of any host.
`/*/key?[group="ABC" and tag="tagname:value"]`	Matches the same item of any host from the ABC group having 'tagname:value' tags.
`/*/key[a,*,c]?[(group="ABC" and tag="Tag1") or (group="DEF" and (tag="Tag2" or tag="Tag3:value"))]`	Matches similar items of any host from the ABC or DEF group with the respective tags.

All referenced items must exist and collect data. Only enabled items on enabled hosts are included in the calculations.

Attention:
If the item key of a referenced item is changed, the filter must be updated manually.

Specifying a parent host group includes the parent group and all nested host groups with their items.

Time period

The **second** parameter allows to specify the time period for aggregation. The time period can only be expressed as time, the amount of values (prefixed with **#**) is not supported.

Supported unit symbols can be used in this parameter for convenience, for example '5m' (five minutes) instead of '300s' (300 seconds) or '1d' (one day) instead of '86400' (86400 seconds).

Time period is ignored by the server if passed with the *last_foreach* function and can thus be omitted:

`last_foreach(/*/key?[group="host group"])`

Time period is not supported with the *exists_foreach* function.

Additional parameters

A third optional parameter is supported by the *bucket_rate_foreach* function:

`bucket_rate_foreach(item filter,time period,<parameter number>)`

where <parameter number> is the position of the "bucket" value in the item key. For example, if the "bucket" value in myItem[aaa,0.2] is '0.2', then its position is 2.

The default value of <parameter number> is '1'.

See aggregate calculations for more details and examples on using foreach functions.

Behavior depending on availability

The following table illustrates how each function behaves in cases of limited availability of host/item and history data.

Function	Disabled host	Unavailable host with data	Unavailable host without data	Disabled item	Unsupported item	Data retrieval error (SQL)
avg_foreach	ignore	return avg	ignore	ignore	ignore	ignore
bucket_rate_foreach	ignore	return bucket rate	ignore	ignore	ignore	ignore
count_foreach	ignore	return count	0	ignore	ignore	ignore
exists_foreach	ignore	1	1	ignore	1	n/a
last_foreach	ignore	return last	ignore	ignore	ignore	ignore
max_foreach	ignore	return max	ignore	ignore	ignore	ignore
min_foreach	ignore	return min	ignore	ignore	ignore	ignore
sum_foreach	ignore	return sum	ignore	ignore	ignore	ignore

If the item is *ignored*, nothing is added to the aggregation.

2 Bitwise functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

The functions are listed without additional information. Click on the function to see the full details.

Function	Description
bitand	The value of ”bitwise AND” of an item value and mask.
bitlshift	The bitwise shift left of an item value.
bitnot	The value of ”bitwise NOT” of an item value.
bitor	The value of ”bitwise OR” of an item value and mask.
bitrshift	The bitwise shift right of an item value.
bitxor	The value of ”bitwise exclusive OR” of an item value and mask.

Function details

Some general notes on function parameters:

- Function parameters are separated by a comma
- Expressions are accepted as parameters
- Optional function parameters (or parameter parts) are indicated by < >

bitand(value,mask)

The value of ”bitwise AND” of an item value and mask.
 Supported value types: *Integer*.

Parameter:

- **value** - the value to check;
- **mask** (mandatory) - a 64-bit unsigned integer (0 - 18446744073709551615).

Although the comparison is done in a bitwise manner, all the values must be supplied and are returned in decimal. For example, checking for the 3rd bit is done by comparing to 4, not 100.

Examples:

```
bitand(last(/host/key),12)=8 or bitand(last(/host/key),12)=4 #3rd or 4th bit set, but not both at the same
bitand(last(/host/key),20)=16 #3rd bit not set and 5th bit set
```

bitlshift(value,bits to shift)

The bitwise shift left of an item value.
 Supported value types: *Integer*.

Parameter:

- **value** - the value to check;
- **bits to shift** (mandatory) - the number of bits to shift.

Although the comparison is done in a bitwise manner, all the values must be supplied and are returned in decimal. For example, checking for the 3rd bit is done by comparing to 4, not 100.

bitnot(value)

The value of ”bitwise NOT” of an item value.
 Supported value types: *Integer*.

Parameter:

- **value** - the value to check.

Although the comparison is done in a bitwise manner, all the values must be supplied and are returned in decimal. For example, checking for the 3rd bit is done by comparing to 4, not 100.

bitor(value,mask)

The value of ”bitwise OR” of an item value and mask.
 Supported value types: *Integer*.

Parameter:

- **value** - the value to check;

- **mask** (mandatory) - a 64-bit unsigned integer (0 - 18446744073709551615).

Although the comparison is done in a bitwise manner, all the values must be supplied and are returned in decimal. For example, checking for the 3rd bit is done by comparing to 4, not 100.

bitrshift(value,bits to shift)

The bitwise shift right of an item value.
 Supported value types: *Integer*.

Parameter:

- **value** - the value to check;
- **bits to shift** (mandatory) - the number of bits to shift.

Although the comparison is done in a bitwise manner, all the values must be supplied and are returned in decimal. For example, checking for the 3rd bit is done by comparing to 4, not 100.

bitxor(value,mask)

The value of ”bitwise exclusive OR” of an item value and mask.
 Supported value types: *Integer*.

Parameter:

- **value** - the value to check;
- **mask** (mandatory) - a 64-bit unsigned integer (0 - 18446744073709551615).

Although the comparison is done in a bitwise manner, all the values must be supplied and are returned in decimal. For example, checking for the 3rd bit is done by comparing to 4, not 100.

See all supported functions.

3 Date and time functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

Attention:
Date and time functions cannot be used in the expression alone; at least one non-time-based function referencing the host item must be present in the expression.

The functions are listed without additional information. Click on the function to see the full details.

Function	Description
date	The current date in YYYYMMDD format.
dayofmonth	The day of month in range of 1 to 31.
dayofweek	The day of week in range of 1 to 7.
now	The number of seconds since the Epoch (00:00:00 UTC, January 1, 1970).
time	The current time in HHMMSS format.

Function details

date

The current date in YYYYMMDD format.

Example:

```
date()<20220101
```

dayofmonth

The day of month in range of 1 to 31.

Example:

```
dayofmonth()=1
```

dayofweek

The day of week in range of 1 to 7 (Mon - 1, Sun - 7).

Example (only weekdays):

`dayofweek()<6`

Example (only weekend):

`dayofweek()>5`

now

The number of seconds since the Epoch (00:00:00 UTC, January 1, 1970).

Example:

`now()<1640998800`

time

The current time in HHMMSS format.

Example (only nighttime, 00:00-06:00):

`time()<060000`

See all supported functions.

4 History functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

The functions are listed without additional information. Click on the function to see the full details.

Function	Description
change	The amount of difference between the previous and latest value.
changecount	The number of changes between adjacent values within the defined evaluation period.
count	The number of values within the defined evaluation period.
countunique	The number of unique values within the defined evaluation period.
find	Find a value match within the defined evaluation period.
first	The first (the oldest) value within the defined evaluation period.
fuzzytime	Check how much the passive agent time differs from the Zabbix server/proxy time.
last	The most recent value.
logeventid	Check if the event ID of the last log entry matches a regular expression.
logseverity	The log severity of the last log entry.
logsource	Check if log source of the last log entry matches a regular expression.
monodec	Check if there has been a monotonous decrease in values.
monoinc	Check if there has been a monotonous increase in values.
nodata	Check for no data received.
percentile	The P-th percentile of a period, where P (percentage) is specified by the third parameter.
rate	The per-second average rate of the increase in a monotonically increasing counter within the defined time period.

Common parameters

- `/host/key` is a common mandatory first parameter for the functions referencing the host item history
- `(sec|#num)<:time shift>` is a common second parameter for the functions referencing the host item history, where:
  - **sec** - maximum evaluation period in seconds (time suffixes can be used), or
  - **#num** - maximum evaluation range in latest collected values (if preceded by a hash mark)
  - **time shift** (optional) allows to move the evaluation point back in time. See more details on specifying time shift.

Function details

Some general notes on function parameters:

- Function parameters are separated by a comma

- Optional function parameters (or parameter parts) are indicated by < >
- Function-specific parameters are described with each function
- `/host/key` and `(sec|#num)<:time shift>` parameters must never be quoted

change(/host/key)

The amount of difference between the previous and latest value.
 Supported value types: *Float*, *Integer*, *String*, *Text*, *Log*.
 For strings returns: 0 - values are equal; 1 - values differ.

Parameters: see common parameters.

Comments:

- Numeric difference will be calculated, as seen with these incoming example values ('previous' and 'latest' value = difference):
'1' and '5' = +4
'3' and '1' = −2
'0' and '-2.5' = −2.5
- See also: abs for comparison.

Examples:

`change(/host/key)>10`

changecount(/host/key,(sec|#num)<:time shift>,<mode>)

The number of changes between adjacent values within the defined evaluation period.
 Supported value types: *Float*, *Integer*, *String*, *Text*, *Log*.

Parameters:

- See common parameters;
- **mode** (must be double-quoted) - possible values: *all* - count all changes (default); *dec* - count decreases; *inc* - count increases

For non-numeric value types, the *mode* parameter is ignored.

Examples:

```
changecount(/host/key,1w) #the number of value changes for the last week until now
changecount(/host/key,#10,"inc") #the number of value increases (relative to the adjacent value) among the
changecount(/host/key,24h,"dec") #the number of value decreases (relative to the adjacent value) for the l
```

count(/host/key,(sec|#num)<:time shift>,<operator>,<pattern>)

The number of values within the defined evaluation period.
 Supported value types: *Float*, *Integer*, *String*, *Text*, *Log*.

Parameters:

- See common parameters;
- **operator** (must be double-quoted). Supported `operators`:
eq - equal (default for integer, float)
ne - not equal
gt - greater
ge - greater or equal
lt - less
le - less or equal
like (default for string, text, log) - matches if contains pattern (case-sensitive)
bitand - bitwise AND
regexp - case-sensitive match of the regular expression given in `pattern`
iregexp - case-insensitive match of the regular expression given in `pattern`
- **pattern** - the required pattern (string arguments must be double-quoted).

Comments:

- Float items match with the precision of 2.22e-16; if database is not upgraded the precision is 0.000001.
- *like* is not supported as operator for integer values;
- *like* and *bitand* are not supported as operators for float values;
- For string, text, and log values only *eq*, *ne*, *like*, *regexp* and *iregexp* operators are supported;
- With *bitand* as operator, the fourth `pattern` parameter can be specified as two numbers, separated by '/': **number_to_compare_with/mask**. count() calculates "bitwise AND" from the value and the *mask* and compares the result to *number_to_compare_with*. If the result of "bitwise AND" is equal to *number_to_compare_with*, the value is counted.
If *number_to_compare_with* and *mask* are equal, only the *mask* need be specified (without '/').
- With *regexp* or *iregexp* as operator, the fourth `pattern` parameter can be an ordinary or global (starting with '@') regular expression. In case of global regular expressions case sensitivity is inherited from global regular expression settings. For the purpose of regexp matching, float values will always be represented with 4 decimal digits after '.'. Also note that for large numbers difference in decimal (stored in database) and binary (used by Zabbix server) representation may affect the 4th decimal digit.

Examples:

```
count(/host/key,10m) #the values for the last 10 minutes until now
count(/host/key,10m,"like","error") #the number of values for the last 10 minutes until now that contain '
count(/host/key,10m,,12) #the number of values for the last 10 minutes until now that equal '12'
count(/host/key,10m,"gt",12) #the number of values for the last 10 minutes until now that are over '12'
```

```
count(/host/key,#10,"gt",12) #the number of values within the last 10 values until now that are over '12'
count(/host/key,10m:now-1d,"gt",12) #the number of values between 24 hours and 10 minutes and 24 hours ago
count(/host/key,10m,"bitand","6/7") #the number of values for the last 10 minutes until now having '110' (
count(/host/key,10m:now-1d) #the number of values between 24 hours and 10 minutes and 24 hours ago from no
```

countunique(/host/key,(sec|#num)<:time shift>,<operator>,<pattern>)

The number of unique values within the defined evaluation period.
 Supported value types: *Float*, *Integer*, *String*, *Text*, *Log*.

Parameters:

- See common parameters;
- **operator** (must be double-quoted). Supported `operators`:
eq - equal (default for integer, float)
ne - not equal
gt - greater
ge - greater or equal
lt - less
le - less or equal
like (default for string, text, log) - matches if contains pattern (case-sensitive)
bitand - bitwise AND
regexp - case-sensitive match of the regular expression given in `pattern`
iregexp - case-insensitive match of the regular expression given in `pattern`
- **pattern** - the required pattern (string arguments must be double-quoted).

Comments:

- Float items match with the precision of 2.22e-16; if database is not upgraded the precision is 0.000001.
- *like* is not supported as operator for integer values;
- *like* and *bitand* are not supported as operators for float values;
- For string, text, and log values only *eq*, *ne*, *like*, *regexp* and *iregexp* operators are supported;
- With *bitand* as operator, the fourth `pattern` parameter can be specified as two numbers, separated by '/': **number_to_compare_with/mask**. countunique() calculates "bitwise AND" from the value and the *mask* and compares the result to *number_to_compare_with*. If the result of "bitwise AND" is equal to *number_to_compare_with*, the value is counted.
If *number_to_compare_with* and *mask* are equal, only the *mask* need be specified (without '/').
- With *regexp* or *iregexp* as operator, the fourth `pattern` parameter can be an ordinary or global (starting with '@') regular expression. In case of global regular expressions case sensitivity is inherited from global regular expression settings. For the purpose of regexp matching, float values will always be represented with 4 decimal digits after '.'. Also note that for large numbers difference in decimal (stored in database) and binary (used by Zabbix server) representation may affect the 4th decimal digit.

Examples:

```
countunique(/host/key,10m) #the number of unique values for the last 10 minutes until now
countunique(/host/key,10m,"like","error") #the number of unique values for the last 10 minutes until now t
countunique(/host/key,10m,,12) #the number of unique values for the last 10 minutes until now that equal '
countunique(/host/key,10m,"gt",12) #the number of unique values for the last 10 minutes until now that are
countunique(/host/key,#10,"gt",12) #the number of unique values within the last 10 values until now that a
countunique(/host/key,10m:now-1d,"gt",12) #the number of unique values between 24 hours and 10 minutes and
countunique(/host/key,10m,"bitand","6/7") #the number of unique values for the last 10 minutes until now h
countunique(/host/key,10m:now-1d) #the number of unique values between 24 hours and 10 minutes and 24 hour
```

find(/host/key,(sec|#num)<:time shift>,<operator>,<pattern>)

Find a value match within the defined evaluation period.
 Supported value types: *Float*, *Integer*, *String*, *Text*, *Log*.
 Returns: 1 - found; 0 - otherwise.

Parameters:

- See common parameters;
- **sec** or **#num** (optional) - defaults to the latest value if not specified
- **operator** (must be double-quoted). Supported `operators`:
eq - equal (default for integer, float)
ne - not equal
gt - greater
ge - greater or equal
lt - less
le - less or equal
like (default for string, text, log) - matches if contains the string given in `pattern` (case-sensitive)
bitand - bitwise AND
regexp - case-sensitive match of the regular expression given in `pattern`
iregexp - case-insensitive match of the regular expression given in `pattern`
- **pattern** - the required pattern (string arguments must be double-quoted); Perl Compatible Regular Expression (PCRE) regular expression if `operator` is *regexp*, *iregexp*.

Comments:

- If more than one value is processed, '1' is returned if there is at least one matching value;
- *like* is not supported as operator for integer values;
- *like* and *bitand* are not supported as operators for float values;
- For string, text, and log values only *eq*, *ne*, *like*, *regexp* and *iregexp* operators are supported;
- With *regexp* or *iregexp* as operator, the fourth `pattern` parameter can be an ordinary or global (starting with '@') regular expression. In case of global regular expressions case sensitivity is inherited from the global regular expression settings.

Example:

```
find(/host/key,10m,"like","error") #find a value that contains 'error' within the last 10 minutes until no
```

first(/host/key,sec<:time shift>)

The first (the oldest) value within the defined evaluation period.
 Supported value types: *Float*, *Integer*, *String*, *Text*, *Log*.

Parameters:

- See common parameters.

See also last().

Example:

```
first(/host/key,1h) #retrieve the oldest value within the last hour until now
```

fuzzytime(/host/key,sec)

Check how much the passive agent time differs from the Zabbix server/proxy time.
 Supported value types: *Float*, *Integer*.
 Returns: 1 - difference between the passive item value (as timestamp) and Zabbix server/proxy timestamp (the clock of value collection) is less than or equal to T seconds; 0 - otherwise.

Parameters:

- See common parameters.

Comments:

- Usually used with the 'system.localtime' item to check that local time is in sync with the local time of Zabbix server. *Note* that 'system.localtime' must be configured as a passive check.
- Can be used also with the `vfs.file.time[/path/file,modify]` key to check that the file did not get updates for long time;
- This function is not recommended for use in complex trigger expressions (with multiple items involved), because it may cause unexpected results (time difference will be measured with the most recent metric), e.g. in `fuzzytime(/Host/system.localtime,60 or last(/Host/trap)<>0`.

Example:

```
fuzzytime(/host/key,60s)=0 #detect a problem if the time difference is over 60 seconds<br><br>
```

last(/host/key,<#num<:time shift»)

The most recent value.
 Supported value types: *Float*, *Integer*, *String*, *Text*, *Log*.

Parameters:

- See common parameters;
- **#num** (optional) - the Nth most recent value.

Comments:

- Take note that a hash-tagged time period (#N) works differently here than with many other functions. For example: `last()` is always equal to `last(#1)`; `last(#3)` - the third most recent value (*not* three latest values);
- Zabbix does not guarantee the exact order of values if more than two values exist within one second in history;
- See also first().

Example:

```
last(/host/key) #retrieve the last value
last(/host/key,#2) #retrieve the previous value
last(/host/key,#1) <> last(/host/key,#2) #the last and previous values differ
```

logeventid(/host/key,<#num<:time shift»,<pattern>)

Check if the event ID of the last log entry matches a regular expression.
 Supported value types: *Log*.
 Returns: 0 - does not match; 1 - matches.

Parameters:

- See common parameters;
- **#num** (optional) - the Nth most recent value;
- **pattern** (optional) - the regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style (string arguments must be double-quoted).

logseverity(/host/key,<#num<:time shift»)

Log severity of the last log entry.
 Supported value types: *Log*.
 Returns: 0 - default severity; N - severity (integer, useful for Windows event logs: 1 - Information, 2 - Warning, 4 - Error, 7 - Failure Audit, 8 - Success Audit, 9 - Critical, 10 - Verbose).

Parameters:

- See common parameters;
- **#num** (optional) - the Nth most recent value.

Zabbix takes log severity from the **Information** field of Windows event log.

logsource(/host/key,<#num<:time shift»,<pattern>)

Check if log source of the last log entry matches a regular expression.
 Supported value types: *Log*.
 Returns: 0 - does not match; 1 - matches.

Parameters:

- See common parameters;
- **#num** (optional) - the Nth most recent value;
- **pattern** (optional) - the regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style (string arguments must be double-quoted).

Normally used for Windows event logs.

Example:

```
logsource(/host/key,,"VMware Server")
```

monodec(/host/key,(sec|#num)<:time shift>,<mode>)

Check if there has been a monotonous decrease in values.
 Supported value types: *Integer*.
 Returns: 1 - if all elements in the time period continuously decrease; 0 - otherwise.

Parameters:

- See common parameters;
- **mode** (must be double-quoted) - *weak* (every value is smaller or the same as the previous one; default) or *strict* (every value has decreased).

Example:

```
monodec(/Host1/system.swap.size[all,free],60s) + monodec(/Host2/system.swap.size[all,free],60s) + monodec(
```

monoinc(/host/key,(sec|#num)<:time shift>,<mode>)

Check if there has been a monotonous increase in values.
 Supported value types: *Integer*.
 Returns: 1 - if all elements in the time period continuously increase; 0 - otherwise.

Parameters:

- See common parameters;
- **mode** (must be double-quoted) - *weak* (every value is bigger or the same as the previous one; default) or *strict* (every value has increased).

Example:

```
monoinc(/Host1/system.localtime,#3,"strict")=0 #check if the system local time has been increasing consist
```

nodata(/host/key,sec,<mode>)

Check for no data received.
 Supported value types: *Integer*, *Float*, *Character*, *Text*, *Log*.
 Returns: 1 - if no data received during the defined period of time; 0 - otherwise.

Parameters:

- See common parameters;
- **sec** - the period should not be less than 30 seconds because the history syncer process calculates this function only every 30 seconds; `nodata(/host/key,0)` is disallowed.
- **mode** - if set to *strict* (double-quoted), this function will be insensitive to proxy availability (see comments for details).

Comments:

- the 'nodata' triggers monitored by proxy are, by default, sensitive to proxy availability - if proxy becomes unavailable, the 'nodata' triggers will not fire immediately after a restored connection, but will skip the data for the delayed period. Note that for passive proxies suppression is activated if connection is restored more than 15 seconds and no less than 2 seconds later.

For active proxies suppression is activated if connection is restored more than 15 seconds later. To turn off sensitiveness to proxy availability, use the third parameter, e.g.: `nodata(/host/key,5m,"strict")`; in this case the function will fire as soon as the evaluation period (five minutes) without data has past.
- This function will display an error if, within the period of the 1st parameter:
- there's no data and Zabbix server was restarted
- there's no data and maintenance was completed
- there's no data and the item was added or re-enabled
- Errors are displayed in the *Info* column in trigger configuration;
- This function may not work properly if there are time differences between Zabbix server, proxy and agent. See also: Time synchronization requirement.

percentile(/host/key,(sec|#num)<:time shift>,percentage)

The P-th percentile of a period, where P (percentage) is specified by the third parameter.
 Supported value types: *Float*, *Integer*.

Parameters:

- See common parameters;
- **percentage** - a floating-point number between 0 and 100 (inclusive) with up to 4 digits after the decimal point.

rate(/host/key,sec<:time shift>)

The per-second average rate of the increase in a monotonically increasing counter within the defined time period.
 Supported value types: *Float*, *Integer*.

Parameters:

- See common parameters.

Functionally corresponds to 'rate' of PromQL.

Example:

```
rate(/host/key,30s) #if the monotonic increase over 30 seconds is 20, this function will return 0.67.
```

See all supported functions.

5 Trend functions

Trend functions, in contrast to history functions, use trend data for calculations.

Trends store hourly aggregate values. Trend functions use these hourly averages, and thus are useful for long-term analysis.

Trend function results are cached so multiple calls to the same function with the same parameters fetch info from the database only once. The trend function cache is controlled by the TrendFunctionCacheSize server parameter.

Triggers that reference trend functions **only** are evaluated once per the smallest time period in the expression. For instance, a trigger like

```
trendavg(/host/key,1d:now/d) > 1 or trendavg(/host/key2,1w:now/w) > 2
```

will be evaluated once per day. If the trigger contains both trend and history (or time-based) functions, it is calculated in accordance with the usual principles.

All functions listed here are supported in:

- Trigger expressions
- Calculated items

The functions are listed without additional information. Click on the function to see the full details.

Function	Description
baselinedev	Returns the number of deviations (by stddevpop algorithm) between the last data period and the same data periods in preceding seasons.
baselinewma	Calculates the baseline by averaging data from the same timeframe in multiple equal time periods ('seasons') using the weighted moving average algorithm.
trendavg	The average of trend values within the defined time period.
trendcount	The number of successfully retrieved history values used to calculate the trend value within the defined time period.
trendmax	The maximum in trend values within the defined time period.
trendmin	The minimum in trend values within the defined time period.
trendstl	Returns the rate of anomalies during the detection period - a decimal value between 0 and 1 that is `((the number of anomaly values)/(total number of values))`.

Function	Description
trendsum	The sum of trend values within the defined time period.

Common parameters

- `/host/key` is a common mandatory first parameter
- `time period:time shift` is a common second parameter, where:
  - **time period** - the time period (minimum '1h'), defined as <N><time unit> where `N` - the number of time units, `time unit` - h (hour), d (day), w (week), M (month) or y (year).
  - **time shift** - the time period offset (see function examples)

Function details

Some general notes on function parameters:

- Function parameters are separated by a comma
- Optional function parameters (or parameter parts) are indicated by < >
- Function-specific parameters are described with each function
- `/host/key` and `time period:time shift` parameters must never be quoted

baselinedev(/host/key,data period:time shift,season unit,num seasons)

Returns the number of deviations (by stddevpop algorithm) between the last data period and the same data periods in preceding seasons.

Parameters:

- See common parameters;
- **data period** - the data gathering period within a season, defined as <N><time unit> where:
`N` - the number of time units
`time unit` - h (hour), d (day), w (week), M (month) or y (year), must be equal to or less than season
- **season unit** - the duration of one season (h, d, w, M, y), cannot be smaller than data period;
- **num seasons** - the number of seasons to evaluate.

Examples:

```
baselinedev(/host/key,1d:now/d,"M",6) #calculating the number of standard deviations (population) between
baselinedev(/host/key,1h:now/h,"d",10)  #calculating the number of standard deviations (population) betwee
```

baselinewma(/host/key,data period:time shift,season unit,num seasons)

Calculates the baseline by averaging data from the same timeframe in multiple equal time periods ('seasons') using the weighted moving average algorithm.

Parameters:

- See common parameters;
- **data period** - the data gathering period within a season, defined as <N><time unit> where:
`N` - the number of time units
`time unit` - h (hour), d (day), w (week), M (month) or y (year), must be equal to or less than season
Time shift - the time period offset, defines the end of data gathering time frame in seasons (see examples);
- **season unit** - the duration of one season (h, d, w, M, y), cannot be smaller than data period;
- **num seasons** - the number of seasons to evaluate.

Examples:

```
baselinewma(/host/key,1h:now/h,"d",3) #calculating the baseline based on the last full hour within a 3-day
baselinewma(/host/key,2h:now/h,"d",3) #calculating the baseline based on the last two hours within a 3-day
baselinewma(/host/key,1d:now/d,"M",4) #calculating the baseline based on the same day of month as 'yesterd
```

trendavg(/host/key,time period:time shift)

The average of trend values within the defined time period.

Parameters:

- See common parameters.

Examples:

```
trendavg(/host/key,1h:now/h) #the average for the previous hour (e.g. 12:00-13:00)
trendavg(/host/key,1h:now/h-1h) #the average for two hours ago (11:00-12:00)
trendavg(/host/key,1h:now/h-2h) #the average for three hours ago (10:00-11:00)
trendavg(/host/key,1M:now/M-1y) #the average for the previous month a year ago
```

trendcount(/host/key,time period:time shift)

The number of successfully retrieved history values used to calculate the trend value within the defined time period.

Parameters:

- See common parameters.

Examples:

```
trendcount(/host/key,1h:now/h) #the value count for the previous hour (e.g. 12:00-13:00)
trendcount(/host/key,1h:now/h-1h) #the value count for two hours ago (11:00-12:00)
trendcount(/host/key,1h:now/h-2h) #the value count for three hours ago (10:00-11:00)
trendcount(/host/key,1M:now/M-1y) #the value count for the previous month a year ago
```

trendmax(/host/key,time period:time shift)

The maximum in trend values within the defined time period.

Parameters:

- See common parameters.

Examples:

```
trendmax(/host/key,1h:now/h) #the maximum for the previous hour (e.g. 12:00-13:00)
trendmax(/host/key,1h:now/h) - trendmin(/host/key,1h:now/h) → calculate the difference between the maximum
trendmax(/host/key,1h:now/h-1h) #the maximum for two hours ago (11:00-12:00)
trendmax(/host/key,1h:now/h-2h) #the maximum for three hours ago (10:00-11:00)
trendmax(/host/key,1M:now/M-1y) #the maximum for the previous month a year ago
```

trendmin(/host/key,time period:time shift)

The minimum in trend values within the defined time period.

Parameters:

- See common parameters.

Examples:

```
trendmin(/host/key,1h:now/h) #the minimum for the previous hour (e.g. 12:00-13:00)
trendmax(/host/key,1h:now/h) - trendmin(/host/key,1h:now/h) → calculate the difference between the maximum
trendmin(/host/key,1h:now/h-1h) #the minimum for two hours ago (11:00-12:00)
trendmin(/host/key,1h:now/h-2h) #the minimum for three hours ago (10:00-11:00)
trendmin(/host/key,1M:now/M-1y) #the minimum for the previous month a year ago
```

trendstl(/host/key,eval period:time shift,detection period,season,<deviations>,<devalg>,<s window>)

Returns the rate of anomalies during the detection period - a decimal value between 0 and 1 that is `((the number of anomaly values)/(total number of values))`.

Parameters:

- See common parameters;
- **eval period** - the time period that must be decomposed (minimum '1h'), defined as <N><time unit> where
`N` - the number of time units
`time unit` - h (hour), d (day), w (week), M (month) or y (year)
- **detection period** - the time period before the end of eval period for which anomalies are calculated (minimum '1h', cannot be longer than eval period), defined as <N><time unit> where
`N` - the number of time units
`time unit` - h (hour), d (day), w (week)
- **season** - the shortest time period where a repeating pattern ("season") is expected (minimum '2h', cannot be longer than eval period, the number of entries in the eval period must be greater than the two times of the resulting frequency (season/h)), defined as <N><time unit> where
`N` - the number of time units
`time unit` - h (hour), d (day), w (week)
- **deviations** - the number of deviations (calculated by devalg) to count as anomaly (can be decimal), (must be greater than or equal to 1, default is 3);
- **devalg** (must be double-quoted) - the deviation algorithm, can be *stddevpop*, *stddevsamp* or *mad* (default);
- **s window** - the span (in lags) of the loess window for seasonal extraction (default is 10 * number of entries in eval period + 1)

Examples:

```
trendstl(/host/key,100h:now/h,10h,2h) #analyse the last 100 hours of trend data, find the anomaly rate for
trendstl(/host/key,100h:now/h-10h,100h,2h,2.1,"mad") #analyse the period of 100 hours of trend data, up to
trendstl(/host/key,100d:now/d-1d,10d,1d,4,,10) #analyse 100 days of trend data up to a day ago, find the a
```

```
trendstl(/host/key,1M:now/M-1y,1d,2h,,"stddevsamp") #analyse the previous month a year ago, find the anoma
```

trendsum(/host/key,time period:time shift)

The sum of trend values within the defined time period.

Parameters:

- See common parameters.

Examples:

```
trendsum(/host/key,1h:now/h) #the sum for the previous hour (e.g. 12:00-13:00)
trendsum(/host/key,1h:now/h-1h) #the sum for two hours ago (11:00-12:00)
trendsum(/host/key,1h:now/h-2h) #the sum for three hours ago (10:00-11:00)
trendsum(/host/key,1M:now/M-1y) #the sum for the previous month a year ago
```

See all supported functions.

6 Mathematical functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

Mathematical functions are supported with float and integer value types, unless stated otherwise.

The functions are listed without additional information. Click on the function to see the full details.

Function	Description
abs	The absolute value of a value.
acos	The arccosine of a value as an angle, expressed in radians.
asin	The arcsine of a value as an angle, expressed in radians.
atan	The arctangent of a value as an angle, expressed in radians.
atan2	The arctangent of the ordinate (value) and abscissa coordinates specified as an angle, expressed in radians.
avg	The average value of the referenced item values.
cbrt	The cube root of a value.
ceil	Round the value up to the nearest greater or equal integer.
cos	The cosine of a value, where the value is an angle expressed in radians.
cosh	The hyperbolic cosine of a value.
cot	The cotangent of a value, where the value is an angle expressed in radians.
degrees	Converts a value from radians to degrees.
e	The Euler's number (2.718281828459045).
exp	The Euler's number at a power of a value.
expm1	The Euler's number at a power of a value minus 1.
floor	Round the value down to the nearest smaller or equal integer.
log	The natural logarithm.
log10	The decimal logarithm.
max	The highest value of the referenced item values.
min	The lowest value of the referenced item values.
mod	The division remainder.
pi	The Pi constant (3.14159265358979).
power	The power of a value.
radians	Converts a value from degrees to radians.
rand	Return a random integer value.
round	Round the value to decimal places.
signum	Returns '-1' if a value is negative, '0' if a value is zero, '1' if a value is positive.
sin	The sine of a value, where the value is an angle expressed in radians.
sinh	The hyperbolical sine of a value, where the value is an angle expressed in radians.
sqrt	The square root of a value.
sum	The sum of the referenced item values.
tan	The tangent of a value.
truncate	Truncate the value to decimal places.

Function details

Some general notes on function parameters:

- Function parameters are separated by a comma
- Expressions are accepted as parameters
- Optional function parameters (or parameter parts) are indicated by < >

abs(value)

The absolute value (from 0) of a value.

Parameter:

- **value** - the value to check

For example, the absolute value of either ’3’ or ’-3’ will be ’3’.

Example:

```
abs(last(/host/key))>10
```

acos(value)

The arccosine of a value as an angle, expressed in radians.

Parameter:

- **value** - the value to check

The value must be between -1 and 1. For example, the arccosine of a value ’0.5’ will be ’2.0943951’.

Example:

```
acos(last(/host/key))
```

asin(value)

The arcsine of a value as an angle, expressed in radians.

Parameter:

- **value** - the value to check

The value must be between -1 and 1. For example, the arcsine of a value ’0.5’ will be ’-0.523598776’.

Example:

```
asin(last(/host/key))
```

atan(value)

The arctangent of a value as an angle, expressed in radians.

Parameter:

- **value** - the value to check

The value must be between -1 and 1. For example, the arctangent of a value ’1’ will be ’0.785398163’.

Example:

```
atan(last(/host/key))
```

atan2(value,abscissa)

The arctangent of the ordinate (value) and abscissa coordinates specified as an angle, expressed in radians.

Parameter:

- **value** - the value to check;
- **abscissa** - the abscissa value.

For example, the arctangent of the ordinate and abscissa coordinates of a value ’1’ will be ’2.21429744’.

Example:

```
atan(last(/host/key),2)
```

avg(<value1>,<value2>,...)

The average value of the referenced item values.

Parameter:

- **valueX** - the value returned by another function that is working with item history.

Example:

```
avg(avg(/host/key),avg(/host2/key2))
```

cbrt(value)

The cube root of a value.

Parameter:

- **value** - the value to check

For example, the cube root of '64' will be '4', of '63' will be '3.97905721'.

Example:

```
cbrt(last(/host/key))
```

ceil(value)

Round the value up to the nearest greater or equal integer.

Parameter:

- **value** - the value to check

For example, '2.4' will be rounded up to '3'. See also floor().

Example:

```
ceil(last(/host/key))
```

cos(value)

The cosine of a value, where the value is an angle expressed in radians.

Parameter:

- **value** - the value to check

For example, the cosine of a value '1' will be '0.54030230586'.

Example:

```
cos(last(/host/key))
```

cosh(value)

The hyperbolic cosine of a value. Returns the value as a real number, not as scientific notation.

Parameter:

- **value** - the value to check

For example, the hyperbolic cosine of a value '1' will be '1.54308063482'.

Example:

```
cosh(last(/host/key))
```

cot(value)

The cotangent of a value, where the value is an angle expressed in radians.

Parameter:

- **value** - the value to check

For example, the cotangent of a value '1' will be '0.54030230586'.

Example:

```
cot(last(/host/key))
```

degrees(value)

Converts a value from radians to degrees.

Parameter:

- **value** - the value to check

For example, a value ’1’ converted to degrees will be ’57.2957795’.

Example:

```
degrees(last(/host/key))
```

e

The Euler’s number (2.718281828459045).

Example:

```
e()
```

exp(value)

The Euler’s number at a power of a value.

Parameter:

- **value** - the value to check

For example, Euler’s number at a power of a value ’2’ will be ’7.38905609893065’.

Example:

```
exp(last(/host/key))
```

expm1(value)

The Euler’s number at a power of a value minus 1.

Parameter:

- **value** - the value to check

For example, Euler’s number at a power of a value ’2’ minus 1 will be ’6.38905609893065’.

Example:

```
expm1(last(/host/key))
```

floor(value)

Round the value down to the nearest smaller or equal integer.

Parameter:

- **value** - the value to check

For example, ’2.6’ will be rounded down to ’2’. See also ceil().

Example:

```
floor(last(/host/key))
```

log(value)

The natural logarithm.

Parameter:

- **value** - the value to check

For example, the natural logarithm of a value ’2’ will be ’0.69314718055994529’.

Example:

```
log(last(/host/key))
```

log10(value)

The decimal logarithm.

Parameter:

- **value** - the value to check

For example, the decimal logarithm of a value ’5’ will be ’0.69897000433’.

Example:

```
log10(last(/host/key))
```

max(<value1>,<value2>,...)

The highest value of the referenced item values.

Parameter:

- **valueX** - the value returned by another function that is working with item history.

Example:

```
max(avg(/host/key),avg(/host2/key2))
```

min(<value1>,<value2>,...)

The lowest value of the referenced item values.

Parameter:

- **valueX** - the value returned by another function that is working with item history.

Example:

```
min(avg(/host/key),avg(/host2/key2))
```

mod(value,denominator)

The division remainder.

Parameter:

- **value** - the value to check;
- **denominator** - the division denominator.

For example, division remainder of a value ’5’ with division denominator ’2’ will be ’1’.

Example:

```
mod(last(/host/key),2)
```

pi

The Pi constant (3.14159265358979).

Example:

```
pi()
```

power(value,power value)

The power of a value.

Parameter:

- **value** - the value to check;
- **power value** - the Nth power to use.

For example, the 3rd power of a value ’2’ will be ’8’.

Example:

```
power(last(/host/key),3)
```

radians(value)

Converts a value from degrees to radians.

Parameter:

- **value** - the value to check

For example, a value ’1’ converted to radians will be ’0.0174532925’.

Example:

```
radians(last(/host/key))
```

rand

Return a random integer value. A pseudo-random generated number using time as seed (enough for mathematical purposes, but not cryptography).

Example:

```
rand()
```

round(value,decimal places)

Round the value to decimal places.

Parameter:

- **value** - the value to check;
- **decimal places** - specify decimal places for rounding (0 is also possible).

For example, a value ’2.5482’ rounded to 2 decimal places will be ’2.55’.

Example:

```
round(last(/host/key),2)
```

signum(value)

Returns ’-1’ if a value is negative, ’0’ if a value is zero, ’1’ if a value is positive.

Parameter:

- **value** - the value to check.

Example:

```
signum(last(/host/key))
```

sin(value)

The sine of a value, where the value is an angle expressed in radians.

Parameter:

- **value** - the value to check

For example, the sine of a value ’1’ will be ’0.8414709848’.

Example:

```
sin(last(/host/key))
```

sinh(value)

The hyperbolical sine of a value, where the value is an angle expressed in radians.

Parameter:

- **value** - the value to check

For example, the hyperbolical sine of a value ’1’ will be ’1.17520119364’.

Example:

```
sinh(last(/host/key))
```

sqrt(value)

The square root of a value.
 This function will fail with a negative value.

Parameter:

- **value** - the value to check

For example, the square root of a value ’3.5’ will be ’1.87082869339’.

Example:

```
sqrt(last(/host/key))
```

sum(<value1>,<value2>,...)

The sum of the referenced item values.

Parameter:

- **valueX** - the value returned by another function that is working with item history.

Example:

```
sum(avg(/host/key),avg(/host2/key2))
```

tan(value)

The tangent of a value.

Parameter:

- **value** - the value to check

For example, the tangent of a value '1' will be '1.55740772465'.

Example:

```
tan(last(/host/key))
```

truncate(value,decimal places)

Truncate the value to decimal places.

Parameter:

- **value** - the value to check;
- **decimal places** - specify decimal places for truncating (0 is also possible).

For example, a value '2.5482' truncated to 2 decimal places will be '2.54'.

Example:

```
truncate(last(/host/key),2)
```

See all supported functions.

7 Operator functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

The functions are listed without additional information. Click on the function to see the full details.

Function	Description
between	Check if the value belongs to the given range.
in	Check if the value is equal to at least one of the listed values.

Function details

Some general notes on function parameters:

- Function parameters are separated by a comma
- Expressions are accepted as parameters

between(value,min,max)

Check if the value belongs to the given range.
 Supported value types: *Integer*, *Float*.
 Returns: 1 - in range; 0 - otherwise.

Parameter:

- **value** - the value to check;
- **min** - the minimum value;
- **max** - the maximum value.

Example:

```
between(last(/host/key),1,10)=1 #trigger if the value is between 1 and 10
```

in(value,value1,value2,...valueN)

Check if the value is equal to at least one of the listed values.
 Supported value types: *Integer*, *Float*, *Character*, *Text*, *Log*.
 Returns: 1 - if equal; 0 - otherwise.

Parameter:

- **value** - the value to check;
- **valueX** - listed values (string values must be double-quoted).

The value is compared to the listed values as numbers, if all of these values can be converted to numeric; otherwise compared as strings.

Example:

```
in(last(/host/key),5,10)=1 #trigger if the last value is equal to 5 or 10
in("text",last(/host/key),last(/host/key,#2))=1 #trigger if "text" is equal to either of the last 2 values
```

See all supported functions.

8 Prediction functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

The functions are listed without additional information. Click on the function to see the full details.

Function	Description
forecast	The future value, max, min, delta or avg of the item.
timeleft	The time in seconds needed for an item to reach the specified threshold.

Common parameters

- `/host/key` is a common mandatory first parameter for the functions referencing the host item history
- `(sec|#num)<:time shift>` is a common second parameter for the functions referencing the host item history, where:
  - **sec** - maximum evaluation period in seconds (time suffixes can be used), or
  - **#num** - maximum evaluation range in latest collected values (if preceded by a hash mark)
  - **time shift** (optional) allows to move the evaluation point back in time. See more details on specifying time shift.

Function details

Some general notes on function parameters:

- Function parameters are separated by a comma
- Optional function parameters (or parameter parts) are indicated by < >
- Function-specific parameters are described with each function
- `/host/key` and `(sec|#num)<:time shift>` parameters must never be quoted

forecast(/host/key,(sec|#num)<:time shift>,time,<fit>,<mode>)

The future value, max, min, delta or avg of the item.
 Supported value types: *Float*, *Integer*.

Parameters:

- See common parameters;
- **time** - the forecasting horizon in seconds (time suffixes can be used); negative values are supported;
- **fit** (optional; must be double-quoted) - the function used to fit historical data. Supported fits:
linear - linear function (default)
polynomialN - polynomial of degree N (1 <= N <= 6)
exponential - exponential function
logarithmic - logarithmic function
power - power function
Note that *polynomial1* is equivalent to *linear*;
- **mode** (optional; must be double-quoted) - the demanded output. Supported modes:
value - value (default)
max - maximum
min - minimum
delta - *max-min*
avg - average
Note that *value* estimates the item value at the moment `now` + `time`; *max*, *min*, *delta* and *avg* investigate the item value estimate on the interval between `now` and `now` + `time`.

Comments:

- If the value to return is larger than 1.7976931348623157E+308 or less than -1.7976931348623157E+308, the return value is cropped to 1.7976931348623157E+308 or -1.7976931348623157E+308 correspondingly;
- Becomes unsupported only if misused in the expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors;
- See also additional information on predictive trigger functions.

Examples:

```
forecast(/host/key,#10,1h) #forecast the item value in one hour based on the last 10 values
forecast(/host/key,1h,30m) #forecast the item value in 30 minutes based on the last hour data
forecast(/host/key,1h:now-1d,12h) #forecast the item value in 12 hours based on one hour one day ago
forecast(/host/key,1h,10m,"exponential") #forecast the item value in 10 minutes based on the last hour dat
forecast(/host/key,1h,2h,"polynomial3","max") #forecast the maximum value the item can reach in the next t
forecast(/host/key,#2,-20m) #estimate the item value 20 minutes ago based on the last two values (this can
```

timeleft(/host/key,(sec|#num)<:time shift>,threshold,<fit>)

The time in seconds needed for an item to reach the specified threshold.
 Supported value types: *Float*, *Integer*.

Parameters:

- See common parameters;
- **threshold** - the value to reach (unit suffixes can be used);
- **fit** (optional; must be double-quoted) - see forecast().

Comments:

- If the value to return is larger than 1.7976931348623157E+308, the return value is cropped to 1.7976931348623157E+308;
- Returns 1.7976931348623157E+308 if the threshold cannot be reached;
- Becomes unsupported only if misused in the expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors;
- See also additional information on predictive trigger functions.

Examples:

```
timeleft(/host/key,#10,0) #the time until the item value reaches zero based on the last 10 values
timeleft(/host/key,1h,100) #the time until the item value reaches 100 based on the last hour data
timeleft(/host/key,1h:now-1d,100) #the time until the item value reaches 100 based on one hour one day ago
timeleft(/host/key,1h,200,"polynomial2") #the time until the item value reaches 200 based on the last hour
```

See all supported functions.

9 String functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

The functions are listed without additional information. Click on the function to see the full details.

Function	Description
ascii	The ASCII code of the leftmost character of the value.
bitlength	The length of value in bits.
bytelength	The length of value in bytes.
char	Return the character by interpreting the value as ASCII code.
concat	The string resulting from concatenating the referenced item values or constant values.
insert	Insert specified characters or spaces into the character string beginning at the specified position in the string.
left	Return the leftmost characters of the value.
length	The length of value in characters.
ltrim	Remove specified characters from the beginning of string.
mid	Return a substring of N characters beginning at the character position specified by 'start'.
repeat	Repeat a string.
replace	Find the pattern in the value and replace with replacement.
right	Return the rightmost characters of the value.
rtrim	Remove specified characters from the end of string.
trim	Remove specified characters from the beginning and end of string.

Function details

Some general notes on function parameters:

- Function parameters are separated by a comma
- Expressions are accepted as parameters
- String parameters must be double-quoted; otherwise they might get misinterpreted

- Optional function parameters (or parameter parts) are indicated by < >

ascii(value)

The ASCII code of the leftmost character of the value.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check

For example, a value like 'Abc' will return '65' (ASCII code for 'A').

Example:

```
ascii(last(/host/key))
```

bitlength(value)

The length of value in bits.
 Supported value types: *String*, *Text*, *Log*, *Integer*.

Parameter:

- **value** - the value to check

Example:

```
bitlength(last(/host/key))
```

bytelength(value)

The length of value in bytes.
 Supported value types: *String*, *Text*, *Log*, *Integer*.

Parameter:

- **value** - the value to check

Example:

```
bytelength(last(/host/key))
```

char(value)

Return the character by interpreting the value as ASCII code.
 Supported value types: *Integer*.

Parameter:

- **value** - the value to check

The value must be in the 0-255 range. For example, a value like '65' (interpreted as ASCII code) will return 'A'.

Example:

```
char(last(/host/key))
```

concat(<value1>,<value2>,...)

The string resulting from concatenating the referenced item values or constant values.
 Supported value types: *String*, *Text*, *Log*, *Float*, *Integer*.

Parameter:

- **valueX** - the value returned by one of the history functions or a constant value (string, integer, or float number). Must contain at least two parameters.

For example, a value like 'Zab' concatenated to 'bix' (the constant string) will return 'Zabbix'.

Examples:

```
concat(last(/host/key),"bix")
concat("1 min: ",last(/host/system.cpu.load[all,avg1]),", 15 min: ",last(/host/system.cpu.load[all,avg15]))
```

insert(value,start,length,replacement)

Insert specified characters or spaces into the character string beginning at the specified position in the string.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check;
- **start** - start position;
- **length** - positions to replace;

- **replacement** - replacement string.

For example, a value like 'Zabbbix' will be replaced by 'Zabbix' if 'bb' (starting position 3, positions to replace 2) is replaced by 'b'.

Example:

```
insert(last(/host/key),3,2,"b")
```

left(value,count)

Return the leftmost characters of the value.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check;
- **count** - the number of characters to return.

For example, you may return 'Zab' from 'Zabbix' by specifying 3 leftmost characters to return. See also right().

Example:

```
left(last(/host/key),3) #return three leftmost characters
```

length(value)

The length of value in characters.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check.

Examples:

```
length(last(/host/key)) #the length of the latest value
length(last(/host/key,#3)) #the length of the third most recent value
length(last(/host/key,#1:now-1d)) #the length of the most recent value one day ago
```

ltrim(value,<chars>)

Remove specified characters from the beginning of string.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check;
- **chars** (optional) - specify the characters to remove.

Whitespace is left-trimmed by default (if no optional characters are specified). See also: rtrim(), trim().

Examples:

```
ltrim(last(/host/key)) #remove whitespace from the beginning of string
ltrim(last(/host/key),"Z") #remove any 'Z' from the beginning of string
ltrim(last(/host/key)," Z") #remove any space and 'Z' from the beginning of string
```

mid(value,start,length)

Return a substring of N characters beginning at the character position specified by 'start'.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check;
- **start** - start position of the substring;
- **length** - positions to return in substring.

For example, it is possible return 'abbi' from a value like 'Zabbix' if starting position is 2, and positions to return is 4.

Example:

```
mid(last(/host/key),2,4)="abbi"
```

repeat(value,count)

Repeat a string.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check;
- **count** - the number of times to repeat.

Example:

```
repeat(last(/host/key),2) #repeat the value two times
```

replace(value,pattern,replacement)

Find the pattern in the value and replace with replacement. All occurrences of the pattern will be replaced.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check;
- **pattern** - the pattern to find;
- **replacement** - the string to replace the pattern with.

Example:

```
replace(last(/host/key),"ibb","abb") - replace all 'ibb' with 'abb'
```

right(value,count)

Return the rightmost characters of the value.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check;
- **count** - the number of characters to return.

For example, you may return 'bix' from 'Zabbix' by specifying 3 rightmost characters to return. See also left().

Example:

```
right(last(/host/key),3) #return three rightmost characters
```

rtrim(value,<chars>)

Remove specified characters from the end of string.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check;
- **chars** (optional) - specify the characters to remove.

Whitespace is right-trimmed by default (if no optional characters are specified). See also: ltrim(), trim().

Examples:

```
rtrim(last(/host/key)) #remove whitespace from the end of string
rtrim(last(/host/key),"x") #remove any 'x' from the end of string
rtrim(last(/host/key),"x ") #remove any 'x' and space from the end of string
```

trim(value,<chars>)

Remove specified characters from the beginning and end of string.
 Supported value types: *String*, *Text*, *Log*.

Parameter:

- **value** - the value to check;
- **chars** (optional) - specify the characters to remove.

Whitespace is trimmed from both sides by default (if no optional characters are specified). See also: ltrim(), rtrim().

Examples:

```
trim(last(/host/key)) - remove whitespace from the beginning and end of string
trim(last(/host/key),"_") - remove '_' from the beginning and end of string
```

See all supported functions.

6 Macros

It is possible to use out-of-the-box Supported macros and User macros supported by location.

1 Supported macros

Overview

This page contains a complete list of built-in macros supported by Zabbix, grouped by application area.

Note:
To see all macros supported in a location (for example, in "map URL"), you may paste the location name into the search box at the bottom of your browser window (accessible by pressing CTRL+F) and do a search for *next*.

Actions

Macro	Supported in	Description
{ACTION.ID}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Numeric ID of the triggered action.*
{ACTION.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications	*Name of the triggered action.*
{ALERT.MESSAGE}	→ Alert script parameters → Webhook parameters	*'Default message' value from action configuration.*
{ALERT.SENDTO}	→ Alert script parameters → Webhook parameters	*'Send to' value from user media configuration.*
{ALERT.SUBJECT}	→ Alert script parameters → Webhook parameters	*'Default subject' value from action configuration.*
{ESC.HISTORY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Internal notifications	*Escalation history. Log of previously sent messages.* Shows previously sent notifications, on which escalation step they were sent and their status (*sent*, *in progress* or *failed*).

Date and time

Macro	Supported in	Description
{DATE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Manual event action scripts	*Current date in yyyy.mm.dd. format.*
{TIME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Trigger event names → Manual event action scripts	*Current time in hh:mm:ss.*

Discovery

Macro	Supported in	Description
{DISCOVERY.DEVICE.IPADDRESS}	→ Discovery notifications and commands	*IP address of the discovered device.* Available always, does not depend on host being added.
{DISCOVERY.DEVICE.DNS}	→ Discovery notifications and commands	*DNS name of the discovered device.* Available always, does not depend on host being added.
{DISCOVERY.DEVICE.STATUS}	→ Discovery notifications and commands	*Status of the discovered device*: can be either UP or DOWN.
{DISCOVERY.DEVICE.UPTIME}	→ Discovery notifications and commands	*Time since the last change of discovery status for a particular device*, with precision down to a second. For example: 1h 29m 01s. For devices with status DOWN, this is the period of their downtime.
{DISCOVERY.RULE.NAME}	→ Discovery notifications and commands	*Name of the discovery rule that discovered the presence or absence of the device or service.*
{DISCOVERY.SERVICE.NAME}	→ Discovery notifications and commands	*Name of the service that was discovered.* For example: HTTP.
{DISCOVERY.SERVICE.PORT}	→ Discovery notifications and commands	*Port of the service that was discovered.* For example: 80.
{DISCOVERY.SERVICE.STATUS}	→ Discovery notifications and commands	*Status of the discovered service*: can be either UP or DOWN.
{DISCOVERY.SERVICE.UPTIME}	→ Discovery notifications and commands	*Time since the last change of discovery status for a particular service*, with precision down to a second. For example: 1h 29m 01s. For services with status DOWN, this is the period of their downtime.

Events

Macro	Supported in	Description
{EVENT.ACK.STATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Acknowledgment status of the event (Yes/No)*.
{EVENT.AGE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Manual event action scripts	*Age of the event that triggered an action*, with precision down to a second. Useful in escalated messages.
{EVENT.DATE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Manual event action scripts	*Date of the event that triggered an action.*
{EVENT.DURATION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Internal notifications → Manual event action scripts	*Duration of the event (time difference between problem and recovery events)*, with precision down to a second. Useful in problem recovery messages. Supported since 5.0.0.

Macro	Supported in	Description
{EVENT.ID}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Trigger URLs → Manual event action scripts	*Numeric ID of the event that triggered an action.*
{EVENT.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Internal notifications → Manual event action scripts	*Name of the problem event that triggered an action.* Supported since 4.0.0.
{EVENT.NSEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Manual event action scripts	*Numeric value of the event severity.* Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster. Supported since 4.0.0.
{EVENT.OBJECT}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Manual event action scripts	*Numeric value of the event object.* Possible values: 0 - Trigger, 1 - Discovered host, 2 - Discovered service, 3 - Autoregistration, 4 - Item, 5 - Low-level discovery rule. Supported since 4.4.0.
{EVENT.OPDATA}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Operational data of the underlying trigger of a problem.* Supported since 4.4.0.
{EVENT.RECOVERY.DATE}	→ Problem recovery notifications and commands → Problem update notifications and commands (if recovery took place) → Service recovery notifications and commands → Manual event action scripts (if recovery took place)	*Date of the recovery event.*
{EVENT.RECOVERY.ID}	→ Problem recovery notifications and commands → Problem update notifications and commands (if recovery took place) → Service recovery notifications and commands → Manual event action scripts (if recovery took place)	*Numeric ID of the recovery event.*
{EVENT.RECOVERY.NAME}	→ Problem recovery notifications and commands → Problem update notifications and commands (if recovery took place) → Service recovery notifications and commands → Manual event action scripts (if recovery took place)	*Name of the recovery event.* Supported since 4.4.1.
{EVENT.RECOVERY.STATUS}	→ Problem recovery notifications and commands → Problem update notifications and commands (if recovery took place) → Service recovery notifications and commands → Manual event action scripts (if recovery took place)	*Verbal value of the recovery event.*
{EVENT.RECOVERY.TAGS}	→ Problem recovery notifications and commands → Problem update notifications and commands (if recovery took place) → Service recovery notifications and commands → Manual event action scripts (if recovery took place)	*A comma separated list of recovery event tags.* Expanded to an empty string if no tags exist. Supported since 3.2.0.

Macro	Supported in	Description
{EVENT.RECOVERY.TAGSJSON}	→ Problem recovery notifications and commands → Problem update notifications and commands (if recovery took place) → Service recovery notifications and commands → Manual event action scripts (if recovery took place)	*A JSON array containing event tag objects.* Expanded to an empty array if no tags exist. Supported since 5.0.0.
{EVENT.RECOVERY.TIME}	→ Problem recovery notifications and commands → Problem update notifications and commands (if recovery took place) → Service recovery notifications and commands → Manual event action scripts (if recovery took place)	*Time of the recovery event.*
{EVENT.RECOVERY.VALUE}	→ Problem recovery notifications and commands → Problem update notifications and commands (if recovery took place) → Service recovery notifications and commands → Manual event action scripts (if recovery took place)	*Numeric value of the recovery event.*
{EVENT.SEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Manual event action scripts	*Name of the event severity.* Supported since 4.0.0.
{EVENT.SOURCE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Manual event action scripts	*Numeric value of the event source.* Possible values: 0 - Trigger, 1 - Discovery, 2 - Autoregistration, 3 - Internal, 4 - Service. Supported since 4.4.0.
{EVENT.STATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Internal notifications → Manual event action scripts	*Verbal value of the event that triggered an action.*
{EVENT.TAGS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Manual event action scripts	*A comma separated list of event tags.* Expanded to an empty string if no tags exist. Supported since 3.2.0.
{EVENT.TAGSJSON}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Manual event action scripts	*A JSON array containing event tag objects.* Expanded to an empty array if no tags exist. Supported since 5.0.0.
{EVENT.TAGS.<tag name>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Webhook media type URL names and URLs → Manual event action scripts	*Event tag value referenced by the tag name.* A tag name containing non-alphanumeric characters (including non-English multibyte-UTF characters) should be double quoted. Quotes and backslashes inside a quoted tag name must be escaped with a backslash. Supported since 4.4.2.

Macro	Supported in	Description
{EVENT.TIME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Manual event action scripts	*Time of the event that triggered an action.*
{EVENT.UPDATE.ACTION}	→ Problem update notifications and commands	*Human-readable name of the action(s)* performed during problem update. Resolves to the following values: *acknowledged*, *commented*, *changed severity from (original severity) to (updated severity)* and *closed* (depending on how many actions are performed in one update). Supported since 4.0.0.
{EVENT.UPDATE.DATE}	→ Problem update notifications and commands → Service update notifications and commands	*Date of event update (acknowledgment, etc).* Deprecated name: {ACK.DATE}
{EVENT.UPDATE.HISTORY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Log of problem updates (acknowledgments, etc).* Deprecated name: {EVENT.ACK.HISTORY}
{EVENT.UPDATE.MESSAGE}	→ Problem update notifications and commands	*Problem update message.* Deprecated name: {ACK.MESSAGE}
{EVENT.UPDATE.NSEVERITY}	→ Service update notifications and commands	*Numeric value of the new event severity set during problem update operation.*
{EVENT.UPDATE.SEVERITY}	→ Service update notifications and commands	*Name of the new event severity set during problem update operation.*
{EVENT.UPDATE.STATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Numeric value of the problem update status.* Possible values: 0 - Webhook was called because of problem/recovery event, 1 - Update operation. Supported since 4.4.0.
{EVENT.UPDATE.TIME}	→ Problem update notifications and commands → Service update notifications and commands	*Time of event update (acknowledgment, etc).* Deprecated name: {ACK.TIME}
{EVENT.VALUE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Service-based notifications and commands → Service update notifications and commands → Service recovery notifications and commands → Internal notifications → Manual event action scripts	*Numeric value of the event that triggered an action (1 for problem, 0 for recovering).*

Cause and symptom events

{EVENT.CAUSE.*} macros are used in the context of a symptom event, for example, in notifications; they return information about the cause event.

The {EVENT.SYMPTOMS} macro is used in the context of the cause event and returns information about symptom events.

Macro	Supported in	Description
{EVENT.CAUSE.ACKSTATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Acknowledgment status of the cause event (Yes/No).*
{EVENT.CAUSE.AGE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Age of the cause event*, with precision down to a second. Useful in escalated messages.
{EVENT.CAUSE.DATE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Date of the cause event.*

Macro	Supported in	Description
{EVENT.CAUSE.DURATION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Duration of the cause event (time difference between problem and recovery events)*, with precision down to a second. Useful in problem recovery messages.
{EVENT.CAUSE.ID}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Numeric ID of the cause event .*
{EVENT.CAUSE.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Name of the cause problem event.*
{EVENT.CAUSE.NSEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Numeric value of the cause event severity.* Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster.
{EVENT.CAUSE.OBJECT}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Numeric value of the cause event object.* Possible values: 0 - Trigger, 1 - Discovered host, 2 - Discovered service, 3 - Autoregistration, 4 - Item, 5 - Low-level discovery rule.
{EVENT.CAUSE.OPDATA}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Operational data of the underlying trigger of the cause problem.*
{EVENT.CAUSE.SEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Name of the cause event severity.*
{EVENT.CAUSE.SOURCE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Numeric value of the cause event source.* Possible values: 0 - Trigger, 1 - Discovery, 2 - Autoregistration, 3 - Internal.
{EVENT.CAUSE.STATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Verbal value of the cause event.*
{EVENT.CAUSE.TAGS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*A comma separated list of cause event tags.* Expanded to an empty string if no tags exist.
{EVENT.CAUSE.TAGSJSON}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*A JSON array containing cause event tag objects.* Expanded to an empty array if no tags exist.
{EVENT.CAUSE.TAG<tag name>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Cause event tag value referenced by the tag name.* A tag name containing non-alphanumeric characters (including non-English multibyte-UTF characters) should be double quoted. Quotes and backslashes inside a quoted tag name must be escaped with a backslash.
{EVENT.CAUSE.TIME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Time of the cause event.*
{EVENT.CAUSE.UPDATE.HISTORY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Log of cause problem updates (acknowledgments, etc).*
{EVENT.CAUSE.VALUE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Numeric value of the cause event (1 for problem, 0 for recovering).*
{EVENT.SYMPTOMS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*The list of symptom events.* Includes the following details: host name, event name, severity, age, service tags and values.

Functions

Macro	Supported in	Description
{FUNCTION.VALUE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts → Event names	*Results of the Nth item-based function in the trigger expression at the time of the event.* Only functions with */host/key* as the first parameter are counted. See indexed macros.
{FUNCTION.RECOVERY.VALUE<1-9>}	→ Event recovery notifications and commands → Problem update notifications and commands → Manual event action scripts	*Results of the Nth item-based function in the recovery expression at the time of the event.* Only functions with */host/key* as the first parameter are counted. See indexed macros.

Hosts

Macro	Supported in	Description
{HOST.CONN}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Map element labels, map URL names and values → Item key parameters[1] → Host interface IP/DNS → Trapper item "Allowed hosts" field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring[4] → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget → Trigger names, event names, operational data and descriptions → Trigger URLs → Tag names and values → Script-type item, item prototype and discovery rule parameter names and values → HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, SSL certificate file, SSL key file, Allowed hosts. → Manual host action scripts (including confirmation text) → Manual event action scripts (including confirmation text) → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Host IP address or DNS name, depending on host settings*[2]. May be used with a numeric index as {HOST.CONN<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{HOST.DESCRIPTION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Map element labels → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Host description.* This macro may be used with a numeric index e.g. {HOST.DESCRIPTION<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{HOST.DNS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Map element labels, map URL names and values → Item key parameters[1] → Host interface IP/DNS → Trapper item ”Allowed hosts” field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring[4] → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget → Trigger names, event names, operational data and descriptions → Trigger URLs → Tag names and values → Script-type item, item prototype and discovery rule parameter names and values → HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, SSL certificate file, SSL key file, Allowed hosts. → Manual host action scripts (including confirmation text) → Manual event action scripts (including confirmation text) → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Host DNS name*[2]. This macro may be used with a numeric index e.g. {HOST.DNS<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{HOST.HOST}	→ Trigger-based notifications and commands → Problem update notifications and commands → Autoregistration notifications and commands → Internal notifications → Item key parameters → Map element labels, map URL names and values → Host interface IP/DNS → Trapper item ”Allowed hosts” field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring[4] → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget → Trigger names, event names, operational data and descriptions → Trigger URLs → Tag names and values → Script-type item, item prototype and discovery rule parameter names and values → HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, SSL certificate file, SSL key file, Allowed hosts. → Manual host action scripts (including confirmation text) → Manual event action scripts (including confirmation text) → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Host name.* This macro may be used with a numeric index e.g. {HOST.HOST<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. `{HOSTNAME<1-9>}` is deprecated.

Macro	Supported in	Description
{HOST.ID}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Map element labels, map URL names and values → URL field of dynamic URL dashboard widget → Trigger URLs → Tag names and values → Manual event action scripts → Manual host action scripts (only for type URL, including confirmation text) → Manual event action scripts (only for type URL, including confirmation text) → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Host ID.* May be used with a numeric index as {HOST.ID<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{HOST.IP}	→ Trigger-based notifications and commands → Problem update notifications and commands → Autoregistration notifications and commands → Internal notifications → Map element labels, map URL names and values → Item key parameters[1] → Host interface IP/DNS → Trapper item "Allowed hosts" field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring[4] → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget → Trigger names, event names, operational data and descriptions → Trigger URLs → Tag names and values → Script-type item, item prototype and discovery rule parameter names and values → HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, SSL certificate file, SSL key file, Allowed hosts. → Manual host action scripts (including confirmation text) → Manual event action scripts (including confirmation text) → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Host IP address*[2]. This macro may be used with a numeric index e.g. {HOST.IP<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. `{IPADDRESS<1-9>}` is deprecated.
{HOST.METADATA}	→ Autoregistration notifications and commands	*Host metadata.* Used only for active agent autoregistration.

Macro	Supported in	Description
{HOST.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Map element labels, map URL names and values → Item key parameters → Host interface IP/DNS → Trapper item "Allowed hosts" field → Database monitoring additional parameters → SSH and Telnet scripts → Web monitoring[4] → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget → Trigger names, event names, operational data and descriptions → Trigger URLs → Tag names and values → Script-type item, item prototype and discovery rule parameter names and values → HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, SSL certificate file, SSL key file, Allowed hosts. → Manual host action scripts (including confirmation text) → Manual event action scripts (including confirmation text) → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Visible host name.* This macro may be used with a numeric index e.g. {HOST.NAME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{HOST.PORT}	→ Trigger-based notifications and commands → Problem update notifications and commands → Autoregistration notifications and commands → Internal notifications → Trigger names, event names, operational data and descriptions → Trigger URLs → JMX item endpoint field → Tag names and values → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Host (agent) port*[2]. This macro may be used with a numeric index e.g. {HOST.PORT<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{HOST.TARGET.CONN}	→ Trigger-based commands → Problem update commands → Discovery commands → Autoregistration commands	*IP address or DNS name of the target host*, depending on host settings. Supported since 5.4.0.
{HOST.TARGET.DNS}	→ Trigger-based commands → Problem update commands → Discovery commands → Autoregistration commands	*DNS name of the target host.* Supported since 5.4.0.
{HOST.TARGET.HOST}	→ Trigger-based commands → Problem update commands → Discovery commands → Autoregistration commands	*Technical name of the target host.* Supported since 5.4.0.
{HOST.TARGET.IP}	→ Trigger-based commands → Problem update commands → Discovery commands → Autoregistration commands	*IP address of the target host.* Supported since 5.4.0.
{HOST.TARGET.NAME}	→ Trigger-based commands → Problem update commands → Discovery commands → Autoregistration commands	*Visible name of the target host.* Supported since 5.4.0.

See also: Host inventory

Host groups

Macro	Supported in	Description
{HOSTGROUP.ID}	→ Map element labels, map URL names and values	*Host group ID.*

Host inventory

Macro	Supported in	Description
{INVENTORY.ALIAS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Alias field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.ALIAS<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.ASSET.TAG}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Asset tag field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.ASSET.TAG<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.CHASSIS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Chassis field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.CHASSIS<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.CONTACT}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Contact field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.CONTACT<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. `{PROFILE.CONTACT<1-9>}` is deprecated.
{INVENTORY.CONTRACT.NUMBER}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Contract number field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.CONTRACT.NUMBER<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{INVENTORY.DEPLOYMENT.STATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Deployment status field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.DEPLOYMENT.STATUS<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.HARDWARE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Hardware field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.HARDWARE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. `{PROFILE.HARDWARE<1-9>}` is deprecated.
{INVENTORY.HARDWARE.FULL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Hardware (Full details) field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.HARDWARE.FULL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.HOST.NETMASK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Host subnet mask field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.HOST.NETMASK<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.HOST.NETWORKS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Host networks field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.HOST.NETWORKS<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.HOST.ROUTER}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Host router field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.HOST.ROUTER<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{INVENTORY.HW.ARCH}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Hardware architecture field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.HW.ARCH<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.HW.DATE.DECOMM}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Date hardware decommissioned field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.HW.DATE.DECOMM<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.HW.DATE.EXPIRY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Date hardware maintenance expires field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.HW.DATE.EXPIRY<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.HW.DATE.INSTALL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Date hardware installed field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.HW.DATE.INSTALL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.HW.DATE.PURCHASE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Date hardware purchased field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.HW.DATE.PURCHASE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.INSTALLER.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Installer name field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.INSTALLER.NAME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{INVENTORY.LOCATION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Location field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.LOCATION<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. {PROFILE.LOCATION<1-9>} is deprecated.
{INVENTORY.LOCATION.LAT}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Location latitude field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.LOCATION.LAT<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.LOCATION.LON}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Location longitude field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.LOCATION.LON<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.MACADDRESS.A}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*MAC address A field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.MACADDRESS.A<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. {PROFILE.MACADDRESS<1-9>} is deprecated.
{INVENTORY.MACADDRESS.B}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*MAC address B field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.MACADDRESS.B<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.MODEL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Model field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.MODEL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{INVENTORY.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Name field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.NAME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. {PROFILE.NAME<1-9>} is deprecated.
{INVENTORY.NOTES}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Notes field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.NOTES<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. {PROFILE.NOTES<1-9>} is deprecated.
{INVENTORY.OOB.IP}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*OOB IP address field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.OOB.IP<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.OOB.NETMASK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*OOB subnet mask field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.OOB.NETMASK<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.OOB.ROUTER}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*OOB router field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.OOB.ROUTER<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.OS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*OS field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.OS<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. {PROFILE.OS<1-9>} is deprecated.

Macro	Supported in	Description
{INVENTORY.OS.FULL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*OS (Full details) field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.OS.FULL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.OS.SHORT}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*OS (Short) field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.OS.SHORT<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.PRIMARY.CELL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Primary POC cell field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.PRIMARY.CELL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.PRIMARY.EMAIL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Primary POC email field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.PRIMARY.EMAIL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.PRIMARY.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Primary POC name field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.PRIMARY.NAME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.PRIMARY.NOTES}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Primary POC notes field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.PRIMARY.NOTES<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{INVENTORY.POC.PRIMARY.PHONE.A}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Primary POC phone A field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.PRIMARY.PHONE.A<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.PRIMARY.PHONE.B}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Primary POC phone B field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.PRIMARY.PHONE.B<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.PRIMARY.SCREEN}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Primary POC screen name field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.PRIMARY.SCREEN<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.SECONDARY.CELL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Secondary POC cell field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.SECONDARY.CELL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.SECONDARY.EMAIL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Secondary POC email field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.SECONDARY.EMAIL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.SECONDARY.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Secondary POC name field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.SECONDARY.NAME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{INVENTORY.POC.SECONDARY.NOTES}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Secondary POC notes field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.SECONDARY.NOTES<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.SECONDARY.PHONE.A}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Secondary POC phone A field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.SECONDARY.PHONE.A<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.SECONDARY.PHONE.B}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Secondary POC phone B field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.SECONDARY.PHONE.B<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.POC.SECONDARY.SCREEN}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Secondary POC screen name field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.POC.SECONDARY.SCREEN<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SERIALNO.A}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Serial number A field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SERIALNO.A<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. `{PROFILE.SERIALNO<1-9>}` is deprecated.
{INVENTORY.SERIALNO.B}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Serial number B field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SERIALNO.B<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{INVENTORY.SITE.ADDRESS.A}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Site address A field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SITE.ADDRESS.A<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SITE.ADDRESS.B}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Site address B field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SITE.ADDRESS.B<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SITE.ADDRESS.C}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Site address C field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SITE.ADDRESS.C<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SITE.CITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Site city field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SITE.CITY<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SITE.COUNTRY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Site country field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SITE.COUNTRY<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SITE.NOTES}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Site notes field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SITE.NOTES<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{INVENTORY.SITE.RACK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Site rack location field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SITE.RACK<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SITE.STATE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Site state/province field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SITE.STATE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SITE.ZIP}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Site ZIP/postal field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SITE.ZIP<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SOFTWARE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Software field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SOFTWARE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. `{PROFILE.SOFTWARE<1-9>}` is deprecated.
{INVENTORY.SOFTWARE.APP.A}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Software application A field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SOFTWARE.APP.A<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SOFTWARE.APP.B}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Software application B field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SOFTWARE.APP.B<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{INVENTORY.SOFTWARE.APP.C}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Software application C field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SOFTWARE.APP.C<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SOFTWARE.APP.D}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Software application D field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SOFTWARE.APP.D<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SOFTWARE.APP.E}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Software application E field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SOFTWARE.APP.E<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.SOFTWARE.FULL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Software (Full details) field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.SOFTWARE.FULL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.TAG}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Tag field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.TAG<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. {PROFILE.TAG<1-9>} is deprecated.
{INVENTORY.TYPE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Type field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.TYPE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. {PROFILE.DEVICETYPE<1-9>} is deprecated.

Macro	Supported in	Description
{INVENTORY.TYPE.FULL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Type (Full details) field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.TYPE.FULL<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.URL.A}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*URL A field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.URL.A<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.URL.B}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*URL B field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.URL.B<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.URL.C}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*URL C field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.URL.C<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{INVENTORY.VENDOR}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Tag names and values → Map element labels, map URL names and values → Script-type items[6] → Manual host action scripts[6] → Manual event action scripts → Description of item value widget → Column of data type *Text* in *Top hosts* widget	*Vendor field in host inventory.* This macro may be used with a numeric index e.g. {INVENTORY.VENDOR<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Items

Macro	Supported in	Description
{ITEM.DESCRIPTION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Manual event action scripts → Description of item value widget	*Description of the Nth item in the trigger expression that caused a notification.* This macro may be used with a numeric index e.g. {ITEM.DESCRIPTION<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{ITEM.DESCRIPTION.ORIG}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Manual event action scripts → Description of item value widget	*Description (with macros unresolved) of the Nth item in the trigger expression that caused a notification.* This macro may be used with a numeric index e.g. {ITEM.DESCRIPTION.ORIG<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. Supported since 5.2.0.
{ITEM.ID}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Script-type item, item prototype and discovery rule parameter names and values → HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, Proxy, SSL certificate file, SSL key file → Manual event action scripts → Description of item value widget	*Numeric ID of the Nth item in the trigger expression that caused a notification.* This macro may be used with a numeric index e.g. {ITEM.ID<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{ITEM.KEY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Script-type item, item prototype and discovery rule parameter names and values → HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, Proxy, SSL certificate file, SSL key file → Manual event action scripts → Description of item value widget	*Key of the Nth item in the trigger expression that caused a notification.* This macro may be used with a numeric index e.g. {ITEM.KEY<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. `{TRIGGER.KEY}` is deprecated.
{ITEM.KEY.ORIG}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Script-type item, item prototype and discovery rule parameter names and values → HTTP agent type item, item prototype and discovery rule fields: URL, Query fields, Request body, Headers, Proxy, SSL certificate file, SSL key file → Manual event action scripts → Description of item value widget	*Original key (with macros not expanded) of the Nth item in the trigger expression that caused a notification* [4]. This macro may be used with a numeric index e.g. {ITEM.KEY.ORIG<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{ITEM.LASTVALUE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, event names, operational data and descriptions → Tag names and values → Trigger URLs → Manual event action scripts → Description of item value widget	*The latest value of the Nth item in the trigger expression that caused a notification.* It will resolve to *UNKNOWN* in the frontend if the latest history value has been collected more than the *Max history display period* time ago (set in the Administration→General menu section). When used in the problem name, the macro will not resolve to the latest item value when viewing problem events; instead, it will keep the item value from the time when the problem happened. When used in notifications, in some cases the macro might not resolve to the latest item value at the moment the trigger fired. For example, if an item quickly receives two values, ”A” and ”B”, and the trigger fires for ”A”, notifications might show ”B” as the latest value due to a slight processing delay - the latest item value changed between the time the trigger fired and when the notification was created. To avoid this, you can use the {ITEM.VALUE} macro, which resolves to the value at the moment the trigger fires, ensuring the correct value is used in the notification. It is alias to `last(/{HOST.HOST}/{ITEM.KEY})`. The resolved value for text/log items is truncated to 20 characters by the frontend in the following locations: - Operational data; - Trigger description; - Trigger URLs; - Trigger URL labels; - Description of the item value widget. To resolve to a full value, you may use macro functions. No values are truncated by the server. Customizing the macro value is supported for this macro; starting with Zabbix 3.2.0. This macro may be used with a numeric index e.g. {ITEM.LASTVALUE<1-9>} to point to the first, second, third, etc. item in a trigger expression. See indexed macros.
{ITEM.LOG.AGE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, operational data and descriptions → Trigger URLs → Event tags and values → Manual event action scripts → Description of item value widget	*Age of the log item event*, with precision down to a second. This macro may be used with a numeric index e.g. {ITEM.LOG.AGE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{ITEM.LOG.DATE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, operational data and descriptions → Trigger URLs → Event tags and values → Manual event action scripts → Description of item value widget	*Date of the log item event.* This macro may be used with a numeric index e.g. {ITEM.LOG.DATE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{ITEM.LOG.EVENTID}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, operational data and descriptions → Trigger URLs → Event tags and values → Manual event action scripts → Description of item value widget	*ID of the event in the event log.* For Windows event log monitoring only. This macro may be used with a numeric index e.g. {ITEM.LOG.EVENTID<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{ITEM.LOG.NSEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, operational data and descriptions → Trigger URLs → Event tags and values → Manual event action scripts → Description of item value widget	*Numeric severity of the event in the event log.* For Windows event log monitoring only. This macro may be used with a numeric index e.g. {ITEM.LOG.NSEVERITY<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{ITEM.LOG.SEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, operational data and descriptions → Trigger URLs → Event tags and values → Manual event action scripts → Description of item value widget	*Verbal severity of the event in the event log.* For Windows event log monitoring only. This macro may be used with a numeric index e.g. {ITEM.LOG.SEVERITY<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{ITEM.LOG.SOURCE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, operational data and descriptions → Trigger URLs → Event tags and values → Manual event action scripts → Description of item value widget	*Source of the event in the event log.* For Windows event log monitoring only. This macro may be used with a numeric index e.g. {ITEM.LOG.SOURCE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{ITEM.LOG.TIME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, operational data and descriptions → Trigger URLs → Event tags and values → Manual event action scripts → Description of item value widget	*Time of the log item event.* This macro may be used with a numeric index e.g. {ITEM.LOG.TIME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{ITEM.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Manual event action scripts → Description of item value widget	*Name of the Nth item in the trigger expression that caused a notification.* This macro may be used with a numeric index e.g. {ITEM.NAME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{ITEM.NAME.ORIG}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Manual event action scripts → Description of item value widget	This macros is deprecated since Zabbix 6.0. It used to resolve to the *original name (i.e. without macros resolved) of the item* in pre-6.0 Zabbix versions when user macros and positional macros were supported in the item name. This macro may be used with a numeric index e.g. {ITEM.NAME.ORIG<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.
{ITEM.STATE}	→ Item-based internal notifications → Description of item value widget	*The latest state of the Nth item in the trigger expression that caused a notification.* Possible values: **Not supported** and **Normal**. This macro may be used with a numeric index e.g. {ITEM.STATE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{ITEM.STATE.ERROR}	→ Item-based internal notifications	*Error message with details why an item became unsupported.* If an item goes into the unsupported state and then immediately gets supported again the error field can be empty.
{ITEM.VALUE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger names, event names, operational data and descriptions → Tag names and values → Trigger URLs → Manual event action scripts → Description of item value widget	Resolved to either: 1) the historical (at-the-time-of-event) value of the Nth item in the trigger expression, if used in the context of trigger status change, for example, when displaying events or sending notifications. 2) the latest value of the Nth item in the trigger expression, if used without the context of trigger status change, for example, when displaying a list of triggers in a pop-up selection window. In this case works the same as {ITEM.LASTVALUE} In the first case it will resolve to *UNKNOWN* if the history value has already been deleted or has never been stored. In the second case, and in the frontend only, it will resolve to *UNKNOWN* if the latest history value has been collected more than the *Max history display period* time ago (set in the Administration→General menu section). The resolved value for text/log items is truncated to 20 characters by the frontend in the following locations: - Operational data; - Trigger description; - Trigger URLs; - Trigger URL labels; - Description of the item value widget. To resolve to a full value, you may use macro functions. No values are truncated by the server. Customizing the macro value is supported for this macro, starting with Zabbix 3.2.0. This macro may be used with a numeric index e.g. {ITEM.VALUE<1-9>} to point to the first, second, third, etc. item in a trigger expression. See indexed macros.
{ITEM.VALUETYPE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Internal notifications → Manual event action scripts → Description of item value widget	*Value type of the Nth item in the trigger expression that caused a notification.* Possible values: 0 - numeric float, 1 - character, 2 - log, 3 - numeric unsigned, 4 - text. This macro may be used with a numeric index e.g. {ITEM.VALUETYPE<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros. Supported since 5.4.0.

Low-level discovery rules

Macro	Supported in	Description
{LLDRULE.DESCRIPTION}	→ Item-based internal notifications	*Description of the low-level discovery rule which caused a notification.*

Macro	Supported in	Description
{LLDRULE.DESCRIPTION.ORIG}	→ LLD-rule based internal notifications	*Description (with macros unresolved) of the low-level discovery rule which caused a notification.* Supported since 5.2.0.
{LLDRULE.ID}	→ LLD-rule based internal notifications	*Numeric ID of the low-level discovery rule which caused a notification.*
{LLDRULE.KEY}	→ LLD-rule based internal notifications	*Key of the low-level discovery rule which caused a notification.*
{LLDRULE.KEY.ORIG}	→ LLD-rule based internal notifications	*Original key (with macros not expanded) of the low-level discovery rule which caused a notification.*
{LLDRULE.NAME}	→ LLD-rule based internal notifications	*Name of the low-level discovery rule (with macros resolved) that caused a notification.*
{LLDRULE.NAME.ORIG}	→ LLD-rule based internal notifications	*Original name (i.e. without macros resolved) of the low-level discovery rule that caused a notification.*
{LLDRULE.STATE}	→ LLD-rule based internal notifications	*The latest state of the low-level discovery rule.* Possible values: **Not supported** and **Normal**.
{LLDRULE.STATE.ERROR}	→ LLD-rule based internal notifications	*Error message with details why an LLD rule became unsupported.* If an LLD rule goes into the unsupported state and then immediately gets supported again the error field can be empty.

Maps

Macro	Supported in	Description
{MAP.ID}	→ Map element labels, map URL names and values	*Network map ID.*
{MAP.NAME}	→ Map element labels, map URL names and values → Text field in map shapes	*Network map name.* Supported since 3.4.0.

Proxies

Macro	Supported in	Description
{PROXY.DESCRIPTION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Manual event action scripts	*Description of the proxy*. Resolves to either: 1) proxy of the Nth item in the trigger expression (in trigger-based notifications). You may use indexed macros here. 2) proxy, which executed discovery (in discovery notifications). Use {PROXY.DESCRIPTION} here, without indexing. 3) proxy to which an active agent registered (in autoregistration notifications). Use {PROXY.DESCRIPTION} here, without indexing. This macro may be used with a numeric index e.g. {PROXY.DESCRIPTION<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Macro	Supported in	Description
{PROXY.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Discovery notifications and commands → Autoregistration notifications and commands → Internal notifications → Manual event action scripts	*Name of the proxy*. Resolves to either: 1) proxy of the Nth item in the trigger expression (in trigger-based notifications). You may use indexed macros here. 2) proxy, which executed discovery (in discovery notifications). Use {PROXY.NAME} here, without indexing. 3) proxy to which an active agent registered (in autoregistration notifications). Use {PROXY.NAME} here, without indexing. This macro may be used with a numeric index e.g. {PROXY.NAME<1-9>} to point to the first, second, third, etc. host in a trigger expression. See indexed macros.

Services

Macro	Supported in	Description
{SERVICE.DESCRIPTION}	→ Service-based notifications and commands → Service update notifications and commands	*Description of the service* (with macros resolved).
{SERVICE.NAME}	→ Service-based notifications and commands → Service update notifications and commands	*Name of the service* (with macros resolved).
{SERVICE.ROOTCAUSE}	→ Service-based notifications and commands → Service update notifications and commands	*List of trigger problem events that caused a service to fail*, sorted by severity and host name. Includes the following details: host name, event name, severity, age, service tags and values.
{SERVICE.TAGS}	→ Service-based notifications and commands → Service update notifications and commands	*A comma separated list of service event tags.* Service event tags can be defined in the service configuration section Tags. Expanded to an empty string if no tags exist.
{SERVICE.TAGSJSON}	→ Service-based notifications and commands → Service update notifications and commands	*A JSON array containing service event tag objects.* Service event tags can be defined in the service configuration section Tags. Expanded to an empty array if no tags exist.
{SERVICE.TAGS.<tag name>}	→ Service-based notifications and commands → Service update notifications and commands	*Service event tag value referenced by the tag name.* Service event tags can be defined in the service configuration section Tags. A tag name containing non-alphanumeric characters (including non-English multibyte-UTF characters) should be double quoted. Quotes and backslashes inside a quoted tag name must be escaped with a backslash.

Triggers

Macro	Supported in	Description
{TRIGGER.DESCRIPTION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*Trigger description.* All macros supported in a trigger description will be expanded if `{TRIGGER.DESCRIPTION}` is used in notification text. `{TRIGGER.COMMENT}` is deprecated.
{TRIGGER.EXPRESSION.EXPLAIN}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts → Event names	*Partially evaluated trigger expression.* Item-based functions are evaluated and replaced by the results at the time of event generation whereas all other functions are displayed as written in the expression. Can be used for debugging trigger expressions.

Macro	Supported in	Description
{TRIGGER.EXPRESSION.RECOVERY.EXPANDED}	→ Problem recovery notifications and commands → Problem update notifications and commands → Manual event action scripts	*Partially evaluated trigger recovery expression.* Item-based functions are evaluated and replaced by the results at the time of event generation whereas all other functions are displayed as written in the expression. Can be used for debugging trigger recovery expressions.
{TRIGGER.EVENTS.ACK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Map element labels → Manual event action scripts	*Number of acknowledged events for a map element in maps, or for the trigger which generated current event in notifications.*
{TRIGGER.EVENTS.PROBLEM.ACK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Map element labels → Manual event action scripts	*Number of acknowledged PROBLEM events for all triggers disregarding their state.*
{TRIGGER.EVENTS.PROBLEM.UNACK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Map element labels → Manual event action scripts	*Number of unacknowledged PROBLEM events for all triggers disregarding their state.*
{TRIGGER.EVENTS.UNACK}	→ Trigger-based notifications and commands → Problem update notifications and commands → Map element labels → Manual event action scripts	*Number of unacknowledged events for a map element in maps, or for the trigger which generated current event in notifications.*
{TRIGGER.HOSTGROUP.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*A sorted (by SQL query), comma-space separated list of host groups in which the trigger is defined.*
{TRIGGER.PROBLEM.EVENTS.ACK}	→ Map element labels	*Number of acknowledged PROBLEM events for triggers in PROBLEM state.*
{TRIGGER.PROBLEM.EVENTS.UNACK}	→ Map element labels	*Number of unacknowledged PROBLEM events for triggers in PROBLEM state.*
{TRIGGER.EXPRESSION}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*Trigger expression.*
{TRIGGER.EXPRESSION.RECOVERY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*Trigger recovery expression* if *OK event generation* in trigger configuration is set to 'Recovery expression'; otherwise an empty string is returned. Supported since 3.2.0.
{TRIGGER.ID}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Map element labels, map URL names and values → Trigger URLs → Trigger tag value → Manual event action scripts	*Numeric trigger ID which triggered this action.* Supported in trigger tag values since 4.4.1.
{TRIGGER.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*Name of the trigger* (with macros resolved). Note that since 4.0.0 {EVENT.NAME} can be used in actions to display the triggered event/problem name with macros resolved.
{TRIGGER.NAME.ORIG}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*Original name of the trigger* (i.e. without macros resolved).
{TRIGGER.NSEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*Numerical trigger severity.* Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster.
{TRIGGER.SEVERITY}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*Trigger severity name.* Can be defined in *Administration → General → Trigger displaying options*.

Macro	Supported in	Description
{TRIGGER.STATE}	→ Trigger-based internal notifications	*The latest state of the trigger.* Possible values: **Unknown** and **Normal**.
{TRIGGER.STATE.ERROR}	→ Trigger-based internal notifications	*Error message with details why a trigger became unsupported.* If a trigger goes into the unsupported state and then immediately gets supported again the error field can be empty.
{TRIGGER.STATUS}	→ Trigger-based notifications and commands → Problem update notifications and commands → Manual event action scripts	*Trigger value at the time of operation step execution.* Can be either PROBLEM or OK. `{STATUS}` is deprecated.
{TRIGGER.TEMPLATE.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*A sorted (by SQL query), comma-space separated list of templates in which the trigger is defined, or *UNKNOWN* if the trigger is defined in a host.*
{TRIGGER.URL}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*Trigger URL.*
{TRIGGER.URL.NAME}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger-based internal notifications → Manual event action scripts	*The label for the trigger URL.*
{TRIGGER.VALUE}	→ Trigger-based notifications and commands → Problem update notifications and commands → Trigger expressions → Manual event action scripts	*Current trigger numeric value*: 0 - trigger is in OK state, 1 - trigger is in PROBLEM state.
{TRIGGERS.UNACK}	→ Map element labels	*Number of unacknowledged triggers for a map element, disregarding trigger state.* A trigger is considered to be unacknowledged if at least one of its PROBLEM events is unacknowledged.
{TRIGGERS.PROBLEM.UNACK}	→ Map element labels	*Number of unacknowledged PROBLEM triggers for a map element.* A trigger is considered to be unacknowledged if at least one of its PROBLEM events is unacknowledged.
{TRIGGERS.ACK}	→ Map element labels	*Number of acknowledged triggers for a map element, disregarding trigger state.* A trigger is considered to be acknowledged if all of it's PROBLEM events are acknowledged.
{TRIGGERS.PROBLEM.ACK}	→ Map element labels	*Number of acknowledged PROBLEM triggers for a map element.* A trigger is considered to be acknowledged if all of it's PROBLEM events are acknowledged.

Users

Macro	Supported in	Description
{USER.FULLNAME}	→ Problem update notifications and commands → Manual host action scripts (including confirmation text) → Manual event action scripts (including confirmation text)	*Name, surname and username of the user* who added event acknowledgment or started the script. Supported for problem updates since 3.4.0, for global scripts since 5.0.2
{USER.NAME}	→ Manual host action scripts (including confirmation text) → Manual event action scripts (including confirmation text)	*Name of the user* who started the script. Supported since 5.0.2.
{USER.SURNAME}	→ Manual host action scripts (including confirmation text) → Manual event action scripts (including confirmation text)	*Surname of the user* who started the script. Supported since 5.0.2.

Macro	Supported in	Description
{USER.USERNAME}[4]	→ Manual host action scripts (including confirmation text) → Manual event action scripts (including confirmation text)	*Username of the user* who started the script. Supported since 5.0.2. {USER.ALIAS}, supported before Zabbix 5.4.0, is now deprecated.

Other macro types

Macro	Supported in	Description
{$MACRO}	→ See: User macros supported by location	*User-definable macros.*
{#MACRO}	→ See: Low-level discovery macros	*Low-level discovery macros.* Customizing the macro value is supported for this macro, starting with Zabbix 4.0.0.
{?EXPRESSION}	→ Trigger event names → Trigger-based notifications and commands → Problem update notifications and commands → Script commands and their webhook parameters → Map element labels[3] → Map shape labels[3] → Link labels in maps[3] → Graph names[5]	See expression macros. Supported since 5.2.0.
$1...$9	→ Trigger names → User parameter commands	*Positional macros/references.*

Footnotes

[1] The {HOST.*} macros supported in item key parameters will resolve to the interface that is selected for the item. When used in items without interfaces they will resolve to either the Zabbix agent, SNMP, JMX or IPMI interface of the host in this order of priority or to 'UNKNOWN' if the host does not have any interface.

[2] In global scripts, interface IP/DNS fields and web scenarios the macro will resolve to the main agent interface, however, if it is not present, the main SNMP interface will be used. If SNMP is also not present, the main JMX interface will be used. If JMX is not present either, the main IPMI interface will be used. If the host does not have any interface, the macro resolves to 'UNKNOWN'.

[3] Only the **avg**, **last**, **max** and **min** functions, with seconds as parameter are supported in this macro in map labels.

[4] {HOST.*} macros are supported in web scenario *Variables*, *Headers*, *SSL certificate file* and *SSL key file* fields and in scenario step *URL*, *Post*, *Headers* and *Required string* fields. Since Zabbix 5.2.2, {HOST.*} macros are no longer supported in web scenario *Name* and web scenario step *Name* fields.

[5] Only the **avg**, **last**, **max** and **min** functions, with seconds as parameter are supported within this macro in graph names. The {HOST.HOST<1-9>} macro can be used as host within the macro. For example:

```
* last(/Cisco switch/ifAlias[{#SNMPINDEX}])
* last(/{HOST.HOST}/ifAlias[{#SNMPINDEX}])
```

[6] Supported in script-type items and manual host action scripts for Zabbix server and Zabbix proxy.

Indexed macros

The indexed macro syntax of {MACRO**<1-9>**} works only in the context of **trigger expressions**. It can be used to reference hosts or functions in the order in which they appear in the expression. Macros like {HOST.IP1}, {HOST.IP2}, {HOST.IP3} will resolve to the IP of the first, second, and third host in the trigger expression (providing the trigger expression contains those hosts). Macros like {FUNCTION.VALUE1}, {FUNCTION.VALUE2}, {FUNCTION.VALUE3} will resolve to the value of the first, second, and third item-based function in the trigger expression at the time of the event (providing the trigger expression contains those functions).

Additionally the {HOST.HOST<1-9>} macro is also supported within the {?func(/host/key,param)} expression macro in **graph names**. For example, {?func(/{HOST.HOST2}/key,param)} in the graph name will refer to the host of the second item in the graph.

Warning:
Indexed macros will not resolve in any other context, except the two cases mentioned here. For other contexts, use macros **without** index (i. e.{HOST.HOST}, {HOST.IP}, etc) instead.

2 User macros supported by location

Overview

This section contains a list of locations, where user-definable macros are supported.

Note:
Only global-level user macros are supported for *Actions*, *Network discovery*, *Proxies* and all locations listed under *Other locations* section of this page. In the mentioned locations, host-level and template-level macros will not be resolved.

Actions

In actions, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Trigger-based notifications and commands		yes
Trigger-based internal notifications		yes
Problem update notifications		yes
Service-based notifications and commands		yes
Service update notifications		yes
Time period condition		no
Operations		
	Default operation step duration	no
	Step duration	no

Hosts/host prototypes

In a host and host prototype configuration, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Interface IP/DNS		DNS only
Interface port		no
SNMP v1, v2		
	SNMP community	yes
SNMP v3		
	Context name	yes
	Security name	yes
	Authentication passphrase	yes
	Privacy passphrase	yes
IPMI		
	Username	yes
	Password	yes
Tags[2]		
	Tag names	yes
	Tag values	yes

Items / item prototypes

In an item or an item prototype configuration, user macros can be used in the following fields:

Location	Multiple macros/mix with text[1]
Item key parameters	yes

Location	Multiple macros/mix with text[1]
Update interval	no
Custom intervals	no
History storage period	no
Trend storage period	no
Description	yes
Calculated item	
Formula	yes
Database monitor	
Username	yes
Password	yes
SQL query	yes
HTTP agent	
URL[3]	yes
Query fields	yes
Timeout	no
Request body	yes
Headers (names and values)	yes
Required status codes	yes
HTTP proxy	yes
HTTP authentication username	yes
HTTP authentication password	yes
SSl certificate file	yes
SSl key file	yes
SSl key password	yes
Allowed hosts	yes
JMX agent	
JMX endpoint	yes
Script item	
Parameter names and values	yes
SNMP agent	
SNMP OID	yes
SSH agent	
Username	yes
Public key file	yes
Private key file	yes
Password	yes
Script	yes

Location	Multiple macros/mix with text[1]
TELNET agent	
Username	yes
Password	yes
Script	yes
Zabbix trapper	
Allowed hosts	yes
Tags[2]	
Tag names	yes
Tag values	yes
Preprocessing steps	
Parameters (including custom scripts)	yes
Custom error-handling parameters (*Set value to* and *Set error to* fields) *Supported since Zabbix 6.4.10*.	yes

Low-level discovery

In a low-level discovery rule, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Key parameters		yes
Update interval		no
Custom interval		no
Keep lost resources period		no
Description		yes
SNMP agent		
	SNMP OID	yes
SSH agent		
	Username	yes
	Public key file	yes
	Private key file	yes
	Password	yes
	Script	yes
TELNET agent		
	Username	yes
	Password	yes
	Script	yes
Zabbix trapper		
	Allowed hosts	yes
Database monitor		
	Username	yes
	Password	yes
	SQL query	yes
JMX agent		
	JMX endpoint	yes
HTTP agent		
	URL[3]	yes
	Query fields	yes
	Timeout	no
	Request body	yes
	Headers (names and values)	yes
	Required status codes	yes
	HTTP authentication username	yes
	HTTP authentication password	yes
Filters		
	Regular expression	yes
Overrides		

Location		Multiple macros/mix with text[1]
	Filters: regular expression	yes
	Operations: update interval (for item prototypes)	no
	Operations: history storage period (for item prototypes)	no
	Operations: trend storage period (for item prototypes)	no

Network discovery

In a network discovery rule, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Update interval		no
SNMP v1, v2		
	SNMP community	yes
	SNMP OID	yes
SNMP v3		
	Context name	yes
	Security name	yes
	Authentication passphrase	yes
	Privacy passphrase	yes
	SNMP OID	yes

Proxies

In a proxy configuration, user macros can be used in the following field:

Location	Multiple macros/mix with text[1]
Interface port (for passive proxy)	no

Templates

In a template configuration, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Tags[2]		
	Tag names	yes
	Tag values	yes

Triggers

In a trigger configuration, user macros can be used in the following fields:

Location	Multiple macros/mix with text[1]
Name	yes
Operational data	yes

Location	Multiple macros/mix with text[1]
Expression (only in constants and function parameters; secret macros are not supported)	yes
Tag for matching	yes
Menu entry name	yes
Menu entry URL[3]	yes
Description	yes
Tags[2]	
Tag names	yes
Tag values	yes

Web scenario

In a web scenario configuration, user macros can be used in the following fields:

Location		Multiple macros/mix with text[1]
Name		yes
Update interval		no
Agent		yes
HTTP proxy		yes
Variables (values only)		yes
Headers (names and values)		yes
Steps		
	Name	yes
	URL[3]	yes
	Variables (values only)	yes
	Headers (names and values)	yes
	Timeout	no
	Required string	yes
	Required status codes	no
Authentication		
	User	yes
	Password	yes
	SSL certificate	yes
	SSL key file	yes

Location		Multiple macros/mix with text[1]
Tags[2]	SSL key password	yes
	Tag names	yes
	Tag values	yes

Other locations

In addition to the locations listed here, user macros can be used in the following fields:

Location	Multiple macros/mix with text[1]
Global scripts (URL, script, SSH, Telnet, IPMI), including confirmation text	yes
Webhooks	
JavaScript script	no
JavaScript script parameter name	no
JavaScript script parameter value	yes
Dashboards	
Column of data type *Text* in *Top hosts* dashboard widget	yes
Description parameter in *Item value* dashboard widget	yes
URL[3] parameter in *URL* dashboard widget	yes
Users → *Users* → *Media*	
When active	no
Administration → *General* → *GUI*	
Working time	no
Administration → *General* → *Connectors*	
URL	yes
Username	yes
Password	yes
Bearer token	yes

Location	Multiple macros/mix with text[1]
Timeout	no
HTTP proxy	yes
SSL certificate file	yes
SSL key file	yes
SSL key password	yes
Alerts → *Media types* → *Message templates*	
Subject	yes
Message	yes
Alerts → *Media types* → *Script*	
Script parameters	yes
Alerts → *Media types* → *Media type*	
Username and *Password* fields for the *Email* media type (when *Authentication* is set to "Username and password"; secret macros recommended)	yes

For a complete list of all macros supported in Zabbix, see supported macros.

Footnotes

[1] If multiple macros in a field or macros mixed with text are not supported for the location, a single macro has to fill the whole field.

[2] Macros used in tag names and values are resolved only during event generation process.

[3] URLs that contain a secret macro will not work, as the macro in them will be resolved as "******".

7 Unit symbols

Overview

Having to use some large numbers, for example '86400' to represent the number of seconds in one day, is both difficult and error-prone. This is why you can use some appropriate unit symbols (or suffixes) to simplify Zabbix trigger expressions and item keys.

Instead of '86400' for the number of seconds you can simply enter '1d'. Suffixes function as multipliers.

Time suffixes

For time you can use:

- **s** - seconds (when used, works the same as the raw value)

- **m** - minutes
- **h** - hours
- **d** - days
- **w** - weeks
- **M** - months (trend functions only)
- **y** - years (trend functions only)

Time suffixes support only integer numbers (so '1h' is supported, '1,5h' or '1.5h' are not; use '90m' instead).

Time suffixes are supported in:

- trigger expression constants and function parameters
- constants of calculated item formulas
- parameters of the **zabbix[queue,<from>,<to>]** internal item
- time period parameter of aggregate calculations
- item configuration ('Update interval', 'Custom intervals', 'History storage period' and 'Trend storage period' fields)
- item prototype configuration ('Update interval', 'Custom intervals', 'History storage period' and 'Trend storage period' fields)
- low-level discovery rule configuration ('Update interval', 'Custom intervals', 'Keep lost resources' fields)
- network discovery configuration ('Update interval' field)
- web scenario configuration ('Update interval', 'Timeout' fields)
- action operation configuration ('Default operation step duration', 'Step duration' fields)
- user profile settings ('Auto-logout', 'Refresh', 'Message timeout' fields)
- graph widget of *Dashboards* ('Time shift' field)
- *Administration* → *Housekeeping* (storage period fields)
- *Administration* → *General* → *Trigger displaying options* ('Display OK triggers for', 'On status change triggers blink for' fields)
- *Administration* → *General* → *Other* ('Login blocking interval' field and fields related to communication with Zabbix server)
- Zabbix server `ha_set_failover_delay=delay` runtime control option

Memory suffixes

Memory size suffixes are supported in:

- trigger expression constants and function parameters
- constants of calculated item formulas

For memory size you can use:

- **K** - kilobyte
- **M** - megabyte
- **G** - gigabyte
- **T** - terabyte

Other uses

Unit symbols are also used for a human-readable representation of data in the frontend.

In both Zabbix server and frontend these symbols are supported:

- **K** - kilo
- **M** - mega
- **G** - giga
- **T** - tera

When item values in B, Bps are displayed in the frontend, base 2 is applied (1K = 1024). Otherwise a base of 10 is used (1K = 1000).

Additionally the frontend also supports the display of:

- **P** - peta
- **E** - exa
- **Z** - zetta
- **Y** - yotta

Usage examples

By using some appropriate suffixes you can write trigger expressions that are easier to understand and maintain, for example these expressions:

```
last(/host/system.uptime)<86400s
avg(/host/system.cpu.load,600s)<10
last(/host/vm.memory.size[available])<20971520
```

could be changed to:

```
last(/host/system.uptime)<1d
avg(/host/system.cpu.load,10m)<10
last(/host/vm.memory.size[available])<20M
```

8 Time period syntax

Overview

To set a time period, the following format has to be used:

```
d-d,hh:mm-hh:mm
```

where the symbols stand for the following:

Symbol	Description
d	Day of the week: 1 - Monday, 2 - Tuesday ,... , 7 - Sunday
hh	Hours: 00-24
mm	Minutes: 00-59

You can specify more than one time period using a semicolon (;) separator:

```
d-d,hh:mm-hh:mm;d-d,hh:mm-hh:mm...
```

Leaving the time period empty equals 1-7,00:00-24:00, which is the default value.

Attention:
The upper limit of a time period is not included. Thus, if you specify 09:00-18:00 the last second included in the time period is 17:59:59.

Examples

Working hours. Monday - Friday from 9:00 till 18:00:

```
1-5,09:00-18:00
```

Working hours plus weekend. Monday - Friday from 9:00 till 18:00 and Saturday, Sunday from 10:00 till 16:00:

```
1-5,09:00-18:00;6-7,10:00-16:00
```

9 Command execution

Zabbix uses common functionality for external checks, user parameters, system.run items, custom alert scripts, remote commands and global scripts.

Execution steps

Note:
By default, all scripts in Zabbix are executed using the *sh* shell, and it is not possible to modify the default shell. To utilize a different shell, you can employ a workaround: create a script file and invoke that script during command execution.

The command/script is executed similarly on both Unix and Windows platforms:

1. Zabbix (the parent process) creates a pipe for communication
2. Zabbix sets the pipe as the output for the to-be-created child process
3. Zabbix creates the child process (runs the command/script)
4. A new process group (in Unix) or a job (in Windows) is created for the child process
5. Zabbix reads from the pipe until timeout occurs or no one is writing to the other end (ALL handles/file descriptors have been closed). Note that the child process can create more processes and exit before they exit or close the handle/file descriptor.
6. If the timeout has not been reached, Zabbix waits until the initial child process exits or timeout occurs

7. If the initial child process exited and the timeout has not been reached, Zabbix checks exit code of the initial child process and compares it to 0 (non-zero value is considered as execution failure, only for custom alert scripts, remote commands and user scripts executed on Zabbix server and Zabbix proxy)
8. At this point it is assumed that everything is done and the whole process tree (i.e. the process group or the job) is terminated

Attention:
Zabbix assumes that a command/script has done processing when the initial child process has exited AND no other process is still keeping the output handle/file descriptor open. When processing is done, ALL created processes are terminated.

All double quotes and backslashes in the command are escaped with backslashes and the command is enclosed in double quotes.

Exit code checking

Exit code are checked with the following conditions:

- Only for custom alert scripts, remote commands and user scripts executed on Zabbix server and Zabbix proxy.
- Any exit code that is different from 0 is considered as execution failure.
- Contents of standard error and standard output for failed executions are collected and available in frontend (where execution result is displayed).
- Additional log entry may be created for remote commands executed on Zabbix agent/proxy by enabling the LogRemoteCommands parameter on agent/proxy.

Possible frontend messages and log entries for failed commands/scripts:

- Contents of standard error and standard output for failed executions (if any).
- "Process exited with code: N." (for empty output, and exit code not equal to 0).
- "Process killed by signal: N." (for process terminated by a signal, on Linux only).
- "Process terminated unexpectedly." (for process terminated for unknown reasons).

See also

- External checks
- User parameters
- system.run items
- Custom alert scripts
- Remote commands
- Global scripts

10 Version compatibility

Supported agents

To be compatible with Zabbix 6.4, Zabbix agent must not be older than version 1.4 and must not be newer than 6.4.

You may need to review the configuration of older agents as some parameters have changed, for example, parameters related to logging for versions before 3.0.

To take full advantage of the latest metrics, improved performance and reduced memory usage, use the latest supported agent.

Notes for Windows XP

- On 32-bit Windows XP, do not use Zabbix agents newer than 6.0.x;
- On Windows XP/Server 2003, do not use agent templates that are newer than Zabbix 4.0.x. The newer templates use English performance counters, which are only supported since Windows Vista/Server 2008.

Supported agents 2

Older Zabbix agents 2 from version 4.4 onwards are compatible with Zabbix 6.4; Zabbix agent 2 must not be newer than 6.4.

Note that when using Zabbix agent 2 versions 4.4 and 5.0, the default interval of 10 minutes is used for refreshing unsupported items.

To take full advantage of the latest metrics, improved performance and reduced memory usage, use the latest supported agent 2.

Supported Zabbix proxies

To be fully compatible with Zabbix 6.4, the proxies must be of the same major version; thus only Zabbix 6.4.x proxies are fully compatible with Zabbix 6.4.x server. However, outdated proxies are also supported, although only partially.

In relation to Zabbix server, proxies can be:

- *Current* (proxy and server have the same major version);
- *Outdated* (proxy version is older than server version, but is partially supported);
- *Unsupported* (proxy version is older than server previous LTS release version *or* proxy version is newer than server major version).

Examples:

Server version	*Current* proxy version	*Outdated* proxy version	*Unsupported* proxy version
6.4	6.4	6.0, 6.2	Older than 6.0; newer than 6.4
7.0	7.0	6.0, 6.2, 6.4	Older than 6.0; newer than 7.0
7.2	7.2	7.0	Older than 7.0; newer than 7.2

Functionality supported by proxies:

Proxy version	Data update	Configuration update	Tasks
Current	Yes	Yes	Yes
Outdated	Yes	No	Remote commands (e.g., shell scripts); Immediate item value checks (i.e., *Execute now*); Note: Preprocessing tests with a real value are not supported.
Unsupported	No	No	No

Warnings about using incompatible Zabbix daemon versions are logged.

Supported XML files

XML files not older than version 1.8 are supported for import in Zabbix 6.4.

Attention:
In the XML export format, trigger dependencies are stored by name only. If there are several triggers with the same name (for example, having different severities and expressions) that have a dependency defined between them, it is not possible to import them. Such dependencies must be manually removed from the XML file and re-added after import.

11 Database error handling

If Zabbix detects that the backend database is not accessible, it will send a notification message and continue the attempts to connect to the database. For some database engines, specific error codes are recognized.

MySQL

- CR_CONN_HOST_ERROR
- CR_SERVER_GONE_ERROR
- CR_CONNECTION_ERROR
- CR_SERVER_LOST
- CR_UNKNOWN_HOST
- ER_SERVER_SHUTDOWN
- ER_ACCESS_DENIED_ERROR
- ER_ILLEGAL_GRANT_FOR_TABLE
- ER_TABLEACCESS_DENIED_ERROR
- ER_UNKNOWN_ERROR

12 Zabbix sender dynamic link library for Windows

Overview

In a Windows environment applications can send data to Zabbix server/proxy by using the Zabbix sender dynamic link library (zabbix_sender.dll) instead of having to launch an external process (zabbix_sender.exe).

zabbix_sender.h and zabbix_sender.lib are required for compiling user applications with zabbix_sender.dll.

Getting it

There are two ways of getting zabbix_sender.dll.

1. Download zabbix_sender.h, zabbix_sender.lib and zabbix_sender.dll files as a ZIP archive.

When choosing download options make sure to select "No encryption" under *Encryption* and "Archive" under *Packaging*. Then download Zabbix agent (not Zabbix agent 2).

The zabbix_sender.h, zabbix_sender.lib and zabbix_sender.dll files will be inside the downloaded ZIP archive in the `bin\dev` directory. Unzip the files where you need it.

See also known issues.

2. Build zabbix_sender.dll from source (see instructions).

The dynamic link library with the development files will be located in the `bin\winXX\dev` directory. To use it, include the zabbix_sender.h header file and link with the zabbix_sender.lib library.

See also

- example of a simple Zabbix sender utility implemented with Zabbix sender dynamic link library to illustrate the library usage;
- zabbix_sender.h file for the interface functions of the Zabbix sender dynamic link library. This file contains documentation explaining the purpose of each interface function, its arguments, and return value.

13 Python library for Zabbix API

Overview

zabbix_utils is a Python library for:

- working with Zabbix API;
- acting like Zabbix sender;
- acting like Zabbix get.

It is supported for Zabbix 5.0, 6.0, 6.4 and later.

14 Service monitoring upgrade

Overview In Zabbix 6.0, service monitoring functionality has been reworked significantly (see What's new in Zabbix 6.0.0 for the list of changes).

This page describes how services and SLAs, defined in earlier Zabbix versions, are changed during an upgrade to Zabbix 6.0 or newer.

Services In older Zabbix versions, services had two types of dependencies: soft and hard. After an upgrade, all dependencies will become equal.

If a service "Child service" has been previously linked to "Parent service 1" via hard dependency and additionally "Parent service 2" via soft dependency, after an upgrade the "Child service" will have two parent services "Parent service 1" and "Parent service 2".

Trigger-based mapping between problems and services has been replaced by tag-based mapping. In Zabbix 6.0 and newer, service configuration form has a new parameter *Problem tags*, which allows specifying one or multiple tag name and value pairs for problem matching. Triggers that have been linked to a service will get a new tag `ServiceLink` : `<trigger ID>:<trigger name>` (tag value will be truncated to 32 characters). Linked services will get `ServiceLink` problem tag with the same value.

Status calculation rules

The 'Status calculation algorithm' will be upgraded using the following rules:

- Do not calculate → Set status to OK
- Problem, if at least one child has a problem → Most critical of child services
- Problem, if all children have problems → Most critical if all children have problems

Note:
If you have upgraded from Zabbix pre-6.0 to Zabbix 6.0.0, 6.0.1 or 6.0.2, see Known issues for Zabbix 6.0 documentation.

SLAs Previously, SLA targets had to be defined for each service separately. Since Zabbix 6.0, SLA has become a separate entity, which contains information about service schedule, expected service level objective (SLO) and downtime periods to exclude from the calculation. Once configured, an SLA can be assigned to multiple services through service tags.

During an upgrade:

- Identical SLAs defined for each service will be grouped and one SLA per each group will be created.
- Each affected service will get a special tag `SLA:<ID>` and the same tag will be specified in the *Service tags* parameter of the corresponding SLA.
- Service creation time, a new metric in SLA reports, will be set to 01/01/2000 00:00 for existing services.

15 Other issues

Login and systemd

We recommend creating a *zabbix* user as system user, that is, without ability to log in. Some users ignore this recommendation and use the same account to log in (e. g. using SSH) to host running Zabbix. This might crash Zabbix daemon on log out. In this case you will get something like the following in Zabbix server log:

```
zabbix_server [27730]: [file:'selfmon.c',line:375] lock failed: [22] Invalid argument
zabbix_server [27716]: [file:'dbconfig.c',line:5266] lock failed: [22] Invalid argument
zabbix_server [27706]: [file:'log.c',line:238] lock failed: [22] Invalid argument
```

and in Zabbix agent log:

```
zabbix_agentd [27796]: [file:'log.c',line:238] lock failed: [22] Invalid argument
```

This happens because of default systemd setting `RemoveIPC=yes` configured in `/etc/systemd/logind.conf`. When you log out of the system the semaphores created by Zabbix previously are removed which causes the crash.

A quote from systemd documentation:

```
RemoveIPC=

Controls whether System V and POSIX IPC objects belonging to the user shall be removed when the
user fully logs out. Takes a boolean argument. If enabled, the user may not consume IPC resources
after the last of the user's sessions terminated. This covers System V semaphores, shared memory
and message queues, as well as POSIX shared memory and message queues. Note that IPC objects of the
root user and other system users are excluded from the effect of this setting. Defaults to "yes".
```

There are 2 solutions to this problem:

1. (recommended) Stop using *zabbix* account for anything else than Zabbix processes, create a dedicated account for other things.
2. (not recommended) Set `RemoveIPC=no` in `/etc/systemd/logind.conf` and reboot the system. Note that `RemoveIPC` is a system-wide parameter, changing it will affect the whole system.

Using Zabbix frontend behind proxy

If Zabbix frontend runs behind proxy server, the cookie path in the proxy configuration file needs to be rewritten in order to match the reverse-proxied path. See examples below. If the cookie path is not rewritten, users may experience authorization issues, when trying to login to Zabbix frontend.

Example configuration for nginx

```
# ..
location / {
# ..
proxy_cookie_path /zabbix /;
proxy_pass http://192.168.0.94/zabbix/;
# ..
```

Example configuration for Apache

```
# ..
ProxyPass "/" http://host/zabbix/
ProxyPassReverse "/" http://host/zabbix/
ProxyPassReverseCookiePath /zabbix /
ProxyPassReverseCookieDomain host zabbix.example.com
# ..
```

16 Agent vs agent 2 comparison

This section describes the differences between the Zabbix agent and the Zabbix agent 2.

Parameter	Zabbix agent	Zabbix agent 2
Programming language	C	Go with some parts in C
Daemonization	yes	by systemd only (yes on Windows)
Supported extensions	Custom loadable modules in C.	Custom plugins in Go.
Requirements		
Supported platforms	Linux, IBM AIX, FreeBSD, NetBSD, OpenBSD, HP-UX, Mac OS X, Solaris: 9, 10, 11, Windows: all desktop and server versions since XP	Linux, Windows: all desktop and server versions, on which an up-to-date supported Go version can be installed.
Supported crypto libraries	GnuTLS 3.1.18 and newer OpenSSL 1.0.1, 1.0.2, 1.1.0, 1.1.1, 3.0.x LibreSSL - tested with versions 2.7.4, 2.8.2 (certain limitations apply, see the Encryption page for details).	Linux: OpenSSL 1.0.1 and later is supported since Zabbix 4.4.8. MS Windows: OpenSSL 1.1.1 or later. The OpenSSL library must have PSK support enabled. LibreSSL is not supported.
Monitoring processes		
Processes	A separate active check process for each server/proxy record.	Single process with automatically created threads. The maximum number of threads is determined by the GOMAXPROCS environment variable.
Metrics	**UNIX**: see a list of supported items. **Windows**: see a list of additional Windows-specific items.	**UNIX**: All metrics supported by Zabbix agent. Additionally, the agent 2 provides Zabbix-native monitoring solution for: Docker, Memcached, MySQL, PostgreSQL, Redis, systemd, and other monitoring targets - see a full list of agent 2 specific items. **Windows**: All metrics supported by Zabbix agent, and also net.tcp.service* checks of HTTPS, LDAP. Additionally, the agent 2 provides Zabbix-native monitoring solution for: PostgreSQL, Redis.
Concurrency	Active checks for single server are executed sequentially.	Checks from different plugins or multiple checks within one plugin can be executed concurrently.
Scheduled/flexible intervals	Supported for passive checks only.	Supported for passive and active checks.
Third-party traps	no	yes
Additional features		
Persistent storage	no	yes
Persistent files for log*[] metrics	yes (only on Unix)	no
Log data upload	Can be performed during log gathering to free the buffer.	Log gathering is stopped when the buffer is full, therefore the BufferSize parameter must be at least MaxLinesPerSecond x 2.
Timeout settings	Defined on an agent level only.	Plugin timeout can override the timeout defined on an agent level.

Parameter	Zabbix agent	Zabbix agent 2
Changes user at runtime	yes (Unix-like systems only)	no (controlled by systemd)
User-configurable ciphersuites	yes	no

See also:

- *Zabbix processes description*: Zabbix agent, Zabbix agent 2
- *Configuration parameters*: Zabbix agent UNIX / Windows, Zabbix agent 2 UNIX / Windows

17 Escaping examples

Overview

This page provides examples of using correct escaping when using regular expressions in various contexts.

> **Note:**
> When using the trigger expression constructor, correct escaping in regular expressions is added automatically.

Examples

User macro with context

Regular expression: `\.+\"[a-z]+<br>` User macro with context: `{$MACRO:regex:"\.+\\"[a-z]+"}`

Notice:

- backslashes are not escaped;
- quotation marks are escaped.

LLD macro function

Regular expression: `\.+\"[a-z]+<br>` LLD macro: `{{#MACRO}.iregsub("\.+\\"[a-z]+", \1)}`

Notice:

- backslashes are not escaped;
- quotation marks are escaped.

LLD macro function inside user macro context

Regular expression: `\.+\"[a-z]+<br>` LLD macro: `{{#MACRO}.iregsub("\.+\\"[a-z]+", \1)}<br>` User macro with context: `{$MACRO:"{{#MACRO}.iregsub(\".+\\\"[a-z]+\", \1)}"}`

Notice:

- backslash escaping for LLD does not change;
- upon inserting the LLD macro into user macro context, we need to put it into string:

1. Quotation marks are added around the macro expression;
2. Quotation marks get escaped; in total, 3 new backslashes are introduced.

String parameter of non-history function

String content: `\.+\"[a-z]+<br>` Expression: `concat("abc", "\\.\\\"[a-z]+")`

Notice:

- String parameters require escaping both for backslashes and quotation marks.

String parameter of history function

String content: `\.+\"[a-z]+<br>` Expression: `find(__ITEM_KEY__,,"regexp","\.+\\"[a-z]+")`

Notice:

- backslashes are not escaped;
- quotation marks are escaped.

LLD macro function inside string parameter of non-history function

Regular expression: `\.+\"[a-z]+<br>` LLD macro: `{{#MACRO}.iregsub("\.+\\"[a-z]+", \1)}<br>` Expression: `concat("abc, "{{#MACRO}.iregsub(\"\\.+\\\\\"[a-z]+\", \\1)}")`

Notice:

- String parameters require escaping both for backslashes and quotation marks;
- Another layer of escaping is added, because the macro will be resolved only after string is unquoted;

LLD macro function inside string parameter of history function

Regular expression: `\.+\"[a-z]+<br>` LLD macro: `{{#MACRO}.iregsub("\.+\\"[a-z]+", \1)}<br>` Expression: `find(__ITEM_KEY__,,"eq","{{#MACRO}.iregsub(\"\.+\\\"[a-z]+\", \1)}")`

Notice:

- backslashes are not escaped;
- quotation marks are escaped.

User macro with context inside string parameter of non-history function

Regular expression: `\.+\"[a-z]+<br>` User macro with context: `{$MACRO:regex:"\.+\\"[a-z]+"}<br>` Expression: `concat("abc, "{$MACRO:regex:\"\\.+\\\\\"[a-z]+\"}")`

Notice:

- Same as in the previous example an additional layer of escaping is needed;
- Backslashes and quotation marks are escaped only for the top-level escaping (by virtue of it being a string parameter).

User macro with context inside string parameter of history function

Regular expression: `\.+\"[a-z]+<br>` User macro with context: `{$MACRO:regex:"\.+\\"[a-z]+"}<br>` Expression: `find(__ITEM_KEY__,,"eq","{$MACRO:regex:\"\.+\\\"[a-z]+\"}")`

Notice:

- backslashes are not escaped;
- quotation marks are escaped.

LLD macro function inside user macro context inside non-history function

Regular expression: `\.+\"[a-z]+<br>` LLD macro: `{{#MACRO}.iregsub("\.+\\"[a-z]+", \1)}<br>` User macro with context: `{$MACRO:"{{#MACRO}.iregsub(\".+\\\"[a-z]+\", \1)}"}<br>` Expression: `concat("abc, "{$MACRO:\"{{#MACRO}.ire \\1)}\"}")`

Notice the three layers of escaping:

1. For LLD macro function, without escaping of backslashes;
2. For User macro with context, without escaping of backslashes;
3. For the string parameter of a function, with escaping of backslashes.

LLD macro function inside user macro context inside history function

Regular expression: `\.+\"[a-z]+<br>` LLD macro: `{{#MACRO}.iregsub("\.+\\"[a-z]+", \1)}<br>` User macro with context: `{$MACRO:"{{#MACRO}.iregsub(\".+\\\"[a-z]+\", \1)}"}<br>` Expression: `find(__ITEM_KEY__,,"eq","{$MACRO:\"{{ \1)}\"}")`

Notice:

- backslashes are not escaped;
- quotation marks are escaped.

User macro with context just inside string

Regular expression: `\.+\"[a-z]+<br>` User macro with context: `{$MACRO:regex:"\.+\\"[a-z]+"}<br>` Inside string of some expression, for example: `func(arg1, arg2, arg3)="{$MACRO:regex:\"\\.+\\\\\"[a-z]+\"}"`

Notice:

- Strings also require backslash escaping;
- Strings also require quotation mark escaping;
- Again a case with 2 levels of escaping:

1. Escaping for user macro context without backslash escaping;
2. Escaping for it being a string with backslash escaping.

22 Quick reference guides

Overview

This documentation section contains quick recipes for setting up Zabbix for some commonly required monitoring goals.

It is designed with the new Zabbix user in mind and can be used as a navigator through other documentation sections that contain information required for resolving the task.

The following quick reference guides are available:

- Monitor Linux with Zabbix agent
- Monitor Windows with Zabbix agent
- Monitor Apache via HTTP
- Monitor MySQL with Zabbix agent 2
- Monitor VMware with Zabbix

1 Monitor Linux with Zabbix agent

Introduction This page walks you through the steps required to start basic monitoring of Linux machines with Zabbix. The steps described in this tutorial can be applied to any Linux-based operating system.

Who this guide is for

This guide is designed for new Zabbix users and contains the minimum set of steps required to enable basic monitoring of your Linux machine. If you are looking for deep customization options or require more advanced configuration, see Configuration section of Zabbix manual.

Prerequisites

Before proceeding with this installation guide, you must download and install Zabbix server and Zabbix frontend according to instructions for your OS.

Install Zabbix agent Zabbix agent is the process responsible for gathering data.

Check your Zabbix server version:

```
zabbix_server -V
```

Install Zabbix agent of the same version (recommended) on the Linux machine that you want to monitor. Based on your monitoring needs, it may be the same machine, where Zabbix server is installed, or a completely different machine.

Choose the most suitable installation method:

- Run as a Docker container - see the list of available images in Zabbix Docker repository.
- Install from Zabbix packages (available for Alma Linux, CentOS, Debian, Oracle Linux, Raspberry Pi OS, RHEL, Rocky Linux, SUSE Linux Enterprise Server, Ubuntu).
- Compile from sources.

Configure Zabbix for monitoring Zabbix agent can collect metrics in active or passive mode (simultaneously).

> **Note:**
> A passive check is a simple data request. Zabbix server or proxy asks for some data (for example, CPU load) and Zabbix agent sends back the result to the server. Active checks require more complex processing. The agent must first retrieve from the server(s) a list of items for independent processing and then bulk send the data back. See Passive and active agent checks for more info.

Monitoring templates provided by Zabbix usually offer two alternatives - a template for Zabbix agent and a template for Zabbix agent (active). With the first option, the agent will collect metrics in passive mode. Such templates will deliver identical monitoring results, but using different communication protocols.

Further Zabbix configuration depends on whether you select a template for active or passive Zabbix agent checks.

Passive checks Zabbix frontend

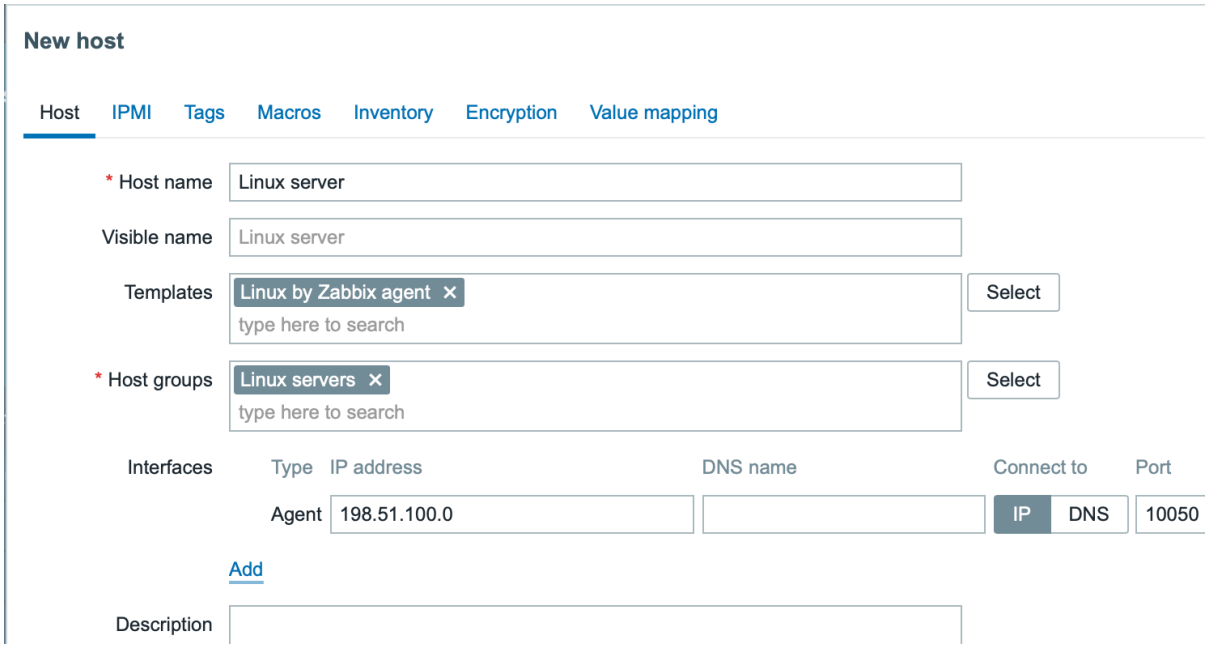
1. Log into Zabbix frontend.

2. Create a host in Zabbix web interface.

This host will represent your Linux machine.

3. In the *Interfaces* parameter, add *Agent* interface and specify the IP address or DNS name of the Linux machine where the agent is installed.

4. In the *Templates* parameter, type or select *Linux by Zabbix agent*.



Zabbix agent

Open Zabbix agent configuration file (by default, the path is */usr/local/etc/zabbix_agentd.conf*):

```
sudo vi /usr/local/etc/zabbix_agentd.conf
```

Add the IP address or DNS name of your Zabbix server to the *Server* parameter.

For example:

```
Server=192.0.2.22
```

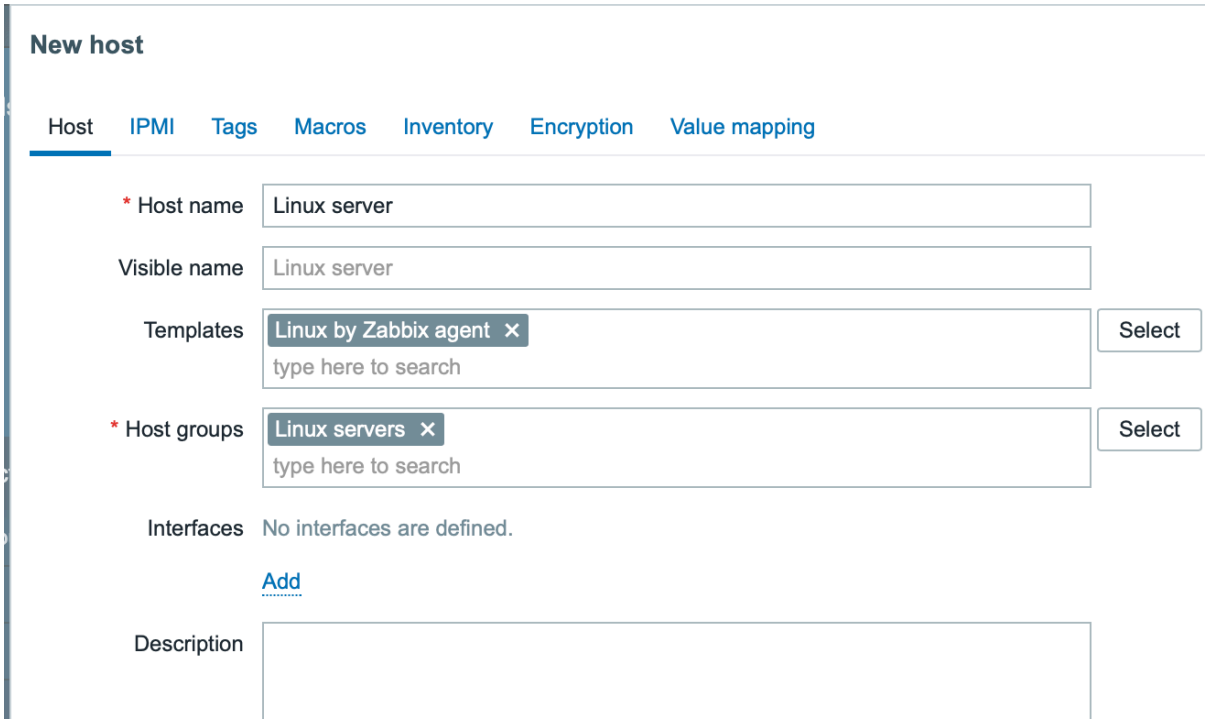
Active checks Zabbix frontend

1. Log in to Zabbix frontend.

2. Create a host in Zabbix web interface.

This host will represent your Linux machine.

3. In the *Templates* parameter, type or select *Linux by Zabbix agent active*.



Zabbix agent

Open Zabbix agent configuration file (by default, the path is */usr/local/etc/zabbix_agentd.conf*):

```
sudo vi /usr/local/etc/zabbix_agentd.conf
```

Add:

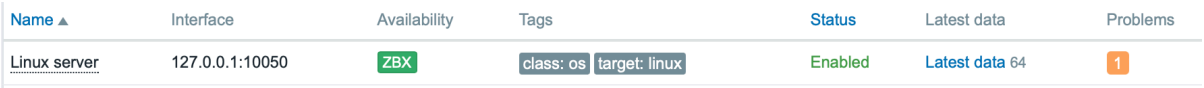
- The name of the host you created in Zabbix web interface to the *Hostname* parameter.
- The IP address or DNS name of your Zabbix server to the *ServerActive* parameter.

For example:

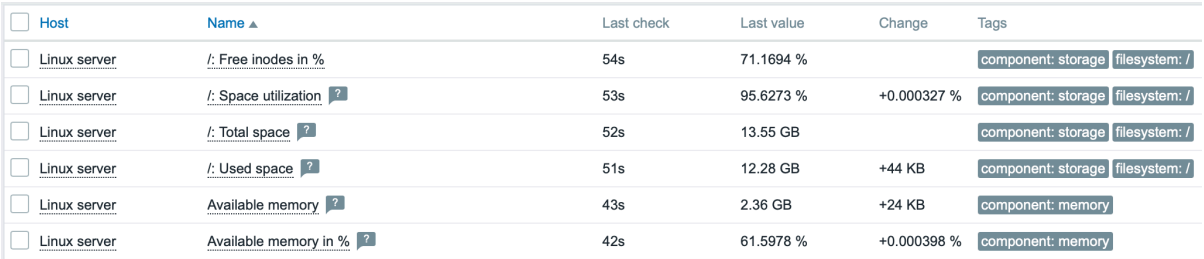
```
ServerActive= 192.0.2.22
Hostname=Linux server
```

View collected metrics Congratulations! At this point, Zabbix is already monitoring your Linux machine.

To view collected metrics, open the *Monitoring->Hosts* menu section and click on the *Latest data* next to the host.

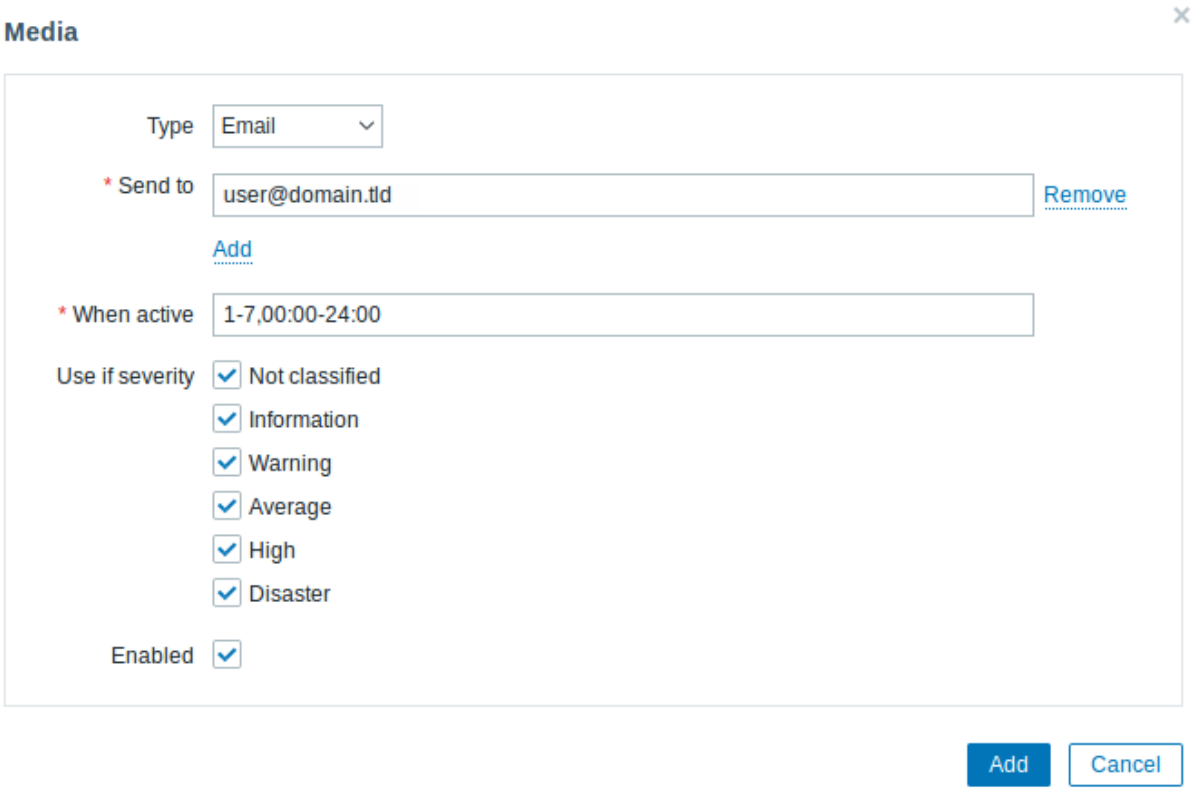


This action will open a list of all the latest metrics collected from Linux server host.



Set up problem alerts Zabbix can notify you about a problem with your infrastructure using a variety of methods. This guide provides configuration steps for sending email alerts.

1. Go to the *User settings -> Profile*, switch to the tab *Media* and add your email.



2. Follow the guide for Receiving problem notification.

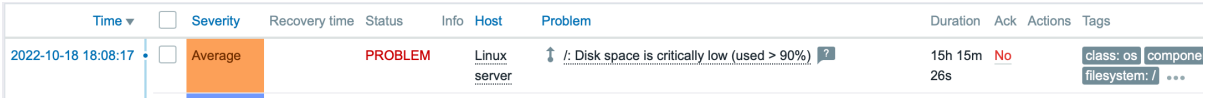
Next time, when Zabbix detects a problem you should receive an alert via email.

Test your configuration On Linux, you can simulate high CPU load and as a result receive a problem alert by running:

```
cat /dev/urandom | md5sum
```

You may need to run several md5sum processes for CPU load to exceed the threshold.

When Zabbix detects the problem, it will appear in the Monitoring->Problems section.



If the alerts are configured, you will also receive the problem notification.

See also:

- Creating an item - how to start monitoring additional metrics (custom monitoring without templates).
- Zabbix agent items, Zabbix agent items for Windows - full list of metrics you can monitor using Zabbix agent on Windows.
- Problem escalations - how to create multi-step alert scenarios (e.g., first send message to the system administrator, then, if a problem is not resolved in 45 minutes, send message to the data center manager).

2 Monitor Windows with Zabbix agent

Introduction This page walks you through the steps required to start basic monitoring of Windows machines with Zabbix.

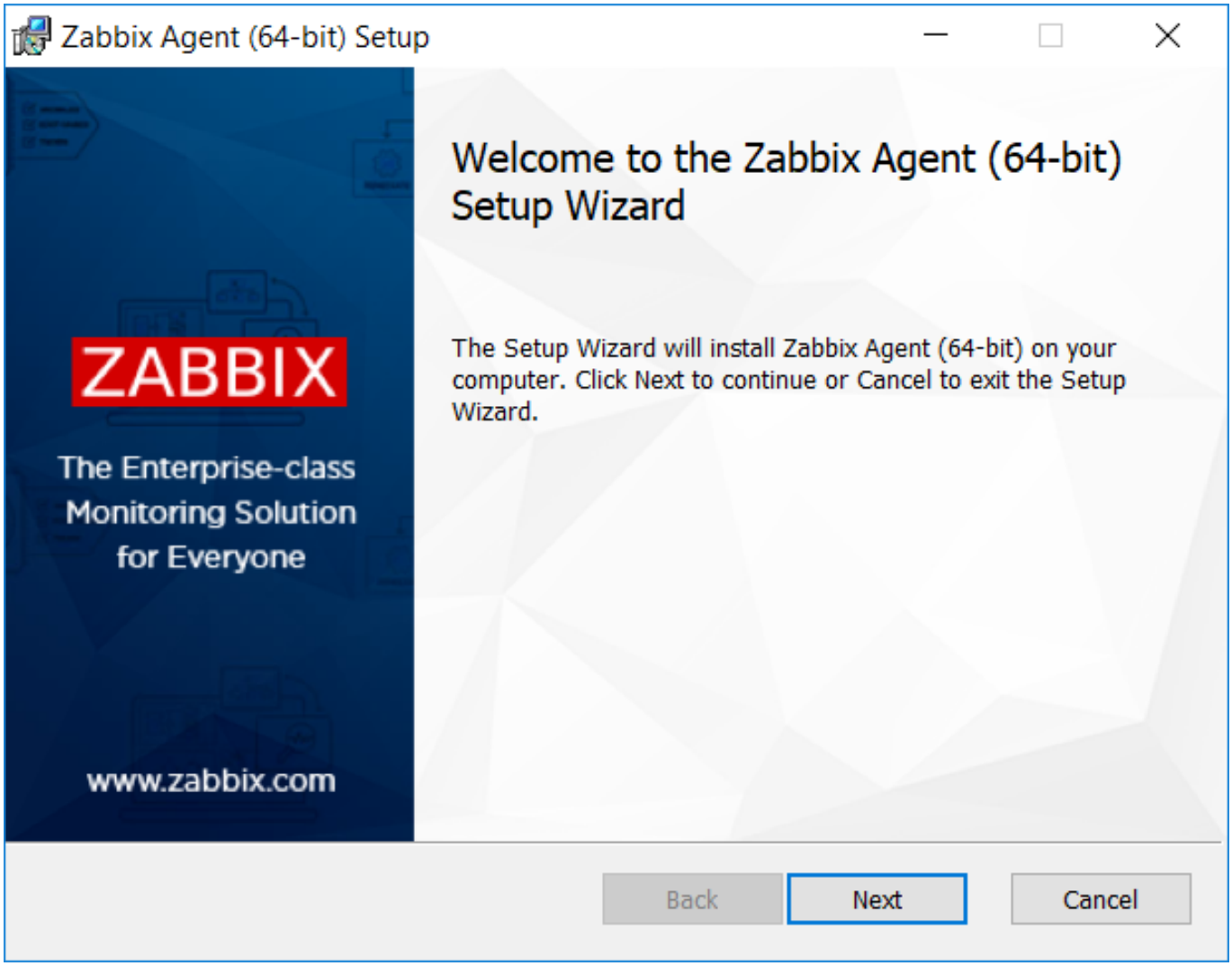
Who this guide is for

This guide is designed for new Zabbix users and contains the minimum set of steps required to enable basic monitoring of your Windows machine. If you are looking for deep customization options or require more advanced configuration, see Configuration section of Zabbix manual.

Prerequisites

Before proceeding with this installation guide, you must download and install Zabbix server and Zabbix frontend according to instructions for your OS.

Install Zabbix agent Zabbix agent is the process responsible for gathering data. You need to install it on the Windows machine that you want to monitor. Follow Zabbix agent installation instructions for Windows.



Configure Zabbix for monitoring Zabbix agent can collect metrics in active or passive mode (simultaneously).

Note:
A passive check is a simple data request. Zabbix server or proxy asks for some data (for example, CPU load) and Zabbix agent sends back the result to the server. Active checks require more complex processing. The agent must first retrieve from the server(s) a list of items for independent processing and then bulk send the data back. See Passive and active agent checks for more info.

Monitoring templates provided by Zabbix usually offer two alternatives - a template for Zabbix agent and a template for Zabbix agent (active). With the first option, the agent will collect metrics in passive mode. Such templates will deliver identical monitoring results, but using different communication protocols.

Further Zabbix configuration depends on whether you select a template for active or passive Zabbix agent checks.

Passive checks Zabbix frontend

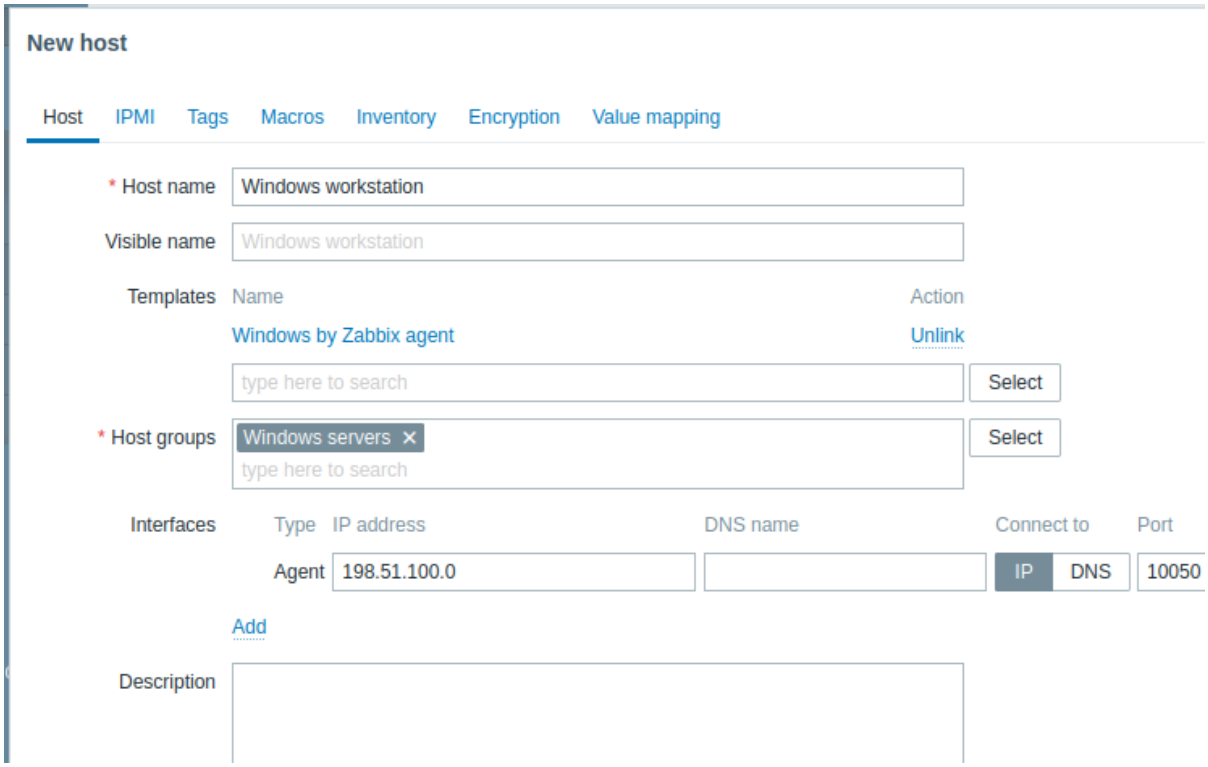
1. Log into Zabbix frontend.

2. Create a host in Zabbix web interface.

This host will represent your Windows machine.

3. In the *Interfaces* parameter, add *Agent* interface and specify the IP address or DNS name of the Windows machine where the agent is installed.

4. In the *Templates* parameter, type or select *Windows by Zabbix agent*.



Zabbix agent

For passive checks Zabbix agent needs to know the IP address or DNS name of Zabbix server. If you have provided correct information during the agent installation, the configuration file is already updated. Otherwise, you need to manually specify it. Go to the `C:\Program files\Zabbix Agent` folder, open the file *zabbix_agentd.conf* and add the IP/DNS of your Zabbix server to the *Server* parameter.

Example:

```
Server=192.0.2.22
```

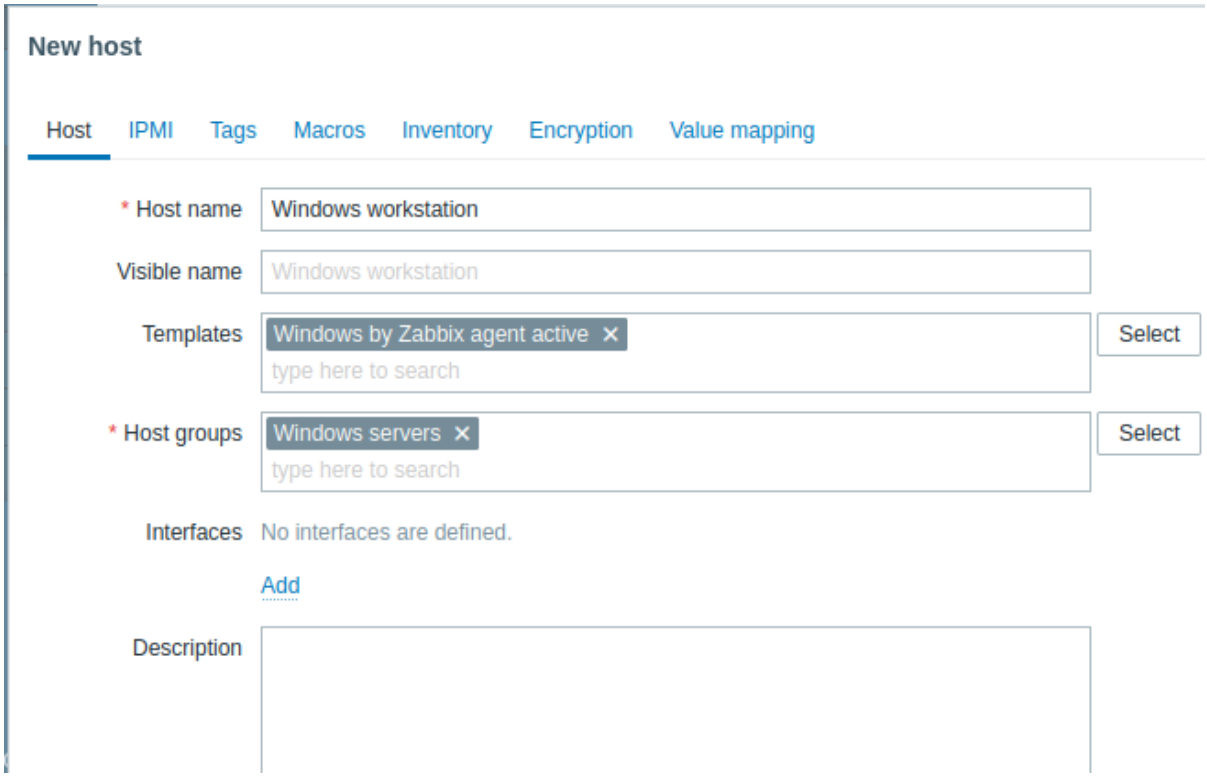
Active checks Zabbix frontend

1. Log in to Zabbix frontend.

2. Create a host in Zabbix web interface.

This host will represent your Windows machine.

3. In the *Templates* parameter, type or select *Windows by Zabbix agent active*.



Zabbix agent

In the `C:\Program files\Zabbix Agent` folder open the file *zabbix_agentd.conf* and add:

- The name of the host you created in Zabbix web interface to the *Hostname* parameter.
- The IP address or DNS name of your Zabbix server machine to the *ServerActive* parameter (might be prefilled if you have provided it during Zabbix agent setup).

Example:

```
ServerActive= 192.0.2.22
Hostname=Windows workstation
```

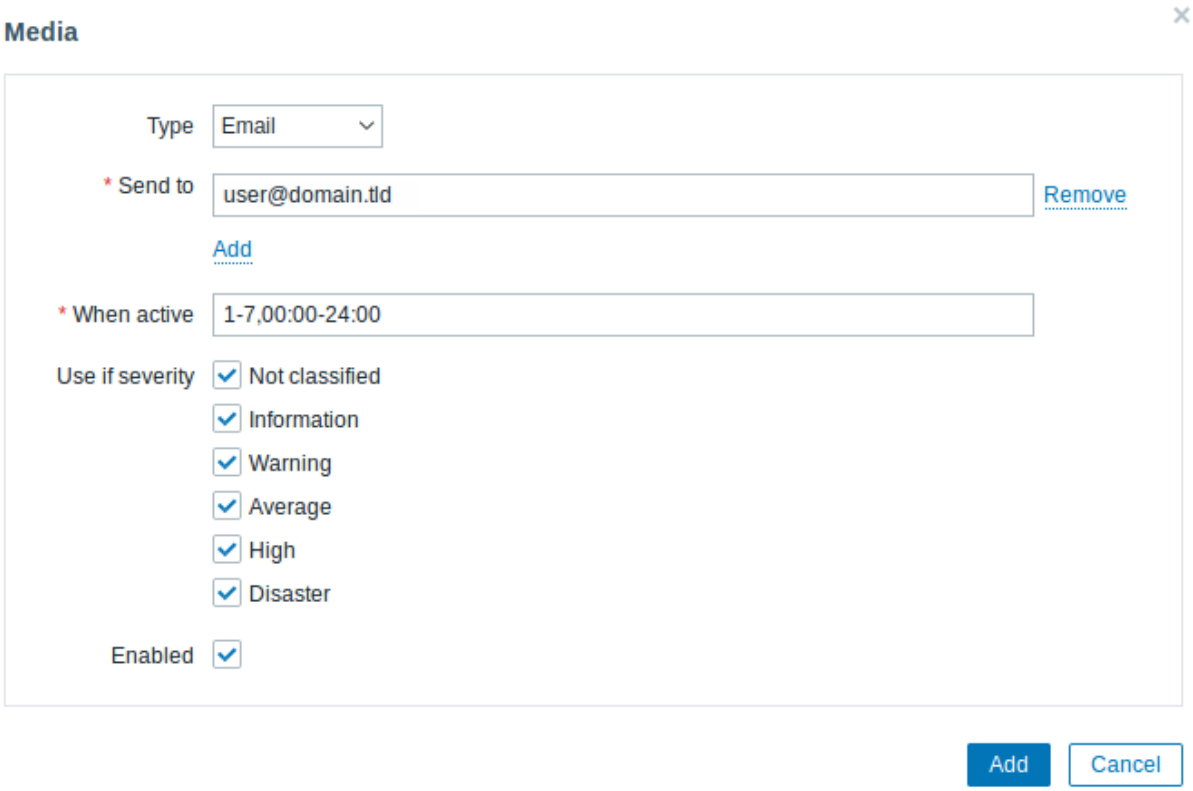
View collected metrics Congratulations! At this point, Zabbix is already monitoring your Windows machine.

To view collected metrics, open the *Monitoring->Hosts* menu section and click on the *Latest data* next to the host.



Set up problem alerts Zabbix can notify you about a problem with your infrastructure using a variety of methods. This guide provides configuration steps for sending email alerts.

1. Go to the *User settings -> Profile*, switch to the tab *Media* and add your email.



2. Follow the guide for Receiving problem notification.

Next time, when Zabbix detects a problem you should receive an alert via email.

Note:
On Windows, you can use CpuStres utility to simulate high CPU load and as a result receive a problem alert.

See also:

- Creating an item - how to start monitoring additional metrics (custom monitoring without templates).
- Zabbix agent items, Zabbix agent items for Windows - full list of metrics you can monitor using Zabbix agent on Windows.
- Problem escalations - how to create multi-step alert scenarios (e.g., first send message to the system administrator, then, if a problem is not resolved in 45 minutes, send message to the data center manager).

3 Monitor Apache via HTTP

Introduction This page shows a quick and simple way to start monitoring an Apache web server without installing any additional software.

Who this guide is for

This guide is designed for new Zabbix users and contains the minimum set of steps required to enable basic monitoring of your Apache installation. If you are looking for deep customization options or require more advanced configuration, see Configuration section of Zabbix manual.

Prerequisites

Before proceeding with this installation guide, you must download and install Zabbix server and Zabbix frontend according to instructions for your OS.

Prepare Apache 1. Check, which Apache version you are using:

On RHEL-based system, run:

```
httpd -v
```

On Debian/Ubuntu, run:

```
apache2 -v
```

2. Make sure that the Status module is enabled in your Apache instance.

On RHEL-based system, run:

```
httpd -M | grep status
status_module (shared)
```

On Debian/Ubuntu, run:

```
apache2ctl -M | grep status
status_module (shared)
```

If you don't see status_module in the list, enable the module by running:

On RHEL-based system, run:

```
LoadModule status_module /usr/lib/apache2/modules/mod_status.so
```

On Debian/Ubuntu, run:

```
sudo /usr/sbin/a2enmod status
```

3. Edit Apache configuration file to allow access to status reports from Zabbix server IP.

On an RHEL-based system: `/etc/httpd/conf.modules.d/status.conf`:

```
sudo vi /etc/httpd/conf.modules.d/status.conf
```

On Debian/Ubuntu: `/etc/apache2/mods-enabled/status.conf`:

```
sudo vi /etc/apache2/mods-enabled/status.conf
```

Add the following lines to the file (**replace 198.51.100.255** with your Zabbix server IP address):

- For Apache 2.2:
  <Location /server-status> SetHandler server-status

  Order Deny,Allow Deny from all Allow from 198.51.100.255 </Location>
- For Apache 2.4:
  <Location "/server-status"> SetHandler server-status Require ip 198.51.100.255 </Location>

4. Restart Apache

On an RHEL-based system, run:

```
sudo systemctl restart httpd
```

On Debian/Ubuntu, run:

```
sudo systemctl restart apache2
```

5. To check, if everything is configured correctly, run (**replace 198.51.100.255** with your Zabbix server IP address):

```
curl 198.51.100.255/server-status
```

The response should contain Apache web server statistics.

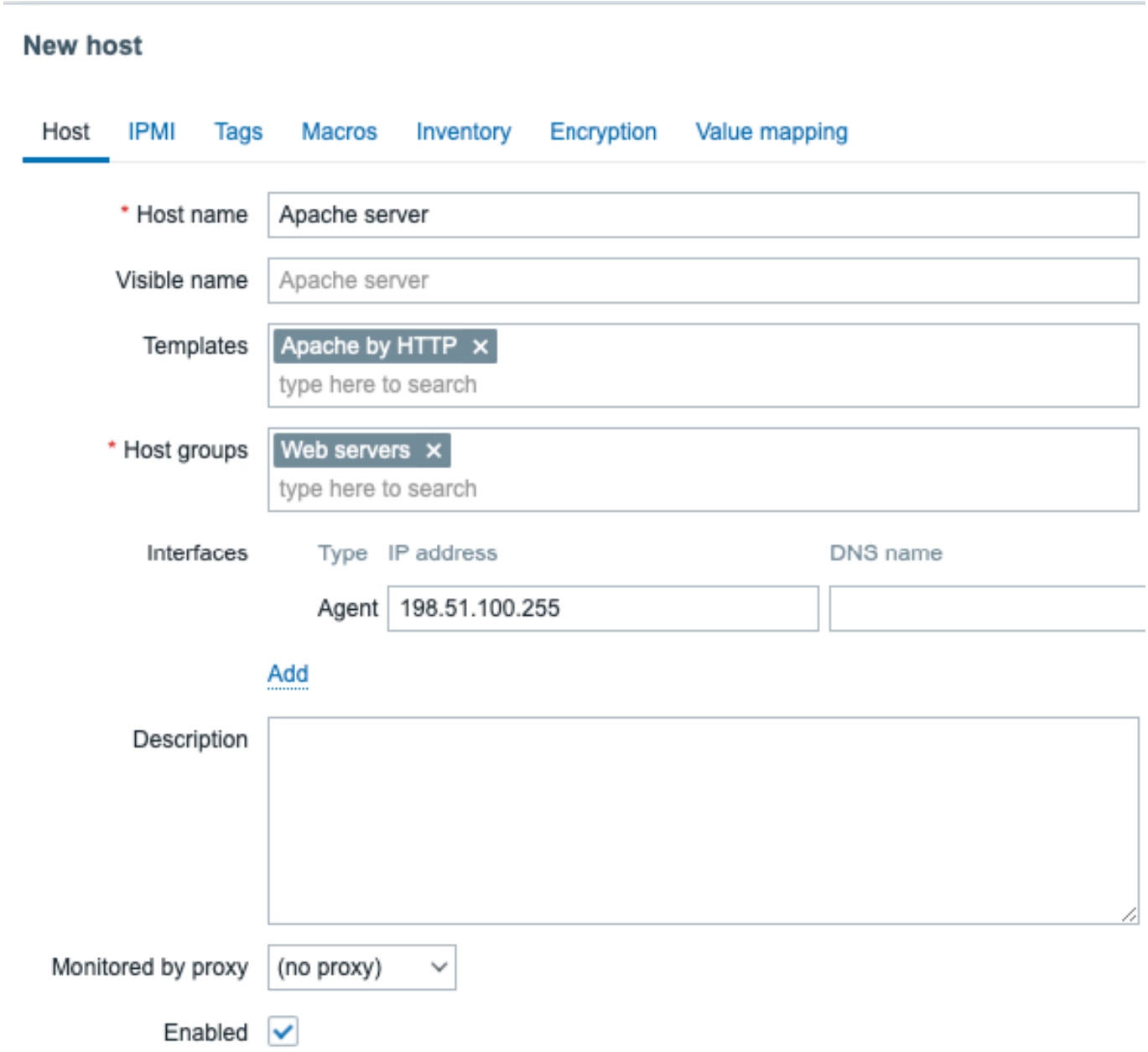
Configure Zabbix for monitoring 1. Log into Zabbix frontend.

2. Create a host in Zabbix web interface.

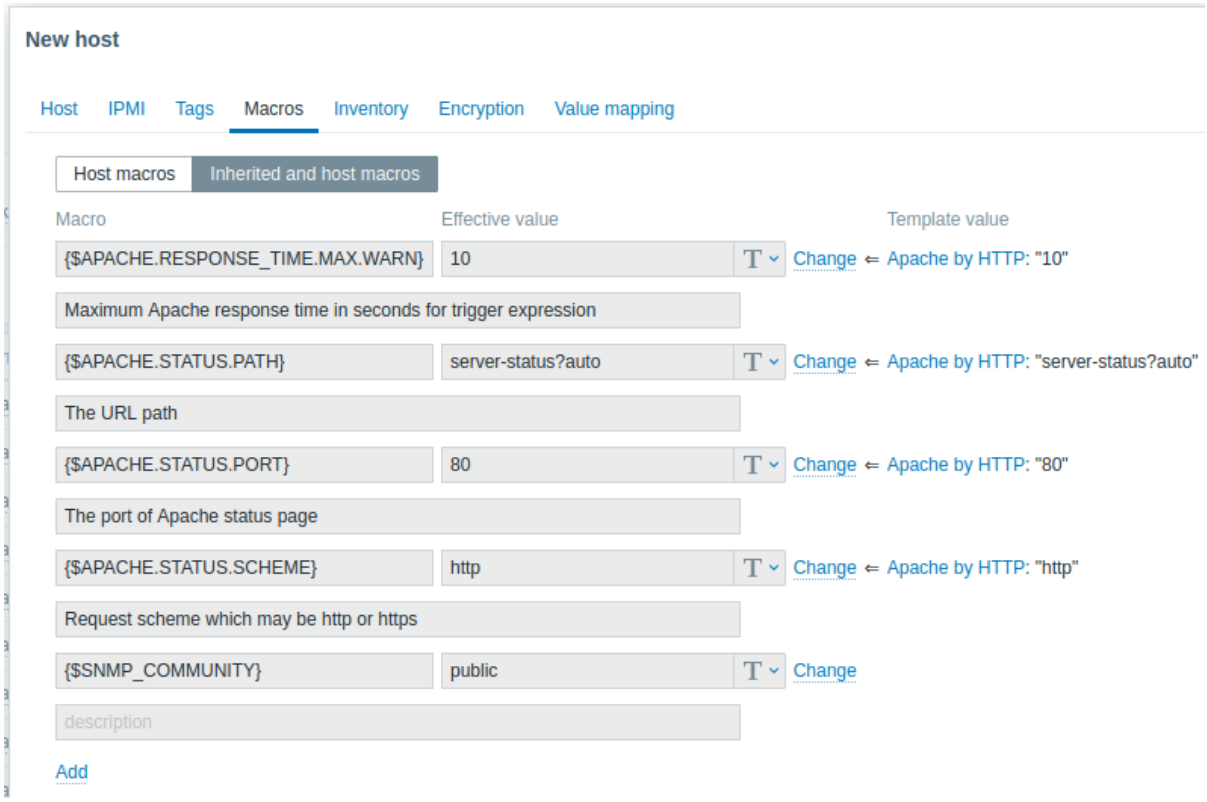
This host will represent your Apache server.

3. In the *Interfaces* parameter, add *Agent* interface and specify your Apache instance IP address. **You don't need to install Zabbix agent on the machine**, the interface will only be used for resolving {HOST.CONN} macro. This macro is used in template items to locate Apache instance.

4. In the *Templates* parameter, type or select *Apache by HTTP*.



5. Switch to the **Macros** tab and select *Inherited and host macros* mode. Check that values of the macros {$APACHE.STATUS.PORT} and {$APACHE.STATUS.SCHEME} suit your installation settings. By default, the port is 80 and the scheme is http. Change macro values if you use different port and/or scheme.

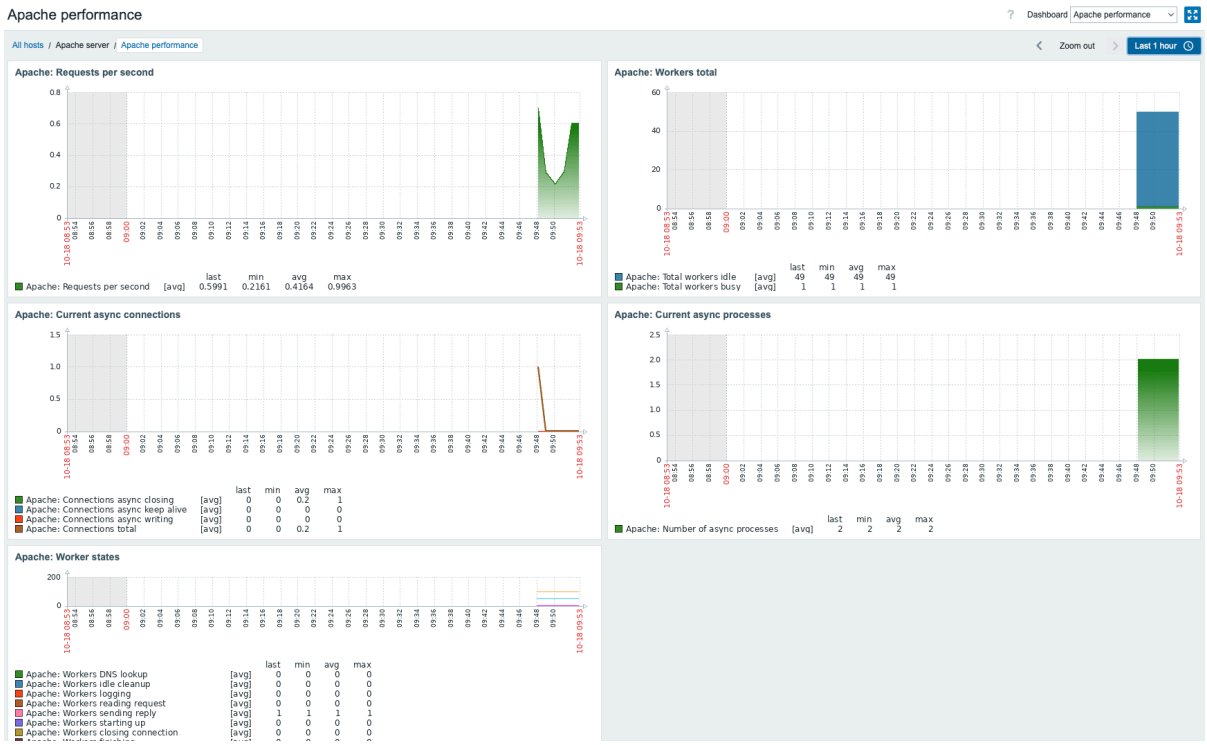


View collected metrics Congratulations! At this point, Zabbix is already monitoring your Apache web server.

To view collected metrics, open the *Monitoring->Hosts* menu section and click on the *Dashboards* next to the host.



This action will take you to the host dashboard with most important metrics collected from Apache /server-status page.

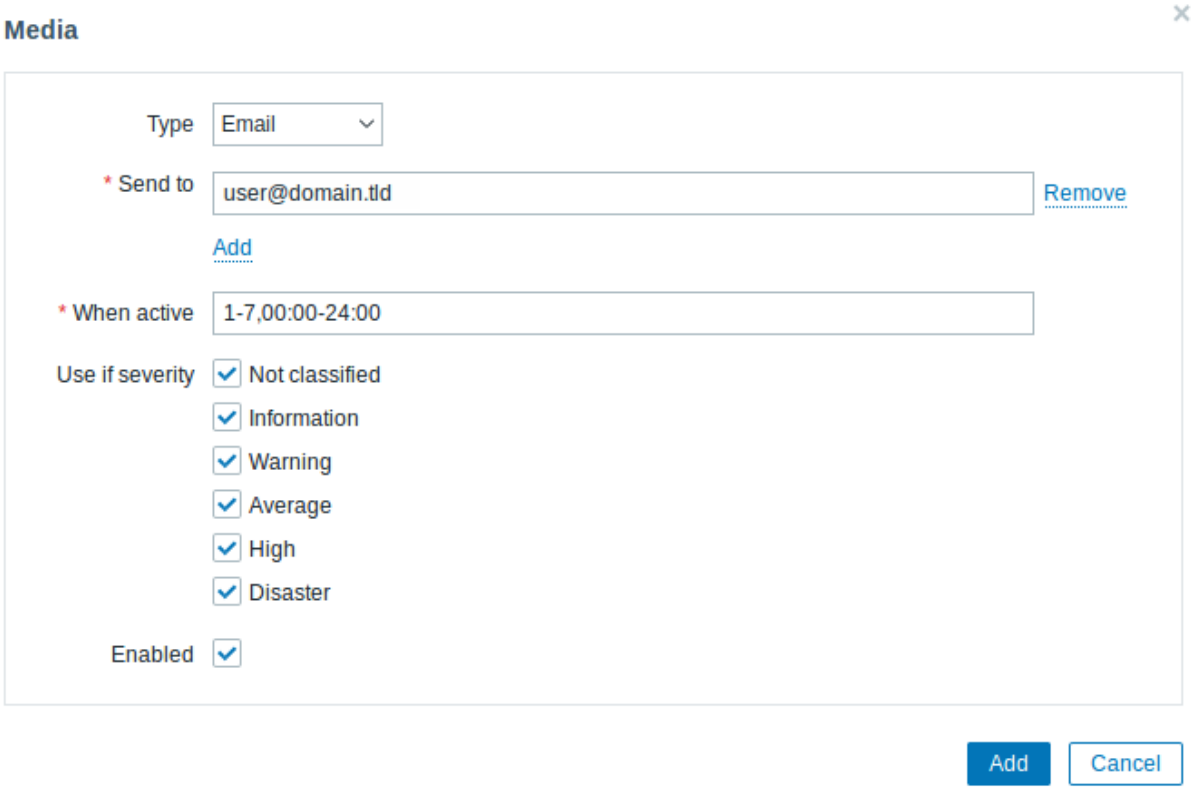


Alternatively, from the *Monitoring->Hosts*, you can click on the *Latest data* to view all the latest collected metrics in a list.



Set up problem alerts Zabbix can notify you about a problem with your infrastructure using a variety of methods. This guide provides configuration steps for sending email alerts.

1. Go to the *User settings -> Profile*, switch to the tab *Media* and add your email.



2. Follow the guide for Receiving problem notification.

Next time, when Zabbix detects a problem you should receive an alert via email.

Test your configuration To simulate real problem and receive a test problem alert:

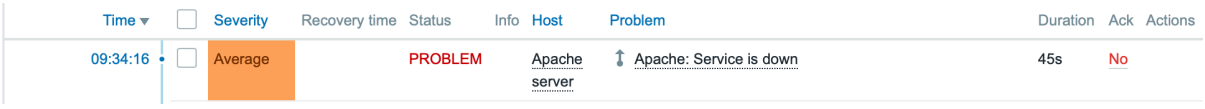
1. Open the *Apache server* host configuration in Zabbix.

2. Switch to the Macros tab and select *Inherited and host macros*.

3. Press *Change* next to {$APACHE.STATUS.PORT} macro and set a different port.

4. Press Update to save host configuration.

5. In a few minutes, Zabbix will detect the problem *Apache service is down*, because now it cannot connect to the instance. It will appear in the Monitoring->Problems section.



If the alerts are configured, you will also receive the problem notification.

6. Change the macro value back to resolve the problem and continue monitoring Apache.

See also:

- Web server hardening - recommended settings for greater web server security.
- Creating an item - how to start monitoring additional metrics.
- HTTP items - how to monitor custom metrics using HTTP agent.

- Problem escalations - how to create multi-step alert scenarios (e.g., first send message to the system administrator, then, if a problem is not resolved in 45 minutes, send message to the data center manager).

4 Monitor MySQL with Zabbix agent 2

Introduction

This page walks you through the steps required to start basic monitoring of a MySQL server.

To monitor a MySQL server, there are several approaches: Zabbix agent, Zabbix agent 2, or the Open Database Connectivity (ODBC) standard. The primary focus of this guide is on monitoring a MySQL server with Zabbix agent 2, which is the **recommended** approach due to its seamless configuration across various setups. However, this page also offers instructions for the other approaches, so feel free to choose the one that best suits your requirements.

Who this guide is for

This guide is designed for new Zabbix users and contains the minimum set of steps required to enable basic monitoring of a MySQL server. If you are looking for deep customization options or require more advanced configuration, see the Configuration section of Zabbix manual.

Prerequisites

Before proceeding with this guide, you need to download and install Zabbix server, Zabbix frontend and Zabbix agent 2 according to the instructions for your OS.

Based on your setup, some of the steps in this guide may slightly differ. This guide is based on the following setup:

- Zabbix version: Zabbix 6.4 (installed from packages)
- OS distribution: Ubuntu
- OS version: 22.04 (Jammy)
- Zabbix components: Server, Frontend, Agent 2
- Database: MySQL
- Web server: Apache

Create MySQL user

To monitor a MySQL server, Zabbix requires access to it and its processes. Your MySQL installation already has a user with the required level of access (the user "zabbix" that was created when installing Zabbix), however, this user has more privileges than necessary for simple monitoring (privileges to DROP databases, DELETE entries from tables, etc.). Therefore, a MySQL user for the purpose of *only* monitoring the MySQL server needs to be created.

1. Connect to the MySQL client, create a "zbx_monitor" user (replace <*password*> for the "zbx_monitor" user with a password of your choice), and GRANT the necessary privileges to the user:

```
mysql -u root -p
# Enter password:

mysql> CREATE USER 'zbx_monitor'@'%' IDENTIFIED BY '<password>';
mysql> GRANT REPLICATION CLIENT,PROCESS,SHOW DATABASES,SHOW VIEW ON *.* TO 'zbx_monitor'@'%';
mysql> quit;
```

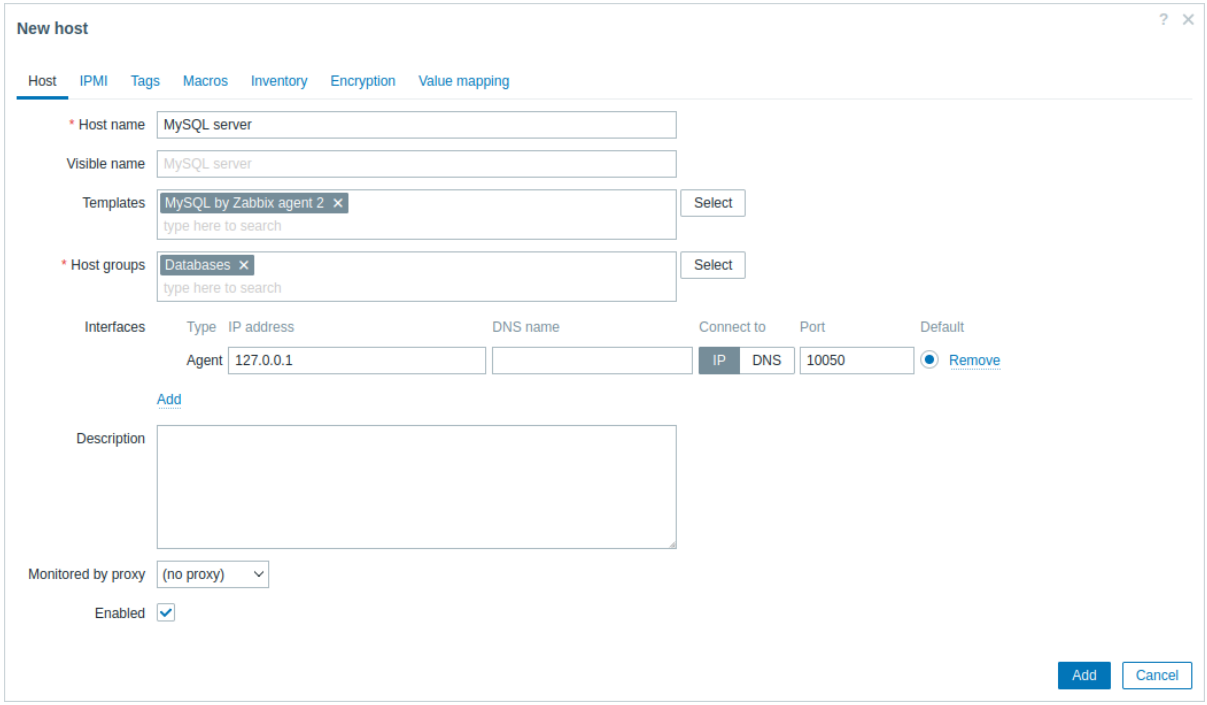
Once the user is created, you can move on to the next step.

Configure Zabbix frontend

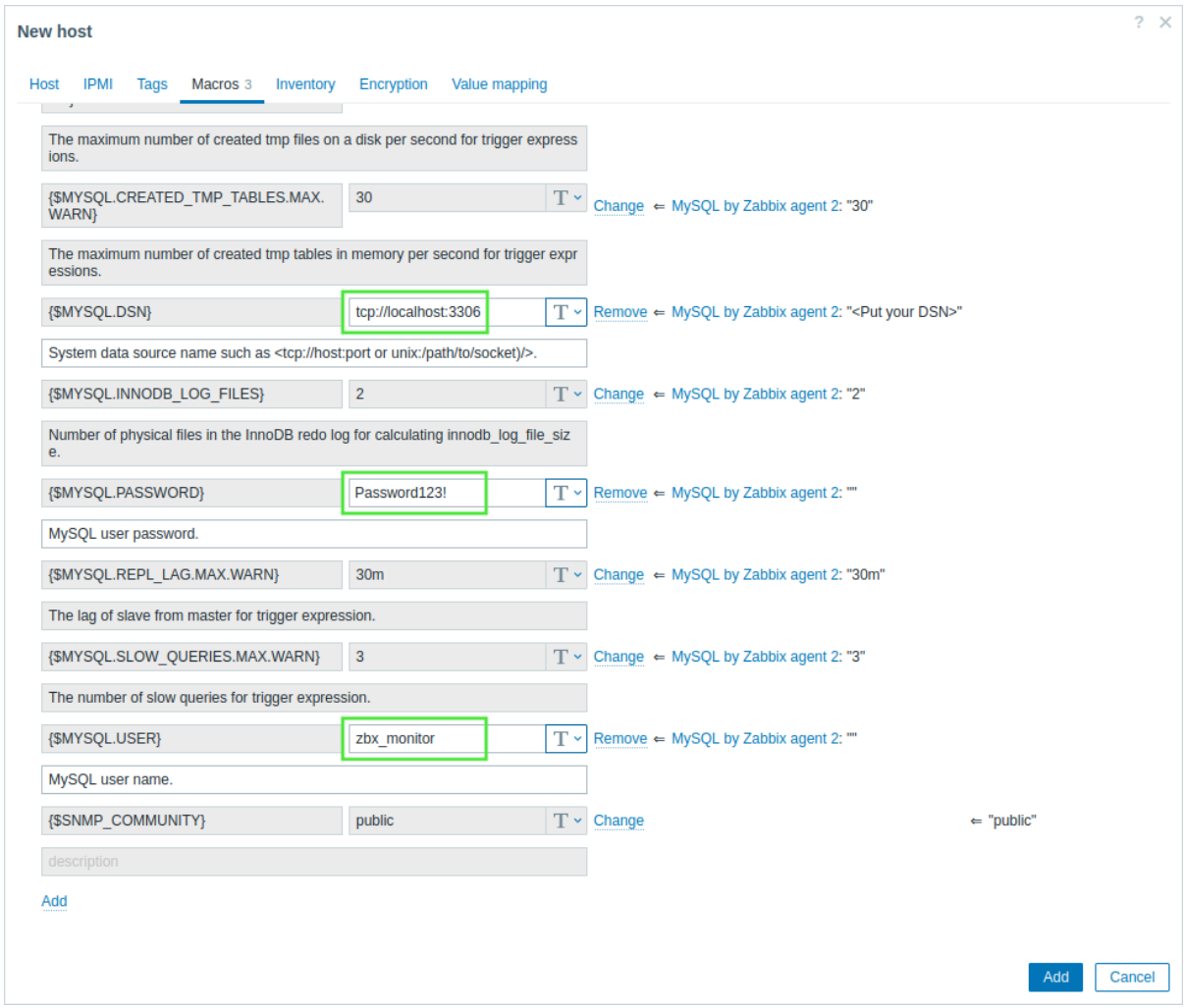
1. Log into Zabbix frontend.

2. Create a host in Zabbix web interface:

- In the *Host name* field, enter a host name (e.g., "MySQL server").
- In the *Templates* field, type or select the template "MySQL by Zabbix agent 2" that will be linked to the host.
- In the *Host groups* field, type or select a host group (e.g., "Databases").
- In the *Interfaces* field, add an interface of type "Agent" and specify your MySQL server IP address. This guide uses "127.0.0.1" (localhost) for monitoring a MySQL server that is installed on the same machine as Zabbix server and Zabbix agent 2.



- In the *Macros* tab, switch to *Inherited and host macros*, look for the following macros and click on *Change* next to the macro value to update it:
  - {$MYSQL.DSN} - set the data source of the MySQL server (the connection string of a named session from the MySQL Zabbix agent 2 plugin configuration file). This guide uses the default data source "tcp://localhost:3306" for monitoring a MySQL server that is installed on the same machine as Zabbix server and Zabbix agent 2.
  - {$MYSQL.PASSWORD} - set the password of the previously created MySQL user "zbx_monitor".
  - {$MYSQL.USER} - set the name of the previously created MySQL user "zbx_monitor".

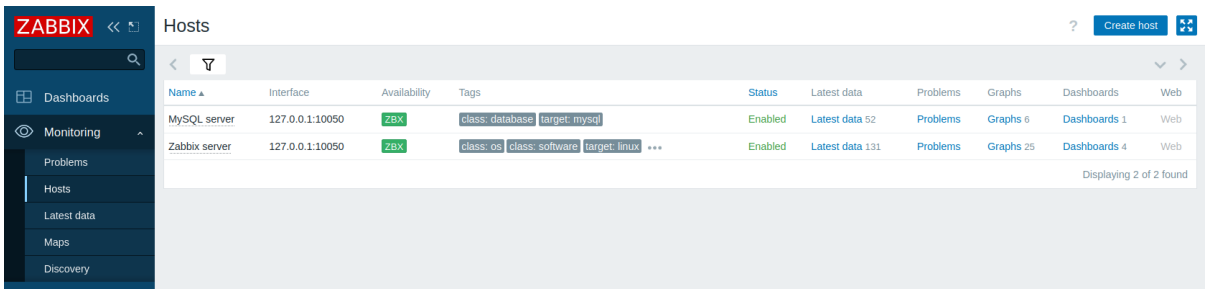


3. Click on *Add* to add the host. This host will represent your MySQL server.

View collected metrics

Congratulations! At this point, Zabbix is already monitoring your MySQL server.

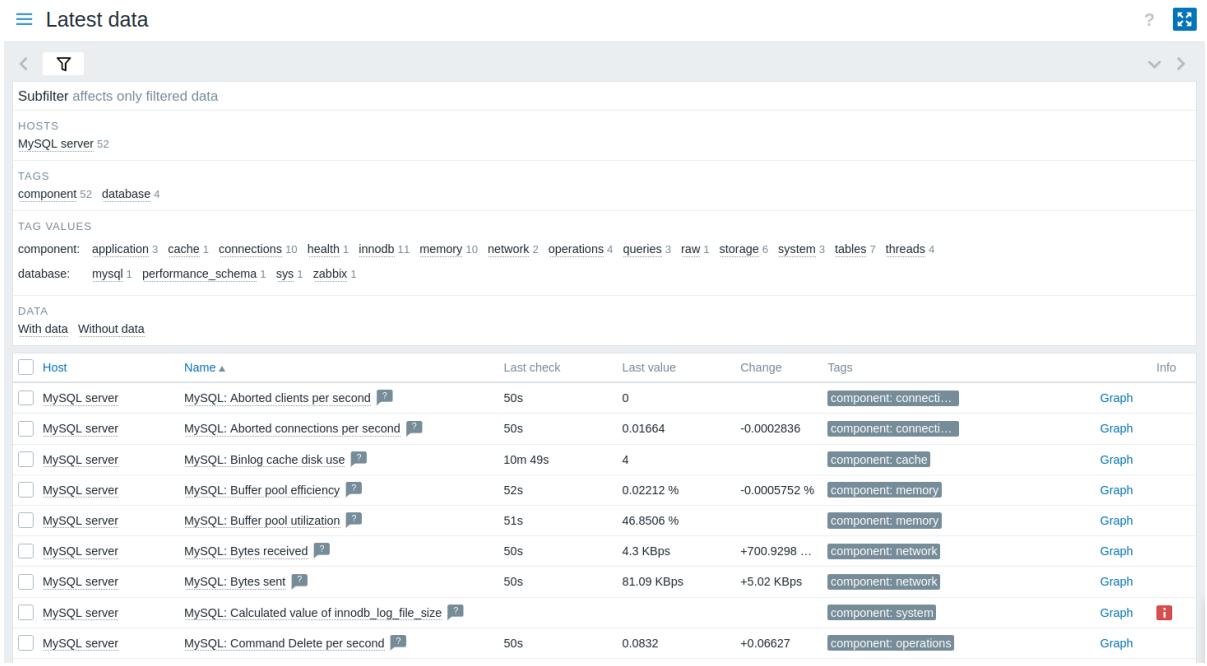
To view collected metrics, navigate to the *Monitoring → Hosts* menu section and click on *Dashboards* next to the host.



This action will take you to the host dashboard (configured on the template level) with the most important metrics collected from the MySQL server.



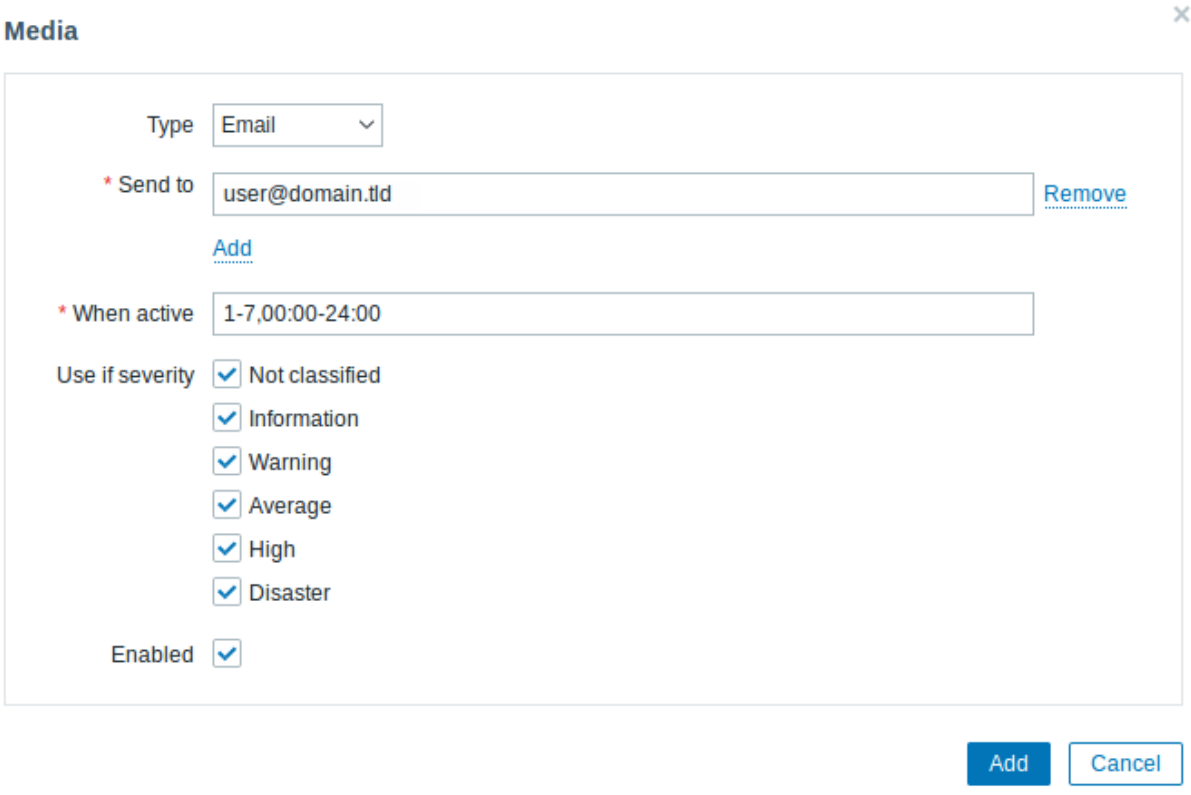
Alternatively, from the *Monitoring → Hosts* menu section, you can click on *Latest data* to view all the latest collected metrics in a list. Note that the item *MySQL: Calculated value of innodb_log_file_size* is expected to have no data, as the value will be calculated from data in the last hour.



Set up problem alerts

Zabbix can notify you about a problem with your infrastructure using a variety of methods. This guide provides basic configuration steps for sending email alerts.

1. Navigate to *User settings* → *Profile*, switch to the *Media* tab and add your email.



2. Follow the guide for Receiving a problem notification.

Next time, when Zabbix detects a problem, you should receive an alert via email.

Test your configuration

To test your configuration, we can simulate a real problem by updating the host configuration in Zabbix frontend.

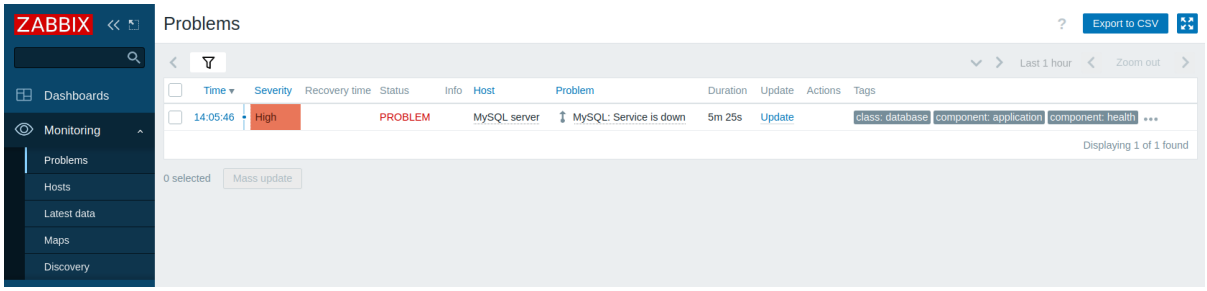
1. Open your MySQL server host configuration in Zabbix.

2. Switch to the *Macros* tab and select *Inherited and host macros*.

3. Click on *Change* next to, for example, the previously configured {$MYSQL.USER} macro value and set a different MySQL user name.

4. Click on *Update* to update the host configuration.

5. In a few moments, Zabbix will detect the problem "MySQL: Service is down", because it will not be able to connect to the MySQL server. The problem will appear in *Monitoring → Problems*.



If alerts are configured, you will also receive the problem notification.

6. Change the macro value back to its previous value to resolve the problem and continue monitoring the MySQL server.

Other approaches to monitor MySQL

Instead of monitoring a MySQL server with Zabbix agent 2, you could also use Zabbix agent or the Open Database Connectivity (ODBC) standard. While using Zabbix agent 2 is recommended, there might be some setups that do not support Zabbix agent 2 or require a custom approach.

The key difference between Zabbix agent and ODBC lies in the data collection method - Zabbix agent is installed directly on the MySQL server and collects data using its built-in functionality, while ODBC relies on an ODBC driver to establish a connection to the MySQL server and retrieve data using SQL queries.

Although many of the configuration steps are similar to monitoring a MySQL server with Zabbix agent 2, there are some significant differences - you need to configure Zabbix agent or ODBC to be able to monitor a MySQL server. The following instructions walk you through these **differences**.

Monitor MySQL with Zabbix agent

To monitor a MySQL server with Zabbix agent, you need to download and install Zabbix server, Zabbix frontend and Zabbix agent according to the instructions for your OS.

Once you have successfully installed the required Zabbix components, you need to create a MySQL user as described in the *Create MySQL user* section.

After you have created the MySQL user, you need to configure Zabbix agent to be able to establish a connection with the MySQL server and monitor it. This includes configuring multiple user parameters for executing custom agent checks, as well as providing Zabbix agent with the necessary credentials for connecting to the MySQL server as the previously created "zbx_monitor" user.

Configure Zabbix agent

1. Navigate to the Zabbix agent additional configurations directory.

```
cd /usr/local/etc/zabbix/zabbix_agentd.d
```

> **Attention:**
> The Zabbix agent additional configurations directory should be located in the same directory as your Zabbix agent configuration file (*zabbix_agentd.conf*). Depending on your OS and Zabbix installation, this directory can have a different location than specified in this guide. For default locations, check the `Include` parameter in the Zabbix agent configuration file.

Instead of defining all of the necessary user parameters for monitoring the MySQL server in the Zabbix agent configuration file, these parameters will be defined in a separate file in the additional configurations directory.

2. Create a *template_db_mysql.conf* file in the Zabbix agent additional configurations directory.

```
vi template_db_mysql.conf
```

3. Copy the contents from the *template_db_mysql.conf* file (located in the Zabbix repository) to the *template_db_mysql.conf* file you created, and save.

4. Restart Zabbix agent to update its configuration.

```
systemctl restart zabbix-agent
```

Once you have configured Zabbix agent user parameters, you can move on to configure the credentials that will allow Zabbix agent to access the MySQL server.

5. Navigate to the Zabbix agent home directory (if it does not exist on your system, you need to create it; default: */var/lib/zabbix*).

```
cd /var/lib/zabbix
```

6. Create a *.my.cnf* file in the Zabbix agent home directory.

```
vi .my.cnf
```

7. Copy the following contents to the *.my.cnf* file (replace *<password>* with the password of the "zbx_monitor" user).

```
[client]
user='zbx_monitor'
password='<password>'
```

Configure Zabbix frontend and test your configuration

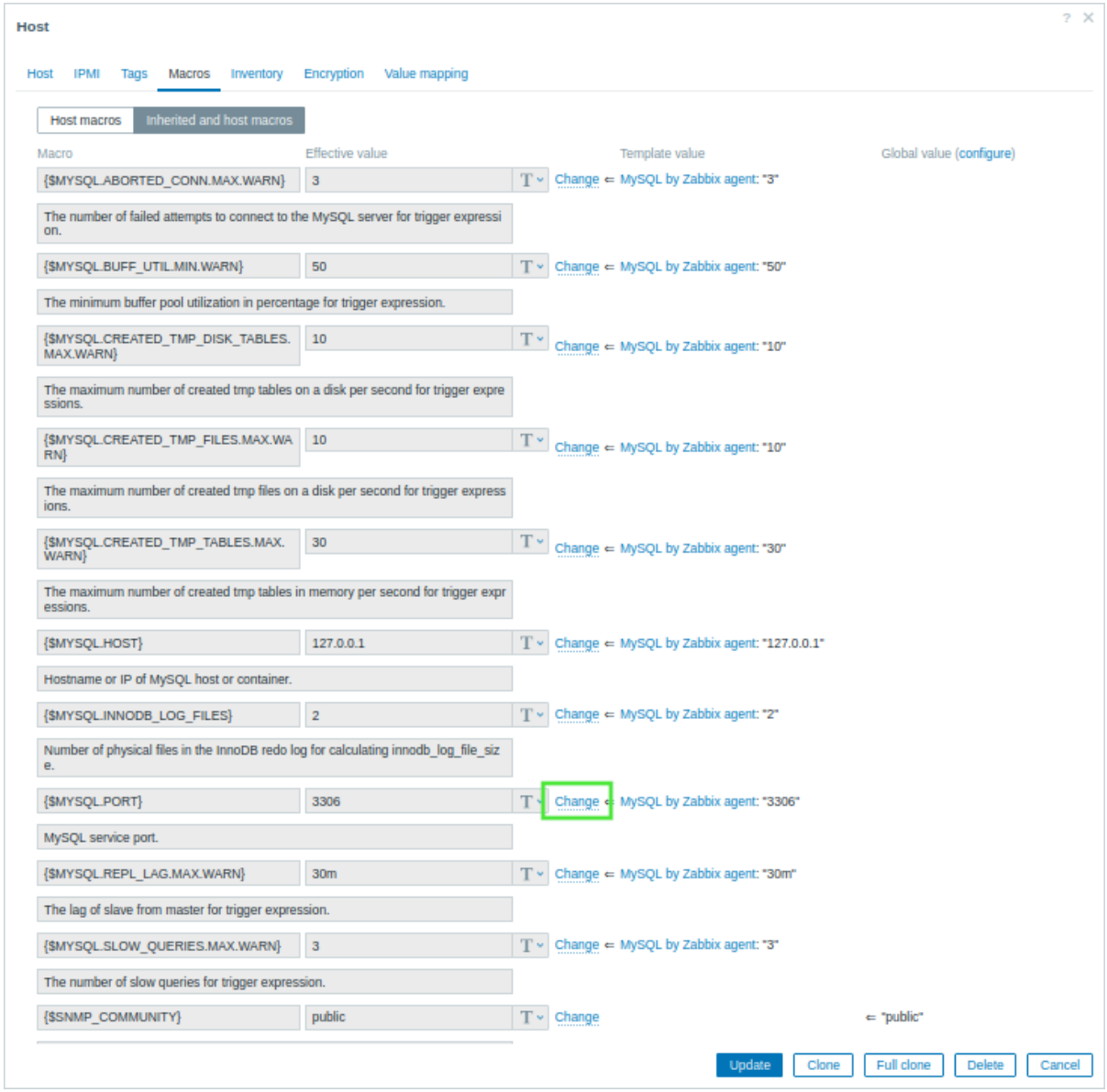
To configure Zabbix frontend, follow the instructions in the *Configure Zabbix frontend* section with the following adjustments:

- In the *Templates* field, type or select the template "MySQL by Zabbix agent" that will be linked to the host.
- Configuring *Macros* is not required.

Once you have configured Zabbix frontend, you can view collected metrics and set up problem alerts.

To test your configuration, follow the instructions in the *Test your configuration* section with the following adjustments:

- In the *Inherited and host macros* section of the MySQL server host configuration, click on *Change* next to the {$MYSQL.PORT} macro value and set a different port (e.g., "6033").



Monitor MySQL with ODBC

To monitor a MySQL server with ODBC, you need to download and install Zabbix server and Zabbix frontend.

Once you have successfully installed the required Zabbix components, you need to create a MySQL user as described in the *Create MySQL user* section.

After you have created the MySQL user, you need to setup ODBC. This includes installing one of the most commonly used open source ODBC API implementations - unixODBC - and a unixODBC driver, as well as editing the ODBC driver configuration file.

Configure ODBC

1. Install unixODBC. The suggested way of installing unixODBC is to use the Linux operating system default package repositories.

```
apt install unixodbc
```

2. Install the MariaDB unixODBC database driver. Although you have a MySQL database, the MariaDB unixODBC driver is used for compatibility issues.

```
apt install odbc-mariadb
```

3. Check the location of the ODBC configuration files *odbcinst.ini* and *odbc.ini*.

```
odbcinst -j
```

The result of executing this command should be similar to the following.

```
unixODBC 2.3.9
DRIVERS............: /etc/odbcinst.ini
SYSTEM DATA SOURCES: /etc/odbc.ini
FILE DATA SOURCES..: /etc/ODBCDataSources
...
```

4. To configure the ODBC driver for monitoring a MySQL database, you need the driver name, which is located in the *odbcinst.ini* file. In the following *odbcinst.ini* file example, the driver name is "MariaDB Unicode".

```
[MariaDB Unicode]
Driver=libmaodbc.so
Description=MariaDB Connector/ODBC(Unicode)
Threading=0
UsageCount=1
```

5. Copy the following contents to the *odbc.ini* file (replace *<password>* with the password of the "zbx_monitor" user). This guide uses "127.0.0.1" (localhost) as the MySQL server address for monitoring a MySQL server that is installed on the same machine as the ODBC driver. Note the data source name (DSN) "test", which will be required when configure Zabbix frontend.

```
[test]
Driver=MariaDB Unicode
Server=127.0.0.1
User=zbx_monitor
Password=<password>
Port=3306
Database=zabbix
```

Configure Zabbix frontend and test your configuration

To configure Zabbix frontend, follow the instructions in the *Configure Zabbix frontend* section with the following adjustments:

- In the *Templates* field, type or select the template "MySQL by ODBC" that will be linked to the host.
- Configuring *Interfaces* is not required.
- The {$MYSQL.DSN} macro value In the *Inherited and host macros* section of the MySQL server host configuration should be set to the DSN name from the *odbc.ini* file.

Once you have configured Zabbix frontend, you can view collected metrics, set up problem alerts and test your configuration.

See also

- Creating an item - how to start monitoring additional metrics.
- Problem escalations - how to create multi-step alert scenarios (e.g., first send message to the system administrator, then, if a problem is not resolved in 45 minutes, send message to the data center manager).
- ODBC monitoring - how to set up ODBC on other Linux distributions, and how to start monitoring additional database-related metrics with ODBC.
- Template *MySQL by Zabbix agent* - additional information about the *MySQL by Zabbix agent* template.
- Template *MySQL by Zabbix agent 2* - additional information about the *MySQL by Zabbix agent 2* template.
- Template *MySQL by ODBC* - additional information about the *MySQL by ODBC* template.

5 Monitor VMware with Zabbix

Introduction

This page walks you through the steps required to start basic monitoring of VMware.

Who this guide is for

This guide is designed for new Zabbix users and contains the minimum set of steps required to enable basic monitoring of VMware. If you are looking for deep customization options or require more advanced configuration, see the Virtual machine monitoring section or the Configuration section of Zabbix manual.

Prerequisites

Before proceeding with this guide, you need to download and install Zabbix server and Zabbix frontend according to the instructions for your OS.

This guide is based on the following setup:

- Zabbix version: 6.4 (installed from packages)
- OS distribution: Ubuntu
- OS version: 22.04 (Jammy)
- Zabbix components: Server, Frontend, Agent
- Database: MySQL
- Web server: Apache

It is assumed that VMware is already configured. This guide does not cover the configuration of VMware.

Configure Zabbix server

To monitor VMware, the *vmware collector* Zabbix processes need to be enabled. For more information on how VMware monitoring is performed, see Virtual machine monitoring.

1. Open the Zabbix server configuration file.

```
vi /etc/zabbix/zabbix_server.conf
```

2. Locate and set the `StartVMwareCollectors` parameter in Zabbix server configuration file to 2 or more (the default value is 0).

```
##### Option: StartVMwareCollectors
###       Number of pre-forked vmware collector instances.
###
### Mandatory: no
### Range: 0-250
### Default:
### StartVMwareCollectors=0

StartVMwareCollectors=2
```

3. Restart Zabbix server.

```
systemctl restart zabbix-server
```

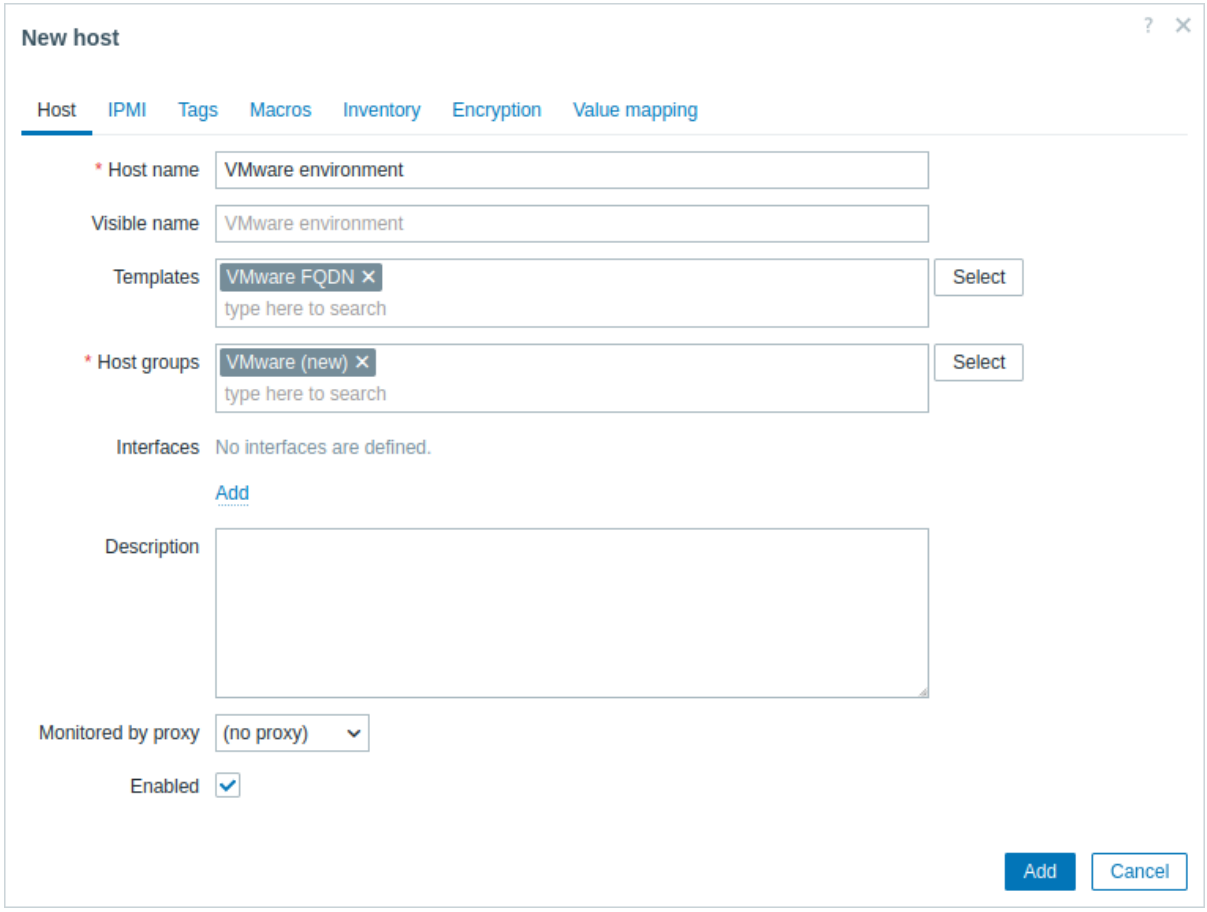
Once the *vmware collector* processes have been started, move on to the next step.

Configure Zabbix frontend

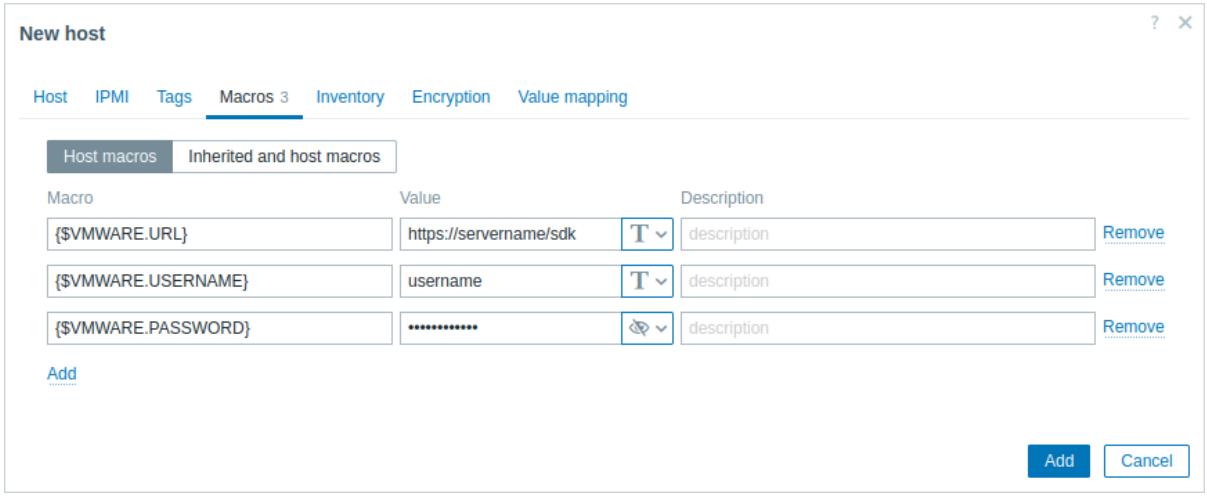
1. Log into Zabbix frontend.

2. Create a host in Zabbix web interface:

- In the *Host name* field, enter a host name (for example, ”VMware environment”).
- In the *Templates* field, type or select the ”VMware FQDN” (or ”VMware”) template. For more information on these templates, see Virtual machine monitoring.
- In the *Host groups* field, type or select a host group (for example, a new host group ”VMware”).



- In the *Macros* tab, set the following host macros:
  - {$VMWARE.URL} - VMware service (vCenter or ESXi hypervisor) SDK URL (https://servername/sdk)
  - {$VMWARE.USERNAME} - VMware service user name
  - {$VMWARE.PASSWORD} - VMware service {$VMWARE.USERNAME} user password



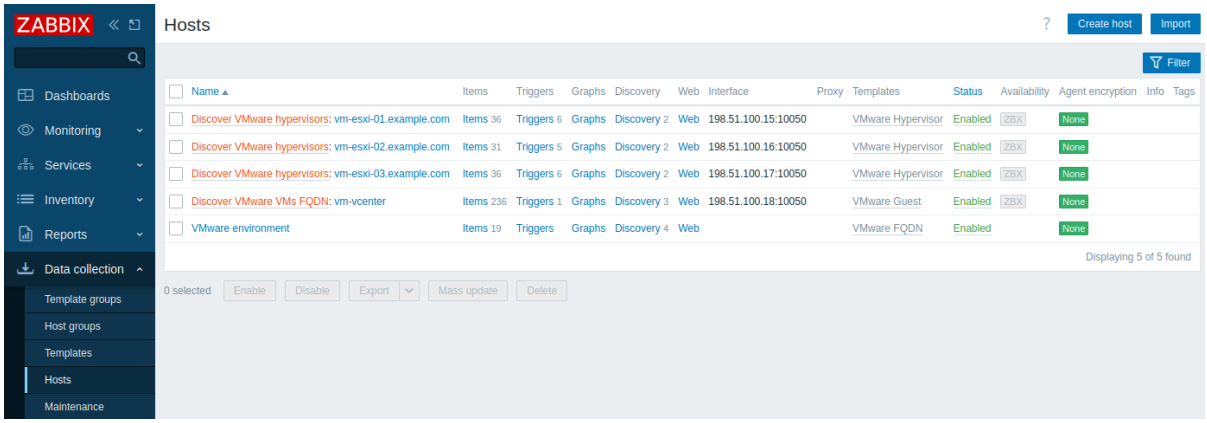
3. Click the *Add* button to create the host. This host will represent your VMware environment.

View collected metrics

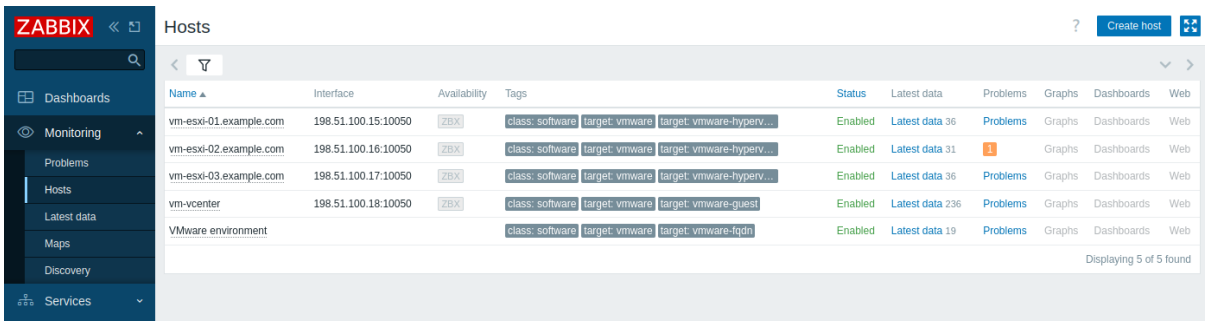
Congratulations! At this point, Zabbix is already monitoring your VMware environment.

Depending on the configuration of your VMware environment, Zabbix may discover and then create hosts for the discovered entities. Note that the discovery and creation of hosts can also be executed manually, if necessary.

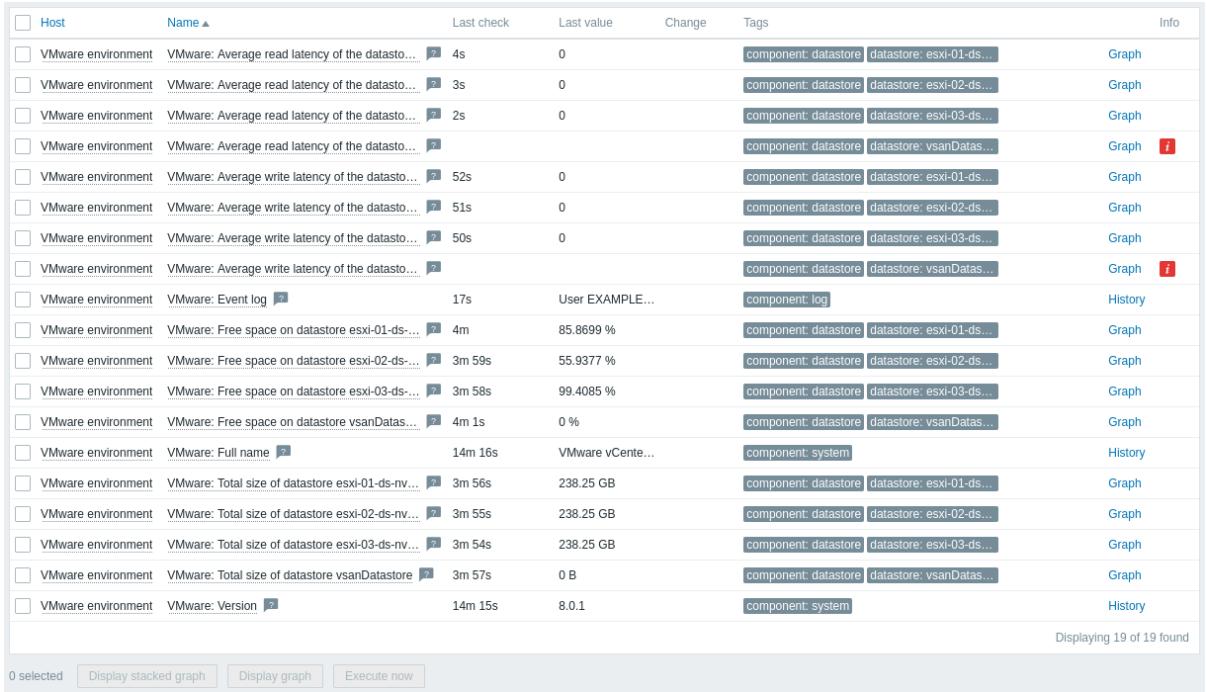
To view created hosts, navigate to the *Data collection → Hosts* menu section.



To view collected metrics, navigate to the *Monitoring → Hosts* menu section and click *Latest data* next to the created "VMware environment" host or one of the hosts that were created for the discovered entities.



This action will open a list of all the latest metrics collected from the selected host.

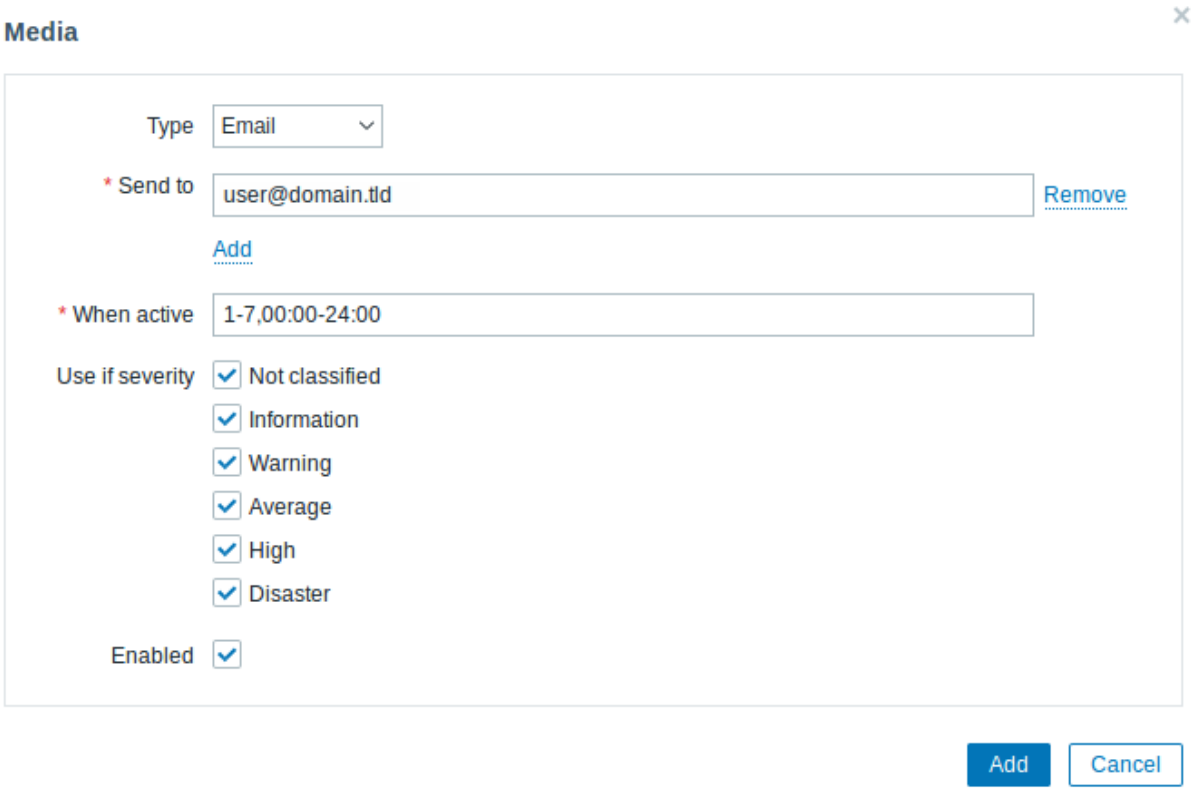


Note that some items have no data and the *Not supported* state. This is because Zabbix cannot find valid performance counters on the specific datastore, as it is not enabled in the VMware environment being monitored.

Set up problem alerts

Zabbix can notify you about a problem with your infrastructure using a variety of methods. This guide provides basic configuration steps for sending email alerts.

1. Navigate to *User settings → Profile*, switch to the *Media* tab and add your email.



2. Follow the guide for Receiving a problem notification.

Next time, when Zabbix detects a problem, you should receive an alert via email.

See also

- Creating an item - how to start monitoring additional metrics.
- Problem escalations - how to create multi-step alert scenarios (e.g., first send message to the system administrator, then, if a problem is not resolved in 45 minutes, send message to the data center manager).
- Virtual machine monitoring - additional information about VMware monitoring (data collection process, server configuration options, troubleshooting guidance, etc.).
- VMware monitoring item keys - a full list of VMware metrics that can be monitored using Zabbix.
- Template *VMware* - additional information about the *VMware* template.
- Template *VMware FQDN* - additional information about the *VMware FQDN* template.

Developer Center

This section contains everything you need to quickly start developing custom Zabbix extensions:

- Frontend modules
- Dashboard widgets
- Plugins for Zabbix agent 2

Copyright notice

Zabbix documentation is NOT distributed under a GPL license. Use of Zabbix documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Zabbix disseminates it (that is, electronically for download on a Zabbix web site) or on a USB or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Zabbix. Zabbix reserves any and all rights to this documentation not expressly granted above.

Modules

What is a PHP frontend module?

- A module is an entity with a unique ID, name, description, author, and other fields defined in it's manifest file, along with PHP, Javascript and other files located in a single directory inside the *modules* directory of your Zabbix frontend installation (for example, *zabbix/ui/modules*).
- A module should conform to simple rules to guarantee correct operation.
- A module must be installed (unpacked) and enabled in the frontend by the administrator.

What a module can be used for

- Adding new functionality via custom frontend sections;
- Creating custom dashboard widget types (see widget modules);
- Overriding or extending the existing functionality.

What a module cannot be used for

- Registering a new API method or modifying an existing one.

How modules work

- An enabled module is launched on each HTTP request, before executing the action code.
- The module will register new actions or redefine the existing ones.
- The module will add new frontend sections and remove or redefine the existing ones.
- The module will hook to frontend events like onBeforeAction and onTerminate, if needed.
- The requested action is finally executed by running the action code - either the default one, or module-defined.

Where to go next Whether you prefer to learn by doing or read the guidelines first, these pages contain the information and steps required to build your own modules:

- Step-by-step tutorials for writing your first module
- Module file structure
- Widget module specifics
- Module examples to reuse

Module file structure

All code related to a module is stored in a single directory inside the **modules** directory of your Zabbix frontend installation (for example, *zabbix/ui/modules*).

```
example_module_directory/          (required)
   manifest.json                   (required)  Metadata and action definition.
   Module.php                                  Module initialization and event handling.
   actions/                                    Action controller files.
       SomethingView.php
       SomethingCreate.php
       SomethingDelete.php
       data_export/
           ExportAsXml.php
           ExportAsExcel.php
   views/                                      View files.
       example.something.view.php
       example.something.delete.php
   assets/                                     Any additional files to be used in views. Must be specified
       js/                                     JavaScript files used in views.
           example.something.view.js.php
       css/                                    CSS files used in views.
           example.something.css
        image.png                              Images used in views.
        example.something.file                 Any file for using in views.
```

Module file tree

Writing a module A sample module writing process consists of the following steps (where available, click on the file or folder name to view additional details about the step):

1. Create a new directory for the module inside **zabbix/ui/modules/**.
2. Add manifest.json file with module metadata.
3. Create *views* folder and define a module view(s).
4. Create *actions* folder and define a module action(s).
5. Create Module.php (or Widget.php for dashboard widgets) file and define initialization and event handling rules.
6. Create *assets* folder for JavaScript files (place into *assets/js*), CSS styles (place into *assets/css*), or any other additional files.
7. Make sure to specify required views, actions and asset files in the manifest.json.
8. Register the module in Zabbix frontend and start using it.

Note:
You can register and enable a module as soon as you create manifest.json file. Once the module is enabled, you can preview all changes made to module files immediately by refreshing Zabbix frontend.

manifest.json

Any module needs the manifest.json file. The file must be located in the module's primary directory (for example, *zabbix/ui/modules/module_name/manifest.json*).

As a bare minimum, manifest.json should specify these fields:

```
{
    "manifest_version": 2.0,
    "id": "my_ip_address",
    "name": "My IP Address",
    "namespace": "MyIPAddress",
    "version": "1.0"
}
```

Parameters supported in **manifest.json** (press on the parameter name for a detailed description):

Parameter	Description	Required
manifest_version	Manifest version of the module.	Yes
id	Unique module ID.	
name	Module name that will be displayed in the Administration section.	
namespace	PHP namespace for module classes.	
version	Module version.	
type	Type of the module. For widget must be set to *widget*	Yes for widgets, otherwise no
widget	Widget configuration. Used for widgets only.	
actions	Actions to register with the module.	
assets	CSS styles and JavaScript files to include.	No
author	Module author.	
config	Default values for custom module options.	
description	Module description.	
url	A link to the module description.	

manifest_version

Manifest version of the module. Currently, supported version is **2.0**.

Type: Double

Example:

```
"manifest_version": 2.0
```

id

Module ID. Must be unique. To avoid future naming conflicts, it is recommended to use prefix for modules (author or company name, or any other). For example, if a module is an example for lessons and the module name is "My module", then the ID will be "example_my_module".

Type: String

Example:

```
"id": "example_my_module"
```

name

Module name that will be displayed in the Administration section.

Type: String

Example:

```
"name": "My module"
```

namespace

PHP namespace for module classes.

Type: String

Example:

```
"namespace": "ClockWidget"
```

version

Module version. The version will be displayed in the Administration section.

Type: String

Example:

```
"version": "1.0"
```

type

Type of the module. Required for widgets and must equal "widget".

Type: String

Default: "module"

Example:

```
"type": "widget"
```

actions

Actions to register with the module. Defining *class* object key for each action is required, other action keys are optional.

Type: Object

Supported object keys if *type* is *module*:

- **write.your.action.name** (object) - action name, should be written in lowercase [a-z], separating words with dots. Supports the keys:
  - **class** (string; required) - action class name.
  - **layout** (string) - action layout. Supported values: *layout.json*, *layout.htmlpage* (default), *null*.
  - **view** (string) - action view.

Example:

```
"actions": {
    "module.example.list": {
        "class": "ExampleList",
        "view": "example.list",
        "layout": "layout.htmlpage"
        }
    }
```

Supported object keys if *type* is *widget*:

- **widget.{id}.view** (object) - file and class name for widget view. Replace **{id}** with the widget's id value (for example, *widget.example_clock.view*). Supports the keys:
  - **class** (string; required) - action class name for widget view mode to extend the default CControllerDashboardWidgetView class. The class source file must be located in the *actions* directory.

    - **view** (string) - widget view. Must be located in the *views* directory. If the view file is *widget.view.php*, which is expected by default, this parameter maybe omitted. If using a different name, specify it here.
- **widget.{id}.edit** (object) - file name for widget configuration view. Replace **{id}** with the widget's id value (for example, *widget.example_clock.edit*). Supports the keys:
    - **class** (string; required) - action class name for widget configuration view mode. The class source file must be located in the *actions* directory.
    - **view** (string) - widget configuration view. Must be located in the *views* directory. If the view file is *widget.edit.php*, which is expected by default, this parameter maybe omitted. If using a different name, specify it here.

Example:

```
"actions": {
    "widget.tophosts.view": {
        "class": "WidgetView"
    },
    "widget.tophosts.column.edit": {
        "class": "ColumnEdit",
        "view": "column.edit",
        "layout": "layout.json"
    }
}
```

assets

CSS styles and JavaScript files to include.

Type: Object

Supported object keys:

- **css** (array) - CSS files to include. The files must be located in the *assets/css*.
- **js** (array) - JavaScript files to include. The files must be located in the *assets/js*.

Example:

```
"assets": {
    "css": ["widget.css"],
    "js": ["class.widget.js"]
}
```

author

Module author. The author will be displayed in the Administration section.

Type: String

Example:

```
"author": "John Smith"
```

config

Default values for the module options. The object may contain any custom keys. If specified, these values will be written into the database during module registration. New variables added later will be written upon the first call. Afterwards, the variable values can only be changed directly in the database.

Type: Object

Example:

```
"config": {
    "username": "Admin",
    "password": "",
    "auth_url": "https://example.com/auth"
}
```

description

Module description.

Type: String

Example:

```
"description": "This is a clock widget."
```

widget

Widget configuration. Used, if *type* is set to *widget*.

Type: Object

Supported object keys:

- **name** (string) - used in the widget list and as default header. If empty, "name" parameter from the module will be used.
- **template_support** (boolean) - determines whether the widget should be available in template dashboards. Supported values: *true*, *false* (default).
- **size** (object) - default widget dimensions. Supports keys:
  - *width* (integer) - default widget width.
  - *height* (integer) - default widget height.
- **form_class** (string) - class with widget fields form. Must be located in the *includes* directory. If the class is *WidgetForm.php*, which is expected by default, this parameter maybe omitted. If using a different name, specify it here.
- **js_class** (string) - name of a JavaScript class for widget view mode to extend the default CWidget class. The class will be loaded with the dashboard. The class source file must be located in the assets/js directory. See also: assets.
- **use_time_selector** (boolean) - determines whether the widget requires dashboard time selector. Supported values: *true*, *false* (default).
- **refresh_rate** (integer) - widget refresh rate in seconds (default: 60).

Example:

```
"widget": {
    "name": "",
    "template_support": true,
    "size": {
        "width": 12,
        "height": 5
    },
    "form_class": "WidgetForm",
    "js_class": "CWidget",
    "use_time_selector": false,
    "refresh_rate": 60
}
```

url

A link to the module description.

Type: String

Example:

```
"url": "http://example.com"
```

Actions

Actions are responsible for 'business logic' of the module. An action usually consists of a controller and an action view.

A module can:

- Call actions that are already defined in Zabbix frontend.
- Override default actions with custom actions.
- Define completely new actions.

To override a default action behavior with some custom behavior, define an action with the same name in the module configuration. When the action is called, the module action will be executed instead of the default Zabbix action.

Action files should be stored in the *actions* folder. The actions need to be specified in the manifest.json.

Controller

Action controller workflow:

1) Check that all parameters passed in an HTTP request are valid:

- Call the controller's *checkInput()* method
- Use validation rules defined in CNewValidator.php
- Call *validateInput()* method

2) Check user permissions.

3) Prepare the data according to passed parameters: if *checkInput()* returns true, Zabbix calls the controller's *doAction()* method.

4) Prepare the **$data** array for the view. Use *CControllerResponseData* and *setResponse()* method to store response in the **$data** array.

Example:

```
/**
 * Validate input parameters.
 *
 * @return bool
 */
protected function checkInput(): bool {
    $ret = $this->validateInput([
        'status' => 'in '.implode(',', [HOST_STATUS_MONITORED, HOST_STATUS_NOT_MONITORED])
    ]);

    if (!$ret) {
        $this->setResponse(new CControllerResponseFatal());
    }

    return $ret;
}

/**
 * Check user permissions.
 *
 * @return bool
 */
protected function checkPermissions() {
    return $this->getUserType() >= USER_TYPE_ZABBIX_ADMIN;
}

/**
 * Execute action and generate response object.
 */
protected function do Action(): void {
    $data = [
        'hosts_count' => API::Host()->get([
            'countOutput' => true,
            'filter' => [
                'status' => $this->getInput('status')
            ]
        ])
    ];

    $this->setResponse(new CControllerResponseData($data));
}
```

Note:
You can view the full list of available controller classes in Zabbix source code.

Views

View file receives the data from a controller and then prepares the HTML look of it.

Note:
Defining view(s) for a frontend module is optional, unless the module is a widget.
Dashboard widgets need at least two views: one for the edit mode and one for the view mode (should be stored in the *views* directory).

It is possible to use pre-defined Zabbix HTML classes (from the */zabbix/ui/include/classes/html*) in the view as well as add new HTML and CSS classes. New classes should be stored in the module's *assets* folder.

Example:

```
...
(new CColHeader(_('Name')))
```

This will add a new column name *Name* and style the top table row as on other Zabbix pages.

Action view

This is a reference file for defining an action view.

```
<?php declare(strict_types = 1);

/**
* @var CView $this
*/

$this->includeJsFile('example.something.view.js.php');

(new CWidget())
    ->setTitle(_('Something view'))
    ->addItem(new CDiv($data['name']))
    ->addItem(new CPartial('module.example.something.reusable', [
        'contacts' => $data['contacts']
    ]))
    ->show();
```

Assets

The folder *assets* may contain any files and subfolders that do not belong to other directories. You can use it for:

- JavaScript styles (must be inside *assets/js*);
- CSS styles (must be inside *assets/css*);
- Images;
- Fonts;
- Anything else you need to include.

assets/js

assets/js directory is reserved and should only contain JavaScript files. To be used by the widget, specify these files in the *manifest.json*.

For example:

```
"assets": {
    "js": ["class.widget.js"]
}
```

assets/css

assets/css is reserved and should only contain CSS style files. To be used by the widget, specify these files in the *manifest.json*.

For example:

```
"assets": {
    "css": ["mywidget.css"]
}
```

CSS styles

CSS files may contain a custom attribute `theme` to define different style for a specific frontend themes.

Available themes and their attribute values:

- **Blue** - [theme='blue-theme']
- **Dark** - [theme='dark-theme']
- **High-contrast light** - [theme='hc-light']
- **High-contrast dark** - [theme='hc-dark']

Example:

```
.widget {
    background-color: red;
}

[theme='dark-theme'] .widget {
    background-color: green;
}
```

Register a new module

This section explains how to add a new module to Zabbix frontend.

Pre-requisites Before proceeding make sure, that:

- The module is located inside the *modules* directory of your Zabbix frontend installation (for example, *zabbix/ui/modules*).
- The module has at least a basic version of manifest.json file.
- You have access to the Administration menu section in Zabbix (requires Super admin user role type).

> **Note:**
> The frontend will not install or even recognize incompatible modules.

Adding a module Open *Administration→General→Modules* page and press *Scan directory*.



Locate your module in the list and activate it.

To activate a module, press on the *Disabled* hyperlink - the module's state will change to *Enabled*.

Press on the module name to view additional information about the module, such as author, version, or short description (if defined in the manifest).

Widget preview Widget modules, once added, become immediately visible in the dashboard widget list.

You can open a dashboard, switch to the edit mode and add the widget to the dashboard as usual.

When you make some changes to the widget, refresh the dashboard to view how the widget looks with the most recent updates.

Widgets

Widgets are Zabbix frontend modules used for the dashboards. Unless otherwise noted, all module guidelines are also applicable to widgets.

However, a widget is notably different from a module. To build a widget:

- specify the type "widget" in the manifest.json file ("type": "widget");
- include at least two views: one for the widget presentation mode and one for the widget configuration mode (example.widget.view.php and example.widget.edit.php);
- and a controller for widget presentation (WidgetView.php);
- use and extend default widget classes.

Configuration

This page describes classes that can be used to create a widget configuration view with custom configuration fields. The widget configuration view is the part of the widget that allows the user to configure widget parameters for presentation.

Widget

Primary widget class, extends the base class of all dashboard widgets - *CWidget*. Required for overriding the default widget behavior.

The *Widget* class should be located in the root directory of the widget (for example, *zabbix/ui/modules/my_custom_widget*).

Widget.php example

```
<?php

namespace Modules\MyCustomWidget;

use Zabbix\Core\CWidget;

class Widget extends CWidget {

    public const MY_CONSTANT = 0;

    public function getTranslationStrings(): array {
        return [
            'class.widget.js' => [
                'No data' => _('No data')
            ]
        ];
    }
}
```

WidgetForm

The *WidgetForm* class extends the default class *CWidgetForm* and contains a set of *CWidgetField* fields that are required for defining widget configuration storage structure in the database and handling input validation.

The *WidgetForm* class should be located in the *includes* directory. If the class has a different name, the name should be specified in the *widget/form_class* parameter in the *manifest.json* file.

includes/WidgetForm.php example

```
<?php

namespace Modules\MyCustomWidget\Includes;

use Modules\MyCustomWidget\Widget;

use Zabbix\Widgets\{
    CWidgetField,
    CWidgetForm
};

use Zabbix\Widgets\Fields\{
    CWidgetFieldMultiSelectItem,
    CWidgetFieldTextBox,
    CWidgetFieldColor
};

class WidgetForm extends CWidgetForm {

    public function addFields(): self {
        return $this
            ->addField(
                (new CWidgetFieldMultiSelectItem('itemid', _('Item')))
                    ->setFlags(CWidgetField::FLAG_NOT_EMPTY | CWidgetField::FLAG_LABEL_ASTERISK)
```

```
                    ->setMultiple(false)
                    ->setFilterParameter('numeric', true)
            )
            ->addField(
                new CWidgetFieldTextBox('description', _('Description'))
            )
            ->addField(
                (new CWidgetFieldColor('chart_color', _('Color')))->setDefault('FF0000')
            );
    }
}
```

CWidgetFormView

The *CWidgetFormView* class is required for specifying the presentation logic of the fields defined in the *WidgetForm* class, determining their appearance and behavior when rendered in the configuration view.

The *CWidgetFormView* class supports the following methods:

- *addField()* - receives an instance of the *CWidgetFieldView* class as a parameter; each *CWidgetField* class, has a respective *CWidgetFieldView* class for using in the widget configuration view.
- *includeJsFile()* - allows to add a JavaScript file to the widget configuration view.
- *addJavaScript()* - allows to add inline JavaScript that will be executed as soon as the widget configuration view is loaded.

The *CWidgetFormView* class should be located in the *views* directory.

views/widget.edit.php example

```
<?php

/**
 * My custom widget form view.
 *
 * @var CView $this
 * @var array $data
 */

use Zabbix\Widgets\Fields\CWidgetFieldGraphDataSet;

(new CWidgetFormView($data))
    ->addField(
        new CWidgetFieldMultiSelectItemView($data['fields']['itemid'], $data['captions']['items']['itemid'
    )
    ->addField(
        new CWidgetFieldTextBoxView($data['fields']['description']),
        'js-advanced-configuration'
    )
    ->addField(
        new CWidgetFieldColorView($data['fields']['chart_color']),
        'js-advanced-configuration'
    )
    ->includeJsFile('widget.edit.js.php')
    ->addJavaScript('my_widget_form.init('.json_encode([
        'color_palette' => CWidgetFieldGraphDataSet::DEFAULT_COLOR_PALETTE
    ], JSON_THROW_ON_ERROR).');')
    ->show();
```

JavaScript

A JavaScript class can be used to add dynamic behavior and interactivity to the widget configuration view. For example, you can initialize a color picker, defined in the *CWidgetFormView* class.

The JavaScript class should be loaded with the form, therefore it should be referenced in the *CWidgetFormView* class by using the methods *includeJsFile()* and *addJavaScript()*.

In the example below, a singleton class instance is immediately created and stored under the *window.my_custom_widget_form* name. Thus, opening the form for the second time will re-create the instance.

The JavaScript class should be located in the *views* directory.

views/widget.edit.js.php example

```
<?php

use Modules\LessonGaugeChart\Widget;

?>

window.widget_lesson_gauge_chart_form = new class {

    init({color_palette}) {
        this._advanced_configuration = document.getElementById('adv_conf');

        this._advanced_configuration.addEventListener('change', () => this.updateForm());

        colorPalette.setThemeColors(color_palette);

        for (const colorpicker of jQuery('.<?= ZBX_STYLE_COLOR_PICKER ?> input')) {
            jQuery(colorpicker).colorpicker();
        }

        const overlay = overlays_stack.getById('widget_properties');

        for (const event of ['overlay.reload', 'overlay.close']) {
            overlay.$dialogue[0].addEventListener(event, () => { jQuery.colorpicker('hide'); });
        }

        this.updateForm();
    }

    updateForm() {
        const show_advanced_configuration = this._advanced_configuration.checked;

        for (const element of this._form.querySelectorAll('.js-advanced-configuration')) {
            element.style.display = show_advanced_configuration ? '' : 'none';
        }
    }
};
```

CWidgetField

The *CWidgetField* class is a base class from which all form field classes (*CWidgetFieldCheckBox*, *CWidgetFieldTextArea*, *CWidgetFieldRadioButtonList*, etc.) are inherited. Classes extending *CWidgetField* are responsible for receiving, saving, and validating widget configuration values.

The following *CWidgetField* classes are available.

CWidgetField class	Database field type	Description
CWidgetFieldCheckBox	int32	Single checkbox.
CWidgetFieldCheckBoxList	array of int32	Multiple checkboxes under a single configuration field.
CWidgetFieldColor	string	Color selection field.
CWidgetFieldDatePicker	string	Date selection field.
CWidgetFieldHostPatternSelect	string	Multiselect field that allows to select one or multiple hosts. Supports defining host name patterns (all matching hosts will be selected).
CWidgetFieldIntegerBox	int32	Field to enter an integer. Can be used to configure minimum and maximum values.
CWidgetFieldLatLng	string	Text box that allows to enter comma-separated latitude, longitude, and map zoom level.
CWidgetFieldMultiSelectAction	ID	Multiselect field for selecting actions (from the list of actions defined in the *Alerts → Actions*).
CWidgetFieldMultiSelectGraph	ID	Multiselect field for selecting custom graphs.

CWidgetField class	Database field type	Description
CWidgetFieldMultiSelectGraphPrototype	ID	Multiselect field for selecting custom graph prototypes.
CWidgetFieldMultiSelectGroup	ID	Multiselect field for selecting host groups.
CWidgetFieldMultiSelectHost	ID	Multiselect field for selecting hosts.
CWidgetFieldMultiSelectItem	ID	Multiselect field for selecting items.
CWidgetFieldMultiSelectItemPrototype	ID	Multiselect field for selecting item prototypes.
CWidgetFieldMultiSelectMediaType	ID	Multiselect field for selecting media types.
CWidgetFieldMultiSelectService	ID	Multiselect field for selecting services.
CWidgetFieldMultiSelectSla	ID	Multiselect field for selecting SLAs.
CWidgetFieldMultiSelectUser	ID	Multiselect field for selecting users.
CWidgetFieldNumericBox	string	Field to enter a float number.
CWidgetFieldRadioButtonList	int32	Radio box group that consists of one or more radio boxes.
CWidgetFieldRangeControl	int32	Slider to select an integer type value.
CWidgetFieldSelect	int32	Dropdown select box.
CWidgetFieldSeverities	array of int32	*CWidgetFieldCheckBoxList* preset with trigger severities.
CWidgetFieldTags	array of (string, int32, string)	Allows to configure one or more tag filter rows.
CWidgetFieldTextArea	string	Text area for entering multi-line text.
CWidgetFieldTextBox	string	Text box for entering single-line text.
CWidgetFieldTimeZone	string	Dropdown with timezones.
CWidgetFieldUrl	string	Text box that allows to enter URLs.

The following *CWidgetField* classes have been created for particular widgets. These classes have very specific use cases, but they can also be reused if needed.

CWidgetField class	Database field type	Description
CWidgetFieldColumnsList	array of (multiple mixed)	For *Top hosts* widget. Create a table with custom columns of allowed types.
CWidgetFieldGraphDataSet	array of (multiple mixed)	For *Graph* widget. Setup dataset configuration and all related options.
CWidgetFieldGraphOverride	array of (multiple mixed)	For *Graph* widget. Setup overrides for specific hosts/items. Any dataset configuration can be overridden.
CWidgetFieldNavTree	string	For *Map navigation tree* widget. Replaces widget view in edit mode with the map selection tree.
CWidgetFieldReference	string	For *Map navigation tree* widget. Creates a unique identifier for this widget on dashboard. It is used to reference this widget from other widgets.
CWidgetFieldSelectResource	ID	For *Map* widget. Allows to select Zabbix network map.
CWidgetFieldThresholds	array of (string, string)	For *Top hosts* widget. Allows to configure color and number pairs.
CWidgetFieldWidgetSelect	string	For *Map* widget. Allows to select a map navigation tree from the current dashboard. Must be used in combination with *CWidgetFieldReference* in the *Map navigation tree* widget.

Presentation

This page describes the components that can be used to create a widget presentation view. The widget presentation view is the part of the widget that receives the data according to its configuration and displays it on the dashboard in a container.

The presentation view consists of three parts:

- Widget action
- Widget view
- JavaScript

Widget action

The widget action class (*WidgetView*) contains methods for operations with widgets in the presentation view mode. The majority of widget actions use and/or extend the default controller class *CControllerDashboardWidgetView*.

The widget action class should be located in the *actions* directory and specified in the *actions* parameter (*actions/widget.{id}.view/class*) in the *manifest.json* file.

actions/WidgetView.php example (implemented in the Zabbix-native *System information* widget)

```
class WidgetView extends CControllerDashboardWidgetView {

    protected function doAction(): void {
        $this->setResponse(new CControllerResponseData([
            'name' => $this->getInput('name', $this->widget->getDefaultName()),
            'system_info' => CSystemInfoHelper::getData(),
            'info_type' => $this->fields_values['info_type'],
            'user_type' => CWebUser::getType(),
            'user' => [
                'debug_mode' => $this->getDebugMode()
            ]
        ]));
    }
}
```

Widget view

The widget view class (*CWidgetView*) is responsible for building the widget presentation view.

The widget view class should be located in the *views* directory. If the file containing the widget view class has a different name than the default (*widget.view.php*), then it must be specified in the *manifest.json* file *actions* parameter (*actions/widget.{id}.view/view*).

views/widget.view.php example

```
<?php

/**
 * My custom widget view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetView($data))
    ->addItem(
        new CTag('h1', true, $data['name'])
    )
    ->show();
```

JavaScript

The JavaScript class is responsible for determining widget behavior, such as updating widget data, resizing the widget, displaying widget elements, etc.

All JavaScript operations use and/or extend the base JavaScript class of all dashboard widgets - *CWidget*. The *CWidget* class contains a set of methods with the default implementation for widget behavior. Depending on widget complexity, these methods can be utilized as is or extended.

The *CWidget* class contains the following methods:

- Methods that define widget lifecycle: *_init()*, *_registerEvents()*, *_doActivate()*, *_doDeactivate()*, *_doDestroy()*, *setEditMode()*.
- Methods that handle updating and displaying widget data: *_promiseUpdate()*, *_getUpdateRequestData()*, *_processUpdateResponse(response)*, *_processUpdateErrorResponse(error)*, *_setContents()*.
- Methods that modify widget appearance: *resize()*, *_hasPadding()*.

The JavaScript class should be located in the *assets/js* directory and specified in the *assets* (*assets/js*) parameter in the *manifest.json* file.

Lifecycle methods

The widget lifecycle methods are invoked by the dashboard, and at different stages of the widget's lifecycle during its existence within the dashboard.

The ***_init()*** method defines the initial state and/or values of the widget, without performing any HTML or data manipulation. This method is invoked when a widget is created (a widget object is instantiated), typically by adding the widget to a dashboard page or loading the dashboard page.

Example:

```
_init() {
    super._init();

    this._time_offset = 0;
    this._interval_id = null;
    this._clock_type = CWidgetClock.TYPE_ANALOG;
    this._time_zone = null;
    this._show_seconds = true;
    this._time_format = 0;
    this._tzone_format = 0;
    this._show = [];
    this._has_contents = false;
    this._is_enabled = true;
}
```

The ***_registerEvents()*** method defines the HTML structure of the widget, without performing any data manipulation. This method is invoked before the first activation of the dashboard page, that is, before the dashboard and its widgets are fully displayed to the user.

Example:

```
_registerEvents() {
    super._registerEvents();

    this._events.resize = () => {
        const padding = 25;
        const header_height = this._view_mode == ZBX_WIDGET_VIEW_MODE_HIDDEN_HEADER
            ? 0
            : this._content_header.offsetHeight;

        this._target.style.setProperty(
            '--content-height',
            `${this._cell_height * this._pos.height - padding * 2 - header_height}px`
        );
    }
}
```

The ***_doActivate()*** method makes the widget active and interactive by enabling custom event listeners (for responding to user actions) and initiating the widget update cycle (for keeping its content up-to-date). This method is invoked when the dashboard page is activated, that is, when it becomes fully displayed in the user interface.

Note that before the *_doActivate()* method is invoked, the widget is in the inactive state (`WIDGET_STATE_INACTIVE`). After successful invocation, the widget transitions to the active state (`WIDGET_STATE_ACTIVE`). In the active state, the widget is responsive, listens to events, updates its content periodically, and can interact with other widgets.

Example:

```
_doActivate() {
    super._doActivate();

    if (this._has_contents) {
        this._activateContentsEvents();
    }
}
```

The ***_doDeactivate()*** method stops any activity and interactivity of the widget by deactivating custom event listeners and stopping the widget update cycle. This method is invoked when the dashboard page is deactivated, that is, switched away or deleted, or when the widget is deleted from the dashboard page.

Note that before the *_doDeactivate()* method is invoked, the widget is in the active state (`WIDGET_STATE_ACTIVE`). After successful invocation, the widget transitions to the inactive state (`WIDGET_STATE_INACTIVE`).

Example:

```
_doDeactivate() {
    super._doDeactivate();

    this._deactivateContentsEvents();
}
```

The ***_doDestroy()*** method performs cleanup tasks before the widget is deleted from the dashboard, which can include closing a database connection that was established during widget initialization, cleaning up temporary data to free up system memory and avoid resource leaks, unregistering event listeners related to resize events or button clicks to prevent unnecessary event handling and memory leaks, etc. This method is invoked when the widget or the dashboard page that contains it is deleted.

Note that before the *_doDestroy()* method is invoked, a widget in an active state (`WIDGET_STATE_ACTIVE`) is always deactivated with the invocation of the *_doDeactivate()* method.

Example:

```
_doDestroy() {
    super._doDestroy();

    if (this._filter_widget) {
        this._filter_widget.off(CWidgetMap.WIDGET_NAVTREE_EVENT_MARK, this._events.mark);
        this._filter_widget.off(CWidgetMap.WIDGET_NAVTREE_EVENT_SELECT, this._events.select);
    }
}
```

The ***setEditMode()*** method defines the appearance and behavior of the widget when the dashboard transitions into editing mode. This method is invoked when the dashboard transitions into editing mode, typically when a user interacts with the widget's *Edit* button or the dashboard's *Edit dashboard* button.

Example:

```
setEditMode() {
    if (this._has_contents) {
        this._deactivateContentsEvents();
        this._removeTree();
    }

    super.setEditMode();

    if (this._has_contents && this._state === WIDGET_STATE_ACTIVE) {
        this._makeTree();
        this._activateTree();
        this._activateContentsEvents();
    }
}
```

Update process methods

The widget update process methods are responsible for retrieving updated data from Zabbix server or any other data source and displaying it in the widget.

The ***_promiseUpdate()*** method initiates the data update process by retrieving data, typically using web requests or API calls. This method is invoked when a dashboard page is displayed and periodically after, until the dashboard page is switched to another dashboard page.

The following is an example of the default implementation of the *_promiseUpdate()* method used by most Zabbix-native widgets. In the default implementation, the *_promiseUpdate()* method follows a general pattern for retrieving data from the server. It creates a new `Curl` object with the appropriate URL and request parameters, sends a `POST` request using the *fetch()* method with the data object constructed by the *_getUpdateRequestData()* method, and processes the response (or an error response) with the *_processUpdateResponse(response)* or *_processUpdateErrorResponse(error)* accordingly. This implementation is suitable for most widgets as they typically retrieve data in a JSON format and handle it in a consistent manner.

```
_promiseUpdate() {
    const curl = new Curl('zabbix.php');
```

```
    curl.setArgument('action', `widget.${this._type}.view`);

    return fetch(curl.getUrl(), {
        method: 'POST',
        headers: {'Content-Type': 'application/json'},
        body: JSON.stringify(this._getUpdateRequestData()),
        signal: this._update_abort_controller.signal
    })
        .then((response) => response.json())
        .then((response) => {
            if ('error' in response) {
                this._processUpdateErrorResponse(response.error);

                return;
            }

            this._processUpdateResponse(response);
        });
}
```

The ***_getUpdateRequestData()*** method prepares the server request data for updating the widget by gathering various properties and their corresponding values (widget identifiers, filter settings, time ranges, etc.) from the widget's state and configuration, and constructing a data object that represents the necessary information to be sent to the server in the update request. This method is invoked only as part of the default *_promiseUpdate()* method, that is, during the widget update process.

Default implementation:

```
_getUpdateRequestData() {
    return {
        templateid: this._dashboard.templateid ?? undefined,
        dashboardid: this._dashboard.dashboardid ?? undefined,
        widgetid: this._widgetid ?? undefined,
        name: this._name !== '' ? this._name : undefined,
        fields: Object.keys(this._fields).length > 0 ? this._fields : undefined,
        view_mode: this._view_mode,
        edit_mode: this._is_edit_mode ? 1 : 0,
        dynamic_hostid: this._dashboard.templateid !== null || this.supportsDynamicHosts()
            ? (this._dynamic_hostid ?? undefined)
            : undefined,
        ...this._content_size
    };
}
```

The ***_processUpdateResponse(response)*** method handles the response received from the server after the update request, and, if the update process has been successful and without errors, clears widget data and displays new contents with the *_setContents()* method. This method is invoked only as part of the default *_promiseUpdate()* method, that is, during the widget update process.

Default implementation:

```
_processUpdateResponse(response) {
    this._setContents({
        name: response.name,
        body: response.body,
        messages: response.messages,
        info: response.info,
        debug: response.debug
    });
}
```

The ***_processUpdateErrorResponse(error)*** method handles the response received from the server after the update request if the response is an error, and displays the error message/-s. This method is invoked only as part of the default *_promiseUpdate()* method, that is, during the widget update process.

Default implementation:

```
_processUpdateErrorResponse(error) {
    this._setErrorContents({error});
}

_setErrorContents({error}) {
    const message_box = makeMessageBox('bad', error.messages, error.title)[0];

    this._content_body.innerHTML = '';
    this._content_body.appendChild(message_box);

    this._removeInfoButtons();
}
```

The ***_setContents()*** method displays widget contents if the widget update process has been successful and without errors, which can include manipulating DOM elements, updating UI components, applying styles or formatting, etc. This method is invoked only as part of the default *_processUpdateResponse(response)* method, that is, during the process of handling the response received from the server after the update request.

Default implementation:

```
_setContents({name, body, messages, info, debug}) {
    this._setHeaderName(name);

    this._content_body.innerHTML = '';

    if (messages !== undefined) {
        const message_box = makeMessageBox('bad', messages)[0];

        this._content_body.appendChild(message_box);
    }

    if (body !== undefined) {
        this._content_body.insertAdjacentHTML('beforeend', body);
    }

    if (debug !== undefined) {
        this._content_body.insertAdjacentHTML('beforeend', debug);
    }

    this._removeInfoButtons();

    if (info !== undefined) {
        this._addInfoButtons(info);
    }
}
```

Presentation modification methods

The widget presentation modification methods are responsible for modifying widget appearance.

The ***resize()*** method is responsible for adjusting widget's visual elements to accommodate the new widget size, which can include rearranging elements, adjusting element dimensions, text truncation, implementing lazy loading to improve responsiveness during resizing, etc. This method is invoked when the widget is resized, for example, when the user manually resizes the widget or when the browser window is resized.

Example:

```
resize() {
    if (this._state === WIDGET_STATE_ACTIVE) {
        this._startUpdating();
    }
}
```

The ***_hasPadding()*** method is responsible for applying an 8px vertical padding at the bottom of the widget when it is configured to show its header. This method is invoked when the dashboard page is activated, that is, when it becomes the displayed page in the user interface.

Default implementation:

```
_hasPadding() {
    return this._view_mode !== ZBX_WIDGET_VIEW_MODE_HIDDEN_HEADER;
}
```

For some widgets it is necessary to use all of the available widget space to configure, for example, a custom background color. The following is an example of the implementation of the *_hasPadding()* method used in the Zabbix-native *Item value* widget.

```
_hasPadding() {
    return false;
}
```

Tutorials

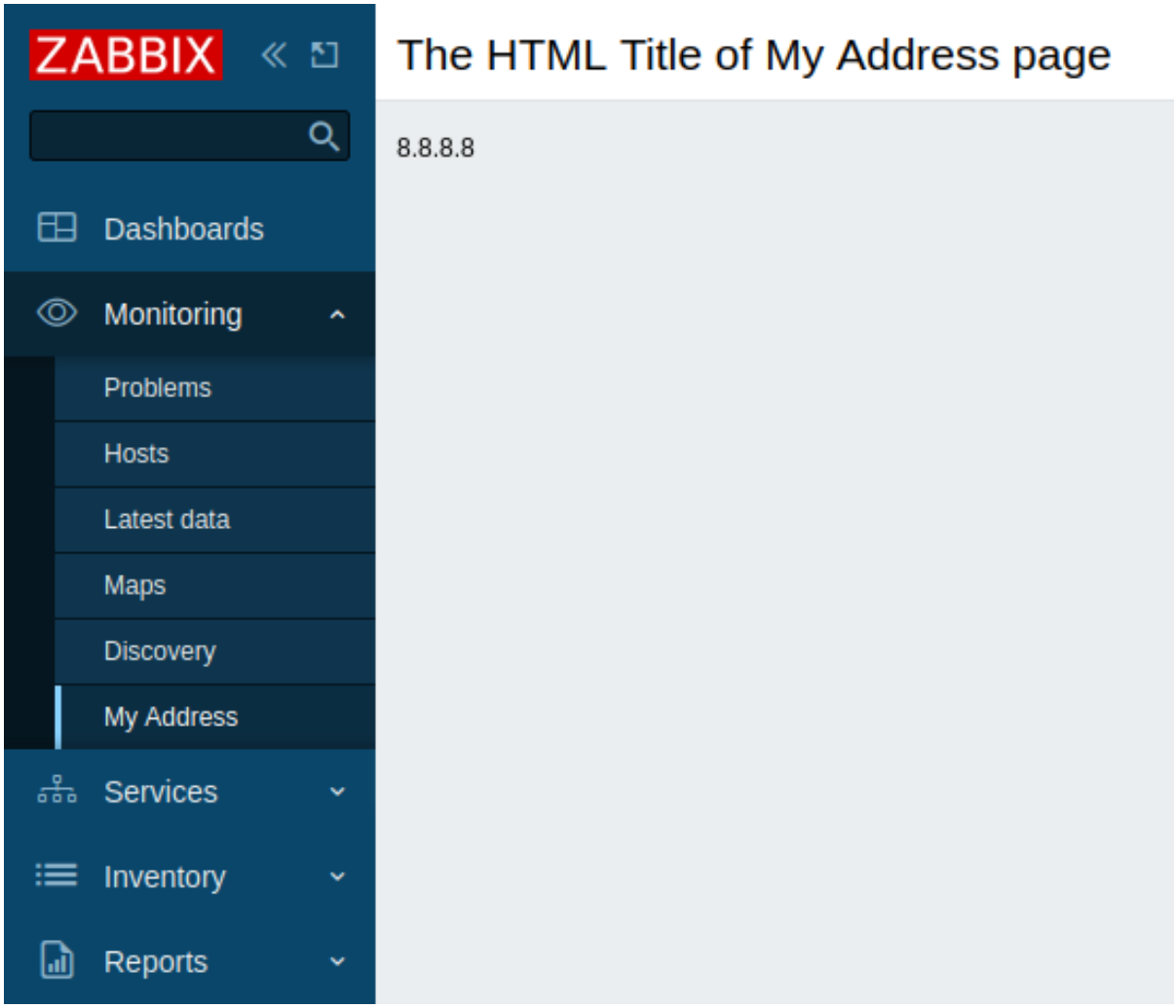
This section contains practical step-by-step tutorials to illustrate how to build a custom module and a widget in Zabbix.

Create a module (tutorial)

This is a step-by-step tutorial that shows how to create a simple Zabbix frontend module. You can download all files of this module as a ZIP archive: MyAddress.zip.

What you'll build

During this tutorial, you will first build a frontend module that adds a new *My Address* menu section and then convert it into a more advanced frontend module that makes an HTTP request to *https://api.seeip.org* and displays the response - the IP address of your computer - on a new page in the newly created *My Address* menu section. Here's how the finished module will look like:



Part I - New menu section

Add a blank module to Zabbix frontend

1. Create a directory *MyAddress* in the *modules* directory of your Zabbix frontend installation (for example, *zabbix/ui/modules*).
2. Create a *manifest.json* file with basic module metadata (see the description of supported parameters).

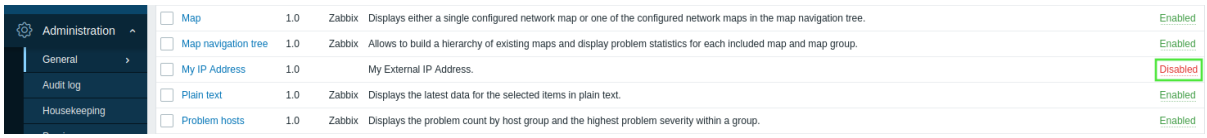
ui/modules/MyAddress/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "my-address",
    "name": "My IP Address",
    "version": "1.0",
    "namespace": "MyAddress",
    "description": "My External IP Address"
}
```

3. In Zabbix frontend, go to *Administration* → *General* → *Modules* section and click on the *Scan directory* button.



4. Locate the new module *My IP Address* in the list and click on the "Disabled" hyperlink to change the module's status from "Disabled" to "Enabled".



The module is now registered in the frontend. However, it is not visible yet, because you still need to define the module functionality. Once you add content to the module directory, you will immediately see the changes in Zabbix frontend upon refreshing the page.

Create a menu section

1. Create a *Module.php* file in the *MyAddress* directory.

This file implements a new *Module* class that extends the default *CModule* class. The *Module* class will insert a new *My Address* menu section into the main menu.

The *setAction()* method specifies an action to be executed upon clicking on the menu section. To start with, you can use the predefined action *userprofile.edit*, which will open the *User profile* page. In part III of this tutorial, you will learn how to create a custom action.

ui/modules/MyAddress/Module.php

```
<?php

namespace Modules\MyAddress;

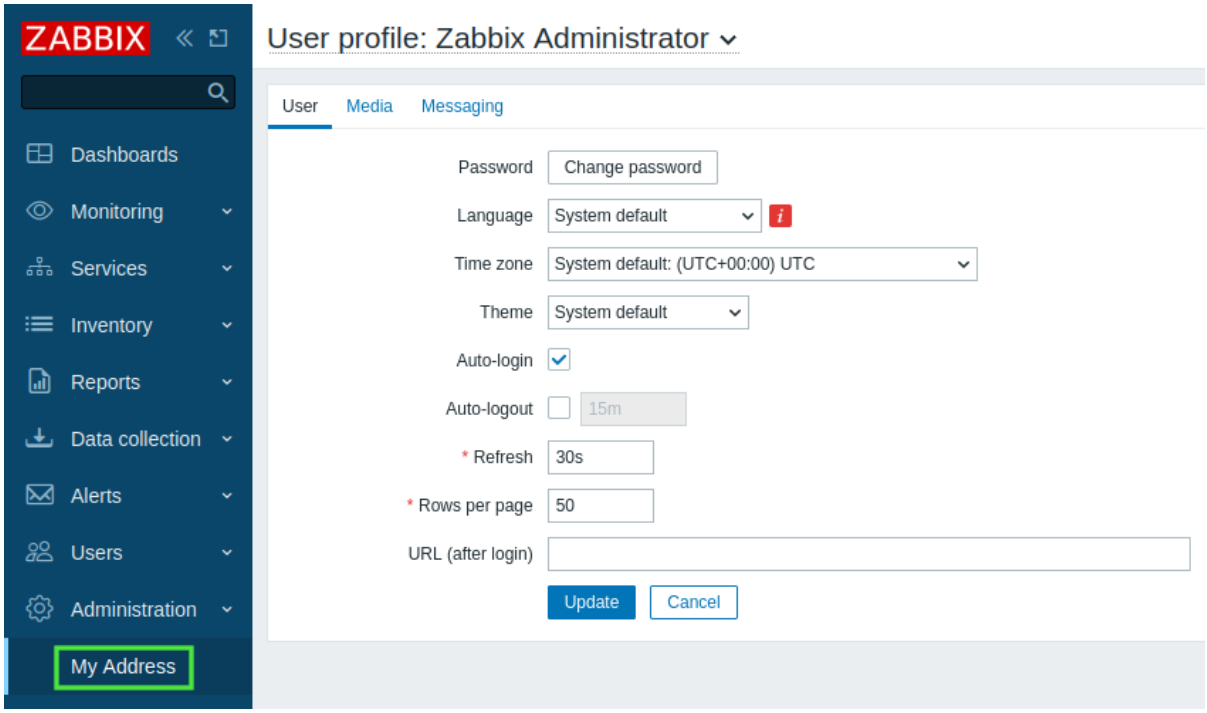
use Zabbix\Core\CModule,
    APP,
    CMenuItem;

class Module extends CModule {

    public function init(): void {
        APP::Component()->get('menu.main')
            ->add((new CMenuItem(_('My Address')))
            ->setAction('userprofile.edit'));
    }
}
```

Note:
You can replace `'userprofile.edit'` with other actions, for example, `'charts.view'` (opens custom graphs), `'problems.view'` (opens *Monitoring* → *Problems*), or `'report.status'` (opens *System information* report).

3. Refresh Zabbix frontend. There is now a new *My Address* section at the bottom of the Zabbix main menu. Click on *My Address* to open the *User profile* page.



Part II - Menu section location change

In this part, you will move the *My Address* menu section to the *Monitoring* section. As a result, users will be able to access two sub-menu pages from the *Monitoring* → *My Address* menu section.

1. Open and edit the *Module.php* file.

ui/modules/MyAddress/Module.php

```
<?php

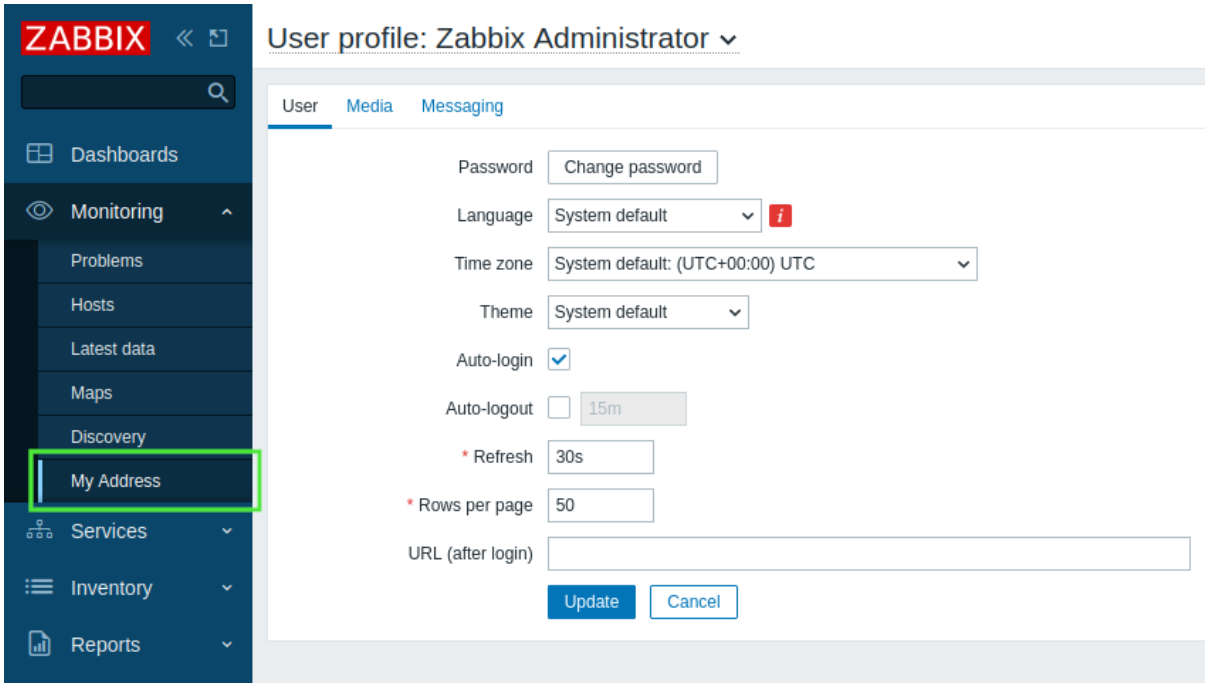
namespace Modules\MyAddress;

use Zabbix\Core\CModule,
    APP,
    CMenuItem;

class Module extends CModule {

    public function init(): void {
        APP::Component()->get('menu.main')
            ->findOrAdd(_('Monitoring'))
                ->getSubmenu()
                    ->insertAfter(_('Discovery'),
                        (new CMenuItem(_('My Address')))->setAction('userprofile.edit')
                    );
    }
}
```

2. Refresh Zabbix frontend. Expand the *Monitoring* menu section and observe that the *My address* section is now located below the *Discovery* section.



Part III - Module action

An action is implemented in two files - *actions/MyAddress.php* and *views/my.address.php*. The ***actions/MyAddress.php*** file takes care of the business logic implementation, while the ***views/my.address.php*** file is responsible for the view.

1. Create a directory *actions* in the *MyAddress* directory.
2. Create a *MyAddress.php* file in the *actions* directory.

The action logic will be defined in the *MyAddress* class. This action class will implement four functions: *init()*, *checkInput()*, *checkPermissions()*, and *doAction()*. Zabbix frontend calls the *doAction()* function when the action is requested. This function is responsible for the business logic of the module.

Attention:
The data must be organized as an associative array. The array can be multidimensional and may contain any data expected by the view.

ui/modules/MyAddress/actions/MyAddress.php

```
<?php

namespace Modules\MyAddress\Actions;

use CController,
    CControllerResponseData;

class MyAddress extends CController {

    public function init(): void {
        $this->disableCsrfValidation();
    }

    protected function checkInput(): bool {
        return true;
    }

    protected function checkPermissions(): bool {
        return true;
    }

    protected function doAction(): void {
        $data = ['my-ip' => file_get_contents("https://api.seeip.org")];
        $response = new CControllerResponseData($data);
```

```
        $this->setResponse($response);
    }
}
```

3. Create a new directory *views* in the *MyAddress* directory.
4. Create a *my.address.php* file in the *views* directory and define the module view.

Note that the variable `$data` is available in the view without specifically defining it. The framework automatically passes the associative array to the view.

ui/modules/MyAddress/views/my.address.php

```
<?php

(new CHtmlPage())
    ->setTitle(_('The HTML Title of My Address Page'))
    ->addItem(new CDiv($data['my-ip']))
    ->show();
```

5. The module action has to be registered in the *manifest.json* file. Open *manifest.json* and add a new object `actions` that contains:

- the action key with the action name written in lowercase (a-z) and with words separated by dots (for example, `my.address`);
- the action class name (`MyAddress`) as a value for the `class` key of the `my.address` object;
- the action view name (`my.address`) as a value for the `view` key of the `my.address` object.

ui/modules/MyAddress/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "my-address",
    "name": "My IP Address",
    "version": "1.0",
    "namespace": "MyAddress",
    "description": "My External IP Address",
    "actions": {
        "my.address": {
            "class": "MyAddress",
            "view": "my.address"
        }
    }
}
```

6. Open *Module.php* and change the action name in the **setAction()** method to *my.address*.

ui/modules/MyAddress/Module.php

```
<?php

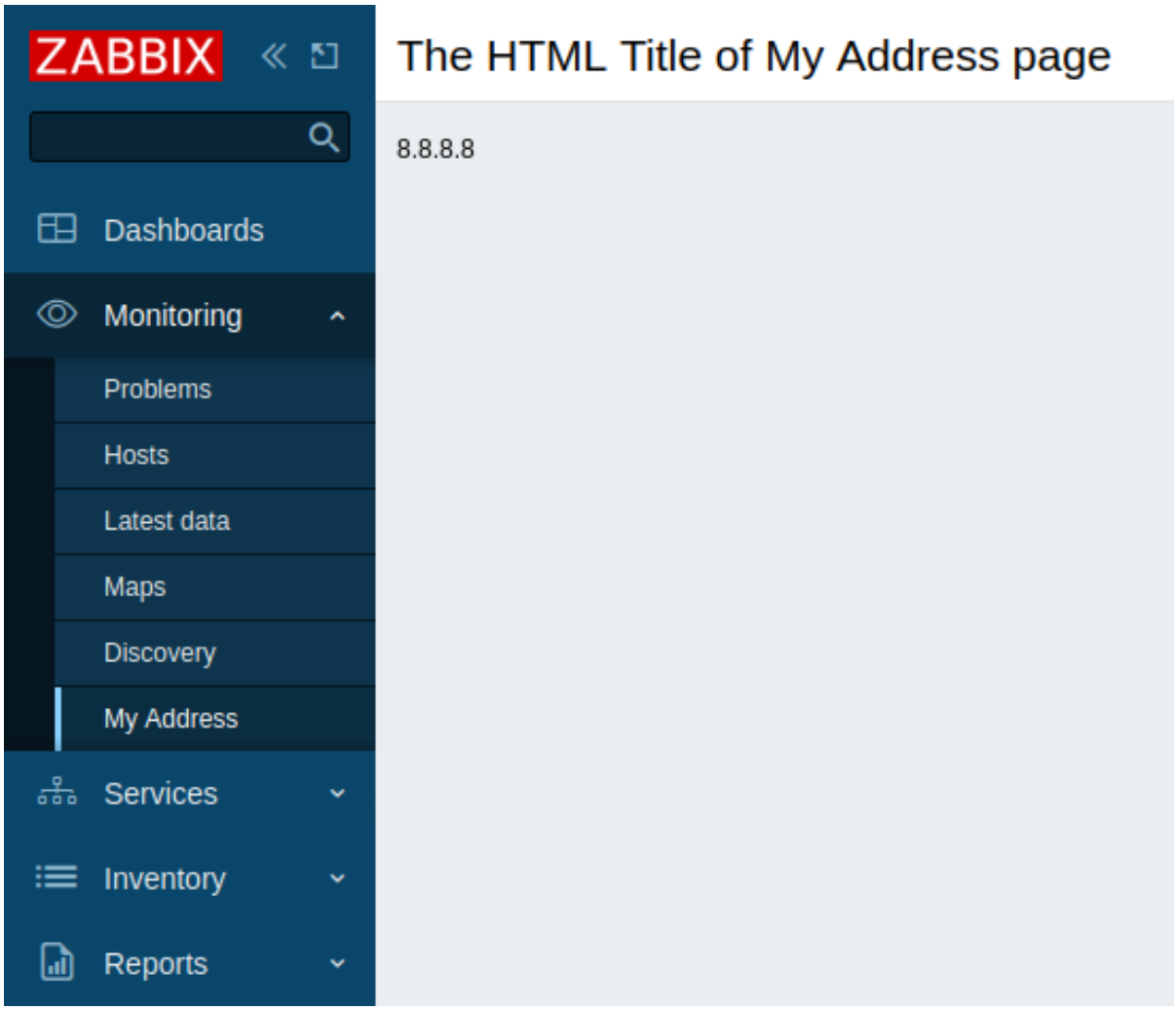
namespace Modules\MyAddress;

use Zabbix\Core\CModule,
    APP,
    CMenuItem;

class Module extends CModule {

    public function init(): void {
        APP::Component()->get('menu.main')
            ->findOrAdd(_('Monitoring'))
                ->getSubmenu()
                    ->insertAfter(_('Discovery'),
                        (new CMenuItem(_('My Address')))->setAction('my.address')
                    );
    }
}
```

7. Refresh Zabbix frontend. Click on the *My address* menu section to see the IP address of your computer.



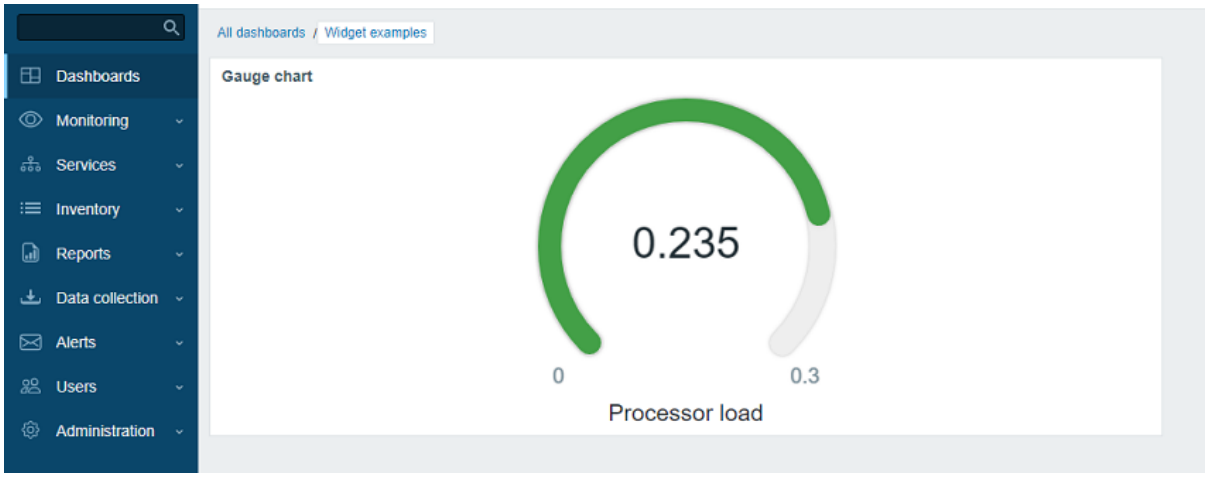
Create a widget (tutorial)

This is a step-by-step tutorial that shows how to create a simple dashboard widget.

> **Attention:**
> The minimum required Zabbix version for this tutorial is 6.4.4.

You can download all files of this widget as a ZIP archive: lesson_gauge_chart.zip.

What you'll build During this tutorial, you will first build a basic "Hello, world!" widget and then convert it into a more advanced widget that displays an item value as a gauge chart. Here's how the finished widget will look like:



Part I - "Hello, world!" In this section you will learn how to create the minimum required widget elements and add a new widget to Zabbix frontend.

Add a blank widget to Zabbix frontend

1. Create a directory *lesson_gauge_chart* in the *modules* directory of your Zabbix frontend installation (for example, *zabbix/ui/modules*).

> **Note:**
> All custom widgets are treated as external modules and must be added to the *modules* directory of your Zabbix frontend installation (for example, *zabbix/ui/modules*). The directory *zabbix/ui/widgets* is reserved for Zabbix built-in widgets and gets updated along with Zabbix UI.

2. Create a *manifest.json* file with basic widget metadata (see the description of supported parameters).

ui/modules/lesson_gauge_chart/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "lesson_gauge_chart",
    "type": "widget",
    "name": "Gauge chart",
    "namespace": "LessonGaugeChart",
    "version": "1.0",
    "author": "Zabbix SIA"
}
```

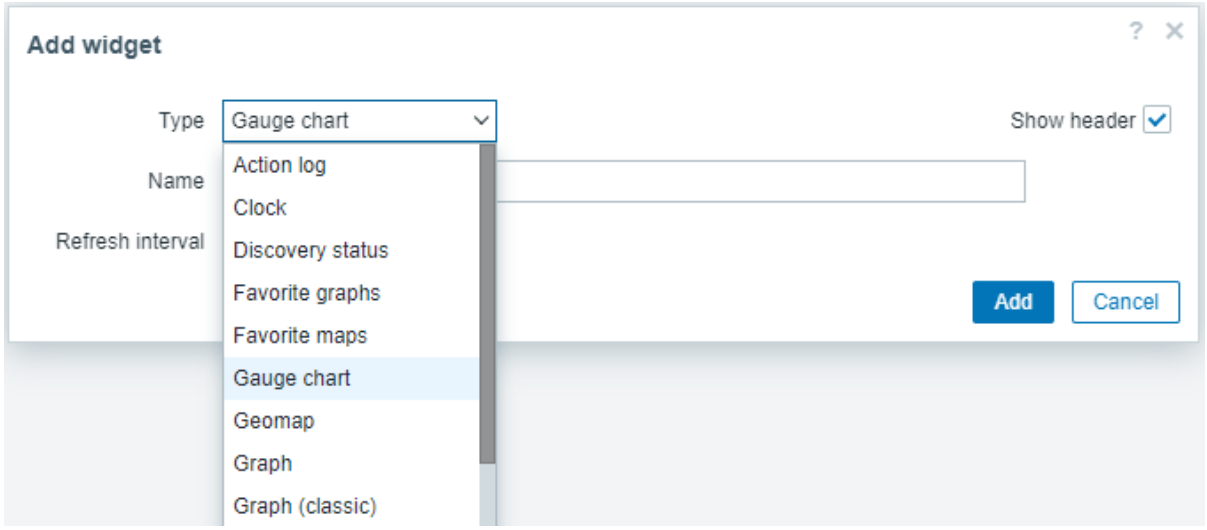
3. In Zabbix frontend, go to *Administration* → *General* → *Modules* section and click on the *Scan directory* button.



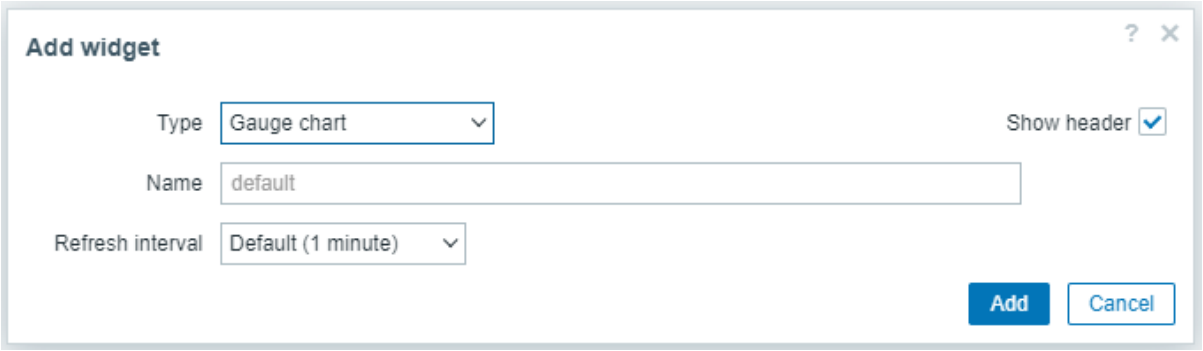
4. Locate the new module *Gauge chart* in the list and click on the "Disabled" hyperlink to change the module's status from "Disabled" to "Enabled".



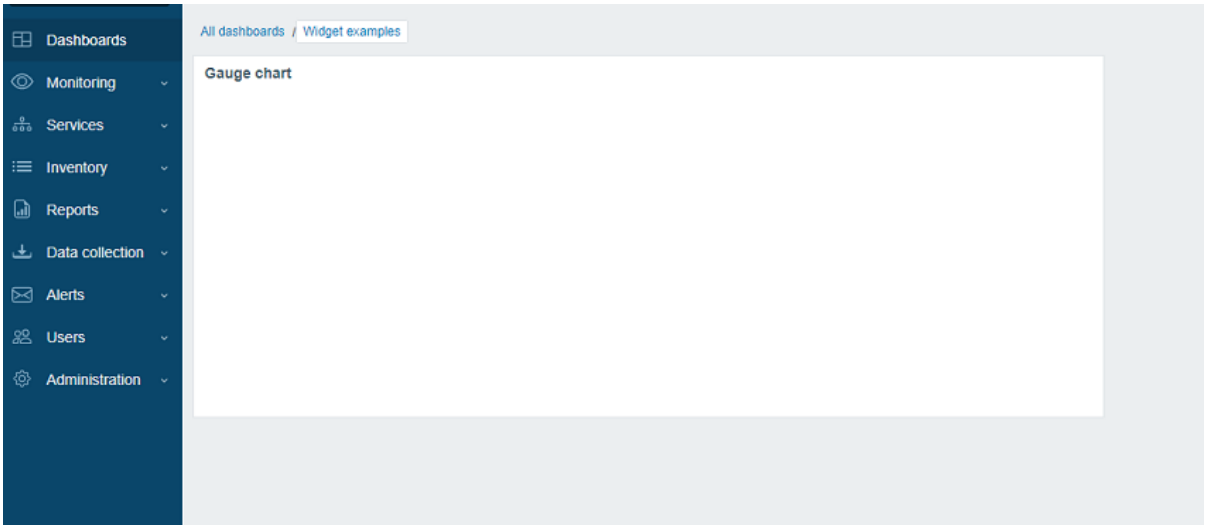
5. Open a dashboard, switch it to the edit mode and add a new widget. In the *Type* field, select "Gauge chart".



6. At this point, the *Gauge chart* widget configuration contains only common widget fields *Name* and *Refresh interval*. Click on *Add* to add the widget to the dashboard.



7. A blank widget should appear on the dashboard. Click on *Save changes* in the top right corner to save the dashboard.



Add a widget view

Note:
The widget's **view** file should be located in the *views* directory (for this tutorial, *ui/modules/lesson_gauge_chart/views/*). If the file has the default name *widget.view.php*, you do not need to register it in the *manifest.json* file. If the file has a different name, specify it in the *actions/widget.lesson_gauge_chart.view* section of the manifest.json file.

1. Create a directory *views* in the *lesson_gauge_chart* directory.
2. Create a *widget.view.php* file in the *views* directory.

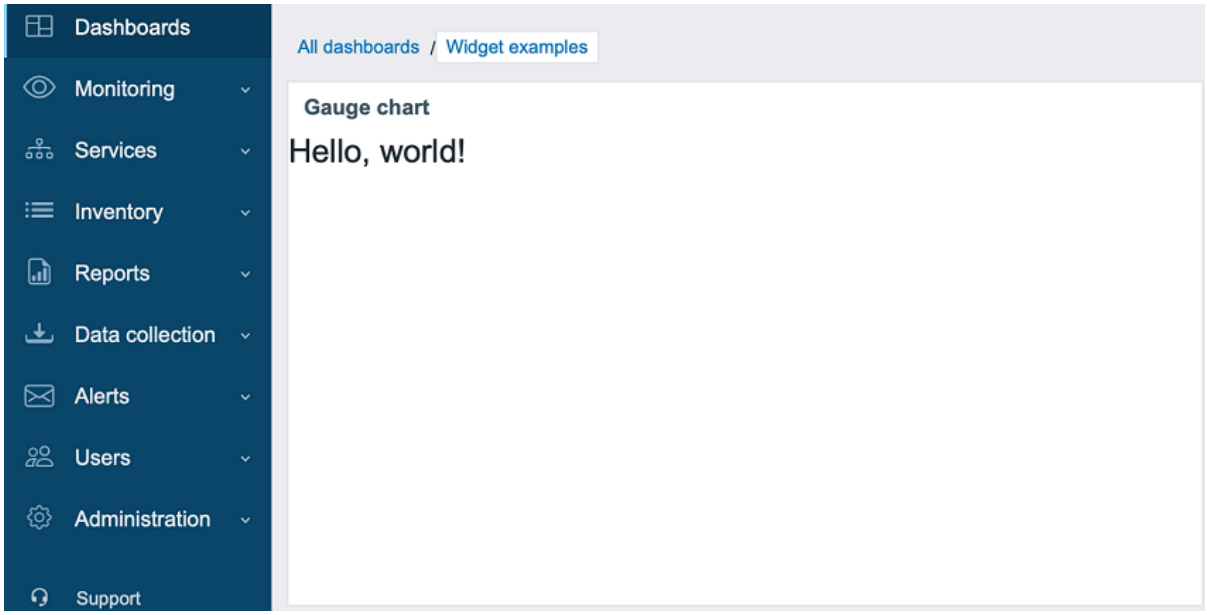
ui/modules/lesson_gauge_chart/views/widget.view.php

```
<?php

/**
 * Gauge chart widget view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetView($data))
    ->addItem(
        new CTag('h1', true, 'Hello, world!')
    )
    ->show();
```

3. Refresh the dashboard. The *Gauge chart* widget now displays "Hello, world!".



Part II - Gauge chart Add settings to a configuration view and use them in a widget view

In this section, you will learn how to add a widget configuration field and show the entered value in the widget view as text.

The widget configuration consists of a form (*Zabbix\Widgets\CWidgetForm*) and a widget form view (*widget.edit.php*). To add fields (*Zabbix\Widgets\CWidgetField*), you need to create a *WidgetForm* class, which will extend *Zabbix\Widgets\CWidgetForm*.

The form contains the set of fields (*Zabbix\Widgets\CWidgetField*) of various types, which are used to validate user-entered values. The form field (*Zabbix\Widgets\CWidgetField*) for each input element type converts the value into a single format to store it in the database.

> **Note:**
> The widget's **form** file should be located in the *includes* directory (for this tutorial, *ui/modules/lesson_gauge_chart/includes/*). If the file has the default name *WidgetForm.php*, you do not need to register it in the *manifest.json* file. If the file has a different name, specify it in the *widget/form_class* section of the manifest.json file.

1. Create a new directory *includes* in the *lesson_gauge_chart* directory.
2. Create a *WidgetForm.php* file in the *includes* directory.

ui/modules/lesson_gauge_chart/includes/WidgetForm.php

```
<?php

namespace Modules\LessonGaugeChart\Includes;

use Zabbix\Widgets\CWidgetForm;

class WidgetForm extends CWidgetForm {
}
```

3. Add a *Description* field to widget configuration form. This is a regular text field, where a user can enter any character set. You can use the *CWidgetFieldTextBox* class for it.

ui/modules/lesson_gauge_chart/includes/WidgetForm.php

```
<?php

namespace Modules\LessonGaugeChart\Includes;

use Zabbix\Widgets\CWidgetForm;

use Zabbix\Widgets\Fields\CWidgetFieldTextBox;

class WidgetForm extends CWidgetForm {
```

```
    public function addFields(): self {
        return $this
            ->addField(
                new CWidgetFieldTextBox('description', _('Description'))
            );
    }
}
```

4. In the *views* directory, create a widget configuration view file *widget.edit.php* and add a view for the new *Description* field. For the *CWidgetFieldTextBox* field class, the view is *CWidgetFieldTextBoxView.*

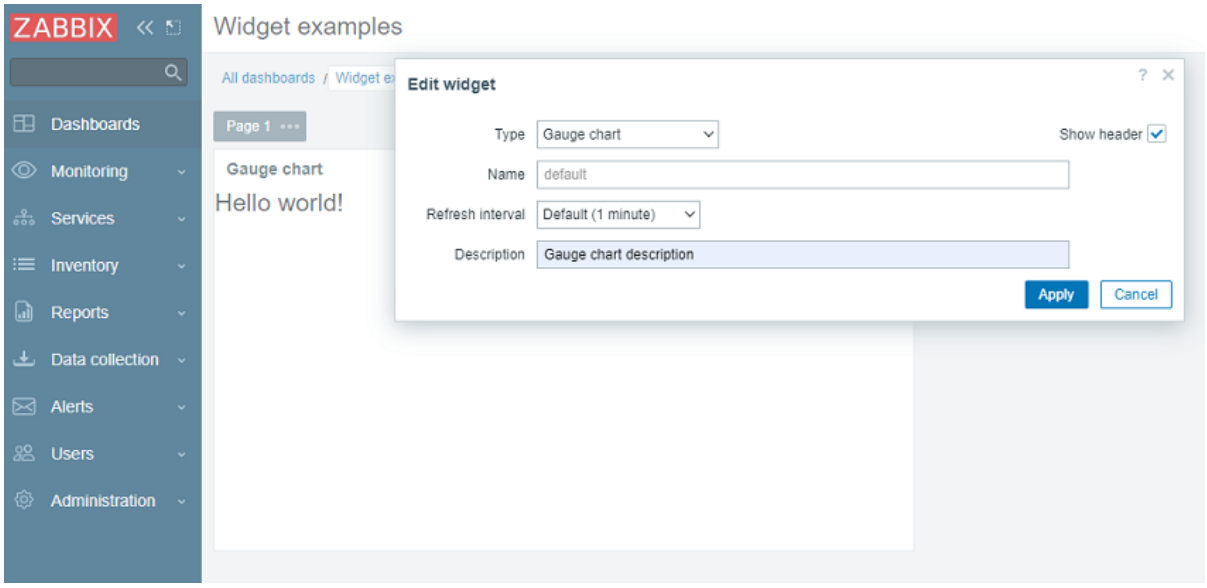
ui/modules/lesson_gauge_chart/views/widget.edit.php

```
<?php

/**
 * Gauge chart widget form view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetFormView($data))
    ->addField(
        new CWidgetFieldTextBoxView($data['fields']['description'])
    )
    ->show();
```

5. Go to the dashboard and click on the gear icon in the widget to open the widget configuration form.
6. The widget configuration form now contains a new *Description* text field. Enter any value, for example, *Gauge chart description*.



7. Click on *Apply* in the widget configuration form. Then click on *Save changes* in the top right corner to save the dashboard. Note that the new description is not visible anywhere, and the widget still displays "Hello, world!".

For the new description to appear in the widget, the *Description* field value needs to be retrieved from the database and passed to the widget view. For that, you need to create an action class.

8. Create a new directory *actions* in the *lesson_gauge_chart* directory.
9. Create a *WidgetView.php* file in the *actions* directory. The *WidgetView* action class will extend the *CControllerDashboardWidgetView* class.

Values of the widget configuration fields are stored in the **$fields_values** property of the action class.

ui/modules/lesson_gauge_chart/actions/WidgetView.php

```php
<?php

namespace Modules\LessonGaugeChart\Actions;

use CControllerDashboardWidgetView,
    CControllerResponseData;

class WidgetView extends CControllerDashboardWidgetView {

    protected function doAction(): void {
        $this->setResponse(new CControllerResponseData([
            'name' => $this->getInput('name', $this->widget->getName()),
            'description' => $this->fields_values['description'],
            'user' => [
                'debug_mode' => $this->getDebugMode()
            ]
        ]));
    }
}
```

10. Open *manifest.json* and register *WidgetView* as an action class in the *actions/widget.lesson_gauge_chart.view* section.

ui/modules/lesson_gauge_chart/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "lesson_gauge_chart",
    "type": "widget",
    "name": "Gauge chart",
    "namespace": "LessonGaugeChart",
    "version": "1.0",
    "author": "Zabbix SIA",
    "actions": {
        "widget.lesson_gauge_chart.view": {
            "class": "WidgetView"
        }
    }
}
```

11. Now you can use the value of the description field, contained in *$data['description']*, in the widget view. Open *views/widget.view.php* and replace the static text "Hello, world!" with *$data['description']*.

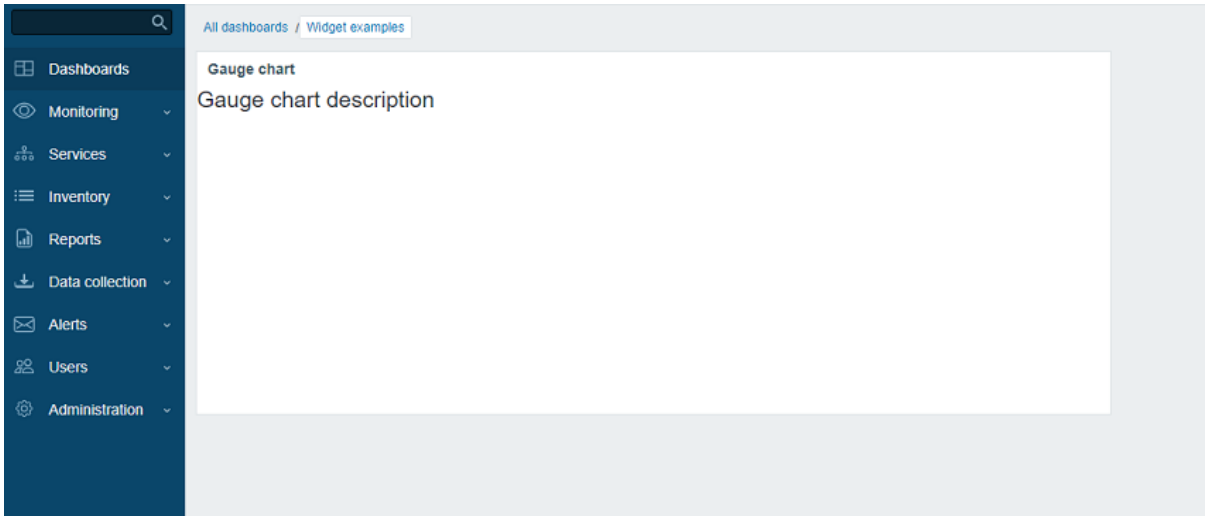
ui/modules/lesson_gauge_chart/views/widget.view.php

```php
<?php

/**
 * Gauge chart widget view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetView($data))
    ->addItem(
        new CTag('h1', true, $data['description'])
    )
    ->show();
```

12. Refresh the dashboard page. You should now see the widget description text instead of "Hello, world!".



Retrieve an item value via API

The widget should show the last value of an item of user's choice. For that, you need to add the ability to select items in the widget configuration.

In this section, you will learn how to add an item selection field to the widget form and how to add the visual part of this field to the configuration view. Then, the widget controller will be able to retrieve item data and its value via an API request. Once received, the value can be displayed in the widget view.

1. Open *includes/WidgetForm.php* and add the *CWidgetFieldMultiSelectItem* field. This will allow selecting an item in the configuration form.

ui/modules/lesson_gauge_chart/includes/WidgetForm.php

```
<?php

namespace Modules\LessonGaugeChart\Includes;

use Zabbix\Widgets\{
    CWidgetField,
    CWidgetForm
};

use Zabbix\Widgets\Fields\{
    CWidgetFieldMultiSelectItem,
    CWidgetFieldTextBox
};

/**
 * Gauge chart widget form.
 */
class WidgetForm extends CWidgetForm {

    public function addFields(): self {
        return $this
            ->addField(
                (new CWidgetFieldMultiSelectItem('itemid', _('Item')))
                    ->setFlags(CWidgetField::FLAG_NOT_EMPTY | CWidgetField::FLAG_LABEL_ASTERISK)
                    ->setMultiple(false)
            )
            ->addField(
                new CWidgetFieldTextBox('description', _('Description'))
            );
    }
}
```

2. Open *views/widget.edit.php* and add the field visual component to the configuration view.

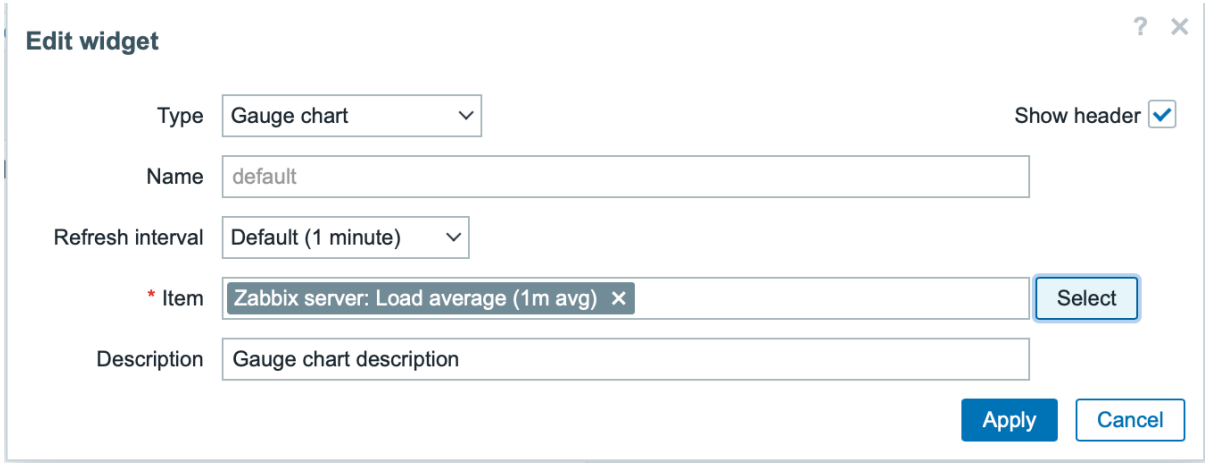
ui/modules/lesson_gauge_chart/views/widget.edit.php

```php
<?php

/**
 * Gauge chart widget form view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetFormView($data))
    ->addField(
        new CWidgetFieldMultiSelectItemView($data['fields']['itemid'], $data['captions']['items']['itemid'
    )
    ->addField(
        new CWidgetFieldTextBoxView($data['fields']['description'])
    )
    ->show();
```

3. Return to the dashboard and click on the gear icon in the widget to open the widget configuration form.
4. The widget configuration form now contains a new input field *Item*. Select the host "Zabbix server" and the item "Load average (1m avg)".



5. Click on *Apply* in the widget configuration form. Then click on *Save changes* in the top right corner to save the dashboard.
6. Open and modify *actions/WidgetView.php*.

From now on, the item ID will be available in the widget controller in *$this->fields_values['itemid']*. The *doAction()* controller method collects the item data (name, value type, units) using the API method *item.get* and the item last value using the API method *history.get*.

ui/modules/lesson_gauge_chart/actions/WidgetView.php

```php
<?php

namespace Modules\LessonGaugeChart\Actions;

use API,
    CControllerDashboardWidgetView,
    CControllerResponseData;

class WidgetView extends CControllerDashboardWidgetView {

    protected function doAction(): void {
        $db_items = API::Item()->get([
            'output' => ['itemid', 'value_type', 'name', 'units'],
            'itemids' => $this->fields_values['itemid'],
            'webitems' => true,
            'filter' => [
                'value_type' => [ITEM_VALUE_TYPE_UINT64, ITEM_VALUE_TYPE_FLOAT]
            ]
```

```
        ]);

        $value = null;

        if ($db_items) {
            $item = $db_items[0];

            $history = API::History()->get([
                'output' => API_OUTPUT_EXTEND,
                'itemids' => $item['itemid'],
                'history' => $item['value_type'],
                'sortfield' => 'clock',
                'sortorder' => ZBX_SORT_DOWN,
                'limit' => 1
            ]);

            if ($history) {
                $value = convertUnitsRaw([
                    'value' => $history[0]['value'],
                    'units' => $item['units']
                ]);
            }
        }

        $this->setResponse(new CControllerResponseData([
            'name' => $this->getInput('name', $this->widget->getName()),
            'value' => $value,
            'description' => $this->fields_values['description'],
            'user' => [
                'debug_mode' => $this->getDebugMode()
            ]
        ]));
    }
}
```

7. Open *views/widget.view.php* and add the item value to the widget view.

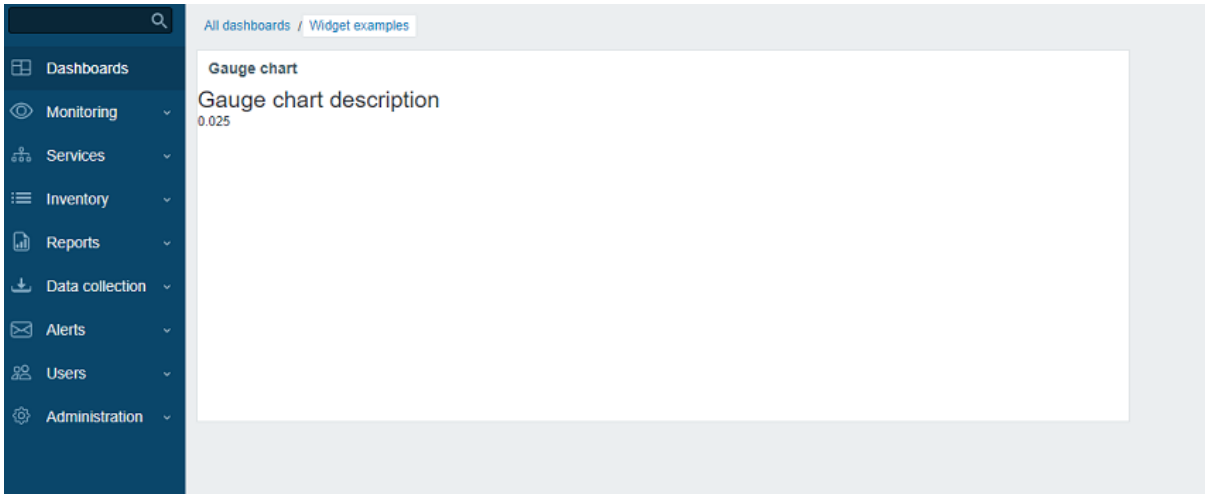
ui/modules/lesson_gauge_chart/views/widget.view.php

```
<?php

/**
 * Gauge chart widget view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetView($data))
    ->addItem([
        new CTag('h1', true, $data['description']),
        new CDiv($data['value'] !== null ? $data['value']['value'] : _('No data'))
    ])
    ->show();
```

8. Refresh the dashboard page. The widget will display the latest item value.



Add JavaScript to the widget

In this section, you will learn how to add Javascript elements to the widget.

You will add:

- A gauge chart made using JavaScript - to show at a glance if the latest value is normal or too high/too low.
- An *Advanced configuration* section for optional parameters, such as color, minimum and maximum values, units and the *Description* field created earlier.

1. Create a *widget.edit.js.php* file in the *views* directory.

JavaScript will be responsible for hiding optional fields behind the *Advanced configuration* checkbox and initializing the color picker in the configuration view. You can add it to the same directory as the configuration view.

Since the JavaScript for the configuration view should be loaded with the form, you will need to include it into the *widget.edit.php* file as shown in the next steps.

ui/modules/lesson_gauge_chart/views/widget.edit.js.php

```
<?php

use Modules\LessonGaugeChart\Widget;

?>

window.widget_lesson_gauge_chart_form = new class {

    init({color_palette}) {
        this._form = document.getElementById('widget-dialogue-form');

        this._advanced_configuration = document.getElementById('adv_conf');
        this._unit_select = document.getElementById('value_units');
        this._unit_value = document.getElementById('value_static_units');

        this._advanced_configuration.addEventListener('change', () => this.updateForm());
        this._unit_select.addEventListener('change', () => this.updateForm());

        colorPalette.setThemeColors(color_palette);

        for (const colorpicker of jQuery('.<?= ZBX_STYLE_COLOR_PICKER ?> input')) {
            jQuery(colorpicker).colorpicker();
        }

        const overlay = overlays_stack.getById('widget_properties');

        for (const event of ['overlay.reload', 'overlay.close']) {
            overlay.$dialogue[0].addEventListener(event, () => { jQuery.colorpicker('hide'); });
        }

        this.updateForm();
```

```
    }

    updateForm() {
        const show_advanced_configuration = this._advanced_configuration.checked;

        for (const element of this._form.querySelectorAll('.js-advanced-configuration')) {
            element.style.display = show_advanced_configuration ? '' : 'none';
        }

        this._unit_value.disabled = this._unit_select.value == <?= Widget::UNIT_AUTO ?>;
    }
};
```

2. Create a *Widget.php* file in the main widget directory *lesson_gauge_chart* to create a new class *Widget*.

The *Widget* class will extend the *CWidget* base class to add/override the default widget settings (in this case - translations). JavaScript, provided below, displays the string "No data" in case of missing data. The "No data" string is present in the Zabbix UI translation files.

If there are any widget constants, it is recommended to also specify them in the *Widget* class.

ui/modules/lesson_gauge_chart/Widget.php

```
<?php

namespace Modules\LessonGaugeChart;

use Zabbix\Core\CWidget;

class Widget extends CWidget {

    public const UNIT_AUTO = 0;
    public const UNIT_STATIC = 1;

    public function getTranslationStrings(): array {
        return [
            'class.widget.js' => [
                'No data' => _('No data')
            ]
        ];
    }
}
```

3. Open *includes/WidgetForm.php* and add the new fields: *Advanced configuration* (checkbox), *Color* (color picker), *Min* (numeric field), *Max* (numeric field), and *Units* (select).

ui/modules/lesson_gauge_chart/includes/WidgetForm.php

```
<?php

namespace Modules\LessonGaugeChart\Includes;

use Modules\LessonGaugeChart\Widget;

use Zabbix\Widgets\{
    CWidgetField,
    CWidgetForm
};

use Zabbix\Widgets\Fields\{
    CWidgetFieldCheckBox,
    CWidgetFieldColor,
    CWidgetFieldMultiSelectItem,
    CWidgetFieldNumericBox,
    CWidgetFieldSelect,
    CWidgetFieldTextBox
```

```
};

/**
 * Gauge chart widget form.
 */
class WidgetForm extends CWidgetForm {

    public function addFields(): self {
        return $this
            ->addField(
                (new CWidgetFieldMultiSelectItem('itemid', _('Item')))
                    ->setFlags(CWidgetField::FLAG_NOT_EMPTY | CWidgetField::FLAG_LABEL_ASTERISK)
                    ->setMultiple(false)
                    ->setFilterParameter('numeric', true)
            )
            ->addField(
                new CWidgetFieldCheckBox('adv_conf', _('Advanced configuration'))
            )
            ->addField(
                (new CWidgetFieldColor('chart_color', _('Color')))->setDefault('FF0000')
            )
            ->addField(
                (new CWidgetFieldNumericBox('value_min', _('Min')))
                    ->setDefault(0)
                    ->setFlags(CWidgetField::FLAG_NOT_EMPTY | CWidgetField::FLAG_LABEL_ASTERISK)
            )
            ->addField(
                (new CWidgetFieldNumericBox('value_max', _('Max')))
                    ->setDefault(100)
                    ->setFlags(CWidgetField::FLAG_NOT_EMPTY | CWidgetField::FLAG_LABEL_ASTERISK)
            )
            ->addField(
                (new CWidgetFieldSelect('value_units', _('Units'), [
                    Widget::UNIT_AUTO => _x('Auto', 'history source selection method'),
                    Widget::UNIT_STATIC => _x('Static', 'history source selection method')
                ]))->setDefault(Widget::UNIT_AUTO)
            )
            ->addField(
                (new CWidgetFieldTextBox('value_static_units'))
            )
            ->addField(
                new CWidgetFieldTextBox('description', _('Description'))
            );
    }
}
```

4. Open *views/widget.edit.php* and add the field visual components to the configuration view.

The *addField()* method of the *CWidgetFormView* class takes a CSS class string as the second parameter. Add the *js-advanced-configuration* string to those fields and their labels, which should be hidden if *Advanced configuration* is not selected.

To add a JavaScript file to the configuration view, use the *includeJsFile()* method. To add inline JavaScript, use the *addJavaScript()* method.

ui/modules/lesson_gauge_chart/views/widget.edit.php

```
<?php

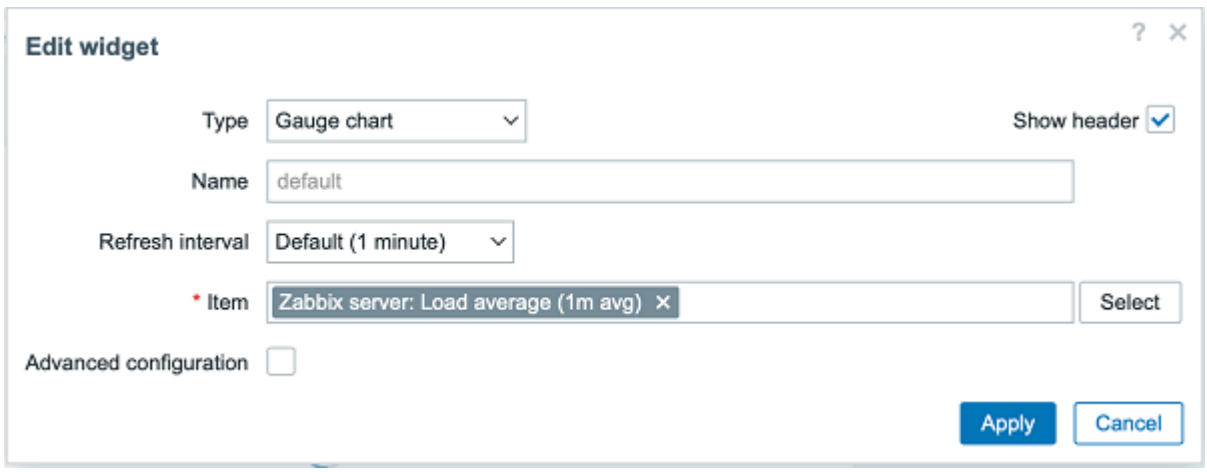
/**
 * Gauge chart widget form view.
 *
 * @var CView $this
 * @var array $data
 */
```

```
use Zabbix\Widgets\Fields\CWidgetFieldGraphDataSet;

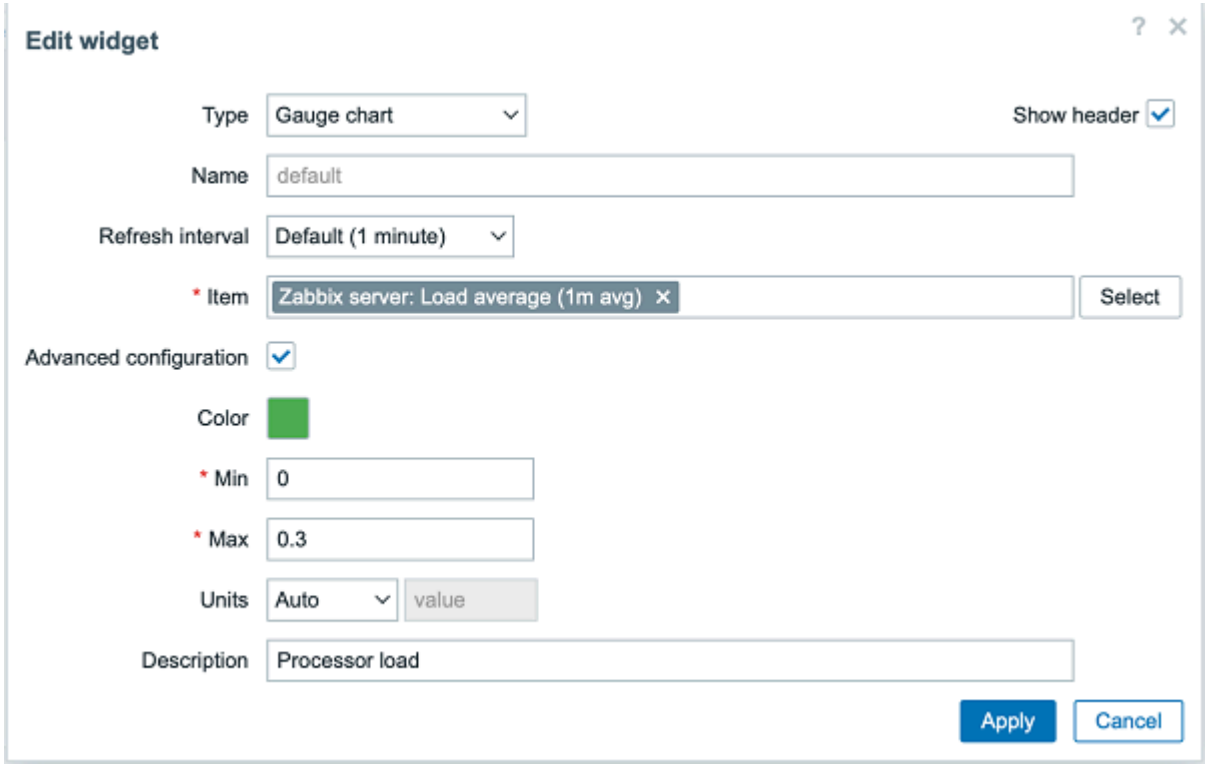
$lefty_units = new CWidgetFieldSelectView($data['fields']['value_units']);
$lefty_static_units = (new CWidgetFieldTextBoxView($data['fields']['value_static_units']))
    ->setPlaceholder(_('value'))
    ->setWidth(ZBX_TEXTAREA_TINY_WIDTH);

(new CWidgetFormView($data))
    ->addField(
        new CWidgetFieldMultiSelectItemView($data['fields']['itemid'], $data['captions']['items']['itemid'
    )
    ->addField(
        new CWidgetFieldCheckBoxView($data['fields']['adv_conf'])
    )
    ->addField(
        new CWidgetFieldColorView($data['fields']['chart_color']),
        'js-advanced-configuration'
    )
    ->addField(
        new CWidgetFieldNumericBoxView($data['fields']['value_min']),
        'js-advanced-configuration'
    )
    ->addField(
        new CWidgetFieldNumericBoxView($data['fields']['value_max']),
        'js-advanced-configuration'
    )
    ->addItem([
        $lefty_units->getLabel()->addClass('js-advanced-configuration'),
        (new CFormField([
            $lefty_units->getView()->addClass(ZBX_STYLE_FORM_INPUT_MARGIN),
            $lefty_static_units->getView()
        ]))->addClass('js-advanced-configuration')
    ])
    ->addField(
        new CWidgetFieldTextBoxView($data['fields']['description']),
        'js-advanced-configuration'
    )
    ->includeJsFile('widget.edit.js.php')
    ->addJavaScript('widget_lesson_gauge_chart_form.init('.json_encode([
        'color_palette' => CWidgetFieldGraphDataSet::DEFAULT_COLOR_PALETTE
    ], JSON_THROW_ON_ERROR).');')
    ->show();
```

5. Return to the dashboard, click on the gear icon in the widget to open the widget configuration form. The widget configuration form now contains a new *Advanced configuration* checkbox.



6. Mark the *Advanced configuration* checkbox to see additional widget configuration fields. Fill in the fields with values and select a widget color.



7. Click on *Apply* in the widget configuration form. Then click on *Save changes* in the top right corner to save the dashboard.
8. Open *actions/WidgetView.php* and update the controller.

The **$this->fields_values** property now contains the values of all the *Advanced configuration* fields. Finalize the controller to enable passing the configuration and selected item value to the widget view.

ui/modules/lesson_gauge_chart/actions/WidgetView.php

```
<?php

namespace Modules\LessonGaugeChart\Actions;

use API,
    CControllerDashboardWidgetView,
    CControllerResponseData;

class WidgetView extends CControllerDashboardWidgetView {

    protected function doAction(): void {
        $db_items = API::Item()->get([
            'output' => ['itemid', 'value_type', 'name', 'units'],
            'itemids' => $this->fields_values['itemid'],
            'webitems' => true,
            'filter' => [
                'value_type' => [ITEM_VALUE_TYPE_UINT64, ITEM_VALUE_TYPE_FLOAT]
            ]
        ]);

        $history_value = null;

        if ($db_items) {
            $item = $db_items[0];

            $history = API::History()->get([
                'output' => API_OUTPUT_EXTEND,
                'itemids' => $item['itemid'],
                'history' => $item['value_type'],
                'sortfield' => 'clock',
                'sortorder' => ZBX_SORT_DOWN,
```

```
                'limit' => 1
            ]);

            if ($history) {
                $history_value = convertUnitsRaw([
                    'value' => $history[0]['value'],
                    'units' => $item['units']
                ]);
            }
        }

        $this->setResponse(new CControllerResponseData([
            'name' => $this->getInput('name', $this->widget->getName()),
            'history' => $history_value,
            'fields_values' => $this->fields_values,
            'user' => [
                'debug_mode' => $this->getDebugMode()
            ]
        ]));
    }
}
```

9. Open and modify *views/widget.view.php*.

You need to create a container for the gauge chart, which you will draw in the next steps, and a container for the description.

To pass values to JavaScript as a JSON object, use the *setVar()* method.

ui/modules/lesson_gauge_chart/views/widget.view.php

```
<?php

/**
 * Gauge chart widget view.
 *
 * @var CView $this
 * @var array $data
 */

(new CWidgetView($data))
    ->addItem([
        (new CDiv())->addClass('chart'),
        $data['fields_values']['description']
            ? (new CDiv($data['fields_values']['description']))->addClass('description')
            : null
    ])
    ->setVar('history', $data['history'])
    ->setVar('fields_values', $data['fields_values'])
    ->show();
```

10. Create a new directory *assets* in the *lesson_gauge_chart* directory. This directory will be used for storing JavaScript, CSS, and potentially any other assets, such as fonts or images.
11. For widget view JavaScript, create a directory *js* in the *assets* directory.
12. Create a *class.widget.js* file in the *assets/js* directory.

This JavaScript widget class will extend the base JavaScript class of all dashboard widgets - *CWidget*.

The dashboard relies on a correct implementation of a widget and communicates any relevant information to the widget through calling the respective JavaScript methods. The dashboard also expects the widget to generate events when some interaction occurs. Thus, the *CWidget* class contains a set of methods with the default implementation of widget behavior, which can be customized by extending the class.

In this case, some customization is necessary, therefore custom logic will be implemented for the following widget behavior:

- widget initialization that is responsible for defining the initial state of the widget (see the *_init()* method);

- displaying widget contents (that is, drawing the gauge chart) if the widget update process has been successful and without errors (see the *_processUpdateResponse(response)* method and the related *_resizeChart()* and *_updatedChart()* methods)
- resizing the widget (see the *resize()* method and the related *_resizeChart()* method)

For other aspects of the gauge chart widget, the default implementation for widget behavior will be used. To learn more about the JavaScript methods of the *CWidget* class, see: JavaScript.

Since this JavaScript is required for the widget view, it should be loaded with the dashboard page. To enable JavaScript loading, you will need to update the *assets/js* and *js_class* parameters of **manifest.json** as shown in the next step.

ui/modules/lesson_gauge_chart/assets/js/class.widget.js

```
class WidgetLessonGaugeChart extends CWidget {

    static UNIT_AUTO = 0;
    static UNIT_STATIC = 1;

    _init() {
        super._init();

        this._refresh_frame = null;
        this._chart_container = null;
        this._canvas = null;
        this._chart_color = null;
        this._min = null;
        this._max = null;
        this._value = null;
        this._last_value = null;
        this._units = '';
    }

    _processUpdateResponse(response) {
        if (response.history === null) {
            this._value = null;
            this._units = '';
        }
        else {
            this._value = Number(response.history.value);
            this._units = response.fields_values.value_units == WidgetLessonGaugeChart.UNIT_AUTO
                ? response.history.units
                : response.fields_values.value_static_units;
        }

        this._chart_color = response.fields_values.chart_color;
        this._min = Number(response.fields_values.value_min);
        this._max = Number(response.fields_values.value_max);

        if (this._canvas === null) {
            super._processUpdateResponse(response);

            this._chart_container = this._content_body.querySelector('.chart');
            this._canvas = document.createElement('canvas');

            this._chart_container.appendChild(this._canvas);

            this._resizeChart();
        }
        else {
            this._updatedChart();
        }
    }

    resize() {
        super.resize();
```

```
        if (this._state === WIDGET_STATE_ACTIVE) {
            this._resizeChart();
        }
    }

    _resizeChart() {
        const ctx = this._canvas.getContext('2d');
        const dpr = window.devicePixelRatio;

        this._canvas.style.display = 'none';
        const size = Math.min(this._chart_container.offsetWidth, this._chart_container.offsetHeight);
        this._canvas.style.display = '';

        this._canvas.width = size * dpr;
        this._canvas.height = size * dpr;

        ctx.scale(dpr, dpr);

        this._canvas.style.width = `${size}px`;
        this._canvas.style.height = `${size}px`;

        this._refresh_frame = null;

        this._updatedChart();
    }

    _updatedChart() {
        if (this._last_value === null) {
            this._last_value = this._min;
        }

        const start_time = Date.now();
        const end_time = start_time + 400;

        const animate = () => {
            const time = Date.now();

            if (time <= end_time) {
                const progress = (time - start_time) / (end_time - start_time);
                const smooth_progress = 0.5 + Math.sin(Math.PI * (progress - 0.5)) / 2;
                let value = this._value !== null ? this._value : this._min;
                value = (this._last_value + (value - this._last_value) * smooth_progress - this._min) / (t

                const ctx = this._canvas.getContext('2d');
                const size = this._canvas.width;
                const char_weight = size / 12;
                const char_shadow = 3;
                const char_x = size / 2;
                const char_y = size / 2;
                const char_radius = (size - char_weight) / 2 - char_shadow;

                const font_ratio = 32 / 100;

                ctx.clearRect(0, 0, size, size);

                ctx.beginPath();
                ctx.shadowBlur = char_shadow;
                ctx.shadowColor = '#bbb';
                ctx.strokeStyle = '#eee';
                ctx.lineWidth = char_weight;
                ctx.lineCap = 'round';
                ctx.arc(char_x, char_y, char_radius, Math.PI * 0.749, Math.PI * 2.251, false);
```

```
                ctx.stroke();

                ctx.beginPath();
                ctx.strokeStyle = `#${this._chart_color}`;
                ctx.lineWidth = char_weight - 2;
                ctx.lineCap = 'round';
                ctx.arc(char_x, char_y, char_radius, Math.PI * 0.75,
                    Math.PI * (0.75 + (1.5 * Math.min(1, Math.max(0, value)))), false
                    );
                ctx.stroke();

                ctx.shadowBlur = 2;
                ctx.fillStyle = '#1f2c33';
                ctx.font = `${(char_radius * font_ratio)|0}px Arial`;
                ctx.textAlign = 'center';
                ctx.textBaseline = 'middle';
                ctx.fillText(`${this._value !== null ? this._value : t('No data')}${this._units}`,
                    char_x, char_y, size - char_shadow * 4 - char_weight * 2
                );

                ctx.fillStyle = '#768d99';
                ctx.font = `${(char_radius * font_ratio * .5)|0}px Arial`;
                ctx.textBaseline = 'top';

                ctx.textAlign = 'left';
                ctx.fillText(`${this._min}${this._min != '' ? this._units : ''}`,
                    char_weight * .75, size - char_weight * 1.25, size / 2 - char_weight
                );

                ctx.textAlign = 'right';
                ctx.fillText(`${this._max}${this._max != '' ? this._units : ''}`,
                    size - char_weight * .75, size - char_weight * 1.25, size / 2 - char_weight
                );

                requestAnimationFrame(animate);
            }
            else {
                this._last_value = this._value;
            }
        };

        requestAnimationFrame(animate);
    }
}
```

13. Open *manifest.json* and add:

- file name (*class.widget.js*) to the array in the *assets/js* section;
- class name (*WidgetLessonGaugeChart*) to the *js_class* parameter in the *widget* section.

The *WidgetLessonGaugeChart* class will now be automatically loaded with the dashboard.

ui/modules/lesson_gauge_chart/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "lesson_gauge_chart",
    "type": "widget",
    "name": "Gauge chart",
    "namespace": "LessonGaugeChart",
    "version": "1.0",
    "author": "Zabbix SIA",
    "actions": {
        "widget.lesson_gauge_chart.view": {
            "class": "WidgetView"
```

```
        }
    },
    "widget": {
        "js_class": "WidgetLessonGaugeChart"
    },
    "assets": {
        "js": ["class.widget.js"]
    }
}
```

Add CSS styles to the widget

In this section you will learn how to add custom CSS styles to make the widget look more appealing.

1. For widget styles, create a new directory *css* in the *assets* directory.
2. Create a *widget.css* file in the *assets/css* directory. To style widget elements, use the selector *div.dashboard-widget-{widget id}*. To configure CSS for the whole widget, use the selector *form.dashboard-widget-{widget id}*

ui/modules/lesson_gauge_chart/assets/css/widget.css

```
div.dashboard-widget-lesson_gauge_chart {
    display: grid;
    grid-template-rows: 1fr;
    padding: 0;
}

div.dashboard-widget-lesson_gauge_chart .chart {
    display: grid;
    align-items: center;
    justify-items: center;
}

div.dashboard-widget-lesson_gauge_chart .chart canvas {
    background: white;
}

div.dashboard-widget-lesson_gauge_chart .description {
    padding-bottom: 8px;
    font-size: 1.750em;
    line-height: 1.2;
    text-align: center;
}

.dashboard-grid-widget-hidden-header div.dashboard-widget-lesson_gauge_chart .chart {
    margin-top: 8px;
}
```

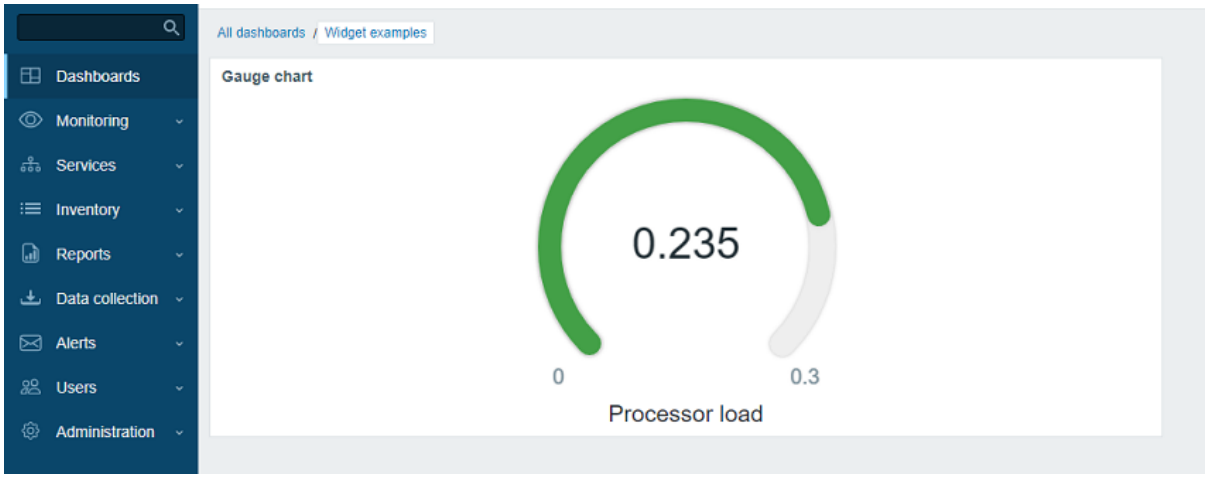
3. Open *manifest.json* and add the CSS file name (*widget.css*) to the array in the *assets/css* section. This will allow the CSS styles defined in *widget.css* to load with the dashboard page.

ui/modules/lesson_gauge_chart/manifest.json

```
{
    "manifest_version": 2.0,
    "id": "lesson_gauge_chart",
    "type": "widget",
    "name": "Gauge chart",
    "namespace": "LessonGaugeChart",
    "version": "1.0",
    "author": "Zabbix SIA",
    "actions": {
        "widget.lesson_gauge_chart.view": {
            "class": "WidgetView"
        }
    },
    "widget": {
```

```
        "js_class": "WidgetLessonGaugeChart"
    },
    "assets": {
        "css": ["widget.css"],
        "js": ["class.widget.js"]
    }
}
```

4. Refresh the dashboard page to see the finished version of the widget.



Examples

This section provides files of sample modules and widgets, which you can use as a base for your custom modules.

To use a module:

1. Download the ZIP archive.
2. Unpack the content into a separate directory inside the *modules* directory of your Zabbix frontend installation (for example, *zabbix/ui/modules*).
3. Register the module in Zabbix frontend.

Module example

- When creating a host group, grant read permissions to configured user groups - hg_auto_perm.zip

Widget examples

- Minimal widget - widget_min.zip
- "Hello, world" widget using CSS only - hello_world_css.zip
- "Hello, world" widget using JavaScript only - hello_world_js.zip
- "Hello, world" widget using PHP - hello_world_php.zip

> **Note:**
> You can also use Zabbix native widgets as examples.

Plugins

Overview Custom loadable plugins extend Zabbix agent 2 functionality. They are compiled separately, but use a package shared with Zabbix agent 2.

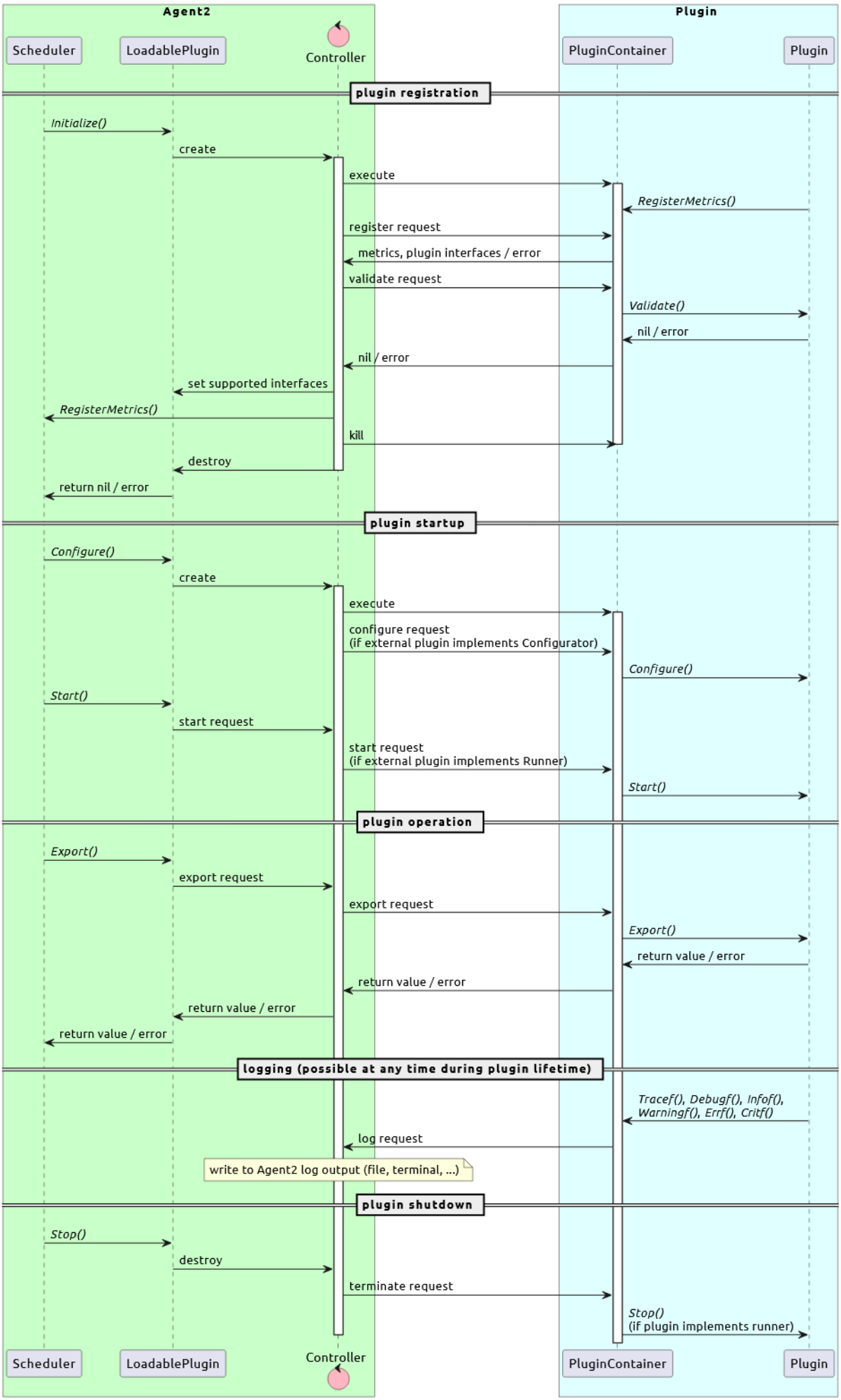
Each plugin is a *Go* package that defines the structure and implements one or several plugin interfaces (*Exporter*, *Configurator*, *Runner*).

Jump to:

- Write your first plugin
- Plugin interfaces

Connection diagram Zabbix agent 2 connects bidirectionally to the plugins using UNIX sockets on Linux and Named Pipes on Windows.

The connection diagram below illustrates the communication process between Zabbix agent 2 and a loadable plugin and the metrics collection process.


Agent2
Plugin
Scheduler
LoadablePlugin
Controller
PluginContainer
Plugin
plugin registration
Initialize()
create
execute
RegisterMetrics()
register request
metrics, plugin interfaces / error
validate request
Validate()
nil / error
nil / error
set supported interfaces
RegisterMetrics()
kill
destroy
return nil / error
plugin startup
Configure()
create
execute
configure request
(if external plugin implements Configurator)
Configure()
Start()
start request
start request
(if external plugin implements Runner)
Start()
plugin operation
Export()
export request
export request
Export()
return value / error
return value / error
return value / error
return value / error
logging (possible at any time during plugin lifetime)
Tracef(), Debugf(), Infof(),
Warningf(), Errf(), Critf()
log request
write to Agent2 log output (file, terminal, ...)
plugin shutdown
Stop()
destroy
terminate request
Stop()
(if plugin implements runner)
Scheduler
LoadablePlugin
Controller
PluginContainer
Plugin

Examples

You can use several empty examples as well as existing loadable plugins supplied by Zabbix as a reference:

- Examples
- MongoDB plugin
- PostgreSQL plugin

Create a plugin (tutorial)

This is a step-by-step tutorial to create a simple loadable plugin for Zabbix agent 2.

What you'll create During this tutorial, you will add a new loadable plugin **MyIP**. The plugin will implement 1 metric called **myip**, which returns the external IP address of the host where Zabbix agent 2 is running.

Part 1: Writing the Go code In this section you will learn how to write the plugin that adds a new metric to Zabbix agent 2.

1. Create a new directory *myip* in */usr/local/zabbix/go/plugins/*.
2. Create the file *main.go* inside *myip* directory and define the name of your Go package.

/usr/local/zabbix/go/plugins/myip/main.go

```
package main
```

> **Note:**
> Keep the file open to add more lines as described in the next steps.

3. Specify the packages to import.
   These are the packages that support the plugin.

/usr/local/zabbix/go/plugins/myip/main.go

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "git.zabbix.com/ap/plugin-support/plugin/container"
    "git.zabbix.com/ap/plugin-support/plugin"
)
```

4. Define the plugin structure.
   Embed the *plugin.Base* structure to gain access to the standard plugin functionality.

/usr/local/zabbix/go/plugins/myip/main.go

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "git.zabbix.com/ap/plugin-support/plugin/container"
    "git.zabbix.com/ap/plugin-support/plugin"
)

type Plugin struct {
    plugin.Base
}

var impl Plugin
```

5. Implement plugin interface Export.
   The Export interface performs a poll and returns a value.

/usr/local/zabbix/go/plugins/myip/main.go

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "git.zabbix.com/ap/plugin-support/plugin/container"
    "git.zabbix.com/ap/plugin-support/plugin"
)

type Plugin struct {
    plugin.Base
}

var impl Plugin

func (p *Plugin) Export(key string, params []string, ctx plugin.ContextProvider) (result interface{}, err
```

6. Add logging.
   Log messages will appear in the Zabbix agent 2 log. You can use one of the logging functions available to plugins: *Critf()*, *Errf()*, *Infof()*, *Warningf()*, *Debugf()*, *Tracef()*.

/usr/local/zabbix/go/plugins/myip/main.go

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "git.zabbix.com/ap/plugin-support/plugin/container"
    "git.zabbix.com/ap/plugin-support/plugin"
)

type Plugin struct {
    plugin.Base
}

var impl Plugin

func (p *Plugin) Export(key string, params []string, ctx plugin.ContextProvider) (result interface{}, err
    p.Infof("received request to handle %s key with %d parameters", key, len(params))
}
```

7. Implement the core plugin logic.
   This logic fetches the response from the specified URL and reads it, then returns the IP address as a response and closes the request.
   In case of an error when executing the GET request or reading a response, the error is returned instead.

/usr/local/zabbix/go/plugins/myip/main.go

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "git.zabbix.com/ap/plugin-support/plugin/container"
    "git.zabbix.com/ap/plugin-support/plugin"
)

type Plugin struct {
```

```
    plugin.Base
}

var impl Plugin

func (p *Plugin) Export(key string, params []string, ctx plugin.ContextProvider) (result interface{}, err
    p.Infof("received request to handle %s key with %d parameters", key, len(params))
    resp, err := http.Get("https://api.ipify.org")
    if err != nil {
        return nil, err
    }

    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return nil, err
    }

    return string(body), nil
}
```

8. Register the metric.
   Zabbix agent 2 initiates running *init()* function upon startup. This function will call *plugin.RegisterMetrics(structure, plugin name, metric name, description)* method to get the plugin data.

The *plugin.RegisterMetrics* method parameter description:

- **structure** - a pointer to plugin implementation; grants access to the plugin structure, including list of available plugin interfaces (for example, *&impl*).
- **name** - plugin name; must be unique (for example, *"Myip"*).
- **metric name** - metric name (for example, *"myip"*). This is the item key used to gather data from a plugin.
- **description** - metric description; must start with a capital letter and end with a period (for example, *""Return the external IP address of the host where agent is running."*).

Note:
To register several metrics, repeat the parameters **metric name** and **description** for each metric.
For example: `plugin.RegisterMetrics(&impl, "Myip", "metric.one", "Metric one description.", "metric.two", "Metric two description.")`

/usr/local/zabbix/go/plugins/myip/main.go

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "git.zabbix.com/ap/plugin-support/plugin/container"
    "git.zabbix.com/ap/plugin-support/plugin"
)

type Plugin struct {
    plugin.Base
}

var impl Plugin

func (p *Plugin) Export(key string, params []string, ctx plugin.ContextProvider) (result interface{}, err
    p.Infof("received request to handle %s key with %d parameters", key, len(params))
    resp, err := http.Get("https://api.ipify.org")
    if err != nil {
        return nil, err
    }
```

```
    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return nil, err
    }

    return string(body), nil
}

func init() {
    plugin.RegisterMetrics(&impl, "Myip", "myip", "Return the external IP address of the host where agent
}
```

9. Define the *main()* function, which will create a new plugin handler instance, assign it to be used for logging by the plugin and then execute the plugin handler.

Attention:
Defining the *main()* function is mandatory.

/usr/local/zabbix/go/plugins/myip/main.go

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"
    "git.zabbix.com/ap/plugin-support/plugin/container"
    "git.zabbix.com/ap/plugin-support/plugin"
)

type Plugin struct {
    plugin.Base
}

var impl Plugin

func (p *Plugin) Export(key string, params []string, ctx plugin.ContextProvider) (result interface{}, err
    p.Infof("received request to handle %s key with %d parameters", key, len(params))
    resp, err := http.Get("https://api.ipify.org")
    if err != nil {
        return nil, err
    }

    defer resp.Body.Close()

    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        return nil, err
    }

    return string(body), nil
}

func init() {
    plugin.RegisterMetrics(&impl, "Myip", "myip", "Return the external IP address of the host where agent
}

func main() {
    h, err := container.NewHandler(impl.Name())
    if err != nil {
        panic(fmt.Sprintf("failed to create plugin handler %s", err.Error()))
```

```
    }
    impl.Logger = &h

    err = h.Execute()
    if err != nil {
        panic(fmt.Sprintf("failed to execute plugin handler %s", err.Error()))
    }
}
```

Part 2: Building the plugin In this section you will learn how to compile the plugin.

1. To create Go files for dependency handling and download the dependencies automatically execute this bash script from the CLI.

```
go mod init myip
GOPROXY=direct go get git.zabbix.com/ap/plugin-support/plugin@branchname
go mod tidy
go build
```

Make sure to specify the correct branch name, i.e. replace `branchname` (see Line 2) with one of the following:

- `release/*` - for the stable release branch, where "*" is the release version (i.e. `6.4`)
- `master` - for the master branch
- `<commit hash>` - for the specific commit version (use the specific commit hash)

The output should be similar to this:

```
go: creating new go.mod: module myip
go: to add module requirements and sums:
    go mod tidy
go: finding module for package git.zabbix.com/ap/plugin-support/plugin/container
go: finding module for package git.zabbix.com/ap/plugin-support/plugin
go: found git.zabbix.com/ap/plugin-support/plugin in git.zabbix.com/ap/plugin-support v0.0.0-2022060810021
go: found git.zabbix.com/ap/plugin-support/plugin/container in git.zabbix.com/ap/plugin-support v0.0.0-202
```

2. Create an executable *myip* for the loadable plugin.
3. Specify the path to the plugin configuration file in the *Plugins.Myip.System.Path* parameter of Zabbix agent 2 configuration file.

> **Attention:**
> Plugin name in the configuration parameter name (*Myip* in this tutorial) must match the plugin name defined in the *plugin.RegisterMetrics()* function.

```
echo 'Plugins.Myip.System.Path=/usr/local/zabbix/go/plugins/myip/myip' > /etc/zabbix_agent2.d/plugins.d/my
```

4. Test the metric:

```
zabbix_agent2 -t myip
```

The response should contain an external IP address of your host.

> **Note:**
> In case of an error, check whether the user *zabbix* has permissions to access */usr/local/zabbix/go/plugins/myip* directory.

Plugin interfaces

This section describes available plugin interfaces.

plugin.Exporter *Exporter* is the simplest interface that performs a poll and returns a value (values), nothing, or error. It accepts a preparsed item key, parameters, and context. Access to all other plugin interfaces is exclusive and no method can be called if a plugin is already performing a task. Also, there is a limit of 100 maximum concurrent *Export()* calls per plugin, which can be reduced according to the requirements for each plugin.

plugin.Configurator *Configurator* interface provides plugin configuration parameters from Zabbix agent 2 configuration files.

plugin.Runner *Runner* interface provides the means for performing initialization when a plugin is started (activated) and deinitialization when a plugin is stopped (deactivated). For example, a plugin can start/stop some background *goroutine* by implementing the Runner interface.

Zabbix manpages

These are Zabbix manpages for Zabbix processes.

zabbix_agent2

Section: Maintenance Commands (8)
Updated: 2019-01-29
Index Return to Main Contents

NAME

zabbix_agent2 - Zabbix agent 2

SYNOPSIS

zabbix_agent2 [**-c** *config-file*]
zabbix_agent2 [**-c** *config-file*] **-p**
zabbix_agent2 [**-c** *config-file*] **-t** *item-key*
zabbix_agent2 [**-c** *config-file*] **-R** *runtime-option*
zabbix_agent2 -h
zabbix_agent2 -V

DESCRIPTION

zabbix_agent2 is an application for monitoring parameters of various services.

OPTIONS

-c, **--config** *config-file*
Use the alternate *config-file* instead of the default one.

-R, **--runtime-control** *runtime-option*
Perform administrative functions according to *runtime-option*.

Runtime control options: userparameter_reload
Reload user parameters from the configuration file

log_level_increase
Increase log level

log_level_decrease
Decrease log level

help
List available runtime control options

metrics
List available metrics

version
Display version

-p, **--print**
Print known items and exit. For each item either generic defaults are used, or specific defaults for testing are supplied. These defaults are listed in square brackets as item key parameters. Returned values are enclosed in square brackets and prefixed with the type of the returned value, separated by a pipe character. For user parameters type is always **t**, as the agent can not determine all possible return values. Items, displayed as working, are not guaranteed to work from the Zabbix server or zabbix_get when querying a running agent daemon as permissions or environment may be different. Returned value types are:

d
Number with a decimal part.

m
Not supported. This could be caused by querying an item that only works in the active mode like a log monitoring item or an item that requires multiple collected values. Permission issues or incorrect user parameters could also result in the not supported state.

s
Text. Maximum length not limited.

t
Text. Same as **s**.

u
Unsigned integer.

-t, **--test** *item-key*
Test single item and exit. See **--print** for output description.

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

FILES

/usr/local/etc/zabbix_agent2.conf
Default location of Zabbix agent 2 configuration file (if not modified during compile time).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agentd(8), **zabbix_get**(8), **zabbix_js**(8), **zabbix_proxy**(8), **zabbix_sender**(8), **zabbix_server**(8)

AUTHOR

Zabbix LLC

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

FILES

SEE ALSO

AUTHOR

This document was created on: 14:07:57 GMT, November 22, 2021

zabbix_agentd

Section: Maintenance Commands (8)
Updated: 2019-01-29
Index Return to Main Contents

NAME

zabbix_agentd - Zabbix agent daemon

SYNOPSIS

zabbix_agentd [**-c** *config-file*]
zabbix_agentd [**-c** *config-file*] **-p**
zabbix_agentd [**-c** *config-file*] **-t** *item-key*
zabbix_agentd [**-c** *config-file*] **-R** *runtime-option*
zabbix_agentd -h
zabbix_agentd -V

DESCRIPTION

zabbix_agentd is a daemon for monitoring various server parameters.

OPTIONS

-c, **--config** *config-file*
Use the alternate *config-file* instead of the default one.

-f, **--foreground**
Run Zabbix agent in foreground.

-R, **--runtime-control** *runtime-option*
Perform administrative functions according to *runtime-option*.

Runtime control options

userparameter_reload
Reload user parameters from the configuration file

log_level_increase[=*target*]
Increase log level, affects all processes if target is not specified

log_level_decrease[=*target*]
Decrease log level, affects all processes if target is not specified

Log level control targets

process-type
All processes of specified type (active checks, collector, listener)

process-type,N
Process type and number (e.g., listener,3)

pid
Process identifier, up to 65535. For larger values specify target as ”process-type,N”

-p, **--print**
Print known items and exit. For each item either generic defaults are used, or specific defaults for testing are supplied. These defaults are listed in square brackets as item key parameters. Returned values are enclosed in square brackets and prefixed with the type of the returned value, separated by a pipe character. For user parameters type is always **t**, as the agent can not determine all possible return values. Items, displayed as working, are not guaranteed to work from the Zabbix server or zabbix_get when querying a running agent daemon as permissions or environment may be different. Returned value types are:

d
Number with a decimal part.

m
Not supported. This could be caused by querying an item that only works in the active mode like a log monitoring item or an item that requires multiple collected values. Permission issues or incorrect user parameters could also result in the not supported state.

s
Text. Maximum length not limited.

t
Text. Same as **s**.

u
Unsigned integer.

-t, **--test** *item-key*
Test single item and exit. See **--print** for output description.

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

FILES

/usr/local/etc/zabbix_agentd.conf
Default location of Zabbix agent configuration file (if not modified during compile time).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agent2(8), **zabbix_get**(1), **zabbix_js**(1), **zabbix_proxy**(8), **zabbix_sender**(1), **zabbix_server**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

FILES

SEE ALSO

AUTHOR

This document was created by man2html, using the manual pages.
Time: 20:50:13 GMT, November 22, 2021

zabbix_get

Section: User Commands (1)
Updated: 2021-06-01
Index Return to Main Contents

NAME

zabbix_get - Zabbix get utility

SYNOPSIS

zabbix_get -s *host-name-or-IP* [**-p** *port-number*] [**-I** *IP-address*] [**-t** *timeout*] **-k** *item-key*
zabbix_get -s *host-name-or-IP* [**-p** *port-number*] [**-I** *IP-address*] [**-t** *timeout*] **--tls-connect cert --tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-agent-cert-issuer** *cert-issuer*] [**--tls-agent-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *item-key*
zabbix_get -s *host-name-or-IP* [**-p** *port-number*] [**-I** *IP-address*] [**-t** *timeout*] **--tls-connect psk --tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *item-key*
zabbix_get -h
zabbix_get -V

DESCRIPTION

zabbix_get is a command line utility for getting data from Zabbix agent.

OPTIONS

-s, **--host** *host-name-or-IP*
Specify host name or IP address of a host.

-p, **--port** *port-number*
Specify port number of agent running on the host. Default is 10050.

-I, **--source-address** *IP-address*
Specify source IP address.

-t, **--timeout** *seconds*
Specify timeout. Valid range: 1-30 seconds (default: 30)

-k, **--key** *item-key*
Specify key of item to retrieve value for.

--tls-connect *value*
How to connect to agent. Values:

unencrypted
connect without encryption (default)

psk
connect using TLS and a pre-shared key

cert
connect using TLS and a certificate

--tls-ca-file *CA-file*
Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification.

--tls-crl-file *CRL-file*
Full pathname of a file containing revoked certificates.

--tls-agent-cert-issuer *cert-issuer*
Allowed agent certificate issuer.

--tls-agent-cert-subject *cert-subject*
Allowed agent certificate subject.

--tls-cert-file *cert-file*
Full pathname of a file containing the certificate or certificate chain.

--tls-key-file *key-file*
Full pathname of a file containing the private key.

--tls-psk-identity *PSK-identity*
PSK-identity string.

--tls-psk-file *PSK-file*
Full pathname of a file containing the pre-shared key.

--tls-cipher13 *cipher-string*
Cipher string for OpenSSL 1.1.1 or newer for TLS 1.3. Override the default ciphersuite selection criteria. This option is not available if OpenSSL version is less than 1.1.1.

--tls-cipher *cipher-string*
GnuTLS priority string (for TLS 1.2 and up) or OpenSSL cipher string (only for TLS 1.2). Override the default ciphersuite selection criteria.

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

EXAMPLES

zabbix_get -s 127.0.0.1 -p 10050 -k "system.cpu.load[all,avg1]"
zabbix_get -s 127.0.0.1 -p 10050 -k "system.cpu.load[all,avg1]" --tls-connect cert --tls-ca-file /home/zabbix/zabbix_ca_file --tls-agent-cert-issuer "CN=Signing CA,OU=IT operations,O=Example Corp,DC=example,DC=com" --tls-agent-cert-subject "CN=server1,OU=IT operations,O=Example Corp,DC=example,DC=com" --tls-cert-file /home/zabbix/zabbix_get.crt --tls-key-file /home/zabbix/zabbix_get.key
zabbix_get -s 127.0.0.1 -p 10050 -k "system.cpu.load[all,avg1]" --tls-connect psk --tls-psk-identity "PSK ID Zabbix agentd" --tls-psk-file /home/zabbix/zabbix_agentd.psk

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agentd(8), **zabbix_proxy**(8), **zabbix_sender**(1), **zabbix_server**(8), **zabbix_js**(1), **zabbix_agent2**(8), **zabbix_web_service**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

EXAMPLES

SEE ALSO

AUTHOR

This document was created on: 08:42:29 GMT, June 11, 2021

zabbix_js

Section: User Commands (1)
Updated: 2019-01-29
Index Return to Main Contents

NAME

zabbix_js - Zabbix JS utility

SYNOPSIS

zabbix_js -s *script-file* **-p** *input-param* [**-l** *log-level*] [**-t** *timeout*]
zabbix_js -s *script-file* **-i** *input-file* [**-l** *log-level*] [**-t** *timeout*]
zabbix_js -h
zabbix_js -V

DESCRIPTION

zabbix_js is a command line utility that can be used for embedded script testing.

OPTIONS

-s, **--script** *script-file*
Specify the file name of the script to execute. If '-' is specified as file name, the script will be read from stdin.

-p, **--param** *input-param*
Specify the input parameter.

-i, **--input** *input-file*
Specify the file name of the input parameter. If '-' is specified as file name, the input will be read from stdin.

-l, **--loglevel** *log-level*
Specify the log level.

-t, **--timeout** *timeout*
Specify the timeout in seconds. Valid range: 1-60 seconds (default: 10)

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

EXAMPLES

zabbix_js -s script-file.js -p example

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agent2(8), **zabbix_agentd**(8), **zabbix_get**(1), **zabbix_proxy**(8), **zabbix_sender**(1), **zabbix_server**(8)

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

EXAMPLES

SEE ALSO

This document was created on: 21:23:35 GMT, March 18, 2020

zabbix_proxy

Section: Maintenance Commands (8)
Updated: 2020-09-04
Index Return to Main Contents

NAME

zabbix_proxy - Zabbix proxy daemon

SYNOPSIS

zabbix_proxy [**-c** *config-file*]
zabbix_proxy [**-c** *config-file*] **-R** *runtime-option*
zabbix_proxy -h
zabbix_proxy -V

DESCRIPTION

zabbix_proxy is a daemon that collects monitoring data from devices and sends it to Zabbix server.

OPTIONS

-c, **--config** *config-file*
Use the alternate *config-file* instead of the default one.

-f, **--foreground**
Run Zabbix proxy in foreground.

-R, **--runtime-control** *runtime-option*
Perform administrative functions according to *runtime-option*.

Runtime control options

config_cache_reload
Reload configuration cache. Ignored if cache is being currently loaded. Active Zabbix proxy will connect to the Zabbix server and request configuration data. Default configuration file (unless **-c** option is specified) will be used to find PID file and signal will be sent to process, listed in PID file.

snmp_cache_reload
Reload SNMP cache.

housekeeper_execute
Execute the housekeeper. Ignored if housekeeper is being currently executed.

diaginfo[=*section*]
Log internal diagnostic information of the specified section. Section can be *historycache*, *preprocessing*. By default diagnostic information of all sections is logged.

log_level_increase[=*target*]
Increase log level, affects all processes if target is not specified.

log_level_decrease[=*target*]
Decrease log level, affects all processes if target is not specified.

Log level control targets

process-type
All processes of specified type (configuration syncer, data sender, discoverer, history syncer, housekeeper, http poller, icmp pinger, ipmi manager, ipmi poller, java poller, poller, self-monitoring, snmp trapper, task manager, trapper, unreachable poller, vmware collector)

process-type,N
Process type and number (e.g., poller,3)

pid
Process identifier, up to 65535. For larger values specify target as ”process-type,N”

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

FILES

/usr/local/etc/zabbix_proxy.conf
Default location of Zabbix proxy configuration file (if not modified during compile time).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agentd(8), **zabbix_get**(1), **zabbix_sender**(1), **zabbix_server**(8), **zabbix_js**(1), **zabbix_agent2**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

FILES

SEE ALSO

AUTHOR

This document was created on: 16:12:22 GMT, September 04, 2020

zabbix_sender

Section: User Commands (1)
Updated: 2021-06-01
Index Return to Main Contents

NAME

zabbix_sender - Zabbix sender utility

SYNOPSIS

zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] **-s** *host* **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] [**-s** *host*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] [**-s** *host*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] [**-s** *host*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] **-s** *host* **--tls-connect cert --tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] [**-s** *host*] **--tls-connect cert --tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] [**-s** *host*] **--tls-connect cert --tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] [**-s** *host*] **--tls-connect cert --tls-ca-file** *CA-file* [**--tls-crl-file** *CRL-file*] [**--tls-server-cert-issuer** *cert-issuer*] [**--tls-server-cert-subject** *cert-subject*] **--tls-cert-file** *cert-file* **--tls-key-file** *key-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] **-s** *host* **--tls-connect psk --tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-z** *server* [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] [**-s** *host*] **--tls-connect psk --tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] [**-s** *host*] **--tls-connect psk --tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] **-k** *key* **-o** *value*
zabbix_sender [**-v**] **-c** *config-file* [**-z** *server*] [**-p** *port*] [**-I** *IP-address*] [**-t** *timeout*] [**-s** *host*] **--tls-connect psk --tls-psk-identity** *PSK-identity* **--tls-psk-file** *PSK-file* [**--tls-cipher13** *cipher-string*] [**--tls-cipher** *cipher-string*] [**-T**] [**-N**] [**-r**] **-i** *input-file*
zabbix_sender -h
zabbix_sender -V

DESCRIPTION

zabbix_sender is a command line utility for sending monitoring data to Zabbix server or proxy. On the Zabbix server an item of type **Zabbix trapper** should be created with corresponding key. Note that incoming values will only be accepted from hosts specified in **Allowed hosts** field for this item.

OPTIONS

-c, **--config** *config-file*
Use *config-file*. **Zabbix sender** reads server details from the agentd configuration file. By default **Zabbix sender** does not

read any configuration file. Only parameters **Hostname**, **ServerActive**, **SourceIP**, **TLSConnect**, **TLSCAFile**, **TLSCRLFile**, **TLSServerCertIssuer**, **TLSServerCertSubject**, **TLSCertFile**, **TLSKeyFile**, **TLSPSKIdentity** and **TLSPSKFile** are supported. All addresses defined in the agent **ServerActive** configuration parameter are used for sending data. If sending of batch data fails to one address, the following batches are not sent to this address.

-z, **--zabbix-server** *server*
Hostname or IP address of Zabbix server. If a host is monitored by a proxy, proxy hostname or IP address should be used instead. When used together with **--config**, overrides the entries of **ServerActive** parameter specified in agentd configuration file.

-p, **--port** *port*
Specify port number of Zabbix server trapper running on the server. Default is 10051. When used together with **--config**, overrides the port entries of **ServerActive** parameter specified in agentd configuration file.

-I, **--source-address** *IP-address*
Specify source IP address. When used together with **--config**, overrides **SourceIP** parameter specified in agentd configuration file.

-t, **--timeout** *seconds*
Specify timeout. Valid range: 1-300 seconds (default: 60)

-s, **--host** *host*
Specify host name the item belongs to (as registered in Zabbix frontend). Host IP address and DNS name will not work. When used together with **--config**, overrides **Hostname** parameter specified in agentd configuration file.

-k, **--key** *key*
Specify item key to send value to.

-o, **--value** *value*
Specify item value.

-i, **--input-file** *input-file*
Load values from input file. Specify **-** as **<input-file>** to read values from standard input. Each line of file contains whitespace delimited: **<hostname> <key> <value>**. Each value must be specified on its own line. Each line must contain 3 whitespace delimited entries: **<hostname> <key> <value>**, where ”hostname” is the name of monitored host as registered in Zabbix frontend, ”key” is target item key and ”value” - the value to send. Specify **-** as **<hostname>** to use hostname from agent configuration file or from **--host** argument.

An example of a line of an input file:

”Linux DB3” db.connections 43

The value type must be correctly set in item configuration of Zabbix frontend. Zabbix sender will send up to 250 values in one connection. Size limit for sending values from an input file depends on the size described in Zabbix communication protocol. Contents of the input file must be in the UTF-8 encoding. All values from the input file are sent in a sequential order top-down. Entries must be formatted using the following rules:

- Quoted and non-quoted entries are supported.
- Double-quote is the quoting character.
- Entries with whitespace must be quoted.
- Double-quote and backslash characters inside quoted entry must be escaped with a backslash.
- Escaping is not supported in non-quoted entries.
- Linefeed escape sequences (\n) are supported in quoted strings.
- Linefeed escape sequences are trimmed from the end of an entry.

-T, **--with-timestamps**
This option can be only used with **--input-file** option.

Each line of the input file must contain 4 whitespace delimited entries: **<hostname> <key> <timestamp> <value>**. Timestamp should be specified in Unix timestamp format. If target item has triggers referencing it, all timestamps must be in an increasing order, otherwise event calculation will not be correct.

An example of a line of the input file:

”Linux DB3” db.connections 1429533600 43

For more details please see option **--input-file**.

If a timestamped value is sent for a host that is in a “no data” maintenance type then this value will be dropped; however, it is possible to send a timestamped value in for an expired maintenance period and it will be accepted.

-N, **--with-ns**
This option can be only used with **--with-timestamps** option.

Each line of the input file must contain 5 whitespace delimited entries: **<hostname> <key> <timestamp> <ns> <value>**.

An example of a line of the input file:

”Linux DB3” db.connections 1429533600 7402561 43

For more details please see option **--input-file**.

-r, **--real-time**
Send values one by one as soon as they are received. This can be used when reading from standard input.

--tls-connect *value*
How to connect to server or proxy. Values:

unencrypted
connect without encryption (default)

psk
connect using TLS and a pre-shared key

cert
connect using TLS and a certificate

--tls-ca-file *CA-file*
Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification.

--tls-crl-file *CRL-file*
Full pathname of a file containing revoked certificates.

--tls-server-cert-issuer *cert-issuer*
Allowed server certificate issuer.

--tls-server-cert-subject *cert-subject*
Allowed server certificate subject.

--tls-cert-file *cert-file*
Full pathname of a file containing the certificate or certificate chain.

--tls-key-file *key-file*
Full pathname of a file containing the private key.

--tls-psk-identity *PSK-identity*
PSK-identity string.

--tls-psk-file *PSK-file*
Full pathname of a file containing the pre-shared key.

--tls-cipher13 *cipher-string*
Cipher string for OpenSSL 1.1.1 or newer for TLS 1.3. Override the default ciphersuite selection criteria. This option is not available if OpenSSL version is less than 1.1.1.

--tls-cipher *cipher-string*
GnuTLS priority string (for TLS 1.2 and up) or OpenSSL cipher string (only for TLS 1.2). Override the default ciphersuite selection criteria.

-v, **--verbose**
Verbose mode, **-vv** for more details.

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

EXIT STATUS

The exit status is 0 if the values were sent and all of them were successfully processed by server. If data was sent, but processing of at least one of the values failed, the exit status is 2. If data sending failed, the exit status is 1.

EXAMPLES

zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -k mysql.queries -o 342.45

Send **342.45** as the value for **mysql.queries** item of monitored host. Use monitored host and Zabbix server defined in agent configuration file.

zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -s "Monitored Host" -k mysql.queries -o 342.45

Send **342.45** as the value for **mysql.queries** item of **Monitored Host** host using Zabbix server defined in agent configuration file.

zabbix_sender -z 192.168.1.113 -i data_values.txt

Send values from file **data_values.txt** to Zabbix server with IP **192.168.1.113**. Host names and keys are defined in the file.

echo "- hw.serial.number 1287872261 SQ4321ASDF" | zabbix_sender -c /usr/local/etc/zabbix_agentd.conf -T -i -

Send a timestamped value from the commandline to Zabbix server, specified in the agent configuration file. Dash in the input data indicates that hostname also should be used from the same configuration file.

echo '"Zabbix server" trapper.item ""' | zabbix_sender -z 192.168.1.113 -p 10000 -i -

Send empty value of an item to the Zabbix server with IP address **192.168.1.113** on port **10000** from the commandline. Empty values must be indicated by empty double quotes.

zabbix_sender -z 192.168.1.113 -s "Monitored Host" -k mysql.queries -o 342.45 --tls-connect cert --tls-ca-file /home/zabbix/zabbix_ca_file --tls-cert-file /home/zabbix/zabbix_agentd.crt --tls-key-file /home/zabbix/zabbix_agentd.key

Send **342.45** as the value for **mysql.queries** item in **Monitored Host** host to server with IP **192.168.1.113** using TLS with certificate.

zabbix_sender -z 192.168.1.113 -s "Monitored Host" -k mysql.queries -o 342.45 --tls-connect psk --tls-psk-identity "PSK ID Zabbix agentd" --tls-psk-file /home/zabbix/zabbix_agentd.psk

Send **342.45** as the value for **mysql.queries** item in **Monitored Host** host to server with IP **192.168.1.113** using TLS with pre-shared key (PSK).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agentd(8), **zabbix_get**(1), **zabbix_proxy**(8), **zabbix_server**(8), **zabbix_js**(1), **zabbix_agent2**(8), **zabbix_web_service**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

EXIT STATUS

EXAMPLES

SEE ALSO

AUTHOR

This document was created on: 08:42:39 GMT, June 11, 2021

zabbix_server

Section: Maintenance Commands (8)
Updated: 2020-09-04
Index Return to Main Contents

NAME

zabbix_server - Zabbix server daemon

SYNOPSIS

zabbix_server [**-c** *config-file*]
zabbix_server [**-c** *config-file*] **-R** *runtime-option*
zabbix_server -h
zabbix_server -V

DESCRIPTION

zabbix_server is the core daemon of Zabbix software.

OPTIONS

-c, **--config** *config-file*
Use the alternate *config-file* instead of the default one.

-f, **--foreground**
Run Zabbix server in foreground.

-R, **--runtime-control** *runtime-option*
Perform administrative functions according to *runtime-option*.

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

Examples of running Zabbix server with command line parameters:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf
shell> zabbix_server --help
shell> zabbix_server -V
```

RUNTIME CONTROL

Runtime control options:

config_cache_reload
Reload configuration cache. Ignored if cache is being currently loaded. Default configuration file (unless **-c** option is specified) will be used to find PID file and signal will be sent to process, listed in PID file.

snmp_cache_reload
Reload SNMP cache, clear the SNMP properties (engine time, engine boots, engine id, credentials) for all hosts.

housekeeper_execute
Execute the housekeeper. Ignored if housekeeper is being currently executed.

trigger_housekeeper_execute
Execute the trigger housekeeper (remove problems for deleted triggers). Ignored if trigger housekeeper for services is being currently executed.

diaginfo[=*section*]
Log internal diagnostic information of the specified section. Section can be *historycache*, *preprocessing*, *alerting*, *lld*, *valuecache*. By default diagnostic information of all sections is logged.

ha_status
Log high availability (HA) cluster status.

ha_remove_node[=*target*]
Remove the high availability (HA) node specified by its name or ID. Note that active/standby nodes cannot be removed.

ha_set_failover_delay[=*delay*]
Set high availability (HA) failover delay. Time suffixes are supported, e.g. 10s, 1m.

proxy_config_cache_reload[=*target*] Reload proxy configuration cache.

secrets_reload
Reload secrets from Vault.

service_cache_reload
Reload the service manager cache.

prof_enable[=*target*]
Enable profiling. Affects all processes if target is not specified. Enabled profiling provides details of all rwlocks/mutexes by function name. Supported since Zabbix 6.0.13.

prof_disable[=*target*]
Disable profiling. Affects all processes if target is not specified. Supported since Zabbix 6.0.13.

log_level_increase[=*target*]
Increase log level, affects all processes if target is not specified

log_level_decrease[=*target*]
Decrease log level, affects all processes if target is not specified

Log level control targets

process-type
All processes of specified type (alerter, alert manager, configuration syncer, discoverer, escalator, history syncer, housekeeper, http poller, icmp pinger, ipmi manager, ipmi poller, java poller, lld manager, lld worker, poller, preprocessing manager, preprocessing worker, proxy poller, self-monitoring, snmp trapper, task manager, timer, trapper, unreachable poller, vmware collector)

process-type,N
Process type and number (e.g., poller,3)

pid
Process identifier, up to 65535. For larger values specify target as ”process-type,N”

FILES

/usr/local/etc/zabbix_server.conf
Default location of Zabbix server configuration file (if not modified during compile time).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agentd(8), **zabbix_get**(1), **zabbix_proxy**(8), **zabbix_sender**(1), **zabbix_js**(1), **zabbix_agent2**(8)

AUTHOR

Alexei Vladishev <alex@zabbix.com>

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

FILES

SEE ALSO

AUTHOR

This document was created on: 16:12:14 GMT, September 04, 2020

zabbix_web_service

Section: Maintenance Commands (8)
Updated: 2019-01-29
Index Return to Main Contents

NAME

zabbix_web_service - Zabbix web service

SYNOPSIS

zabbix_web_service [**-c** *config-file*]
zabbix_web_service -h
zabbix_web_service -V

DESCRIPTION

zabbix_web_service is an application for providing web services to Zabbix components.

OPTIONS

-c, **--config** *config-file*
Use the alternate *config-file* instead of the default one.

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

FILES

/usr/local/etc/zabbix_web_service.conf
Default location of Zabbix web service configuration file (if not modified during compile time).

SEE ALSO

Documentation https://www.zabbix.com/manuals

zabbix_agentd(8), **zabbix_get**(1), **zabbix_proxy**(8), **zabbix_sender**(1), **zabbix_server**(8), **zabbix_js**(1), **zabbix_agent2**(8)

AUTHOR

Zabbix LLC

Index

NAME

SYNOPSIS

DESCRIPTION

OPTIONS

FILES

SEE ALSO

AUTHOR

This document was created on: 12:58:30 GMT, June 11, 2021