Documentation 6.2

ZABBIX

02.01.2026

Contents

Zabbix Manual **5**

Copyright notice . . . 5

1. Introduction . . . 5

1 Manual structure . . . 5

2 What is Zabbix . . . 6

3 Zabbix features . . . 6

4 Zabbix overview . . . 7

5 What's new in Zabbix 6.2.0 . . . 8

6 What's new in Zabbix 6.2.1 . . . 13

7 What's new in Zabbix 6.2.2 . . . 13

8 What's new in Zabbix 6.2.3 . . . 14

9 What's new in Zabbix 6.2.4 . . . 14

10 What's new in Zabbix 6.2.5 . . . 15

11 What's new in Zabbix 6.2.6 . . . 15

12 What's new in Zabbix 6.2.7 . . . 16

13 What's new in Zabbix 6.2.8 . . . 17

14 What's new in Zabbix 6.2.9 . . . 17

2. Definitions . . . 17

3. Zabbix processes . . . 19

1 Server . . . 20

2 Agent . . . 26

3 Agent 2 . . . 29

4 Proxy . . . 31

5 Java gateway . . . 34

6 Sender . . . 38

7 Get . . . 38

8 JS . . . 39

9 Web service . . . 39

4. Installation . . . 40

1 Getting Zabbix . . . 40

2 Requirements . . . 40

3 Installation from sources . . . 54

4 Installation from packages . . . 63

Zabbix unstable repository **79**

5 Installation from containers . . . 79

6 Web interface installation . . . 86

7 Upgrade procedure . . . 91

8 Known issues . . . 104

9 Template changes . . . 108

10 Upgrade notes for 6.2.0 . . . 109

11 Upgrade notes for 6.2.1 . . . 111

12 Upgrade notes for 6.2.2 . . . 111

13 Upgrade notes for 6.2.3 . . . 111

14 Upgrade notes for 6.2.4 . . . 111

15 Upgrade notes for 6.2.5 . . . 111

16 Upgrade notes for 6.2.6 . . . 111

17 Upgrade notes for 6.2.7 . . . 112

18 Upgrade notes for 6.2.8 . . . 112

19 Upgrade notes for 6.2.9 . . . 112

5. Quickstart . . . 112

1 Login and configuring user . . . 112
2 New host . . . 116
3 New item . . . 117
4 New trigger . . . 119
5 Receiving problem notification . . . 121
6 New template . . . 125
6. Zabbix appliance . . . 127
7. Configuration . . . 130
1 Hosts and host groups . . . 139
2 Items . . . 149
3 Triggers . . . 347
4 Events . . . 365
5 Event correlation . . . 368
6 Tagging . . . 374
7 Visualization . . . 377
8 Templates and template groups . . . 404
9 Templates out of the box . . . 405
10 Notifications upon events . . . 414
11 Macros . . . 457
12 Users and user groups . . . 468
13 Storage of secrets . . . 475
14 Scheduled reports . . . 481
8. Service monitoring . . . 484
1 Service tree . . . 485
2 Service actions . . . 489
3 SLA . . . 490
4 Setup example . . . 491
9. Web monitoring . . . 496
1 Web monitoring items . . . 505
2 Real-life scenario . . . 506
10. Virtual machine monitoring . . . 514
VMware monitoring item keys . . . 517
Virtual machine discovery key fields . . . 532
JSON examples for VMware items . . . 534
11. Maintenance . . . 539
12. Regular expressions . . . 543
13. Problem acknowledgment . . . 548
1. Problem suppression . . . 550
14. Configuration export/import . . . 551
1 Template groups . . . 553
2 Host groups . . . 553
3 Templates . . . 554
4 Hosts . . . 574
5 Network maps . . . 592
6 Media types . . . 599
15. Discovery . . . 606
1 Network discovery . . . 606
2 Active agent autoregistration . . . 614
3 Low-level discovery . . . 617
16. Distributed monitoring . . . 669
1 Proxies . . . 669
17. Encryption . . . 673
1 Using certificates . . . 680
2 Using pre-shared keys . . . 686
3 Troubleshooting . . . 688
18. Web interface . . . 691
1 Menu . . . 691
2 Frontend sections . . . 693
3 User settings . . . 842
4 Global search . . . 846
5 Frontend maintenance mode . . . 847
6 Page parameters . . . 848
7 Definitions . . . 849

8 Creating your own theme . . . 850
9 Debug mode . . . 850
10 Cookies used by Zabbix . . . 851
11 Time zones . . . 852
19. API . . . 853
Method reference . . . 858
Appendix 1. Reference commentary . . . 1442
Appendix 2. Changes from 6.0 to 6.2 . . . 1447
Zabbix API changes in 6.2 . . . 1451
20. Modules . . . 1452
21. Appendixes . . . 1458
1 Frequently asked questions / Troubleshooting . . . 1458
2 Installation and setup . . . 1459
3 Process configuration . . . 1497
4 Protocols . . . 1547
5 Items . . . 1572
6 Supported functions . . . 1599
7 Macros . . . 1630
8 Unit symbols . . . 1665
9 Time period syntax . . . 1667
10 Command execution . . . 1667
11 Version compatibility . . . 1668
12 Database error handling . . . 1669
13 Zabbix sender dynamic link library for Windows . . . 1669
14 Service monitoring upgrade . . . 1670
15 Other issues . . . 1670
16 Agent vs agent 2 comparison . . . 1671

Zabbix manpages **1672**

zabbix_agent2 . . . 1672
NAME . . . 1672
SYNOPSIS . . . 1673
DESCRIPTION . . . 1673
OPTIONS . . . 1673
FILES . . . 1674
SEE ALSO . . . 1674
AUTHOR . . . 1674
Index . . . 1674
zabbix_agentd . . . 1674
NAME . . . 1674
SYNOPSIS . . . 1675
DESCRIPTION . . . 1675
OPTIONS . . . 1675
FILES . . . 1676
SEE ALSO . . . 1676
AUTHOR . . . 1676
Index . . . 1676
zabbix_get . . . 1676
NAME . . . 1676
SYNOPSIS . . . 1677
DESCRIPTION . . . 1677
OPTIONS . . . 1677
EXAMPLES . . . 1678
SEE ALSO . . . 1678
AUTHOR . . . 1678
Index . . . 1678
zabbix_js . . . 1678
NAME . . . 1679
SYNOPSIS . . . 1679
DESCRIPTION . . . 1679
OPTIONS . . . 1679
EXAMPLES . . . 1679
SEE ALSO . . . 1679

Index . . . 1679
zabbix_proxy . . . 1680
NAME . . . 1680
SYNOPSIS . . . 1680
DESCRIPTION . . . 1680
OPTIONS . . . 1680
FILES . . . 1681
SEE ALSO . . . 1681
AUTHOR . . . 1681
Index . . . 1681
zabbix_sender . . . 1681
NAME . . . 1681
SYNOPSIS . . . 1682
DESCRIPTION . . . 1682
OPTIONS . . . 1682
EXIT STATUS . . . 1684
EXAMPLES . . . 1684
SEE ALSO . . . 1685
AUTHOR . . . 1685
Index . . . 1685
zabbix_server . . . 1686
NAME . . . 1686
SYNOPSIS . . . 1686
DESCRIPTION . . . 1686
OPTIONS . . . 1686
FILES . . . 1687
SEE ALSO . . . 1687
AUTHOR . . . 1687
Index . . . 1687
zabbix_web_service . . . 1687
NAME . . . 1687
SYNOPSIS . . . 1688
DESCRIPTION . . . 1688
OPTIONS . . . 1688
FILES . . . 1688
SEE ALSO . . . 1688
AUTHOR . . . 1688
Index . . . 1688

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 e-mail 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).

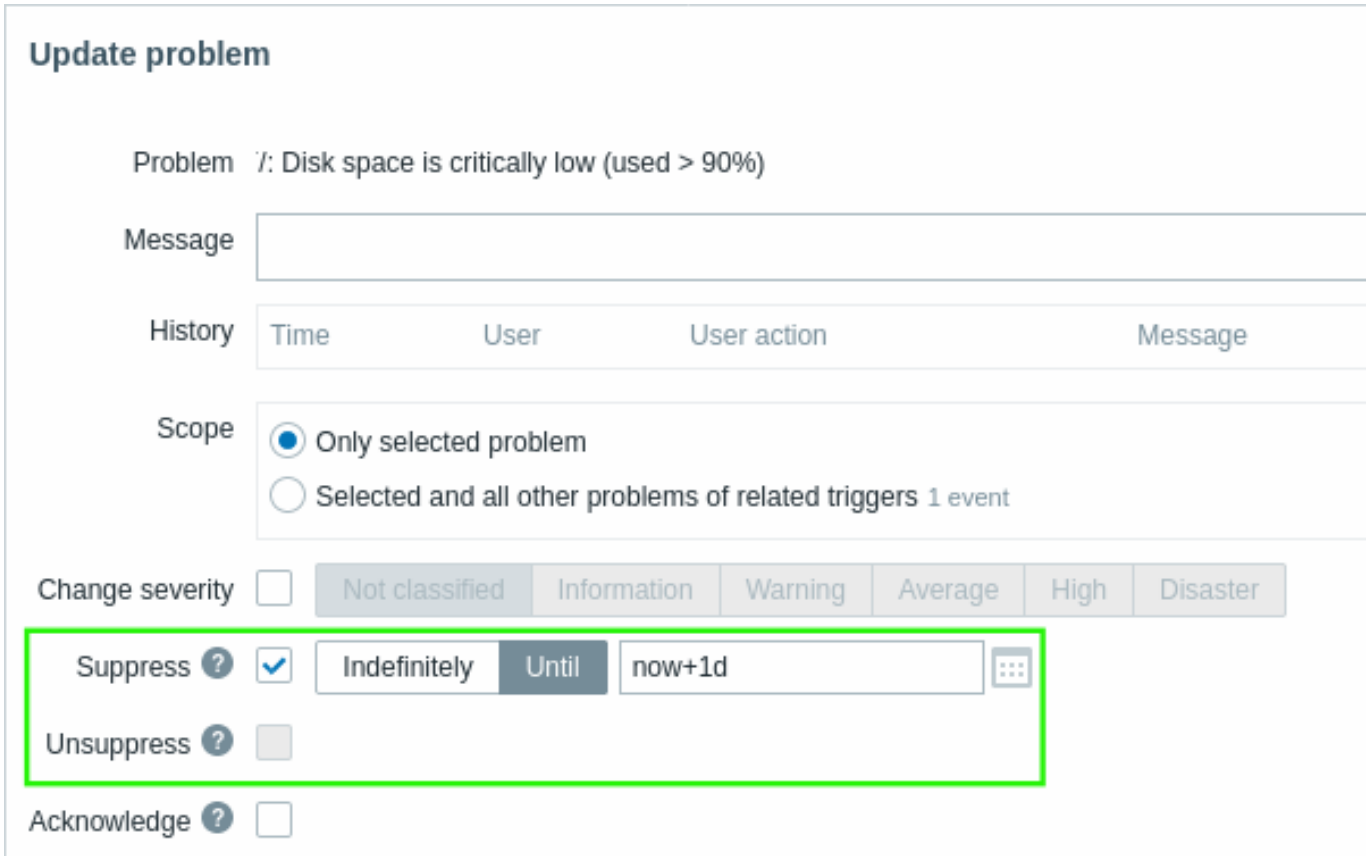
Data flow

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 6.2.0

Single problem suppression In the new version you may prioritize the problem list by hiding those issues that can be dealt with at a later time. This is done by suppressing single problems for a defined time period.

A problem can be suppressed through the problem update window that is opened when acknowledging a problem:



For more details see Problem suppression.

Multiple LDAP sources It is now possible to define multiple LDAP servers for LDAP authentication. This is useful when different LDAP servers are used to authenticate different groups of users.

Once the servers are configured in Zabbix, it becomes possible to select the required LDAP server for the respective user group, in user group configuration.

Storage of secrets It is now possible to store some sensitive information from Zabbix in CyberArk Vault CV2. Similarly to storing secrets in HashiCorp Vault, introduced in Zabbix 5.2, CyberArk Vault can be used for:

- user macro values

- database access credentials

Zabbix provides read-only access to the secrets in vault.

See also: CyberArk configuration

Secure password hashing In Zabbix 5.0 the password hashing algorithm was changed from MD5 to the more secure bcrypt. However, MD5 cryptography remained supported to ensure smooth upgrades from previous versions. MD5 hashing was only used for some users upon the first login after an upgrade - to convert passwords with not reliable hashes from MD5 to bcrypt. Now support of MD5 cryptography has been dropped completely.

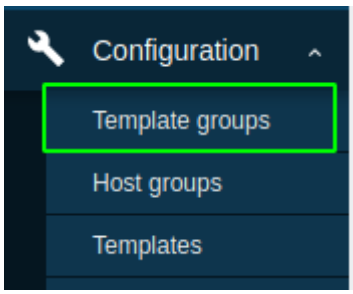
Reload proxy configuration It is now possible to force the reload of configuration for proxies from the server. It can be done in two ways:

- by Zabbix server runtime control command (e.g. `zabbix_server -R proxy_config_cache_reload`)
- from the frontend (in the list of proxies or the proxy editing form)

It is also now possible for passive proxies to request configuration from the server using the `config_cache_reload` proxy runtime control command.

Separate groups for templates Previously, host groups were used to organize both hosts and templates. Now this functionality has been split into template groups, which may contain templates only, and host groups, which may contain hosts only.

A new subsection **Template groups** has been added to the *Configuration* menu.



User role and user group permissions are now defined separately for host groups and template groups.

AWS EC2 monitoring A new template *AWS EC2 by HTTP* has been added allowing to quickly deploy Zabbix monitoring of AWS EC2 and attached AWS EBS volumes by HTTP.

You can get this template:

- In *Configuration* → *Templates* in new installations;
- When upgrading from previous versions, the latest templates can be downloaded from the Zabbix Git repository and manually imported into Zabbix in the *Configuration* → *Templates* section. If a template with the same name already exists, check the *Delete missing* option before importing to achieve a clean import. This way the items that have been excluded from the updated template will be removed (note, that history of the deleted items will be lost).

Configuration cache Faster configuration sync

Incremental configuration cache synchronization has been added for hosts, host tags, items, item tags, item preprocessing, triggers, trigger tags and functions to lessen synchronization time and database load when configuration is being updated on an already running Zabbix server or Zabbix proxy.

User macro cache

To reduce configuration cache locking and, therefore, improve performance, user macro values are now stored in a separate user macro cache instead of the configuration cache.

Items Immediate checks for new items

Previously, newly added items were first checked at a random time within their update interval. Now new items and discovery rules will be checked within 60 seconds of their creation, unless they have Scheduling or Flexible update interval with the *Update interval* parameter set to 0.

Windows registry monitoring

Windows registry monitoring is now supported out-of-the-box in Zabbix. Two new keys have been added to the Windows Zabbix agent and agent 2:

- **registry.data**[key,<value name>] - return data for the specified value name in the Windows Registry key
- **registry.get**[key,<mode>,<name regexp>] - list of Windows Registry values or keys located at given key; returns JSON

See also: Windows Zabbix agent items

Low-level discovery of OS processes

Zabbix now provides a native solution to discover running OS processes. A new item **proc.get[]** can be used in discovery rules to return a list of running processes/threads or summarized data grouped by process name.

See also:

- Zabbix agent item keys
- Process parameters returned by proc.get item

Extended VMware monitoring

Multiple new items are now available for discovering and monitoring VMware:

- vSphere Distributed Switch ports:**vmware.dvswitch.discovery[]**, **vmware.dvswitch.fetchports.get[]**;
- Virtual machines: **vmware.vm.state[]**, **vmware.vm.tools[]**, **vmware.vm.snapshot.get[]**, **vmware.vm.consolidationneeded[]**, **vmware.vm.attribute[]**;
- Hypervisors: **vmware.hv.connectionstate[]**, **vmware.hv.hw.serialnumber[]**, **vmware.hv.hw.sensors.get[]**, **vmware.hv.net.if.discovery[]**, **vmware.hv.network.linkspeed[]**;
- Resource pools: **vmware.rp.cpu.usage[]**, **vmware.rp.memory[]**.

The items **vmware.hv.network.in[]** and **vmware.hv.network.out[]**, used for monitoring hypervisor network traffic, now support additional NIC counters. Four new mode options have been added to both items: packets, dropped, errors, broadcast.

The item **vmware.vm.discovery[]**, used for virtual machine discovery, now returns additional discovery fields, including user-defined custom attribute values.

The items **vmware.hv.discovery[]**, used for hypervisor discovery, and **vmware.cluster.discovery[]**, used for clusters discovery, now return information about resource pools.

Active checks affect host availability Active Zabbix agent items now also affect host availability as seen in *Monitoring -> Hosts* or *Configuration -> Hosts*.

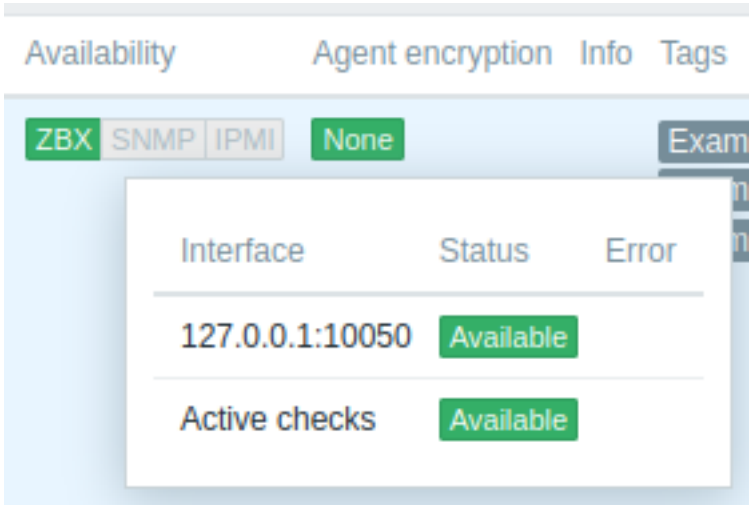
To determine active check availability heartbeat messages are now sent in the active check thread. The frequency of the heartbeat messages is set by the new `HeartbeatFrequency` parameter in Zabbix agent and agent 2 configurations (60 seconds by default, 0-3600 range). Active checks are considered unavailable when the active check heartbeat is older than 2 x HeartbeatFrequency seconds.

> **Attention:**
> This functionality will only work if the latest version of Zabbix agent or Zabbix agent 2 is used. Agents of older versions are not sending any heartbeats, so the availability of their hosts will remain unknown.

Active agent availability is counted towards the total agent availability in the same way as a passive interface is, for example:

- if a passive agent interface is available, and active checks are available - the total availability is green (all available)
- if a passive agent interface is available, but active checks are not available - the total availability is yellow (mixed)
- if a passive agent interface is available, but active checks are unknown - the total availability is gray (unknown)

Additionally, active agent availability is listed, as a separate row, in the popup list of available agent interfaces. This popup is opened when hovering over the host availability icon for the Zabbix agent interface in *Monitoring -> Hosts* or *Configuration -> Hosts*.



A new `zabbix[host,active_agent,available]` internal item allows to monitor the availability of active checks.

Discovered host customization For hosts, created from host prototypes, the following parameters can now be modified after discovery:

- Tags

- Macros
- Templates

It is possible to link additional templates and add more tags and macros as well as update or remove existing tags and macros. Templates inherited from a host prototype cannot be unlinked from a discovered host.

Graph widget The vector graph widget has been improved and it is now possible to:

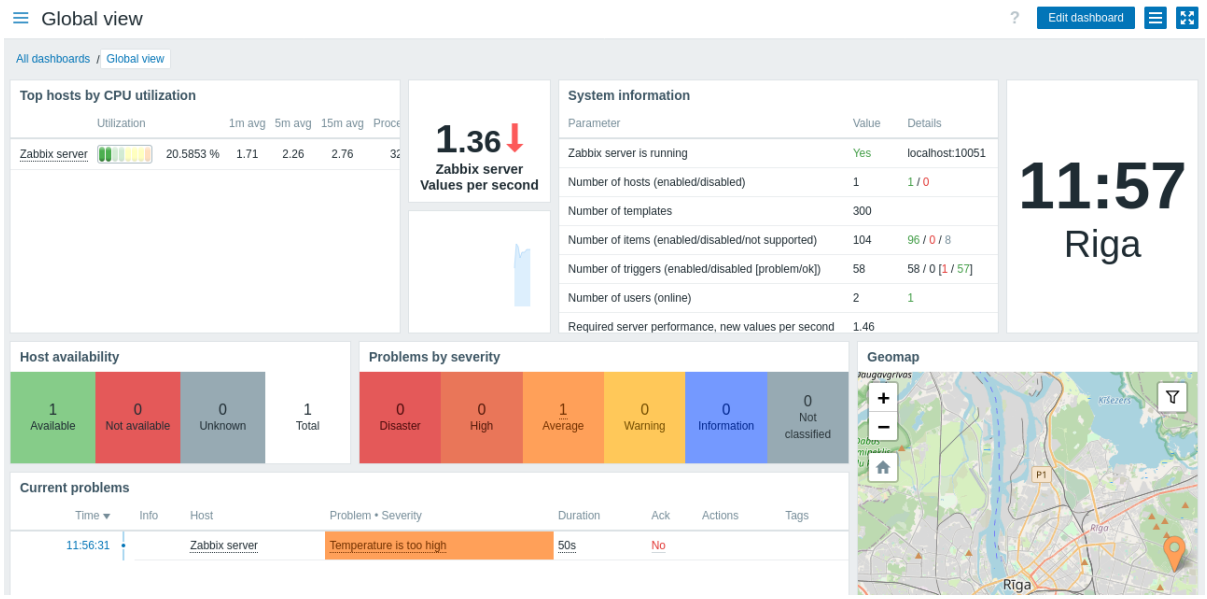
- display stacked graph (with filled areas)
- add a list of items
- clone a data set
- show lines for minimum, maximum and average values
- show the trigger line for simple triggers
- show percentiles
- show working time
- show minimum, maximum and average item values in the legend
- have columns in the legend

Frontend Minimum required PHP version

The minimum required PHP version has been raised from 7.2.5 to 7.4.0.

Default dashboard updated

The ”Global view” default dashboard for new Zabbix installations has been updated to include the latest dashboard widgets.



Digital clock widget

The Clock dashboard widget has been updated to display a digital clock as well.



Direct links to documentation

All frontend forms now have direct links to the corresponding parts of the documentation.

This is implemented by adding a help link to the frontend form headers:



Retrieve latest item value in latest data

The option to retrieve the latest item value immediately is now available in the latest data page, both as an ”Execute now” button below the list of items and as an option in the item menu when clicking on the item name.

In previous versions the same functionality was available in the Configuration section (item/discovery rule form and lists) only.

As another improvement, it is now also possible to ”Execute now” dependent items.

The option to ”execute now” depends on host permissions and user role settings; for more details see:

- Latest data
- Execute now

Filter settings remembered

In several Monitoring pages (*Problems*, *Hosts*, *Latest data*) current filter settings are now remembered in the user profile. When the user opens the page again, the filter settings will have stayed the same.

Additionally, the marking of a changed (but not saved) favorite filter is now a green dot next to the filter name, instead of the filter name in italics.

Miscellaneous

- The form for API token creation and editing is now opened in a modal (popup) window.
- Locale "British English" (en_GB) is available again in Zabbix frontend.
- German and Vietnamese languages are now enabled.

Macros

- {INVENTORY.*} macros are now supported in script-type items and manual host action scripts for Zabbix server and Zabbix proxy.

HMAC function for JavaScript A new function has been added to the JavaScript engine allowing to return HMAC hash:

- `hmac('<hash type>',key,string)`

This is useful for cases when hash-based message authentication code (HMAC) is required for signing requests. MD5 and SHA256 hash types are supported, e. g.:

- `hmac('md5',key,string)`
- `hmac('sha256',key,string)`

Breaking changes PHP version

PHP versions below 7.4.0 are no longer supported. The minimum required PHP version has been raised from 7.2.5 to 7.4.0.

6 What's new in Zabbix 6.2.1

MariaDB 10.8 support The maximum supported version for MariaDB is now 10.8.X.

TimescaleDB 2.6 support The maximum supported version for TimescaleDB is now 2.6.

Templates A new template *HPE Synergy by HTTP* is available.

See setup instructions for HTTP templates.

You can get this template:

- In *Configuration* → *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 *Configuration* → *Templates* you can import them manually into Zabbix.

Extended VMware monitoring New items are now available for:

- Monitoring VMware alarms: **vmware.alarms.get[]**, **vmware.cluster.alarms.get[]**, **vmware.datastore.alarms.get[]**, **vmware.dc.alarms.get[]**, **vmware.hv.alarms.get[]**, **vmware.vm.alarms.get[]**.
- Collecting tag information from different VMware components: **vmware.cluster.tags.get[]**, **vmware.datastore.tags.get[]**, **vmware.dc.tags.get[]**, **vmware.hv.tags.get[]**, **vmware.vm.tags.get[]**.
- Returning property values: **vmware.cluster.property[]**, **vmware.datastore.property[]**, **vmware.hv.property[]**, **vmware.vm.property[]**

Multiple VMware discovery items now also return a JSON array containing tags.

The items **vmware.datastore.discovery[]** and **vmware.hv.datastore.discovery[]** now also return {#DATASTORE.UUID} macro with a datacenter identifier.

7 What's new in Zabbix 6.2.2

Items The `proc.get[]` agent item on FreeBSD now also returns the jail name in the `jname` property.

Month abbreviated with capital letter A ”month” is now abbreviated with the capital ”M” in the frontend. Previously it was abbreviated with the small ”m”, overlapping with the abbreviation of a minute.

Templates New templates are available:

- *AWS RDS instance by HTTP*
- *AWS S3 bucket by HTTP*
- *Azure by HTTP*
- *OPNsense by SNMP*

See setup instructions for HTTP templates.

You can get these templates:

- In *Configuration* → *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 *Configuration* → *Templates* you can import them manually into Zabbix.

TimescaleDB 2.7 support The maximum supported version for TimescaleDB is now 2.7.

RHEL packages renamed RHEL packages have been renamed by adding a ”release” word in the name:

Naming	Package name
Old	`zabbix-agent-6.2.1-1.el9.x86_64.rpm`
New	`zabbix-agent-6.2.2-release1.el9.x86_64.rpm`

There is no functional change associated with this change.

This is necessary as preparation for providing packages of minor version (i.e. 6.2.x) release candidates, expected to start with 6.2.3. The naming change will ensure that for someone who has both stable and unstable repositories enabled on their system, repository updates will be received in the correct order. This naming change is for RHEL packages only.

8 What’s new in Zabbix 6.2.3

Expression macros {ITEM.KEY<1-9>} macros are now supported inside expression macros.

Templates A new *AWS by HTTP* template is available now.

See setup instructions for HTTP templates.

Packages SQL scripts have been moved from the `/usr/share/doc` directory to `/usr/share` in Zabbix packages.

9 What’s new in Zabbix 6.2.4

TimescaleDB 2.8 support The maximum supported version for TimescaleDB is now 2.8.

PostgreSQL 15 support PostgreSQL 15 is now supported. Note that TimescaleDB **does not** support PostgreSQL 15 yet.

Possible to build Zabbix agent 2 offline Zabbix agent 2 now can be built offline. The source tarball now includes the `src/go/vendor` directory, which should make sure that golang is not forced to download dependency modules automatically. It is still possible to update to the latest modules manually by using `go mod tidy` or `go get` commands.

PostgreSQL plugin loadable The PostgreSQL plugin is now loadable in Zabbix agent 2 (previously built-in).

See also: PostgreSQL loadable plugin repository

Extended VMware monitoring New items

New items are now available for VMware monitoring:

- **vmware.datastore.perfcounter[]**
- **vmware.hv.diskinfo.get[]**

Additional discovery fields

Items, used for VMware discovery, now return additional discovery fields:

- **vmware.datastore.discovery[]** returns {#DATASTORE.TYPE} macro.
- **vmware.hv.datastore.discovery[]** returns {#DATASTORE.TYPE} macro and `datastore_extent` array.

vmware.vm.discovery output format updated

The **vmware.vm.discovery** item now returns the *vm_customattribute* array instead of *vm.customattribute*. This formatting change has been implemented to simplify further processing of the returned JSON (for example, in preprocessing steps or LLD macros).

Tag reading from VMware

VMware items reporting various VMware tags (e.g., `vmware.datastore.tags.get[]`) are now supported since vSphere version 6.5 (previously vSphere 7.0 Update 2).

Templates The template *Azure by HTTP* has been updated and now includes metrics to monitor Microsoft Azure MySQL servers out-of-the-box.

You can get this template:

- In Configuration → 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 Configuration → Templates you can import them manually into Zabbix.

Frontend Miscellaneous

- Warnings about incorrect housekeeping configuration for TimescaleDB are now displayed if history or trend tables contain compressed chunks, but *Override item history period* or *Override item trend period* options are disabled. For more information, see TimescaleDB setup.

10 What's new in Zabbix 6.2.5

Templates The template *Azure by HTTP* has been updated and now includes metrics to monitor Microsoft PostgreSQL flexible servers and Microsoft PostgreSQL single server out-of-the-box.

You can get this template:

- In Configuration → 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 Configuration → Templates you can import them manually into Zabbix.

Reporting file systems with zero inodes `vfs.fs.get` agent items are now capable of reporting file systems with the inode count equal to zero, which can be the case for file systems with dynamic inodes (e.g. btrfs).

Additionally `vfs.fs.inode` items now will not become unsupported in such cases with mode set to 'pfree' or 'pused'. Instead the pfree/pused values for such file systems will be reported as "100" and "0" respectively.

11 What's new in Zabbix 6.2.6

Improved performance of history syncers The performance of history syncers has been improved by introducing a new read-write lock. This reduces locking between history syncers, trappers and proxy pollers by using a shared read lock while accessing the configuration cache. The new lock can be write locked only by the configuration syncer performing a configuration cache reload.

12 What's new in Zabbix 6.2.7

Breaking changes Loadable plugin versioning

Loadable plugins for Zabbix agent 2 now use the same versioning system as Zabbix itself. The following version changes have been made:

- MongoDB 1.2.0 -> MongoDB 6.2.7
- PostgreSQL 1.2.1 -> PostgreSQL 6.2.7

These plugins are supported for any minor version of Zabbix 6.2. Note that source code repository for each plugin now contains a dedicated *release/6.2* branch (previously, there was only *master* branch).

MariaDB 10.10 support The maximum supported version for MariaDB is now 10.10.X.

TimescaleDB 2.9 support The maximum supported version for TimescaleDB is now 2.9.

Configuration import Previously, the import process would fail on a UUID mismatch of an importable entity (host group, item, graph, etc). For example, it was not possible to import a host group if a host group with the same name already existed on the host.

In the new version the import will not fail because of a UUID mismatch; instead the entity will be matched by the uniquenes criteria such as entity ID (name). The entity will be imported, and the UUID will be updated to the UUID of the imported entity.

In another improvement, when removing template linkage (the *Delete missing* option for template linkage) through the imported template or host, the inherited entities of the unlinked template are no longer removed (template gets unlinked, not unlinked and cleared), unless these entities are missing in the import file **and** the *Delete missing* option for the specific entity is marked.

Due to this change, the warning message when marking the *Delete missing* option for template linkage will no longer be displayed.

Query separate tablespaces in Oracle databases with Zabbix agent 2 The following Zabbix agent 2 items, supported for the Oracle plugin, now have additional optional parameters:

- oracle.diskgroups.stats[<existingParameters>,**<diskgroup>**]
- oracle.archive.info[<existingParameters>,**<destination>**]
- oracle.cdb.info[<existingParameters>,**<database>**]
- oracle.pdb.info[<existingParameters>,**<database>**]
- oracle.ts.stats[<existingParameters>,**<tablespace>**,**<type>**]

These parameters allow to query separate instances of data instead of all data, thus improving performance.

Retrieving additional information with docker.container_info[] The `docker.container_info[]` Zabbix agent 2 item now supports the option to retrieve either partial (short) or full low-level information about a Docker container.

Encrypted MongoDB plugin connection MongoDB plugin now supports TLS encryption when connecting to MongoDB using named sessions.

Updated plugin (MongoDB plugin 1.2.1) is included into Zabbix official packages starting from Zabbix 6.2.7. Note that MongoDB is a loadable plugin and can be installed separately either from packages or from sources. The plugin will work with any minor version of Zabbix 6.2. For more details see MongoDB plugin.

Webhook integrations A new media type *LINE* is now available allowing to use the webhook feature for sending notifications about Zabbix events to *LINE* messenger.

Templates New templates

New templates are available now:

- *Control-M enterprise manager by HTTP*
- *Control-M server by HTTP*
- *Veeam Backup Enterprise Manager by HTTP*
- *Veeam Backup and Replication by HTTP*

See setup instructions for HTTP templates.

Updated templates

The template *Oracle by Zabbix agent 2* has been updated (multiple static items removed; multiple item prototypes added) according to the changes made to multiple Zabbix agent 2 items.

For more information about the updates, see Template changes.

You can get this template:

- In *Configuration* → *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 *Configuration* → *Templates*, you can import them manually into Zabbix.

Golang library for Windows updated Golang library used by Zabbix agent 2 in conjunction with MongoDB or PostgreSQL plugins to monitor Windows now is `github.com/Microsoft/go-winio`, version 0.6.0 (previously `github.com/natefinch/npipe`). See also Golang libraries, MongoDB plugin dependencies, and PostgreSQL plugin dependencies.

Open file descriptor limit increase for Zabbix agent 2 The systemd service file shipped in Zabbix agent 2 packages now declares the open file descriptor limit of 8196. Previously, the system default limit of 1024 has been used. The new limit is sufficient for the default Zabbix agent 2 configuration. If you have a non-standard agent 2 configuration, for example, use additional plugins or extended features, this limit may need to be manually increased further. In this case, adjust the `LimitNOFILE` parameter in the systemd unit file.

13 What's new in Zabbix 6.2.8

PHP support The maximum supported version for PHP is now 8.2.

Limits for JavaScript objects in preprocessing The following limits for JavaScript objects in preprocessing have been introduced:

- The total size of all messages that can be logged with the `log()` method has been limited to 8 MB per script execution.
- The initialization of multiple `HttpRequest` objects has been limited to 10 per script execution.
- The total length of header fields that can be added to a single `HttpRequest` object with the `addHeader()` method has been limited to 128 Kbytes (special characters and header names included).

14 What's new in Zabbix 6.2.9

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.

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 off of 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.

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.*

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.*

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 off of 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.

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.

agent autoregistration

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

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> service zabbix-server start
```

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

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.	

Option	Description	Target
diaginfo[=<**target**>]	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)
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.	
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. 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.	

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
- `discoverer` - process for discovery of devices
- `escalator` - process for escalation of actions
- `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
- `problem housekeeper` - process for removing problems of deleted triggers
- `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
- `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
- `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**.

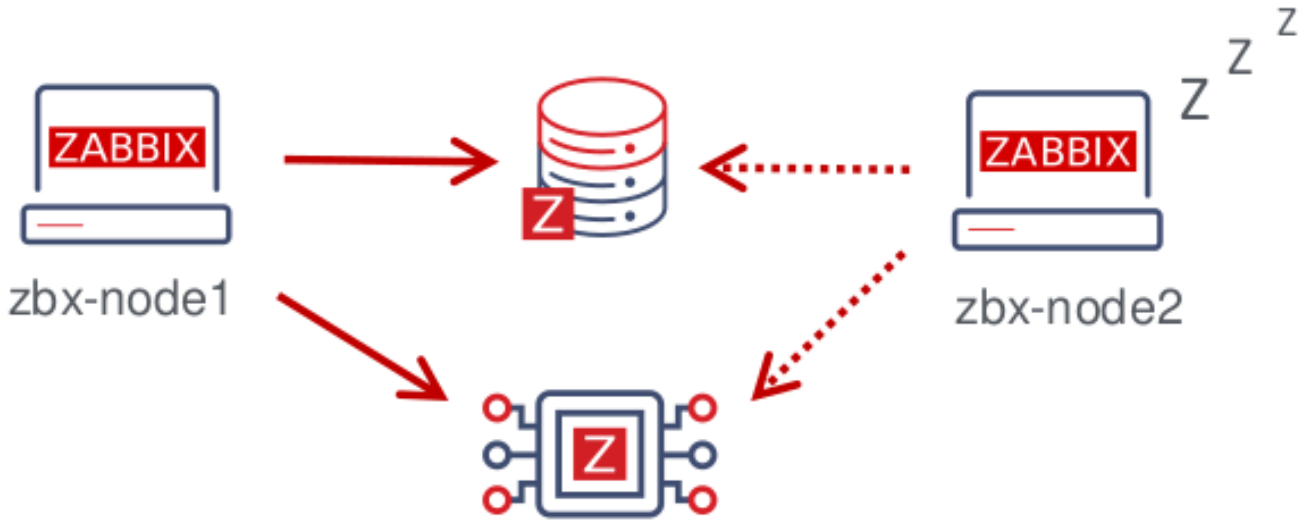
```
Server=zabbix-node-01,zabbix-node-02
```

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 extremely efficient because of 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

Zabbix agent is supported on the following platforms:

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

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> service zabbix-agent start
```

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

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'.

Option	Description	Target
log_level_decrease[=<target>]	Decrease log level. If target is not specified, all processes are affected.	
userparameter_reload	Reload user parameters from the current configuration file. Note that UserParameter is the only agent configuration option that will be reloaded.	

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 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.

See also: Plugin development guidelines.

Supported platforms

Zabbix agent 2 is supported on the following platforms:

- Windows (all desktop and server versions since XP; also available as a pre-compiled binary)
- Linux (also available in distribution packages)

Installation

Zabbix agent 2 is available in pre-compiled Zabbix packages. To compile Zabbix agent 2 from sources you have to specify the `--enable-agent2` configure option.

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 number and the 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 user parameters from the current configuration file. Note that UserParameter is the only agent configuration option that will be reloaded.
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.
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> service zabbix-proxy start
```

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

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[=<**target**>]	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.	

Option	Description	Target
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.	

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
- `heartbeat sender` - proxy heartbeat sender
- `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 temporarily will not accept any new requests.

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
RHEL/CentOS 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-0.9.27.jar
lib/logback-classic-0.9.27.jar
lib/slf4j-api-1.6.1.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/CentOS packages

Overview

If installed from RHEL/CentOS 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:

```
# service zabbix-java-gateway restart
```

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:

```
# service zabbix-java-gateway restart
```

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.

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 pooling 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.

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.

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 is 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 of.
  -h --help                        Give this help.
  -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

Zabbix web service is available in pre-compiled Zabbix packages available for download at Zabbix website. To compile Zabbix web service from sources, specify the `--enable-webservice` configure option.

See also:

- Configuration file options for zabbix_web_service;
- Setting up scheduled reports

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
```

2 Requirements

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 box 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	-

Platform	Server	Agent	Agent2
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).

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

Mandatory requirements are needed always. Optional requirements are needed for the support of the specific function.

Software	Mandatory status	Supported versions	Comments
MySQL/Percona	One of	8.0.X	Required if MySQL (or Percona) is used as Zabbix backend database. InnoDB engine is required. We recommend using the C API (libmysqlclient) library for building server/proxy. For MySQL versions 8.0.0-8.0.28, see also: Possible deadlocks.
MariaDB		10.5.00-10.11.X	InnoDB engine is required. We recommend using the MariaDB Connector/C library for building server/proxy. Added support for MariaDB versions: - 10.8.X since Zabbix 6.2.1; - 10.10.X since Zabbix 6.2.7; - 10.11.X since Zabbix 6.2.9. See also: Possible deadlocks.
Oracle		19c - 21c	Required if Oracle is used as Zabbix backend database.
PostgreSQL		13.0-15.X	Required if PostgreSQL is used as Zabbix backend database. Added support for PostgreSQL versions: - 15.X since Zabbix 6.2.4.
TimescaleDB for PostgreSQL		2.0.1-2.10	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. Added support for TimescaleDB versions: - 2.6 since Zabbix 6.2.1; - 2.7 since Zabbix 6.2.2; - 2.8 since Zabbix 6.2.4; - 2.9 since Zabbix 6.2.7; - 2.10 since Zabbix 6.2.9.
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.

Software	Mandatory status	Supported versions	Comments
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.

Mandatory requirements are needed always. Optional requirements are needed for the support of the specific function.

Software	Mandatory status	Version	Comments
Apache	yes	1.3.12 or later	
PHP		7.4.0 - 8.2.X	It is recommended to use PHP 8.0 or newer, because PHP 7.4 is no longer supported by the vendor. Added support for PHP versions: - 8.2 since Zabbix 6.2.8.
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*).
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.
OneLogin's 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

Mandatory requirements are needed always. Optional requirements are needed for the support of the 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.
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); 0.6.0 or higher (libssh). libssh is supported since Zabbix 4.4.6.

Requirement	Mandatory status	Description
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.
libnetsnmp		Required for SNMP support. Version 5.3.0 or higher.
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.

Golang 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 golang.
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 plugin.
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 plugin.
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	
golang.org/x/sys		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	Comments
android-json	Yes	Version 4.3r1 or higher. 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		Version 1.2.9 or higher.
logback-core		Version 1.2.9 or higher.
slf4j-api		Version 1.7.32 or higher.

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 table below provides information about OpenJDK versions used for building Zabbix packages by distribution:

Distribution	OpenJDK version
RHEL/CentOS 8	1.8.0
RHEL/CentOS 7	1.8.0
SLES 15	11.0.4
SLES 12	1.8.0
Debian 10	11.0.8
Ubuntu 20.04	11.0.8
Ubuntu 18.04	11.0.8

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.
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	10053
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.

Plugins

1 PostgreSQL plugin dependencies

Overview

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

Golang 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.
github.com/jackc/chunkreader/v2	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/v2		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 MongoDB plugin mongo-driver lib 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.

Golang 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.

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 *Administration → 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/Centos, install mod_ssl package:

```
dnf install mod_ssl
```

Create directory for SSL keys:

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

Create 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. We will use *example.com* in this article.

```
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 Apache SSL configuration:

```
/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

Add a virtual host to Apache configuration and set permanent redirect for document root to Zabbix SSL URL. Do not forget to replace *example.com* with the actual name of the server.

/etc/httpd/conf/httpd.conf

```
#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 to enable HSTS policy on the web server.

For example, to enable HSTS policy for your Zabbix frontend in Apache configuration:

/etc/httpd/conf/httpd.conf

add the following directive to your virtual host's configuration:

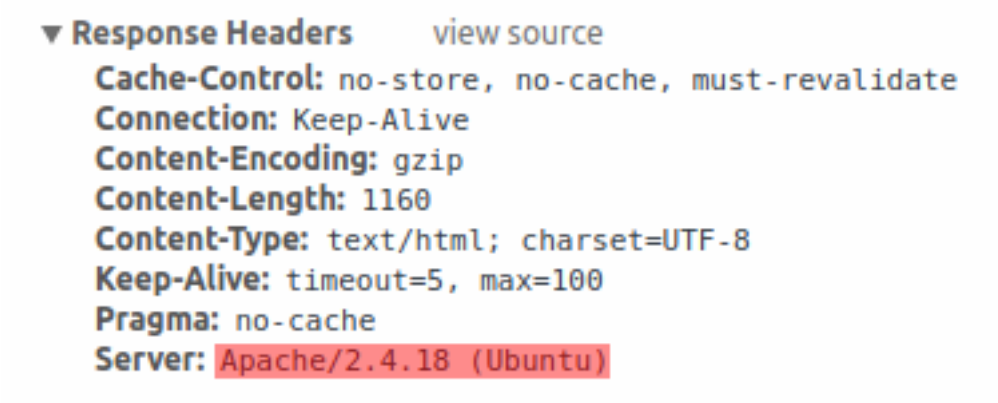
```
<VirtualHost *:443>
   Header set Strict-Transport-Security "max-age=31536000"
</VirtualHost>
```

Restart the Apache service to apply the changes:

```
systemctl restart httpd.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
```

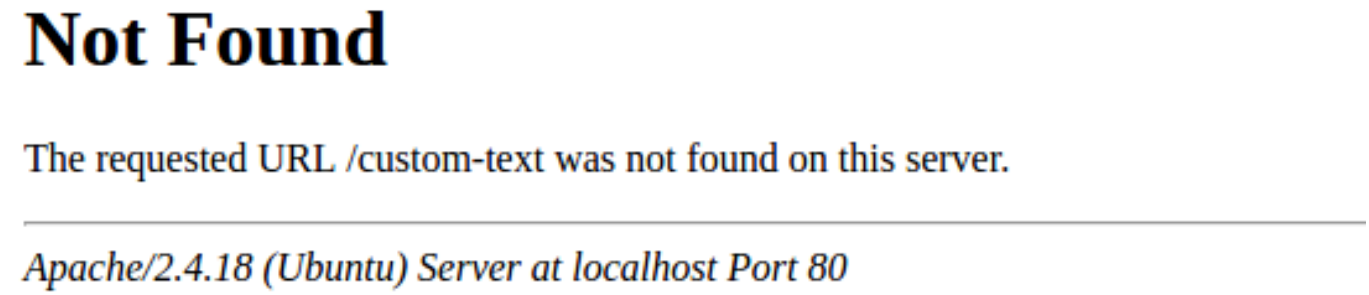
Web server restart is required for configuration file changes to be applied.

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:

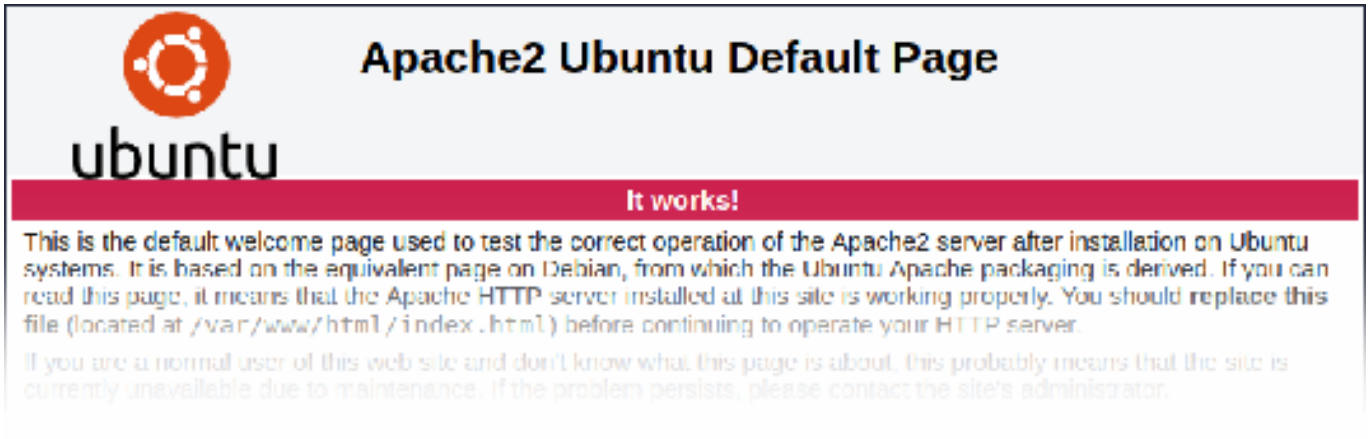


Default error pages should be replaced/removed as part of the web server hardening process. The ”ErrorDocument” directive can be used to define a custom error page/text for Apache web server (used as an example).

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 response header* set to `SAMEORIGIN`, meaning 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 Admins can modify *iframe sandboxing* and *X-Frame-Options HTTP response header* parameters as needed. Please carefully weigh the risks and benefits before changing default settings. Turning sandboxing or X-Frame-Options 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 `.htacess` 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-6.2.0.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"
```

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 golang.org 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, CentOS)

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.

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 PCRE library version 8.XX from pcre.org (http://ftp.pcre.org/pub/pcre/) 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
```

Installing OpenSSL development libraries

1. Download 32 and 64 bit builds from https://curl.se/windows/
2. Extract files into *c:\dev\openssl32* and *c:\dev\openssl* directories accordingly.
3. After that remove extracted **.dll.a* (dll call wrapper libraries) as MinGW prioritizes them before static libraries.

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.

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
```

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 PCRE or PCRE2 (supported since Zabbix 6.0) library from pcre.org repository: (https://github.com/PhilipHazel/pcre2/releases/download/pcre2-10.39/pcre2-10.39.zip)
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 PCREINCDIR=E:\pcre2-10.39-install\include PCRELIBDIR=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:\Ope TLSLIBDIR=C:\OpenSSL-Win64-111-static\lib PCREINCDIR=E:\pcre2-10.39-install\include PCRELIBDIR=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:\OpenS PCREINCDIR=E:\pcre2-10.39-install\include PCRELIBDIR=E:\pcre2-10.39-install\lib E:\zabbix-4.4 nmake /K -f Makefile_agent TLS=openssl TLSINCDIR=C:\OpenSSL-Win64-111-static\include TLSLIBDIR=C:\OpenSSL-Win64-111-static\lib PCREINCDIR=E:\pcre2-10.39-install\include PCRELIBDIR=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 <code>`

CFLAGS = $(CFLAGS) /DHAVE_OPENSSL /DHAVE_OPENSSL_WITH_PSK</code>and replace it with CFLAGS = $(CFLAGS) /DHAVE_OPENSSL

`* In ''Makefile_common.inc'' add ''/NODEFAULTLIB:LIBCMT'' i.e. find <code>`

/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO /PDB:$(TARGETDIR)\$(TARGETNAME).pdb</code>and replace it with `/MANIFESTUAC:"level='asInvoker' uiAccess='false'" /DYNAMICBASE:NO /PDB:$(TARGETDIR)\$(TARGETNAME) /NODEFAULTLIB:LIBCMT`

4 Installation from packages

From Zabbix official repository

Zabbix SIA provides official RPM and DEB packages for:

- Red Hat Enterprise Linux/CentOS
- 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.

Note, that though some OS distributions (in particular, Debian-based distributions) provide their own Zabbix packages, these packages are not supported by Zabbix. Zabbix packages provided by third parties can be out of date and may lack the latest features and bug fixes. It is recommended to use only official packages from repo.zabbix.com. If you have previously used unofficial Zabbix packages, see notes about upgrading Zabbix packages from OS repositories.

1 Red Hat Enterprise Linux/CentOS

Overview

Official Zabbix 6.2 packages for Red Hat Enterprise Linux, CentOS, and Oracle Linux are available on Zabbix website.

Packages are available with either MySQL/PostgreSQL database and Apache/Nginx web server support.

Zabbix agent packages and utilities *Zabbix get* and *Zabbix sender* are available on Zabbix Official Repository for RHEL 9, RHEL 8, RHEL 7, RHEL 6, and RHEL 5.

Zabbix Official 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*
```

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 allo
# 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:

```
# service httpd restart
```

In addition, Zabbix provides the zabbix-selinux-policy package as part of source RPM packages for RHEL 8 and RHEL 7. 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.conf:

```
# vi /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:

```
# service zabbix-proxy start
# 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/CentOS versions 9, 7, 6 and 5.

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 - $basearch
baseurl=http://repo.zabbix.com/zabbix/6.2/rhel/7/$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 6.2 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/CentOS.

After the frontend and SELinux configuration is done, restart the Apache web server:

```
# service apache2 restart
```

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.conf:

```
# vi /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/CentOS 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 6.2 packages for SUSE Linux Enterprise Server are available on Zabbix website.

Zabbix agent packages and utilities *Zabbix get* and *Zabbix sender* are available on Zabbix Official Repository for SLES 15 and SLES 12.

> **Note:**
> *Verify CA* encryption mode doesn't work on SLES 12 (all minor OS versions) with MySQL due to older MySQL libraries.

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/6.2/sles/15/x86_64/zabbix-release-6.2-1.sles15.noa
# zypper --gpg-auto-import-keys refresh 'Zabbix Official Repository'
```

SLES 12:

```
# rpm -Uvh --nosignature https://repo.zabbix.com/zabbix/6.2/sles/12/x86_64/zabbix-release-6.2-1.sles12.noa
# 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 MySQL support:

```
# zypper install zabbix-server-mysql zabbix-web-mysql zabbix-apache-conf zabbix-agent
```

Substitute 'apache' in the command with 'nginx' if using the package for Nginx web server. See also: Nginx setup for Zabbix on SLES 12/15.

Substitute 'zabbix-agent' with 'zabbix-agent2' in these commands if using Zabbix agent 2 (only SLES 15 SP1+).

To install Zabbix proxy with MySQL support:

```
# zypper install zabbix-proxy-mysql zabbix-sql-scripts
```

Substitute 'mysql' in the commands with 'pgsql' to use PostgreSQL.

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

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

> **Warning:**
> Separate databases are needed for Zabbix server and Zabbix proxy; they cannot use the same database. Therefore, if they are installed on the same host, their databases must be created with different names!

Create the database using the provided instructions for MySQL or PostgreSQL.

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 zabb
```

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 /etc/zabbix/zabbix_server.conf (and zabbix_proxy.conf) to use their respective databases. For example:

```
# vi /etc/zabbix/zabbix_server.conf
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:

- For Apache the configuration file is located in /etc/apache2/conf.d/zabbix.conf. Some PHP settings are already configured. But it's necessary to uncomment the "date.timezone" setting and set the right timezone for you.

```
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. Some PHP settings are already configured. But it's necessary to set the right date.timezone setting for you.

```
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 which 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
```

Substitute 'apache2' with 'nginx' for Nginx web server.

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/6.2/sles/15/x86_64/debuginfo/
gpgcheck=1
gpgkey=http://repo.zabbix.com/zabbix/6.2/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.

The minimum requirement for MSI installation is:

- Windows XP x64 and Server 2003 for Zabbix agent;
- Windows 7 x32 for Zabbix agent 2.

The Zabbix get and sender utilities can also be installed, either together with Zabbix agent/agent 2 or separately.

A 32-bit package cannot be installed on a 64-bit Windows.

All packages come with TLS support, however, configuring TLS is optional.

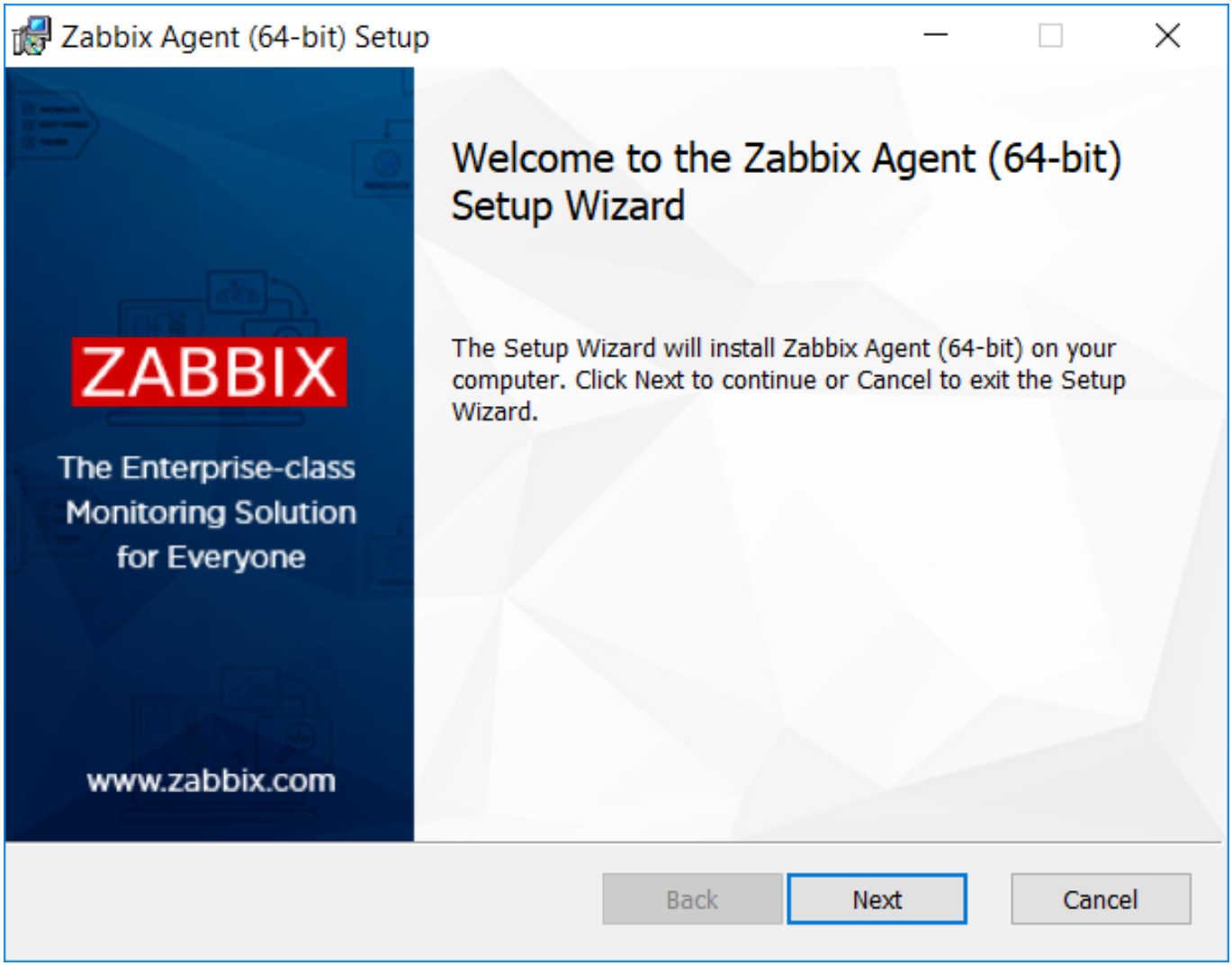
Both UI and command-line based installation is supported.

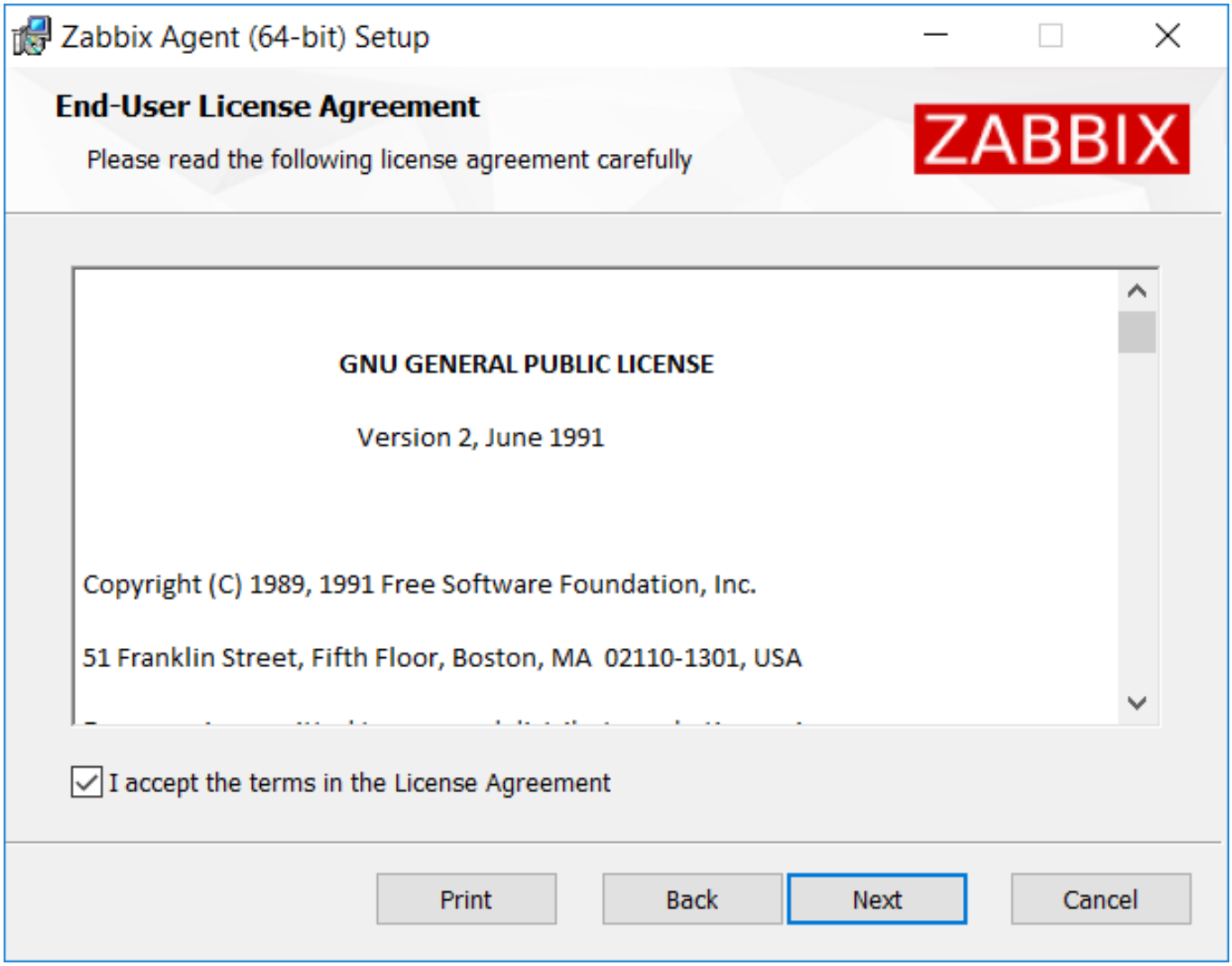
Note:
Although Zabbix installation from MSI installer packages is fully supported, it is recommended to install at least *Microsoft .NET Framework 2* for proper error handling. See Microsoft Download .NET Framework.

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.

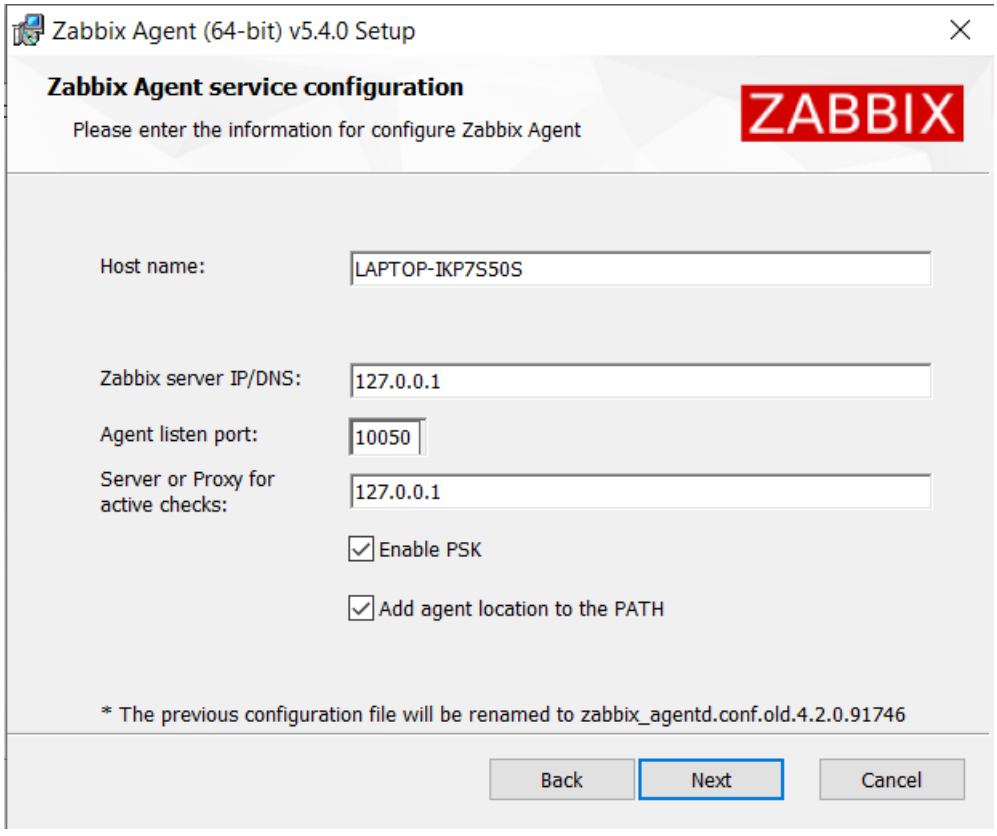
Installation steps

To install, double-click the downloaded MSI file.


Zabbix Agent (64-bit) Setup
ZABBIX
The Enterprise-class Monitoring Solution for Everyone
www.zabbix.com
Welcome to the Zabbix Agent (64-bit) Setup Wizard
The Setup Wizard will install Zabbix Agent (64-bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel

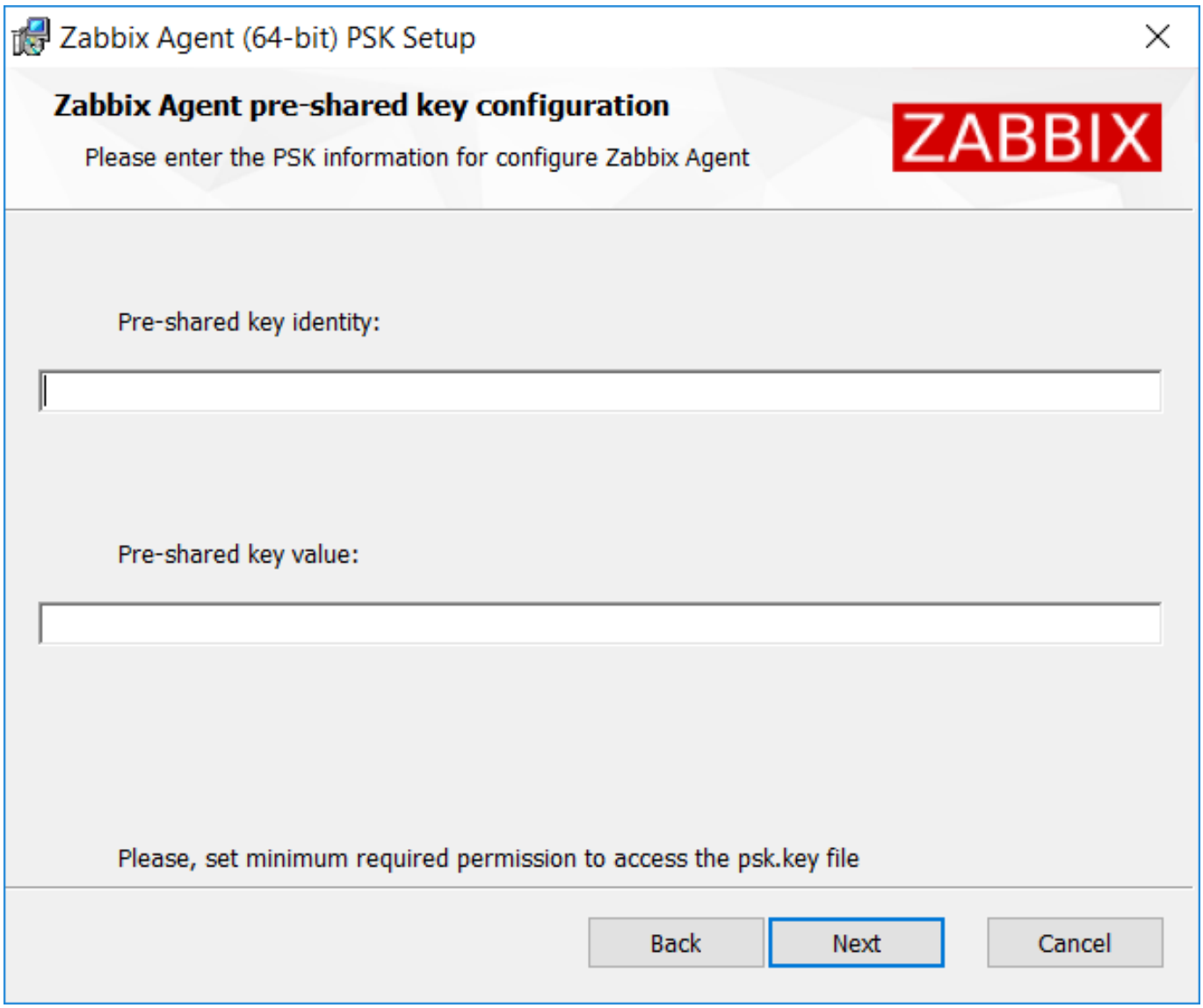


Accept the license to proceed to the next step.

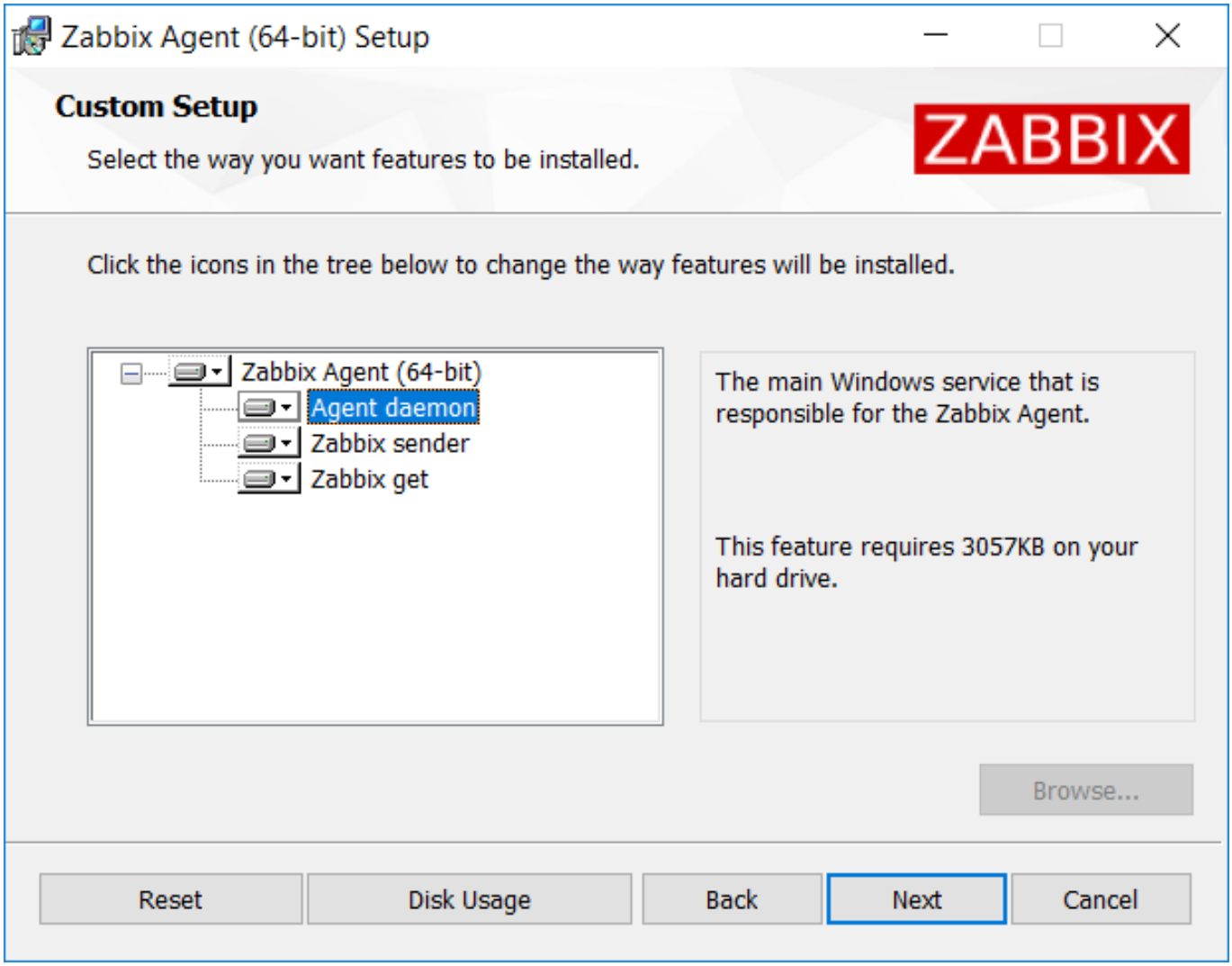


Specify the following parameters.

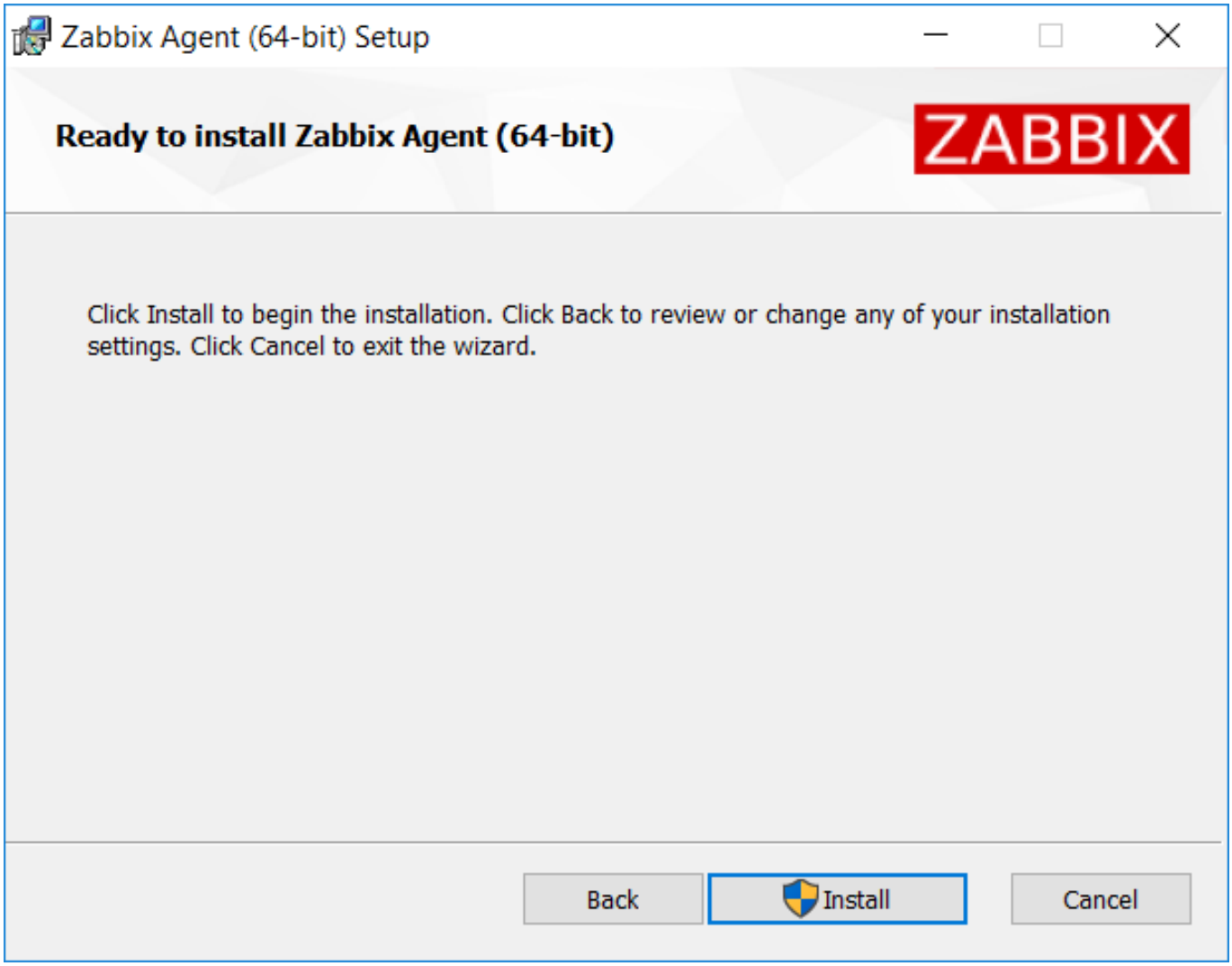
Parameter	Description
Host name	Specify host name.
Zabbix server IP/DNS	Specify IP/DNS of Zabbix server.
Agent listen port	Specify agent listen port (10050 by default).
Server or Proxy for active checks	Specify IP/DNS of Zabbix server/proxy for active agent checks.
Enable PSK	Mark the checkbox to enable TLS support via pre-shared keys.
Add agent location to the PATH	Add agent location to the PATH variable.



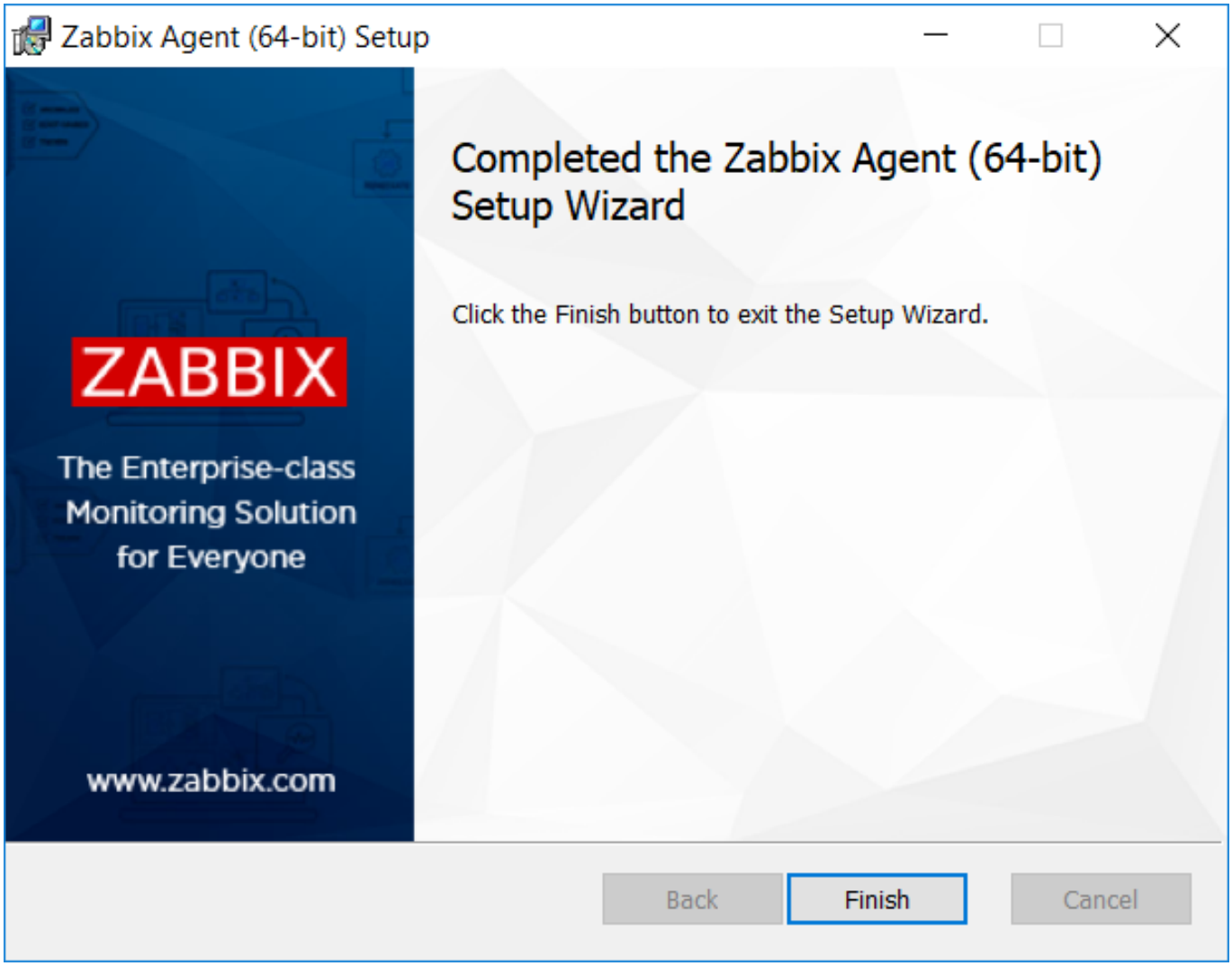
Enter pre-shared key identity and value. This step is only available if you checked *Enable PSK* in the previous step.



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 set of parameters is supported by created MSIs:

Number	Parameter	Description
1	LOGTYPE	
2	LOGFILE	
3	SERVER	
4	LISTENPORT	
5	SERVERACTIVE	
6	HOSTNAME	
7	TIMEOUT	
8	TLSCONNECT	
9	TLSACCEPT	
10	TLSPSKIDENTITY	
11	TLSPSKFILE	
12	TLSPSKVALUE	
13	TLSCAFILE	
14	TLSCRLFILE	
15	TLSSERVERCERTISSUER	
16	TLSSERVERCERTSUBJECT	
17	TLSCERTFILE	
18	TLSKEYFILE	
19	LISTENIP	
20	HOSTINTERFACE	
21	HOSTMETADATA	
22	HOSTMETADATAITEM	

Number	Parameter	Description
23	STATUSPORT	Zabbix agent 2 only.
24	ENABLEPERSISTENTBUFFER	Zabbix agent 2 only.
25	PERSISTENTBUFFERPERIOD	Zabbix agent 2 only.
26	PERSISTENTBUFFERFILE	Zabbix agent 2 only.
27	INSTALLFOLDER	
28	ENABLEPATH	
29	SKIP	`SKIP=fw` - do not install firewall exception rule
30	INCLUDE	Sequence of includes separated by `;`
31	ALLOWDENYKEY	Sequence of "AllowKey" and "DenyKey" parameters separated by `;`. Use `\\;` to escape the delimiter.
32	ADDPROGRAM	A comma-delimited list of programs to install. Possible values: `AgentProgram`, `GetProgram`, `SenderProgram` E.g., ADDPROGRAM=AgentProgram,GetProgram
33	ADDLOCAL	A comma-delimited list of programs to install. Possible values: `AgentProgram`, `GetProgram`, `SenderProgram` E.g., ADDLOCAL=AgentProgram,SenderProgram
34	CONF	Specify path to custom configuration file, e.g., `CONF=c:\full\path\to\user.conf`

To install you may run, for example:

```
SET INSTALLFOLDER=C:\Program Files\za

msiexec /l*v log.txt /i zabbix_agent-6.2.0-x86.msi /qn^
 LOGTYPE=file^
 LOGFILE="%INSTALLFOLDER%\za.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]"
```

or

```
msiexec /l*v log.txt /i zabbix_agent-6.2.0-x86.msi /qn^
 SERVER=192.168.6.76^
 TLSCONNECT=psk^
 TLSACCEPT=psk^
 TLSPSKIDENTITY=MyPSKID^
 TLSPSKVALUE=1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

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-6.2.0-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

Packages for minor Zabbix version (i.e. Zabbix 6.2.x) release candidates are provided starting with Zabbix 6.2.3.

The instructions below are for enabling unstable Zabbix release repositories (disabled by default).

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) - $basearch baseurl=https://repo.zabbix.com/zabbix/6.1/rhel/8/$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/6.1/debian bullseye main deb-src https://repo.zabbix.com/zabbix/6.1/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/6.1/sles/15/x86_64/ gpgcheck=1 gpgkey=https://repo.zabbix.com/zabbix/6.1/sles/15/x86_64/repodata/repomd.xml.key enabled=1 update=1

5 Installation from containers

Docker Zabbix provides Docker images for each Zabbix component as portable and self-sufficient containers to speed up deployment and update procedure.

Zabbix components come with MySQL and PostgreSQL database support, Apache2 and Nginx web server support. These images are separated into different images.

Docker base images

Zabbix components are provided on Ubuntu, Alpine Linux and CentOS base images:

Image	Version
alpine	3.17
ubuntu	22.04 (jammy)
centos	8

All images are configured to rebuild latest images if base images are updated.

Docker file sources

Everyone can follow Docker file changes using the Zabbix official repository on github.com. You can fork the project or make your own images based on official Docker files.

Structure

All Zabbix components are available in the following Docker repositories:

- Zabbix agent - zabbix/zabbix-agent
- Zabbix server
  - Zabbix server with MySQL database support - zabbix/zabbix-server-mysql
  - Zabbix server with PostgreSQL database support - zabbix/zabbix-server-pgsql
- Zabbix web-interface
  - Zabbix web-interface based on Apache2 web server with MySQL database support - zabbix/zabbix-web-apache-mysql
  - Zabbix web-interface based on Apache2 web server with PostgreSQL database support - zabbix/zabbix-web-apache-pgsql
  - Zabbix web-interface based on Nginx web server with MySQL database support - zabbix/zabbix-web-nginx-mysql
  - Zabbix web-interface based on Nginx web server with PostgreSQL database support - zabbix/zabbix-web-nginx-pgsql
- Zabbix proxy
  - Zabbix proxy with SQLite3 database support - zabbix/zabbix-proxy-sqlite3
  - Zabbix proxy with MySQL database support - zabbix/zabbix-proxy-mysql
- Zabbix Java Gateway - zabbix/zabbix-java-gateway

Additionally there is SNMP trap support. It is provided as additional repository (zabbix/zabbix-snmptraps) based on Ubuntu Trusty only. It could be linked with Zabbix server and Zabbix proxy.

Versions

Each repository of Zabbix components contains the following tags:

- `latest` - latest stable version of a Zabbix component based on Alpine Linux image
- `alpine-latest` - latest stable version of a Zabbix component based on Alpine Linux image
- `ubuntu-latest` - latest stable version of a Zabbix component based on Ubuntu image
- `alpine-6.2-latest` - latest minor version of a Zabbix 6.2 component based on Alpine Linux image
- `ubuntu-6.2-latest` - latest minor version of a Zabbix 6.2 component based on Ubuntu image
- `alpine-6.2.*` - different minor versions of a Zabbix 6.2 component based on Alpine Linux image, where `*` is the minor version of Zabbix component
- `ubuntu-6.2.*` - different minor versions of a Zabbix 6.2 component based on Ubuntu image, where `*` is the minor version of Zabbix component

Usage

Environment variables

All Zabbix component images provide environment variables to control configuration. These environment variables are listed in each 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 and Zabbix proxy configuration files.

> **Attention:**
> Some of configuration options are not allowed to change. For example, `PIDFile` and `LogType`.

Some of components have specific environment variables, which do not exist in official Zabbix configuration files:

Variable	Components	Description
`DB_SERVER_HOST`	Server Proxy Web interface	This variable is IP or DNS name of MySQL or PostgreSQL server. By default, value is `mysql-server` or `postgres-server` for MySQL or PostgreSQL respectively
`DB_SERVER_PORT`	Server Proxy Web interface	This variable is port of MySQL or PostgreSQL server. By default, value is '3306' or '5432' respectively.
`MYSQL_USER`	Server Proxy Web-interface	MySQL database user. By default, value is 'zabbix'.
`MYSQL_PASSWORD`	Server Proxy Web interface	MySQL database password. By default, value is 'zabbix'.
`MYSQL_DATABASE`	Server Proxy Web interface	Zabbix database name. By default, value is 'zabbix' for Zabbix server and 'zabbix_proxy' for Zabbix proxy.
`POSTGRES_USER`	Server Web interface	PostgreSQL database user. By default, value is 'zabbix'.
`POSTGRES_PASSWORD`	Server Web interface	PostgreSQL database password. By default, value is 'zabbix'.
`POSTGRES_DB`	Server Web interface	Zabbix database name. By default, value is 'zabbix' for Zabbix server and 'zabbix_proxy' for Zabbix proxy.
`PHP_TZ`	Web-interface	Timezone in PHP format. Full list of supported timezones are available on php.net. By default, value is 'Europe/Riga'.
`ZBX_SERVER_NAME`	Web interface	Visible Zabbix installation name in right top corner of the web interface. By default, value is 'Zabbix Docker'
`ZBX_JAVAGATEWAY_ENABLE`	Server Proxy	Enables communication with Zabbix Java gateway to collect Java related checks. By default, value is "false"
`ZBX_ENABLE_SNMP_TRAPS`	Server Proxy	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 use some mount points. These mount points are different and depend on Zabbix component type:

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	
/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 use Zabbix official repositories in Docker Hub.

Usage 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 \
      --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-6.2-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-6.2-latest
```

Note:
Zabbix server instance exposes 10051/TCP port (Zabbix trapper) to host machine.

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-6.2-latest
```

Note:
Zabbix web interface instance exposes 80/TCP port (HTTP) to host machine.

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-6.2-latest
```

Note:
Zabbix snmptrap instance exposes the 162/UDP port (SNMP traps) to host machine.

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-6.2-latest
```

Note:
Zabbix server instance exposes the 10051/TCP port (Zabbix trapper) to host machine.

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-6.2-latest
```

Note:
Zabbix web interface instance exposes the 443/TCP port (HTTPS) to host machine.
Directory */etc/ssl/nginx* must contain certificate with required name.

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 \
    -eZBX_SERVER_HOST="127.0.0.1,localhost" \
    --restart=always \
    --pod=zabbix \
    -d registry.connect.redhat.com/zabbix/zabbix-agent-50: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
```

3. 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-50
```

4. 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-50
```

5. 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-50
```

Note:
Pod `zabbix` exposes 80/TCP port (HTTP) to host machine from 8080/TCP of `zabbix-web-mysql` container.

Docker Compose Zabbix provides compose files also for defining and running multi-container Zabbix components in Docker. These compose files are available in Zabbix docker official repository on github.com: https://github.com/zabbix/zabbix-docker. These compose files are added as examples, they are overloaded. For example, they contain proxies with MySQL and SQLite3 support.

There are a few different versions of compose files:

File name	Description
`docker-compose_v3_alpine_mysql_latest.yaml`	The compose file runs the latest version of Zabbix 6.2 components on Alpine Linux with MySQL database support.
`docker-compose_v3_alpine_mysql_local.yaml`	The compose file builds the latest version of Zabbix 6.2 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 6.2 components on Alpine Linux with PostgreSQL database support.
`docker-compose_v3_alpine_pgsql_local.yaml`	The compose file builds the latest version of Zabbix 6.2 and runs Zabbix components on Alpine Linux with PostgreSQL database support.
`docker-compose_v3_centos_mysql_latest.yaml`	The compose file runs the latest version of Zabbix 6.2 components on CentOS 8 with MySQL database support.
`docker-compose_v3_centos_mysql_local.yaml`	The compose file builds the latest version of Zabbix 6.2 and runs Zabbix components on CentOS 8 with MySQL database support.
`docker-compose_v3_centos_pgsql_latest.yaml`	The compose file runs the latest version of Zabbix 6.2 components on CentOS 8 with PostgreSQL database support.
`docker-compose_v3_centos_pgsql_local.yaml`	The compose file builds the latest version of Zabbix 6.2 and runs Zabbix components on CentOS 8 with PostgreSQL database support.
`docker-compose_v3_ubuntu_mysql_latest.yaml`	The compose file runs the latest version of Zabbix 6.2 components on Ubuntu 20.04 with MySQL database support.
`docker-compose_v3_ubuntu_mysql_local.yaml`	The compose file builds the latest version of Zabbix 6.2 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 6.2 components on Ubuntu 20.04 with PostgreSQL database support.
`docker-compose_v3_ubuntu_pgsql_local.yaml`	The compose file builds the latest version of Zabbix 6.2 and runs Zabbix components on Ubuntu 20.04 with PostgreSQL database support.

Attention:
Available Docker compose files support version 3 of Docker Compose.

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 above 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 files

In the same directory with compose files on github.com you can find files with default environment variables for each component in compose file. These environment files are named like `.env_<type of component>`.

Examples

Note:
The commands listed below are for Docker Compose V2. If you are using Docker Compose V1, replace `docker compose -f` with `docker-compose -f`

Example 1

```
# git checkout 6.2
# docker compose -f ./docker-compose_v3_alpine_mysql_latest.yaml up -d
```

The command will download latest Zabbix 6.2 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 6.2
# docker compose -f ./docker-compose_v3_ubuntu_mysql_local.yaml up -d
```

The command will download base image Ubuntu 20.04 (focal), then build Zabbix 6.2 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.

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>*

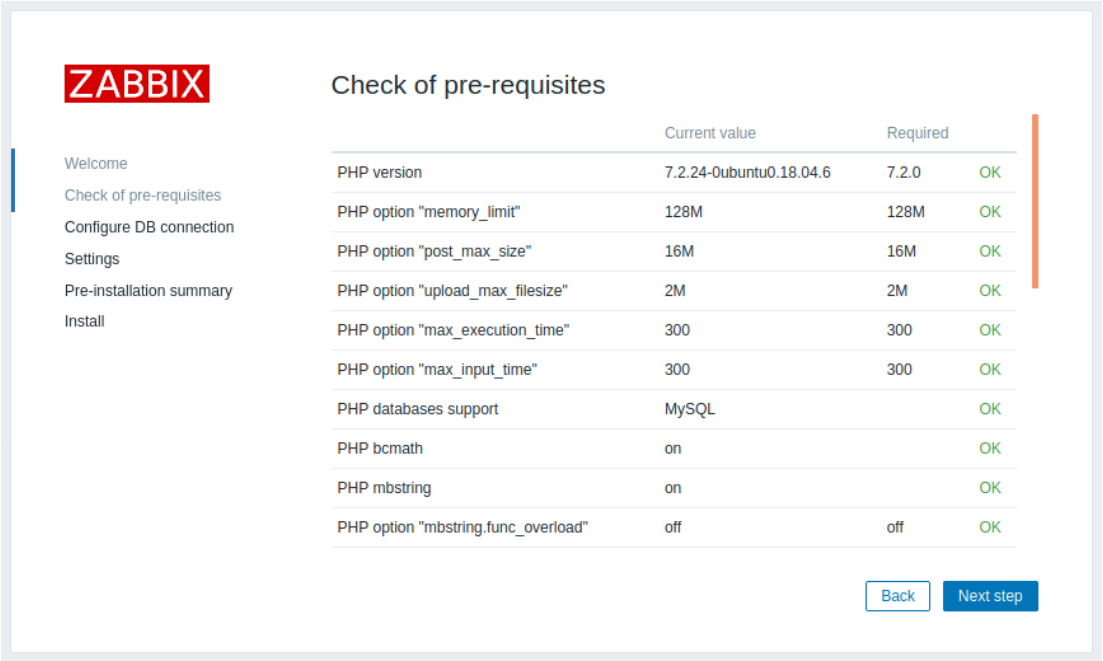
You should see the first screen of the frontend installation wizard.

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.



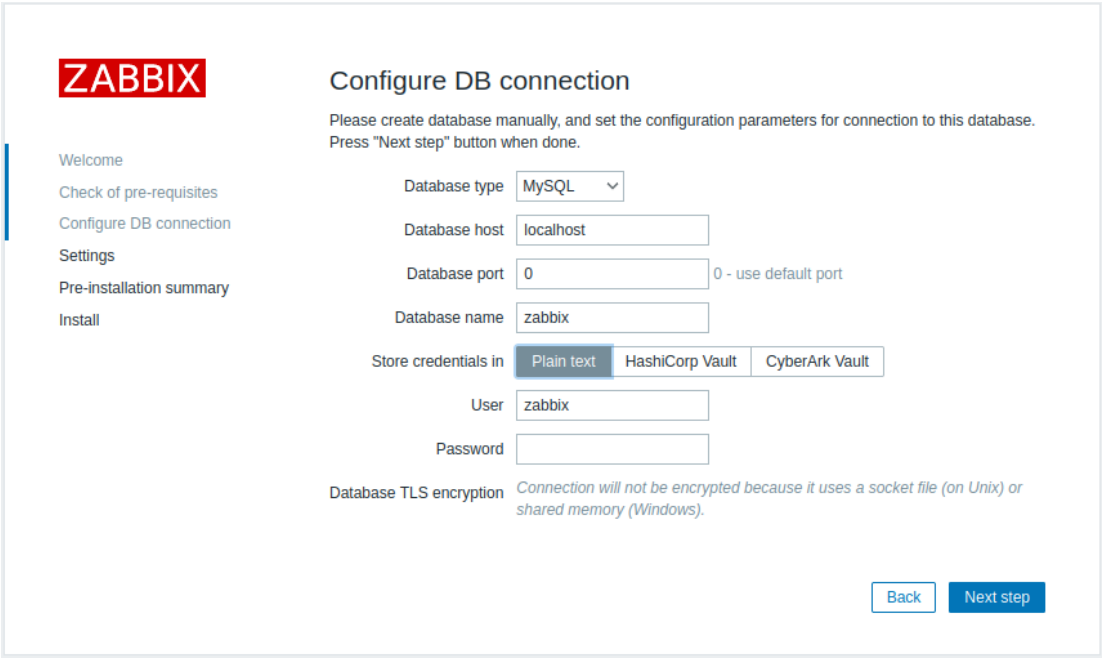
Pre-requisite	Minimum value	Description
PHP version	7.4.0	
PHP memory_limit option	128MB	In php.ini: memory_limit = 128M
PHP post_max_size option	16MB	In php.ini: post_max_size = 16M
PHP upload_max_filesize option	2MB	In php.ini: upload_max_filesize = 2M
PHP max_execution_time option	300 seconds (values 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

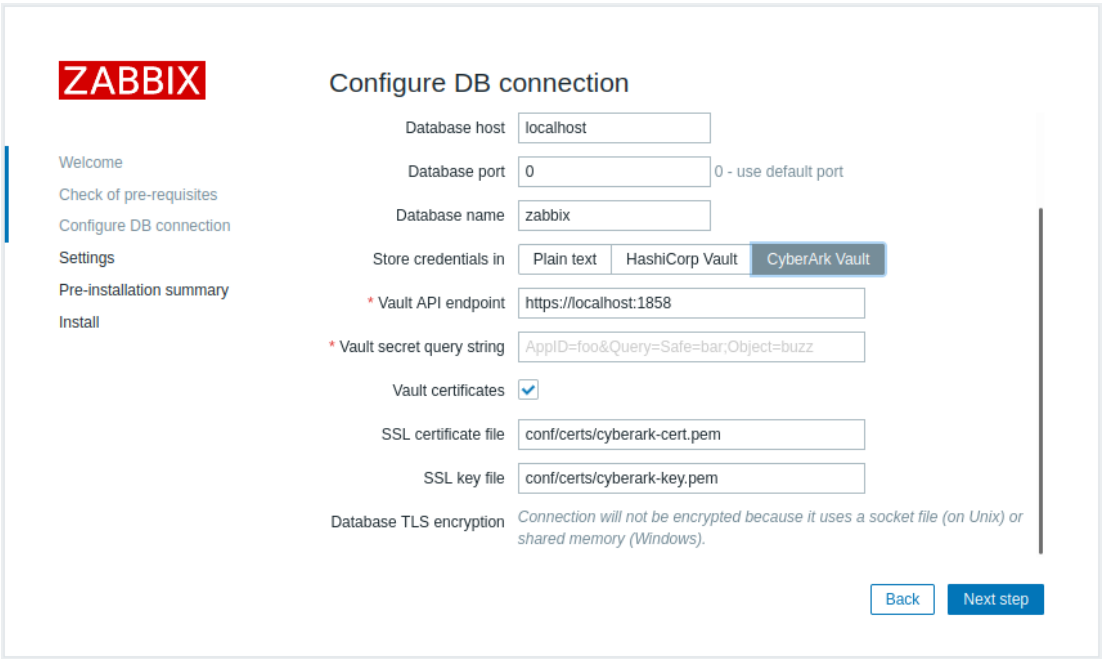
Enter details for connecting to the database. Zabbix database must already be created.



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:

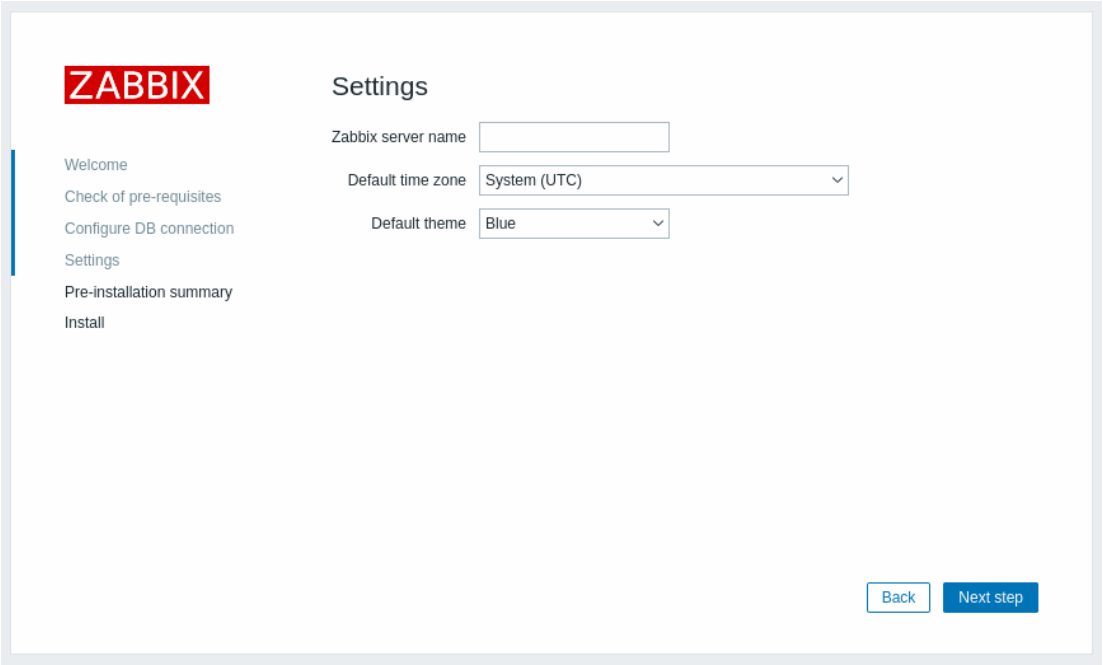
- for HashiCorp Vault: Vault API endpoint, secret path and authentication token;
- for CyberArk Vault: Vault API endpoint, secret query string and certificates. Upon marking *Vault certificates* checkbox, two new fields for specifying paths to SSL certificate file and SSL key file will appear.



Settings

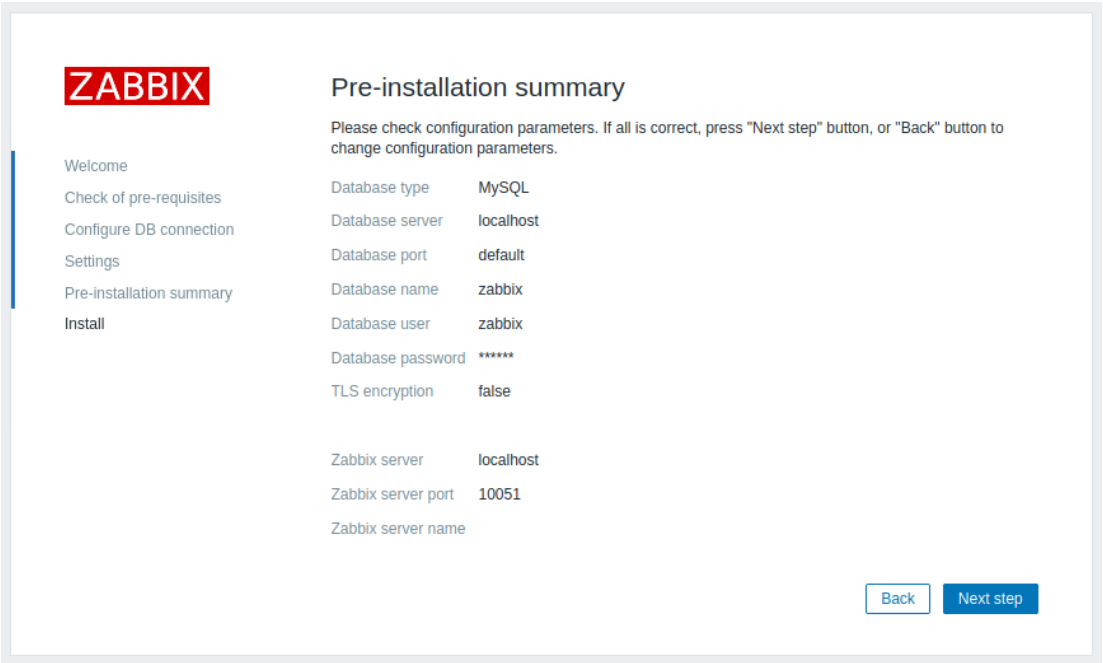
Entering a name for Zabbix server is optional, however, if submitted, it will be displayed in the menu bar and page titles.

Set the default time zone and theme for the frontend.



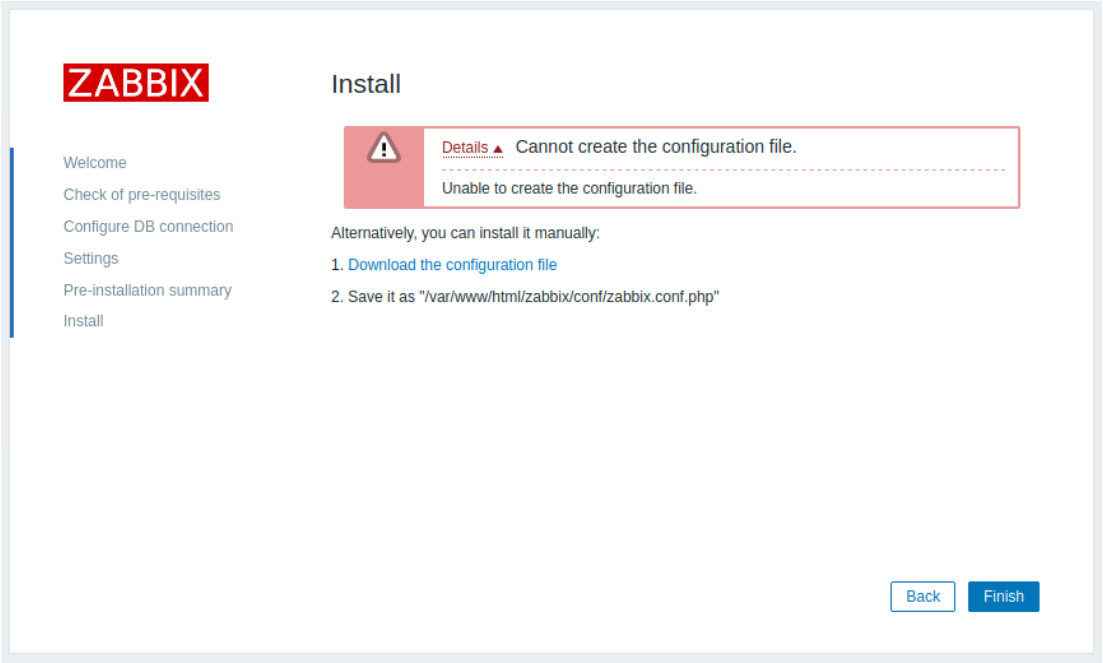
Pre-installation summary

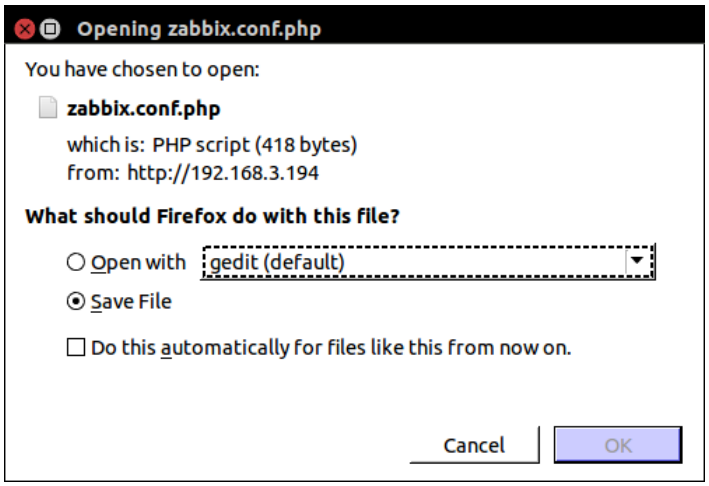
Review a summary of settings.



Install

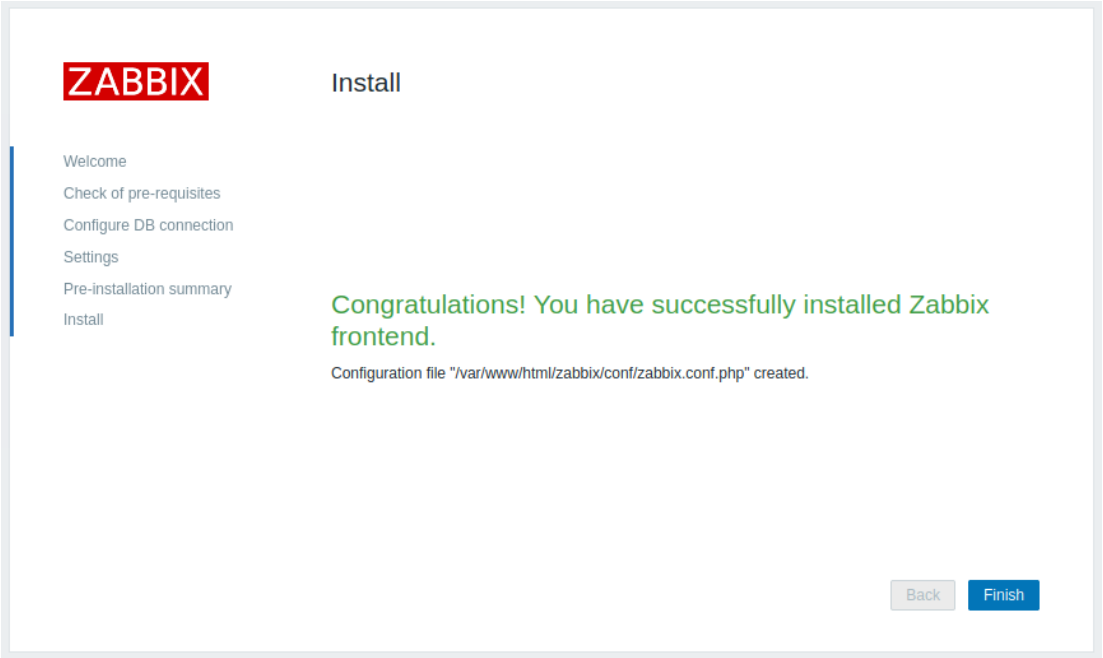
If installing Zabbix from sources, download the configuration file and place it under conf/ in the webserver HTML documents subdirectory where you copied Zabbix PHP files to.





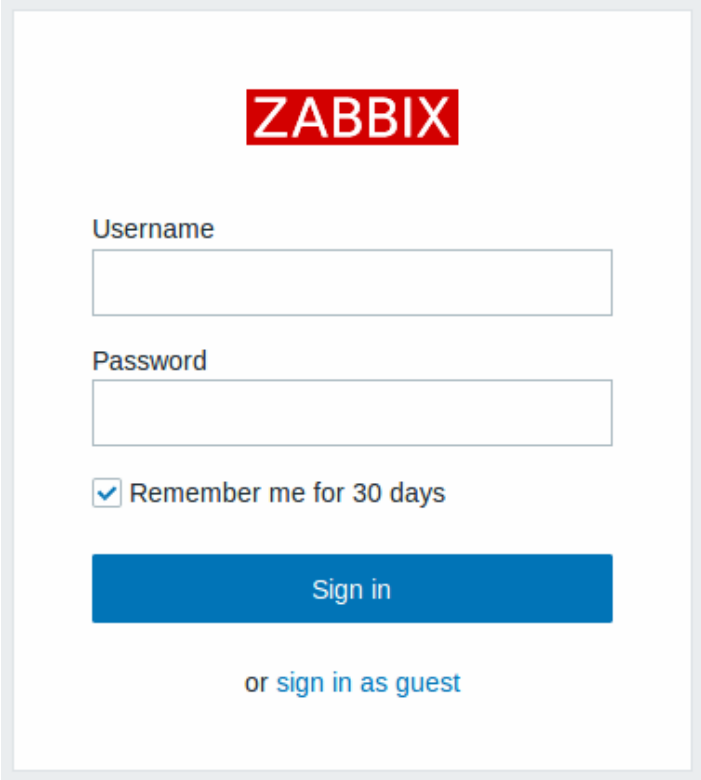
Note:
Providing the webserver user has write access to conf/ directory the configuration file would be saved automatically and it would be possible to proceed to the next step right away.

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 **6.2**:

- using packages:
  - for Red Hat Enterprise Linux/CentOS
  - for Debian/Ubuntu
- using sources

See also upgrade instructions for servers in a **high-availability** (HA) cluster.

Direct upgrade to Zabbix 6.2.x is possible from Zabbix **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.

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/CentOS
- 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/CentOS

Overview

This section provides the steps required for a successful upgrade from Zabbix **6.0**.x to Zabbix **6.2**.x using official Zabbix packages for Red Hat Enterprise Linux/CentOS.

While upgrading Zabbix agents is not mandatory (but recommended), Zabbix server and proxies must be of the same major version. Therefore, in a server-proxy setup, Zabbix server and all proxies have to be stopped and upgraded. Keeping proxies running during server upgrade no longer will bring any benefit as during proxy upgrade their old data will be discarded and no new data will be gathered until proxy configuration is synced with server.

Note that with SQLite database on proxies, history data from proxies before the upgrade will be lost, because SQLite database upgrade is not supported and the SQLite database file has to be manually removed. When proxy is started for the first time and the SQLite database file is missing, proxy creates it automatically.

Depending on database size the database upgrade to version 6.2 may take a long time.

> **Warning:**
> Before the upgrade make sure to read the relevant **upgrade notes!**

The following upgrade notes are available:

Upgrade from	Read full upgrade notes	Most important changes between versions
6.0.x LTS	For: Zabbix 6.2	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.

Upgrade from	Read full upgrade notes	Most important changes between versions
5.4.x	For:	Minimum required database versions upped.
	Zabbix 6.0	Server/proxy will not start if outdated database.
	Zabbix 6.2	Audit log records lost because of database structure change.
5.2.x	For:	Minimum required database versions upped.
	Zabbix 5.4	Aggregate items removed as a separate type.
	Zabbix 6.0	
	Zabbix 6.2	
5.0.x LTS	For:	Minimum required PHP version upped from 7.2.0 to 7.2.5.
	Zabbix 5.2	Password hashing algorithm changed from MD5 to bcrypt.
	Zabbix 5.4	
	Zabbix 6.0	
	Zabbix 6.2	
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	
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	
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 server.
	Zabbix 4.4	
	Zabbix 5.0	
	Zabbix 5.2	
	Zabbix 5.4	
	Zabbix 6.0	
	Zabbix 6.2	
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	
3.2.x	For:	SQLite support as backend database dropped for Zabbix server/frontend.
	Zabbix 3.4	
	Zabbix 4.0	Perl Compatible Regular Expressions (PCRE) supported instead of POSIX extended.
	Zabbix 4.2	
	Zabbix 4.4	'libpcre' and 'libevent' libraries mandatory for Zabbix server.
	Zabbix 5.0	Exit code checks added for user parameters, remote commands and system.run[] items without the 'nowait' flag as well as Zabbix server executed scripts.
	Zabbix 5.2	
	Zabbix 5.4	
	Zabbix 6.0	Zabbix Java gateway has to be upgraded to support new functionality.
	Zabbix 6.2	

Upgrade from	Read full upgrade notes	Most important changes between versions
3.0.x LTS	For: 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	Database upgrade may be slow, depending on the history table size.
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	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	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	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'

You may also want to check the requirements for 6.2.

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.

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 the proxy, stop proxy too.

```
# systemctl stop zabbix-proxy
```

> **Attention:**
> It is no longer possible to start the upgraded server and have older and unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled, as the server will ignore data from unupgraded proxies.

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

To proceed with the upgrade your current repository package has to be updated.

```
# rpm -Uvh https://repo.zabbix.com/zabbix/6.2/rhel/8/x86_64/zabbix-release-6.2-1.el8.noarch.rpm
```

Switch the DNF module version for PHP:

```
# dnf module switch-to php:7.4
```

5 Upgrade Zabbix components

To upgrade Zabbix components you may run something like:

```
# dnf upgrade 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` in the command.

To upgrade the web frontend with Apache **on RHEL 8** correctly, also run:

```
# dnf install zabbix-apache-conf
```

6 Review component configuration parameters

See the upgrade notes for details on mandatory changes.

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 minor versions of 6.2.x (for example, from 6.2.1 to 6.2.3). Upgrading between minor versions is easy.

To execute Zabbix minor version upgrade it is required to run:

```
$ sudo dnf upgrade 'zabbix-*'
```

To execute Zabbix server minor version upgrade run:

```
$ sudo dnf upgrade 'zabbix-server-*'
```

To execute Zabbix agent minor version upgrade run:

```
$ sudo dnf upgrade 'zabbix-agent-*'
```

or, for Zabbix agent 2:

```
$ sudo dnf upgrade 'zabbix-agent2-*'
```

2 Debian/Ubuntu

Overview

This section provides the steps required for a successful upgrade from Zabbix **6.0**.x to Zabbix **6.2**.x using official Zabbix packages for Debian/Ubuntu.

While upgrading Zabbix agents is not mandatory (but recommended), Zabbix server and proxies must be of the same major version. Therefore, in a server-proxy setup, Zabbix server and all proxies have to be stopped and upgraded. Keeping proxies running during server upgrade no longer will bring any benefit as during proxy upgrade their old data will be discarded and no new data will be gathered until proxy configuration is synced with server.

Note that with SQLite database on proxies, history data from proxies before the upgrade will be lost, because SQLite database upgrade is not supported and the SQLite database file has to be manually removed. When proxy is started for the first time and the SQLite database file is missing, proxy creates it automatically.

Depending on database size the database upgrade to version 6.2 may take a long time.

> **Warning:**
> Before the upgrade make sure to read the relevant **upgrade notes!**

The following upgrade notes are available:

Upgrade from	Read full upgrade notes	Most important changes between versions
6.0.x LTS	For: Zabbix 6.2	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	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	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	Minimum required PHP version upped from 7.2.0 to 7.2.5. Password hashing algorithm changed from MD5 to bcrypt.
4.4.x	For: Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2	Support of IBM DB2 dropped. Minimum required PHP version upped from 5.4.0 to 7.2.0. Minimum required database versions upped. Changed Zabbix PHP file directory.

Upgrade from	Read full upgrade notes	Most important changes between versions
4.2.x	For: Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2	Jabber, Ez Texting media types removed.
4.0.x LTS	For: Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2	Older proxies no longer can report data to an upgraded server. Newer agents no longer will be able to work with an older Zabbix server.
3.4.x	For: Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2	'libpthread' and 'zlib' libraries now mandatory. Support for plain text protocol dropped and header is mandatory. Pre-1.4 version Zabbix agents are no longer supported. The Server parameter in passive proxy configuration now mandatory.
3.2.x	For: 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	SQLite support as backend database dropped for Zabbix server/frontend. Perl Compatible Regular Expressions (PCRE) supported instead of POSIX extended. 'libpcre' and 'libevent' libraries mandatory for Zabbix server. Exit code checks added for user parameters, remote commands and system.run[] items without the 'nowait' flag as well as Zabbix server executed scripts. Zabbix Java gateway has to be upgraded to support new functionality.
3.0.x LTS	For: 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	Database upgrade may be slow, depending on the history table size.
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	Minimum required PHP version upped from 5.3.0 to 5.4.0 LogFile agent parameter must be specified

Upgrade from	Read full upgrade notes	Most important changes between versions
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	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	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'

You may also want to check the requirements for 6.2.

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.

Upgrade procedure

1 Stop Zabbix processes

Stop Zabbix server to make sure that no new data is inserted into database.

```
# service zabbix-server stop
```

If upgrading Zabbix proxy, stop proxy too.

```
# service zabbix-proxy stop
```

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

To proceed with the update your current repository package has to be uninstalled.

```
# rm -Rf /etc/apt/sources.list.d/zabbix.list
```

Then install the new repository configuration package.

On **Debian 11** run:

```
# wget https://repo.zabbix.com/zabbix/6.2/debian/pool/main/z/zabbix-release/zabbix-release_6.2-1+debian11_
# dpkg -i zabbix-release_6.2-1+debian11_all.deb
```

On **Debian 10** run:

```
# wget https://repo.zabbix.com/zabbix/6.2/debian/pool/main/z/zabbix-release/zabbix-release_6.2-1+debian10_
# dpkg -i zabbix-release_6.2-1+debian10_all.deb
```

On **Debian 9** run:

```
# wget https://repo.zabbix.com/zabbix/6.2/debian/pool/main/z/zabbix-release/zabbix-release_6.2-1+debian9_a
# dpkg -i zabbix-release_6.2-1+debian9_all.deb
```

On **Ubuntu 20.04** run:

```
# wget https://repo.zabbix.com/zabbix/6.2/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.2-1+ubuntu20.
# dpkg -i zabbix-release_6.2-1+ubuntu20.04_all.deb
```

On **Ubuntu 18.04** run:

```
# wget https://repo.zabbix.com/zabbix/6.2/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.2-1+ubuntu18.
# dpkg -i zabbix-release_6.2-1+ubuntu18.04_all.deb
```

On **Ubuntu 16.04** run:

```
# wget https://repo.zabbix.com/zabbix/6.2/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.2-1+ubuntu16.
# dpkg -i zabbix-release_6.2-1+ubuntu16.04_all.deb
```

On **Ubuntu 14.04** run:

```
# wget https://repo.zabbix.com/zabbix/5.4/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.2-1+ubuntu14.
# dpkg -i zabbix-release_6.2-1+ubuntu14.04_all.deb
```

Update the repository information.

```
# apt-get update
```

5 Upgrade Zabbix components

To upgrade Zabbix components you may run something like:

```
# apt-get 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` in the command.

Then, to upgrade the web frontend with Apache correctly, also run:

```
# apt-get install zabbix-apache-conf
```

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

See the upgrade notes for details on mandatory changes (if any).

For new optional parameters, see the What's new section.

7 Start Zabbix processes

Start the updated Zabbix components.

```
# service zabbix-server start
# service zabbix-proxy start
# service zabbix-agent start
```

```
# service zabbix-agent2 start
```

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 minor versions of 6.2.x (for example, from 6.2.1 to 6.2.3). It is easy.

To upgrade Zabbix minor version please run:

```
$ sudo apt install --only-upgrade 'zabbix.*'
```

To upgrade Zabbix server minor version please run:

```
$ sudo apt install --only-upgrade 'zabbix-server.*'
```

To upgrade Zabbix agent minor version please run:

```
$ sudo apt install --only-upgrade 'zabbix-agent.*'
```

or, for Zabbix agent 2:

```
$ sudo apt install --only-upgrade 'zabbix-agent2.*'
```

Upgrade from sources

Overview

This section provides the steps required for a successful upgrade from Zabbix **6.0**.x to Zabbix **6.2**.x using official Zabbix sources.

While upgrading Zabbix agents is not mandatory (but recommended), Zabbix server and proxies must be of the same major version. Therefore, in a server-proxy setup, Zabbix server and all proxies have to be stopped and upgraded. Keeping proxies running no longer will bring any benefit as during proxy upgrade their old data will be discarded and no new data will be gathered until proxy configuration is synced with server.

> **Attention:**
> It is no longer possible to start the upgraded server and have older and unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled, as the server will ignore data from unupgraded proxies.

Note that with SQLite database on proxies, history data from proxies before the upgrade will be lost, because SQLite database upgrade is not supported and the SQLite database file has to be manually removed. When proxy is started for the first time and the SQLite database file is missing, proxy creates it automatically.

Depending on database size the database upgrade to version 6.2 may take a long time.

> **Warning:**
> Before the upgrade make sure to read the relevant **upgrade notes!**

The following upgrade notes are available:

Upgrade from	Read full upgrade notes	Most important changes between versions
6.0.x LTS	For: Zabbix 6.2	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	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	Minimum required database versions upped. Aggregate items removed as a separate type.

Upgrade from	Read full upgrade notes	Most important changes between versions
5.0.x LTS	For: Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2	Minimum required PHP version upped from 7.2.0 to 7.2.5. Password hashing algorithm changed from MD5 to bcrypt.
4.4.x	For: Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2	Support of IBM DB2 dropped. Minimum required PHP version upped from 5.4.0 to 7.2.0. Minimum required database versions upped. Changed Zabbix PHP file directory.
4.2.x	For: Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2	Jabber, Ez Texting media types removed.
4.0.x LTS	For: Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2	Older proxies no longer can report data to an upgraded server. Newer agents no longer will be able to work with an older Zabbix server.
3.4.x	For: Zabbix 4.0 Zabbix 4.2 Zabbix 4.4 Zabbix 5.0 Zabbix 5.2 Zabbix 5.4 Zabbix 6.0 Zabbix 6.2	'libpthread' and 'zlib' libraries now mandatory. Support for plain text protocol dropped and header is mandatory. Pre-1.4 version Zabbix agents are no longer supported. The Server parameter in passive proxy configuration now mandatory.
3.2.x	For: 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	SQLite support as backend database dropped for Zabbix server/frontend. Perl Compatible Regular Expressions (PCRE) supported instead of POSIX extended. 'libpcre' and 'libevent' libraries mandatory for Zabbix server. Exit code checks added for user parameters, remote commands and system.run[] items without the 'nowait' flag as well as Zabbix server executed scripts. Zabbix Java gateway has to be upgraded to support new functionality.
3.0.x LTS	For: 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	Database upgrade may be slow, depending on the history table size.

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	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	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	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'

You may also want to check the requirements for 6.2.

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

See the upgrade notes for details on mandatory changes.

For new optional parameters, see the What's new section.

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 7.2.5. 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 6.2.x (for example from 6.2.1 to 6.2.3) 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.

8 Known issues

Incorrect permissions in packages

Some repository files in the zabbix-release package have incorrect permissions (755 instead of 644) in the 6.2 release in these locations:

- `/etc/apt/sources.list.d/zabbix.list`
- `/etc/apt/sources.list.d/zabbix-agent2-plugins.list`
- `/etc/apt/trusted.gpg.d/zabbix-official-repo.gpg`

This has been fixed in zabbix-release-6.2-4 packages. Despite that, running `apt update && apt upgrade` will not fix the permission issue, but fresh installations will have correct permissions.

The user may manually change permissions (chmod) for these files from 755 to 644. This should not have any impact on overall operation.

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 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).

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.

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 CentOS 8 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 CentOS 6, CentOS 7 and possibly other related 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.

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/CentOS 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; if you are using CentOS 8.0-8.2, you may wait for CentOS 8.3 or use a package from EPEL.

Please also see ZBX-17192 for more information.

Alerter process crash in Centos/RHEL 7

Instances of a Zabbix server alerter process crash have been encountered in Centos/RHEL 7. Please see ZBX-10461 for details.

Compiling Zabbix agent on HP-UX

If you install the PCRE library from a popular HP-UX package site http://hpux.connect.org.uk, for example from file `pcre-8.42-ia64_64-11.3` you get only the 64-bit version of the library installed in the /usr/local/lib/hpux64 directory.

In this case, for successful agent compilation customized options need to be used for the ”configure” script, e.g.:

```
CFLAGS="+DD64" ./configure --enable-agent --with-libpcre-include=/usr/local/include --with-libpcre-lib=/us
```

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.

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 is caused by a known issue in MySQL 5.6/5.7 versions. A workaround to this is disabling the index_condition_pushdown optimizer in MySQL. For an extended discussion, see ZBX-10652.

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`.

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/CentOS 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;
}
```

Issues in Zabbix 6.2.5

Server-proxy compatibility

This version has the following server-proxy compatibility issues:

- Zabbix server 6.2.5 will not work with a Zabbix proxy below/above 6.2.5;
- Zabbix proxy 6.2.5 will not work with a Zabbix server below/above 6.2.5.

JSONPath parsing errors

JSONPath parsing errors occur in case of leading whitespace and empty array/object. Fixed in Zabbix 6.2.6.

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 *Configuration* → *Templates* you can import them manually into Zabbix. If templates with the same names already exist, the *Delete missing* options 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).

CHANGES IN 6.2.0

New templates

See the list of new templates in Zabbix 6.2.0

CHANGES IN 6.2.1

A new template *HPE Synergy by HTTP* is available.

The templates *HashiCorp Consul Node by HTTP* and *HashiCorp Consul Cluster by HTTP* now support Consul namespaces.

CHANGES IN 6.2.2

New templates:

- *AWS RDS instance by HTTP*
- *AWS S3 bucket by HTTP*
- *Azure by HTTP*
- *OPNsense by SNMP*

See setup instructions for HTTP templates.

CHANGES IN 6.2.3

A new *AWS by HTTP* template is available.

See setup instructions for HTTP templates.

CHANGES IN 6.2.4

The template *Azure by HTTP* has been updated and now includes metrics to monitor Microsoft Azure MySQL servers out-of-the-box.

CHANGES IN 6.2.5

The template *Azure by HTTP* has been updated and now includes metrics to monitor Microsoft PostgreSQL flexible servers and Microsoft PostgreSQL single server out-of-the-box.

CHANGES IN 6.2.7

New templates are available:

- *Control-M enterprise manager by HTTP*
- *Control-M server by HTTP*
- *Veeam Backup Enterprise Manager by HTTP*
- *Veeam Backup and Replication by HTTP*

See setup instructions for HTTP templates.

The template *Oracle by Zabbix agent 2* has been updated:

- The following static items, that requested data for all existing relevant DB objects in a single query, have been removed:
  - "Oracle: Get archive log info"
  - "Oracle: Get ASM stats"
  - "Oracle: Get CDB and No-CDB info"
  - "Oracle: Get PDB info"
  - "Oracle: Get tablespaces stats"
- The following agent item prototypes have been added to the corresponding discovery rules:
  - Archive log discovery rule: "Archivelog '{#DEST_NAME}': Get archive log info"
  - ASM disk groups discovery: "ASM '{#DGNAME}': Get ASM stats"
  - Database discovery: "Oracle Database '{#DBNAME}': Get CDB and No-CDB info"
  - PDB discovery: "Oracle Database '{#DBNAME}': Get PDB info"
  - Tablespace discovery: "Oracle TBS '{#TABLESPACE}': Get tablespace stats"

10 Upgrade notes for 6.2.0

These notes are for upgrading from Zabbix 6.0.x to Zabbix 6.2.0. All notes are grouped into:

- `Critical` - the most critical information related to the upgrade process and the changes in Zabbix functionality
- `Informational` - all remaining information describing the changes in Zabbix functionality

It is possible to upgrade to Zabbix 6.2.0 from versions before Zabbix 6.0.0. See the upgrade procedure section for all relevant information about upgrading from previous Zabbix versions.

Critical

Faster configuration sync Incremental configuration cache synchronization has been added for hosts, host tags, items, item tags, item preprocessing, triggers, trigger tags and functions to lessen synchronization time and database load when configuration is being updated on an already running Zabbix server or Zabbix proxy. As a result of this change, deterministic triggers need to be created during upgrade.

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. To set the variable using MySQL console, run:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
```

Once the upgrade has been successfully completed, `log_bin_trust_function_creators` can be disabled:

```
mysql> SET GLOBAL log_bin_trust_function_creators = 0;
```

Triggers are also created for PostgreSQL and Oracle database.

Minimum required PHP version The minimum required PHP version has been raised from 7.2.5 to 7.4.0.

Internal items for history/trends removed The following internal items, deprecated since Zabbix 6.0, have now been removed:

- zabbix[history]
- zabbix[history_log]
- zabbix[history_str]
- zabbix[history_text]
- zabbix[history_uint]
- zabbix[trends]
- zabbix[trends_uint]

History pollers removed from Zabbix proxy History pollers have been removed from Zabbix proxy.

Internal items that used to require a database connection (such as zabbix[proxy,,lastaccass], zabbix[proxy,,delay] and zabbix[proxy_history]) and were polled by history pollers on Zabbix server or Zabbix proxy have been reconfigured to be polled by regular pollers and use data from the configuration cache instead.

Secure password hashing In Zabbix 5.0 password hashing algorithm has been changed from MD5 to the more secure bcrypt. However, in Zabbix versions 5.0 - 6.0, MD5 hashing was still used upon the first user login after an upgrade - to convert passwords with hashes not exceeding 32 bytes from MD5 to bcrypt. Now support of MD5 cryptography has been dropped completely.

If you're upgrading from Zabbix versions before 5.0, users with passwords hashed by MD5 won't be able to log in. In this case, a Super administrator can change passwords of the affected users. If a Super administrator also cannot log in, run the following SQL query to apply the default password to the user (replace 'Admin' with the required username):

```
UPDATE users SET passwd = '$2a$10$ZXIvHAEP2ZM.dLXTm6uPHOMVlARXX7cqjbhM6Fn0cANzkCQBWpMrS' WHERE username =
```

After running this query, the user's password will be set to *zabbix*. Make sure to change the default password on the first login.

Storage of secrets In addition to HashiCorp Vault, Zabbix now supports storage of secrets in CyberArk Vault. To distinguish between secret management platforms, a new parameter *$DB['VAULT']* has been added to zabbix.conf.php.

If your Zabbix installation has been configured to work with HashiCorp Vault, after the upgrade you will need to manually update the configuration file. To continue using the HashiCorp integration, add to zabbix.conf.php the variable:

```
$DB['VAULT'] = 'HashiCorp';
```

Additionally, the database credentials will no longer be cached by default. Instead, Zabbix will make a call to the vault API every time when establishing a database connection. To enable storing of retrieved credentials in a local cache, you now need to manually set the option `$DB['VAULT_CACHE'] = true`.

For more info, see Storage of secrets.

CurlHttpRequest removed The `CurlHttpRequest` additional JavaScript object that was renamed to `HttpRequest` in Zabbix 5.4, and had been deprecated since, has now been removed.

API changes

See the list of API changes in Zabbix 6.2.0.

Informational

Immediate checks for new items Previously, newly added items were first checked at a random time within their update interval. Now new items, web scenarios and discovery rules, will be executed within 60 seconds of their creation, unless they have Scheduling or Flexible update interval with the *Update interval* parameter set to 0.

Separate groups for templates Host group functionality has been split into template groups, which may contain templates only and host groups, which may contain hosts only. During the upgrade all host groups that contained templates only will be automatically converted into template groups. Host and template groups will retain the same UUIDs.

If a host group contains both hosts and templates, such group will be split into two groups with the same name: a template group and a host group. If a group doesn't contain hosts, but is referenced somewhere in the hosts-related configuration (for example, used by a host prototype or in an action operation), an empty host group will be created.

After the upgrade, all users whose role allows access to *Configuration -> Host groups* menu section will get access to the new *Configuration -> Template groups* section. For user groups, existing permission sets will be automatically split into host and template group permissions.

11 Upgrade notes for 6.2.1

Symlink name expansion Symlink name and full path of the symlink are now returned in `vfs.dir.get[]` and `vfs.file.get[]` items, instead of resolving to the symlink target.

12 Upgrade notes for 6.2.2

This minor version doesn't have any upgrade notes.

13 Upgrade notes for 6.2.3

This minor version doesn't have any upgrade notes.

14 Upgrade notes for 6.2.4

Breaking changes

PostgreSQL plugin moved to loadable plugins The PostgreSQL plugin is no longer built-in in Zabbix agent 2. Instead it is now a loadable plugin in the agent 2.

This change may/will break automations with Ansible, Chef, etc, because it is no longer possible to pull the plugin repository directly.

See also: PostgreSQL loadable plugin repository

vmware.vm.discovery output format updated The **vmware.vm.discovery** item now returns the *vm_customattribute* array instead of *vm.customattribute*. This formatting change has been implemented to simplify further processing of the returned JSON (for example, in preprocessing steps or LLD macros).

15 Upgrade notes for 6.2.5

Breaking changes

Server-proxy compatibility This version has the following server-proxy compatibility issues:

- Zabbix server 6.2.5 will not work with a Zabbix proxy below/above 6.2.5;
- Zabbix proxy 6.2.5 will not work with a Zabbix server below/above 6.2.5.

JSONPath parsing errors JSONPath parsing errors occur in this version in case of leading whitespace and empty array/object. Fixed in Zabbix 6.2.6.

16 Upgrade notes for 6.2.6

Improved performance of history syncers The performance of history syncers has been improved by introducing a new read-write lock. This reduces locking between history syncers, trappers and proxy pollers by using a shared read lock while accessing the configuration cache. The new lock can be write locked only by the configuration syncer performing a configuration cache reload.

17 Upgrade notes for 6.2.7

Breaking changes Loadable plugin versioning

Loadable plugins for Zabbix agent 2 now use the same versioning system as Zabbix itself. The following version changes have been made:

- MongoDB 1.2.0 -> MongoDB 6.2.7
- PostgreSQL 1.2.1 -> PostgreSQL 6.2.7

These plugins are supported for any minor version of Zabbix 6.2. Note that source code repository for each plugin now contains a dedicated *release/6.2* branch (previously, there was only *master* branch).

Configuration import See changes in the configuration import process.

Query separate tablespaces in Oracle databases with Zabbix agent 2 The following Zabbix agent 2 items, supported for the Oracle plugin, now have additional optional parameters:

- oracle.diskgroups.stats[<existingParameters>,**<diskgroup>**]
- oracle.archive.info[<existingParameters>,**<destination>**]
- oracle.cdb.info[<existingParameters>,**<database>**]
- oracle.pdb.info[<existingParameters>,**<database>**]
- oracle.ts.stats[<existingParameters>,**<tablespace>**,**<type>**]

These parameters allow to query separate instances of data instead of all data, thus improving performance.

Open file descriptor limit increase for Zabbix agent 2 The systemd service file shipped in Zabbix agent 2 packages now declares the open file descriptor limit of 8196. Previously, the system default limit of 1024 has been used. The new limit is sufficient for the default Zabbix agent 2 configuration. If you have a non-standard agent 2 configuration, for example, use additional plugins or extended features, this limit may need to be manually increased further. In this case, adjust the `LimitNOFILE` parameter in the systemd unit file.

18 Upgrade notes for 6.2.8

Limits for JavaScript objects in preprocessing The following limits for JavaScript objects in preprocessing have been introduced:

- The total size of all messages that can be logged with the `log()` method has been limited to 8 MB per script execution.
- The initialization of multiple `HttpRequest` objects has been limited to 10 per script execution.
- The total length of header fields that can be added to a single `HttpRequest` object with the `addHeader()` method has been limited to 128 Kbytes (special characters and header names included).

19 Upgrade notes for 6.2.9

This minor version doesn't have any upgrade notes.

5. Quickstart

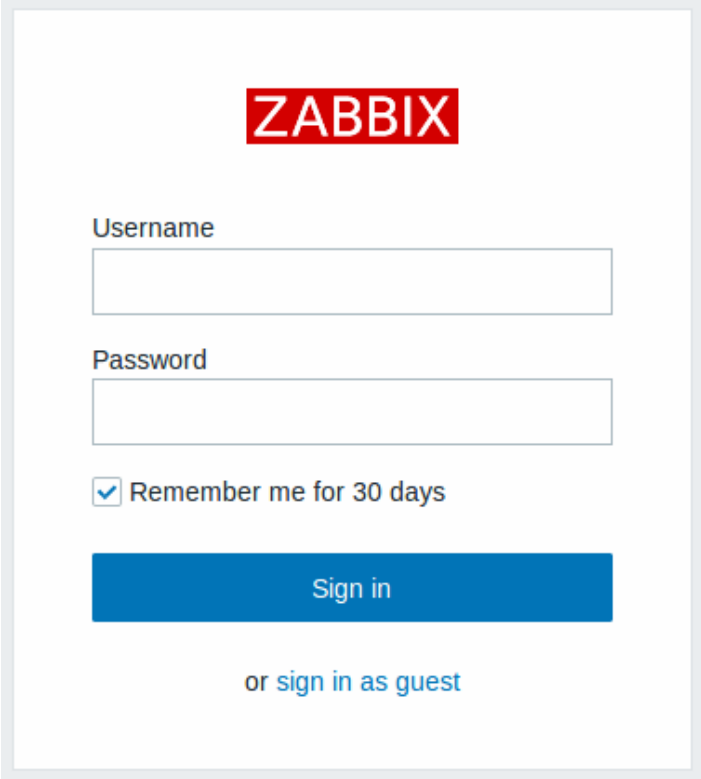
Please use the sidebar to access content in the Quickstart section.

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 *Configuration* and *Administration* menus 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.

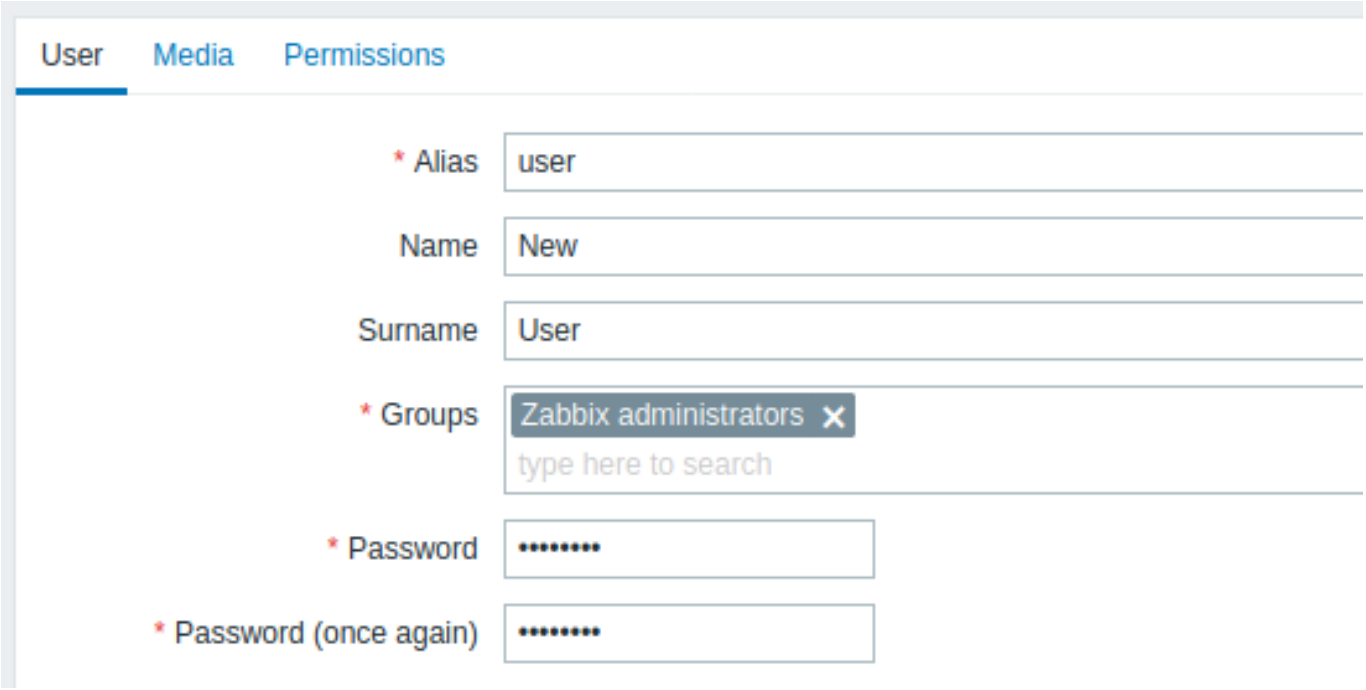
Adding user

To view information about users, go to *Administration* → *Users*.



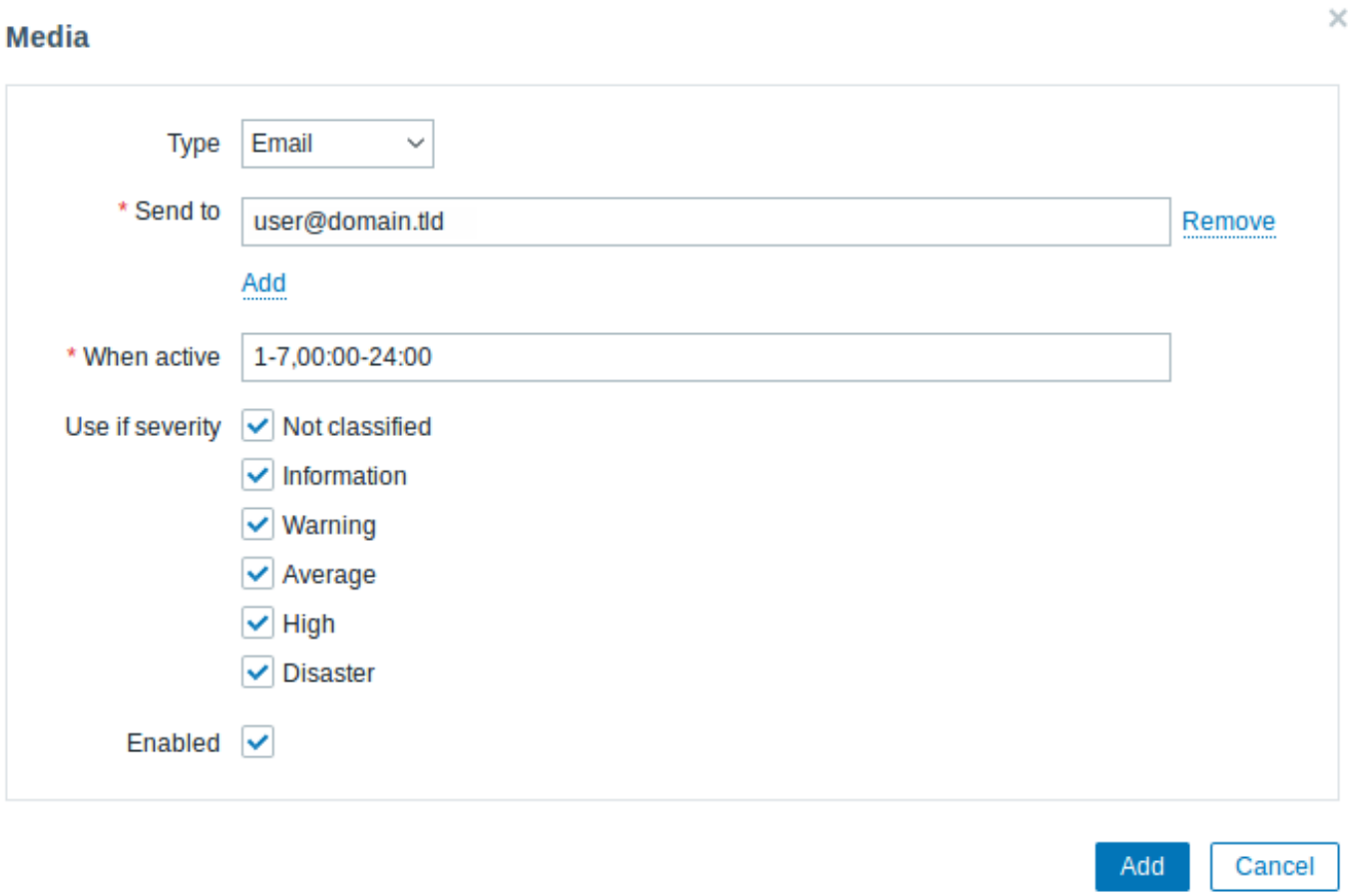
To add a new user, click on *Create user*.

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*.



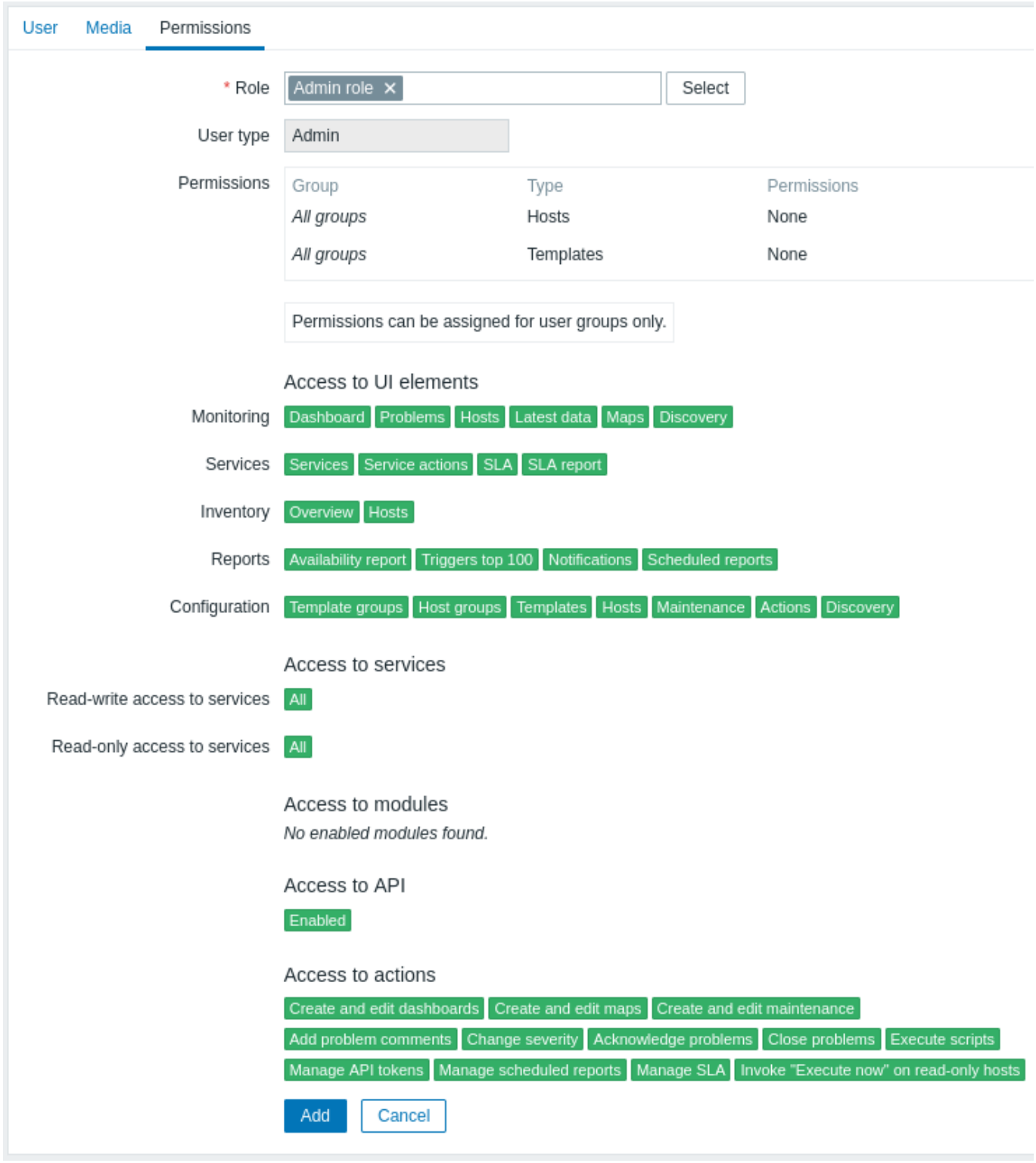
In this pop-up, enter an e-mail address for the user.

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:

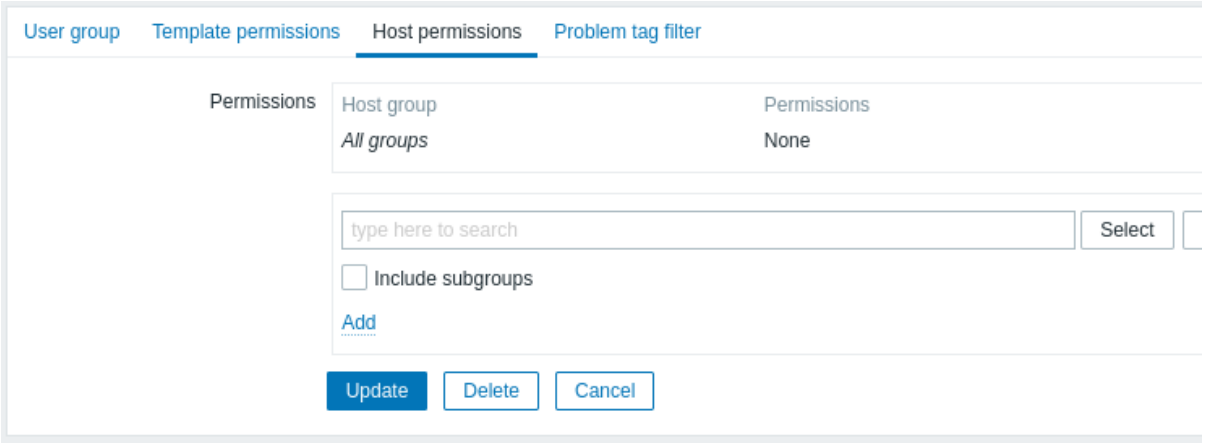


Click *Add* in the user properties form to save the user. The new user appears in the userlist.

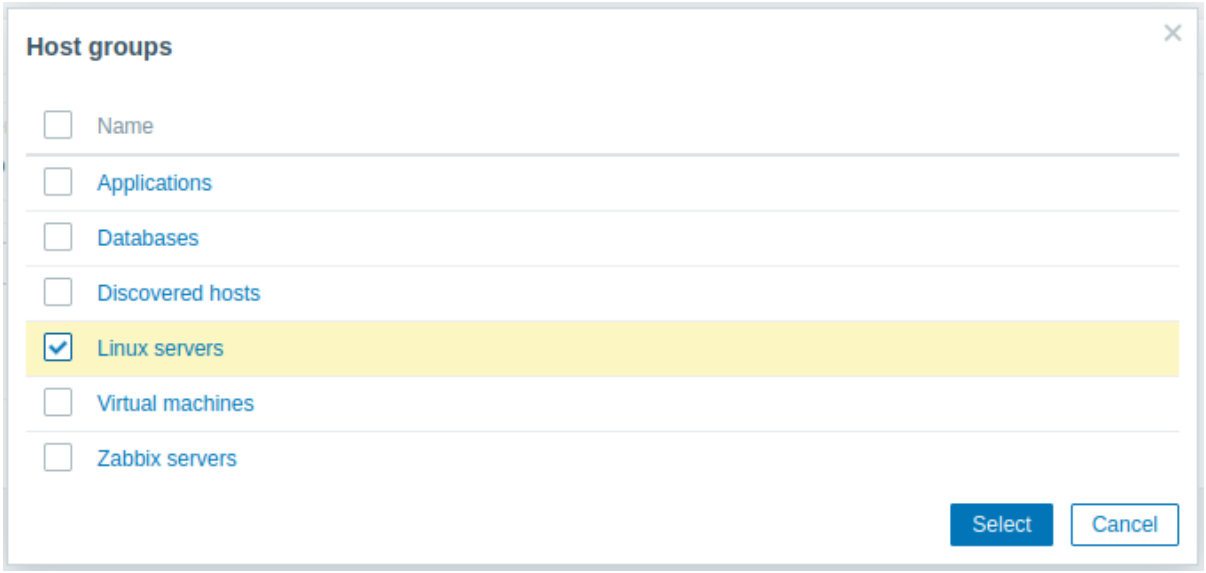


Adding permissions

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 user 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

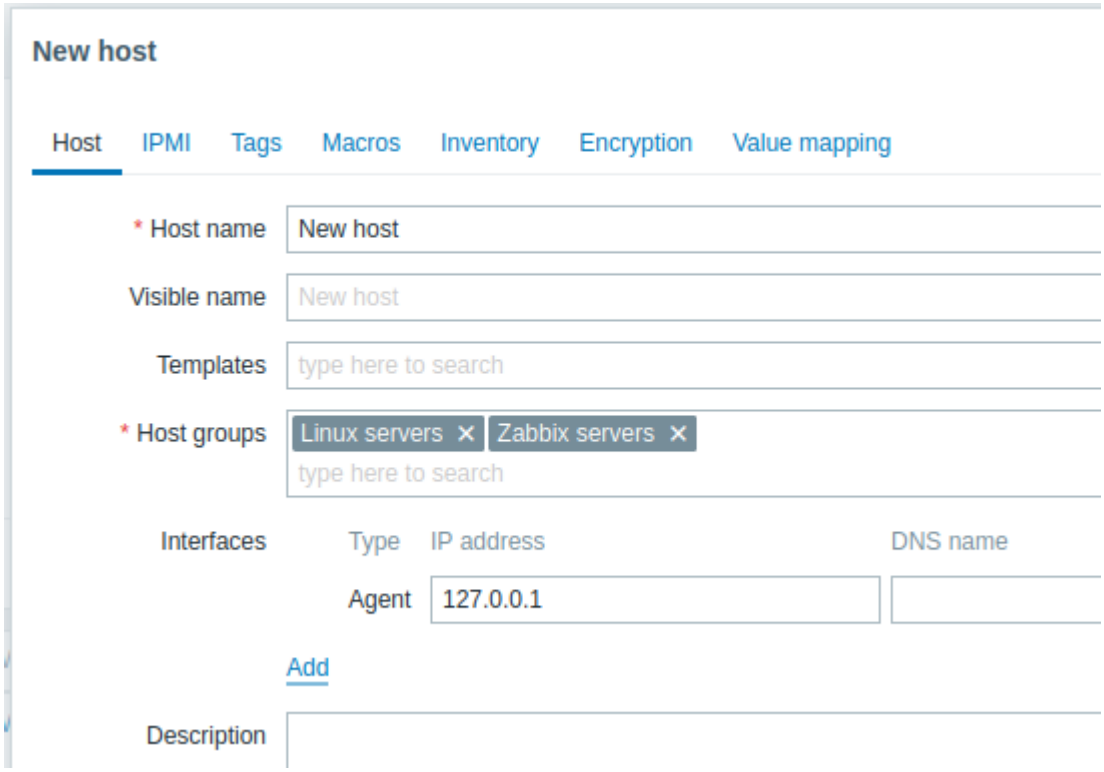
In this section you will learn how to set up a new host.

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 *Configuration* → *Hosts* and *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.

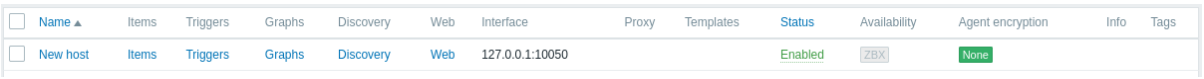
Note:
All access permissions are assigned to host groups, not individual hosts. That is why a host must belong to at least one group.

Interfaces: IP address

- Although not a required field technically, you may want to enter the IP address of the host. Note that if this is the Zabbix server IP address, it must be specified in the Zabbix agent configuration file 'Server' directive.

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:

- ZBX - host status has not been established; no metric check has happened yet
- ZBX - host is available, a metric check has been successful
- ZBX - 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

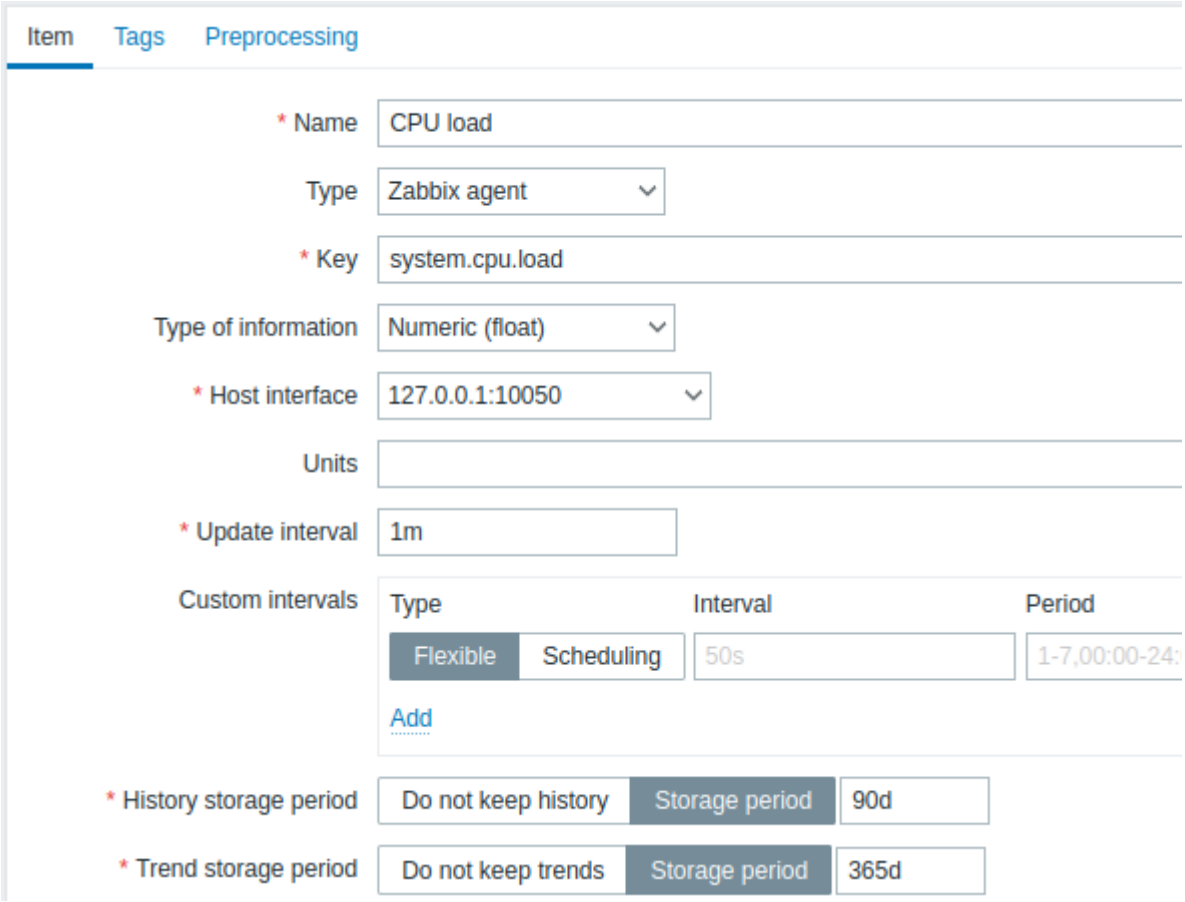
In this section, you will learn how to set up an item.

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 *Configuration* → *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.

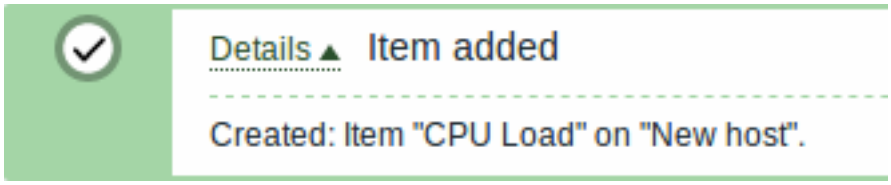
Type of information

- 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.

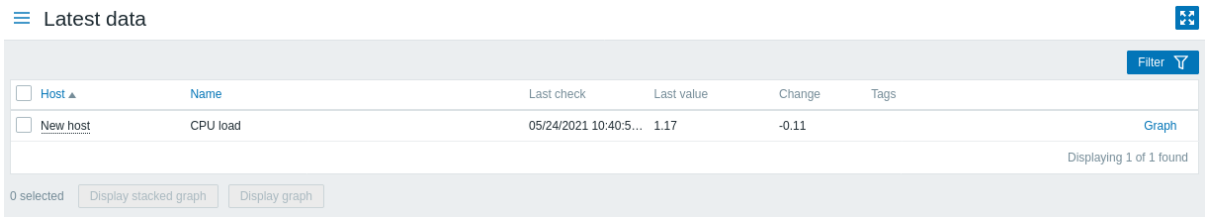
Other options will suit us with their defaults for now.

When done, click *Add*. The new item should appear in the item list. Click on *Details* above the list to view what exactly was done.



Seeing 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.

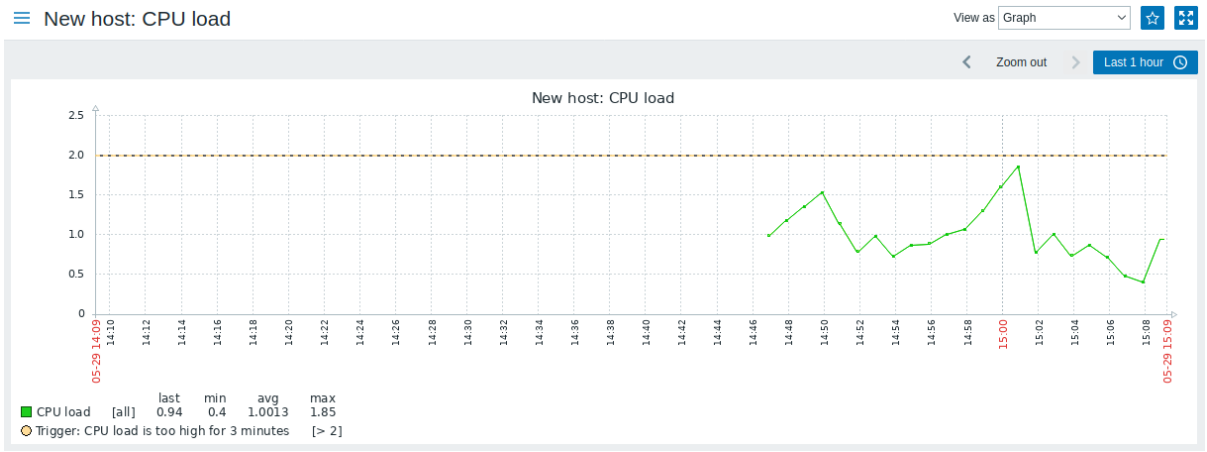
If you do not see information about the item as in the screenshot, make sure that:

- 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
- a host is selected in the host dropdown, the item is active

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

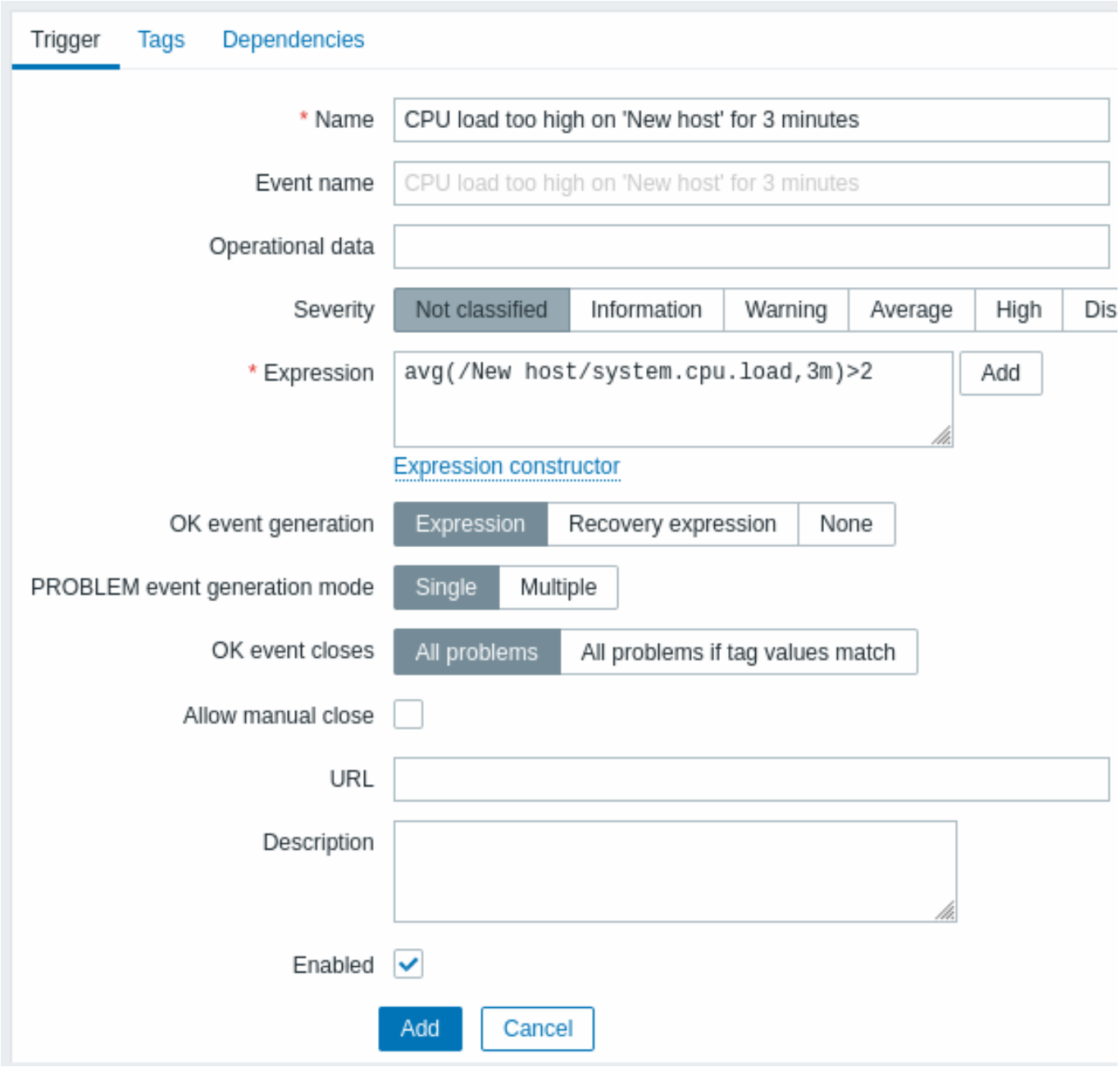
In this section you will learn how to set up a trigger.

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 *Configuration* → *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

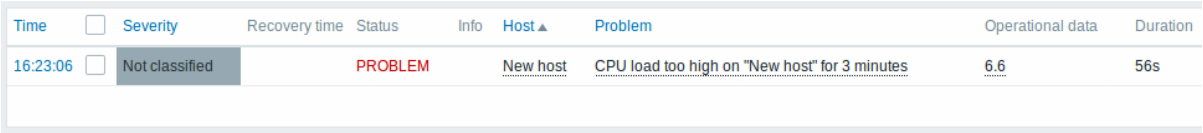
This is the trigger expression. Make sure that the expression is entered right, down to the last symbol. The item key here (system.cpu.load) is used to refer to the item. This particular expression basically says that the problem threshold is exceeded when the CPU load average value for 3 minutes is 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.

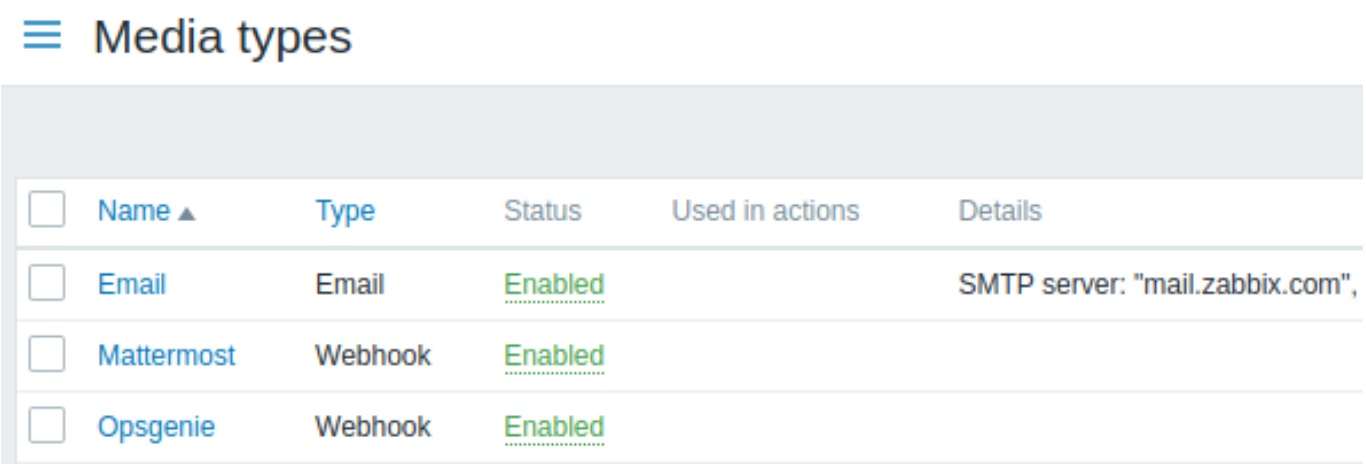
With items collecting data and triggers designed to ”fire” upon problem situations, it would also be useful to have some alerting mechanism in place that would notify us about important events even when we are not directly looking at Zabbix frontend.

This is what notifications do. E-mail being the most popular delivery method for problem notifications, we will learn how to set up an e-mail notification.

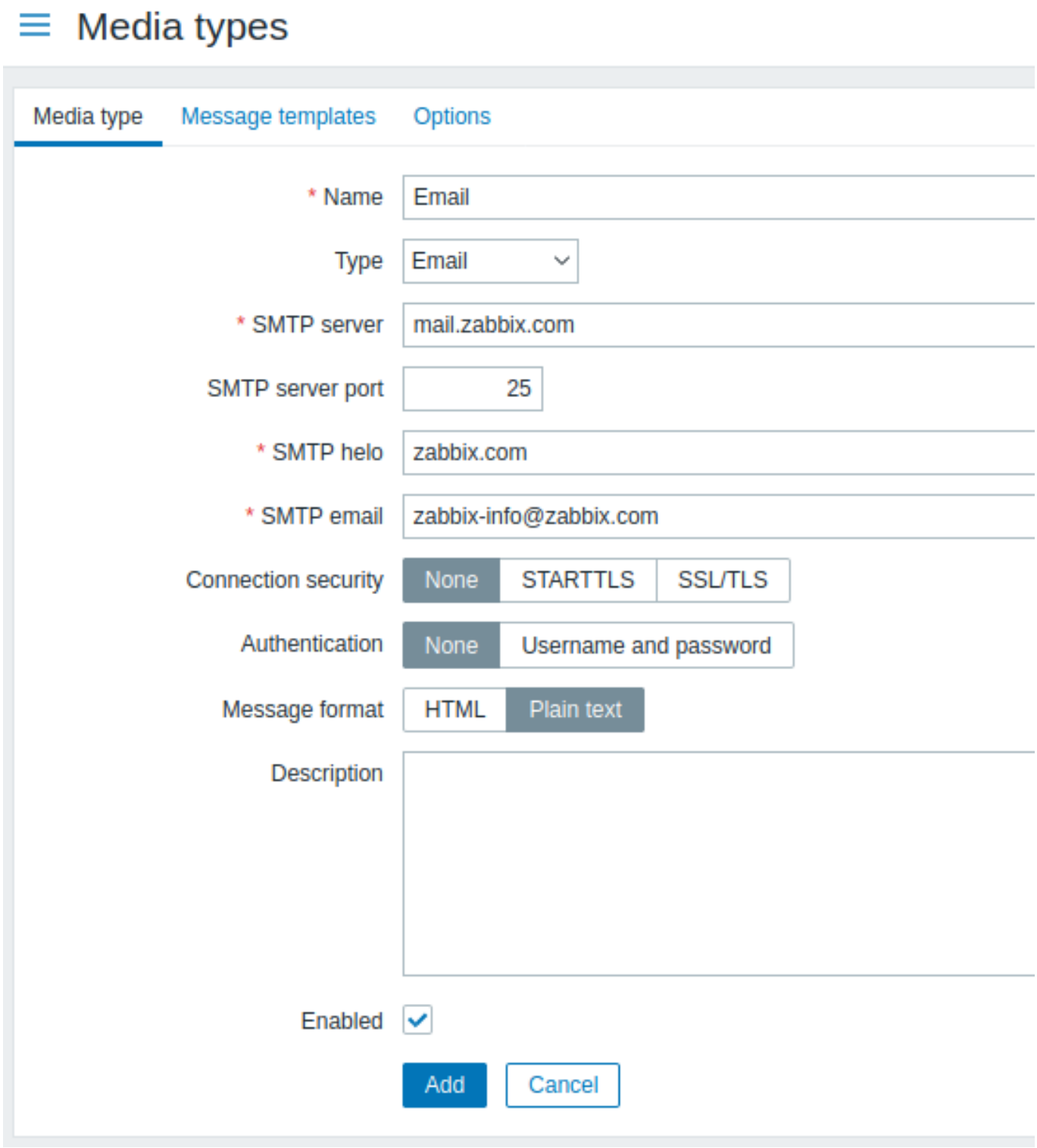
E-mail settings

Initially there are several predefined notification delivery methods in Zabbix. E-mail is one of those.

To configure e-mail settings, go to *Administration* → *Media types* and click on *Email* in the list of pre-defined media types.



This will present us with the e-mail settings definition form.



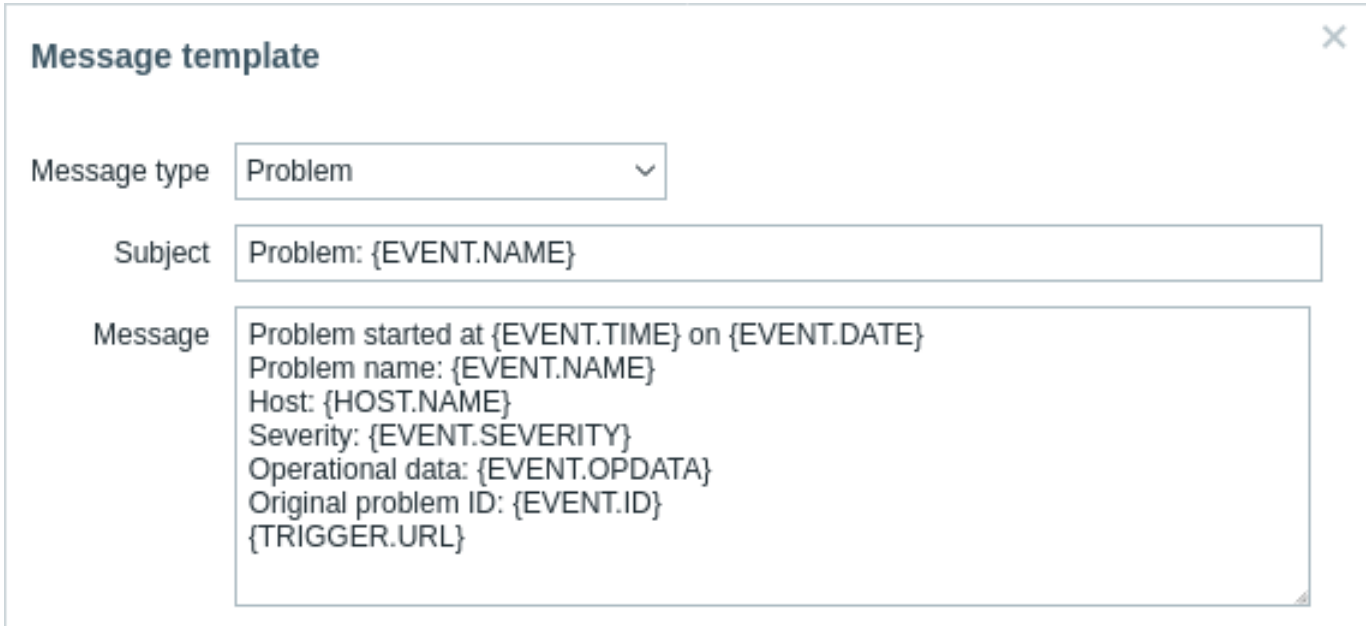
All mandatory input fields are marked with a red asterisk.

In the *Media type* tab, set the values of SMTP server, SMTP helo and SMTP e-mail 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.

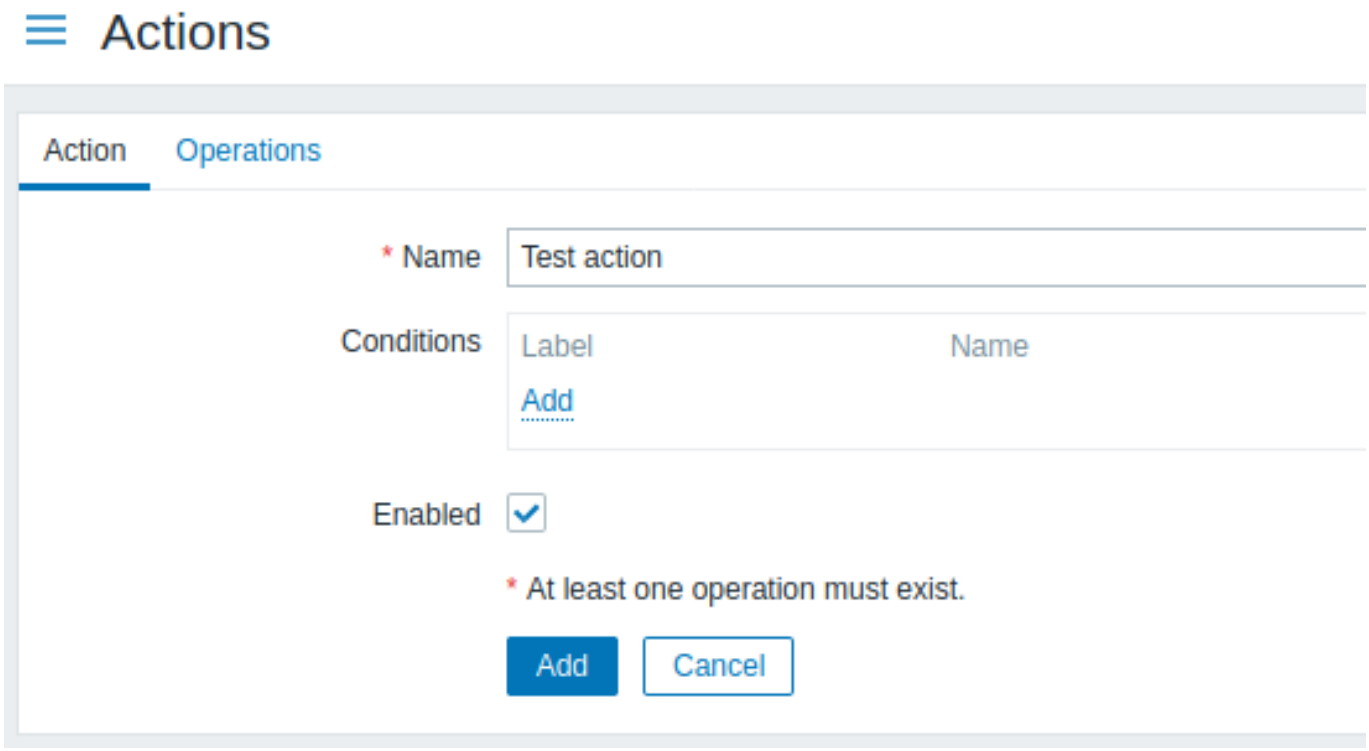


Click on *Add* when ready and save the form.

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 *Configuration* → *Actions* and click on *Create action*.

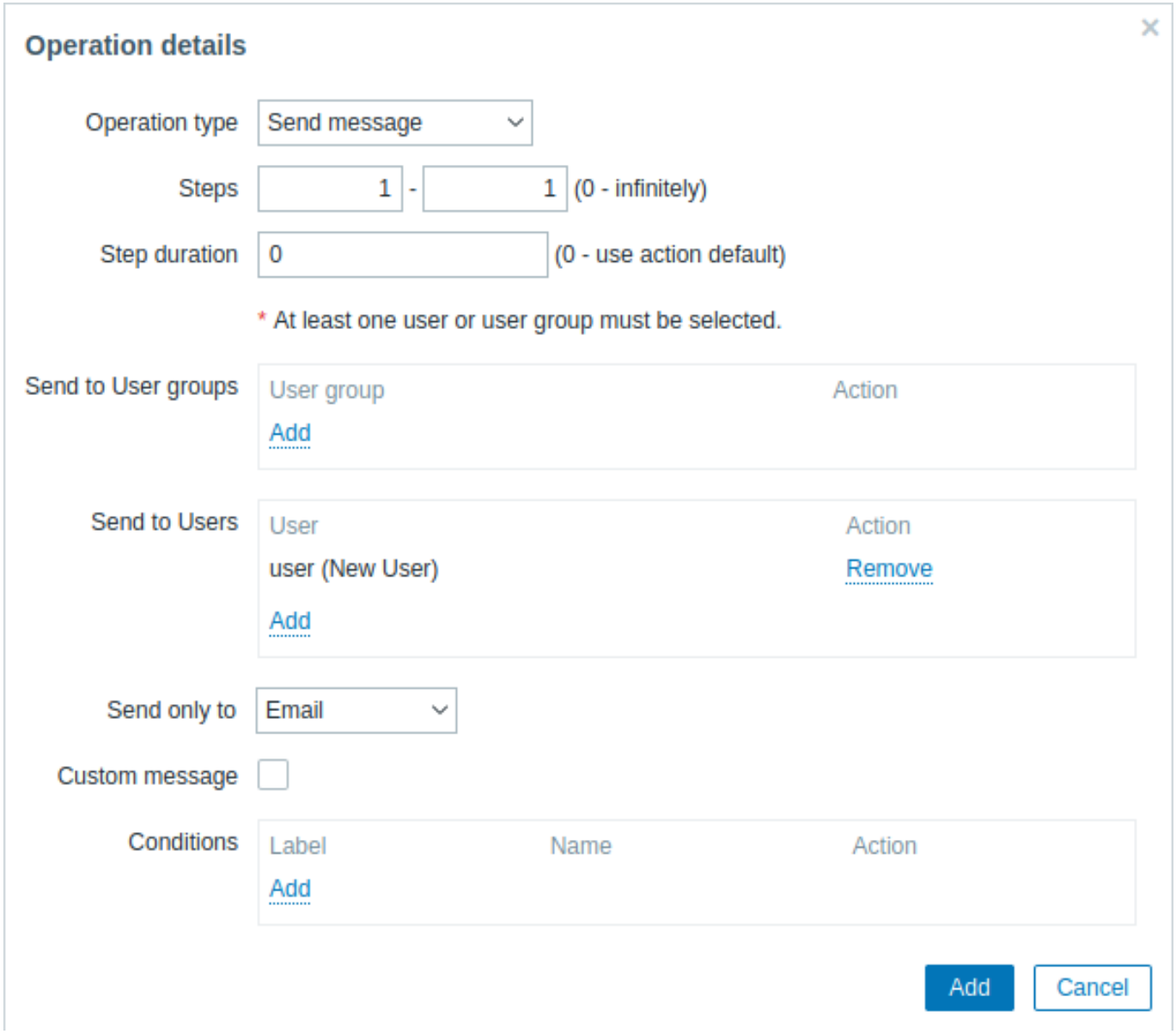


All mandatory input fields are marked with a red asterisk.

In this form, enter a name for the action.

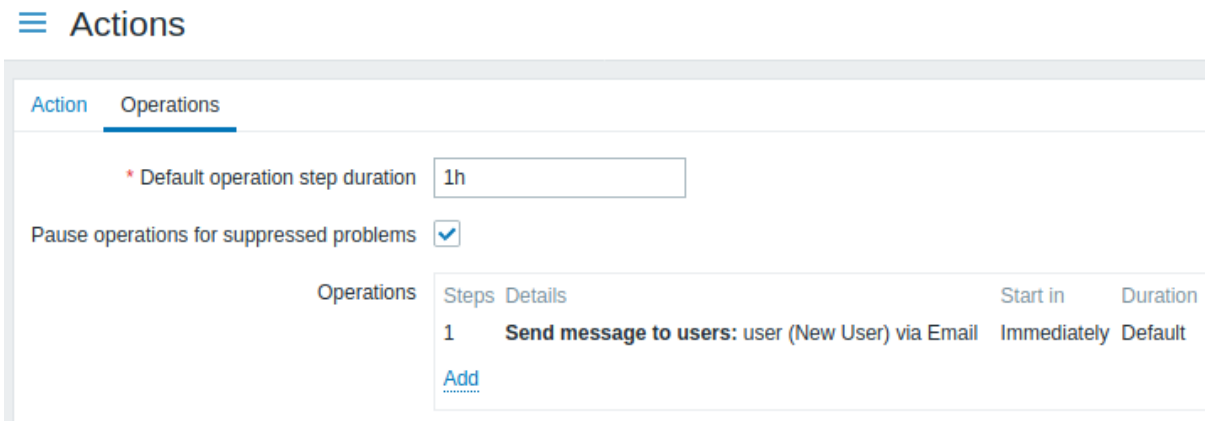
In the most simple case, if we do not add any more specific conditions, the action will be taken upon any trigger change from 'Ok' to 'Problem'.

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 e-mail

Attention:
If notifications do not work:

- verify once again that both the e-mail 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.

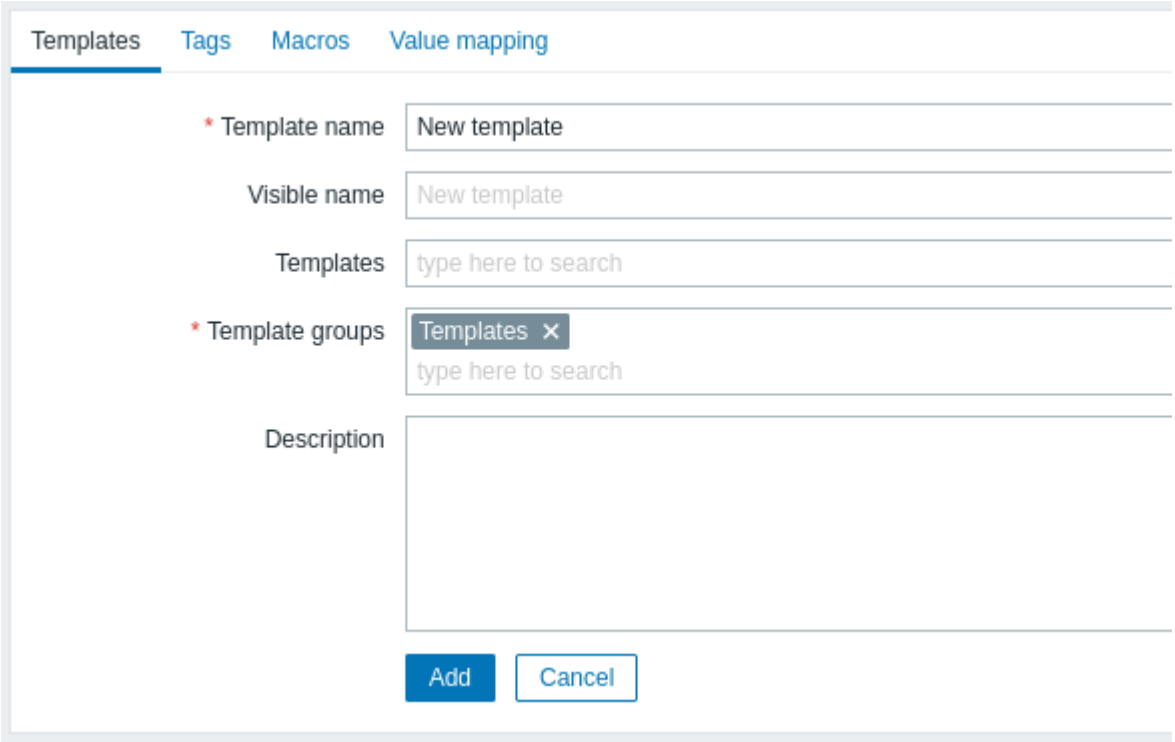
While all of these steps offer a great deal of flexibility in themselves, it may appear like a lot of steps to take if needed for, say, a thousand hosts. Some automation would be handy.

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 *Configuration* → *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

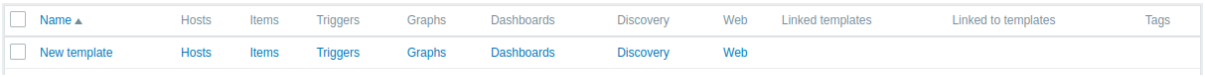
- Enter a template name. Alpha-numericals, spaces and underscores are allowed.

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.



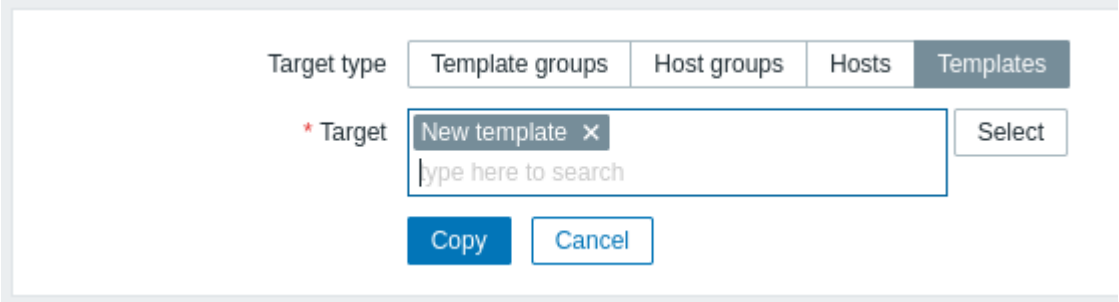
As you may see, the template is there, but it holds nothing in it - no items, triggers or other entities.

Adding item to template

To add an item to the template, go to the item list for 'New host'. In *Configuration* → *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 item to



All mandatory input fields are marked with a red asterisk.

- click on *Copy*

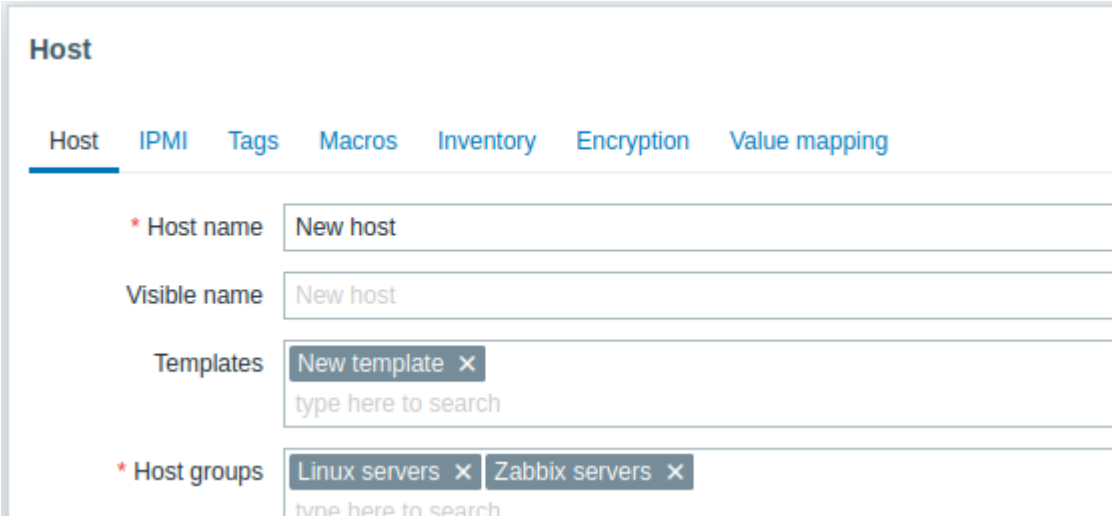
If you now go to *Configuration* → *Templates*, 'New template' should have one new item in it.

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 *Configuration* → *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 CentOS 8 (x86_64).

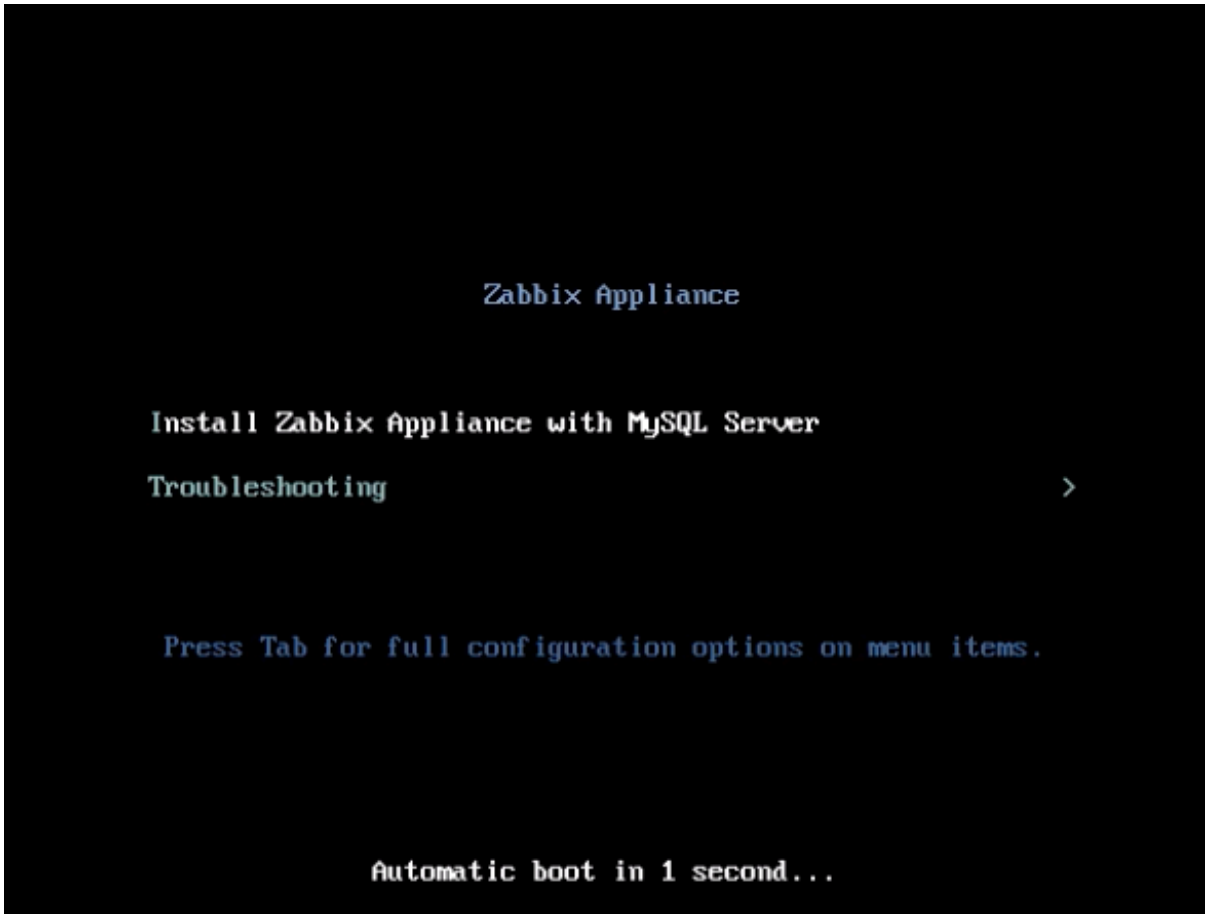
Zabbix appliance installation CD can be used for instant deployment of Zabbix server (MySQL).

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 CentOS 8 configuration The appliance is based on AlmaLinux 8.

1.1 Repositories

Official Zabbix repository has been added to */etc/yum.repos.d*:

```
[zabbix]
name=Zabbix Official Repository - $basearch
baseurl=http://repo.zabbix.com/zabbix/6.2/rhel/8/$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 converter.

7.2 HDD/flash image (raw)

```
dd if=./zabbix_appliance_5.2.0.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

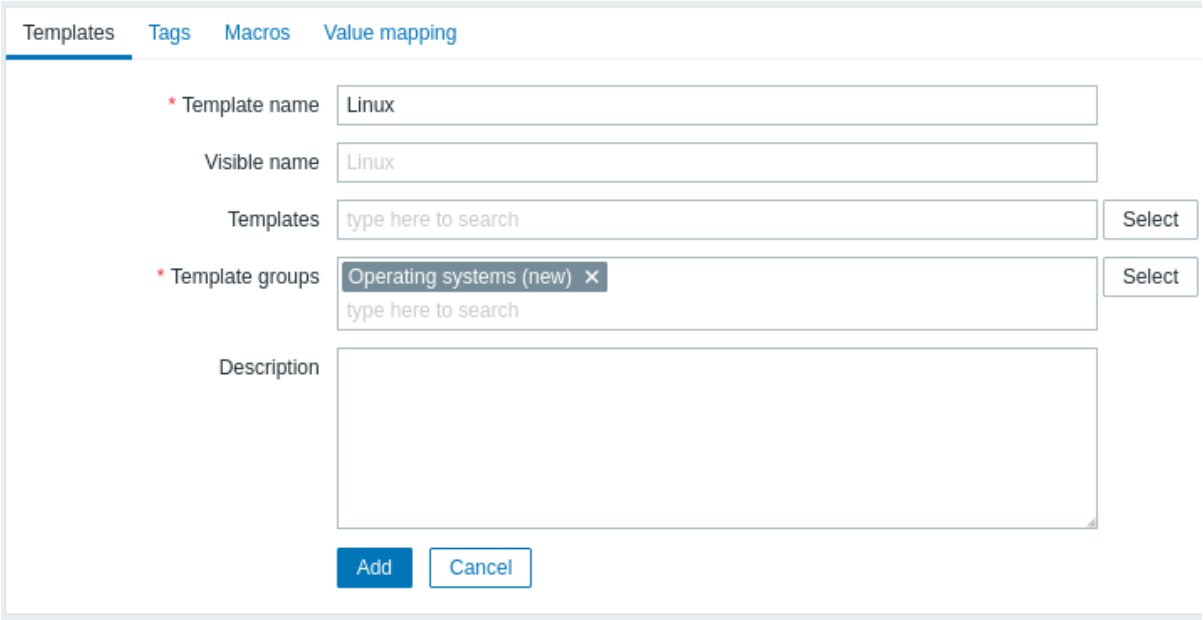
Configuring a template requires that you first create a template by defining its general parameters and then you add entities (items, triggers, graphs, etc.) to it.

Creating a template

To create a template, do the following:

- Go to *Configuration → Templates*
- Click on *Create template*
- Edit template attributes

The **Templates** tab contains general template attributes.

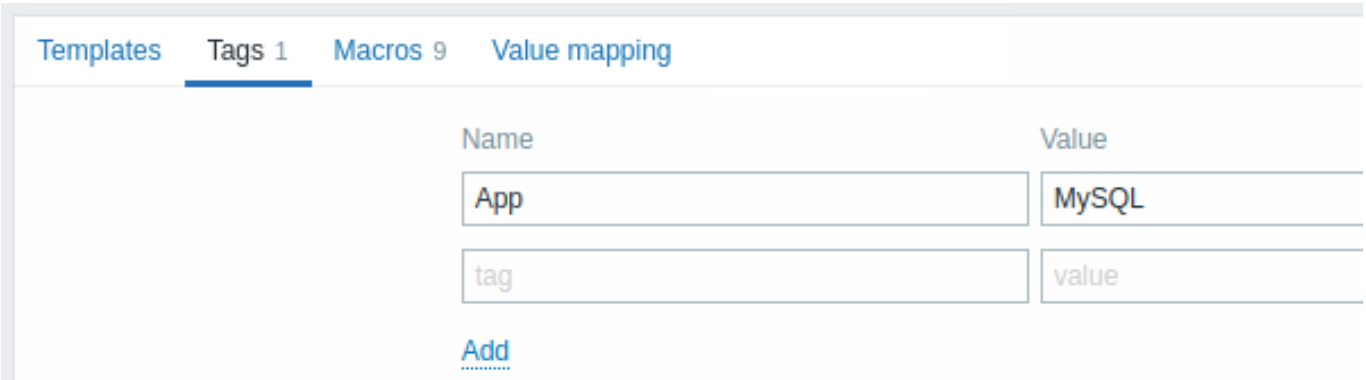


All mandatory input fields are marked with a red asterisk.

Template attributes:

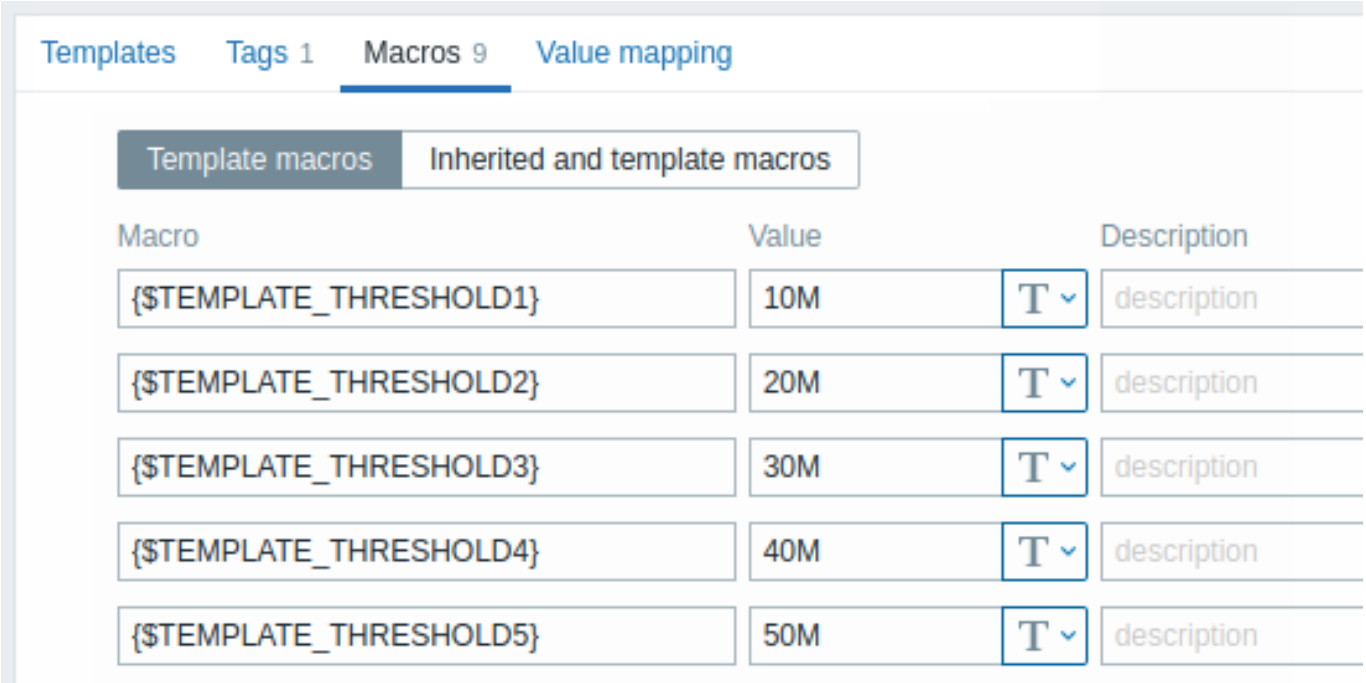
Parameter	Description
Template name	Unique template name. Alphanumerics, spaces, dots, dashes, and underscores are allowed. However, leading and trailing spaces are disallowed.
Visible name	If you set this name, it will be the one visible in lists, maps, etc.
Templates	Link one or more "nested" templates to this template. All entities (items, triggers, graphs, 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 popup 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 items, triggers, and graphs *Unlink and clear* - unlink the template and remove all its items, triggers, and graphs
Template groups	Groups the template belongs to.
Description	Enter the template description.

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.

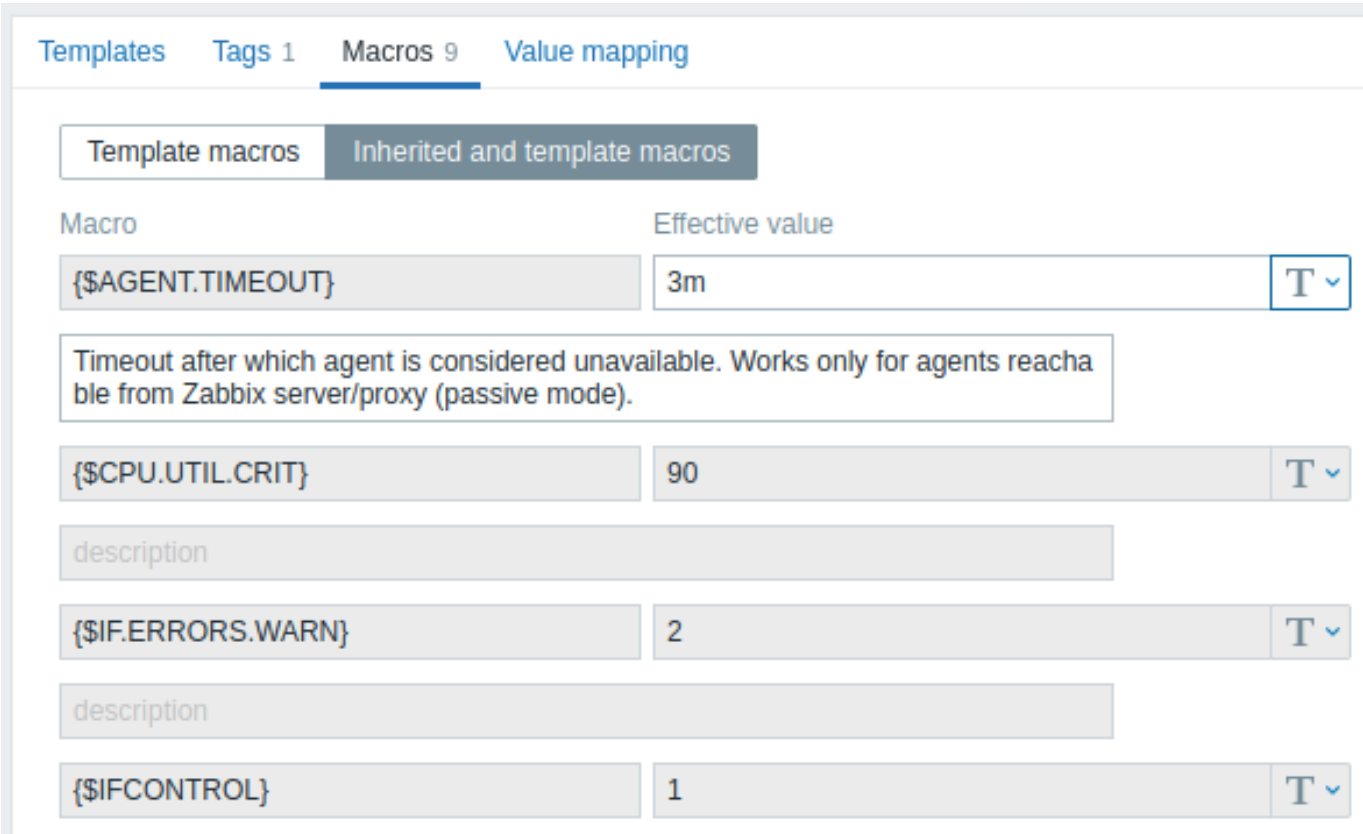


User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags.

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.



You may also view here macros from linked templates and global macros if you select the *Inherited and template macros* option. That is where all defined user macros for the template 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 nested template/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.
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.
Delete	Delete the template; entities of the template (items, triggers, etc) remain with the linked hosts.
Clear history and trends	Delete the template and all its entities from linked hosts.
Cancel	Cancel the editing of template properties.

With a template created, it is time to add some entities to it.

Attention:
Items have to be added to a template first. Triggers and graphs cannot be added without the corresponding item.

Adding items, triggers, graphs

To add items to the template, do the following:

- Go to *Configuration* → *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 *Configuration* → *Templates*, do the following:

- Click on *Dashboards* in the row of the template
- Configure a dashboard following the guidelines of configuring dashboards

Attention:
The widgets that can be included in a template dashboard are: classic graph, graph prototype, clock, 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 *Configuration* → *Templates*, do the following:

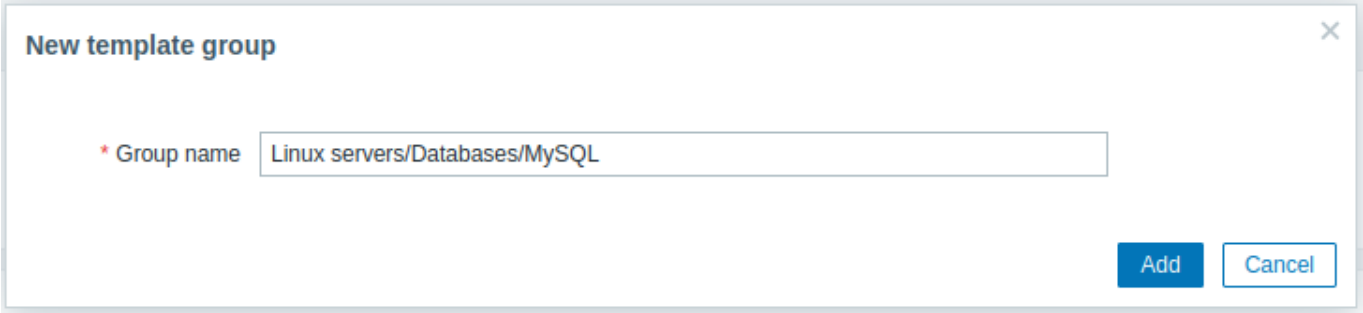
- Click on *Web* in the row of the template
- Configure a web scenario following the usual method of configuring web scenarios

Creating a template group

Attention:
Only Super Admin users can create template groups.

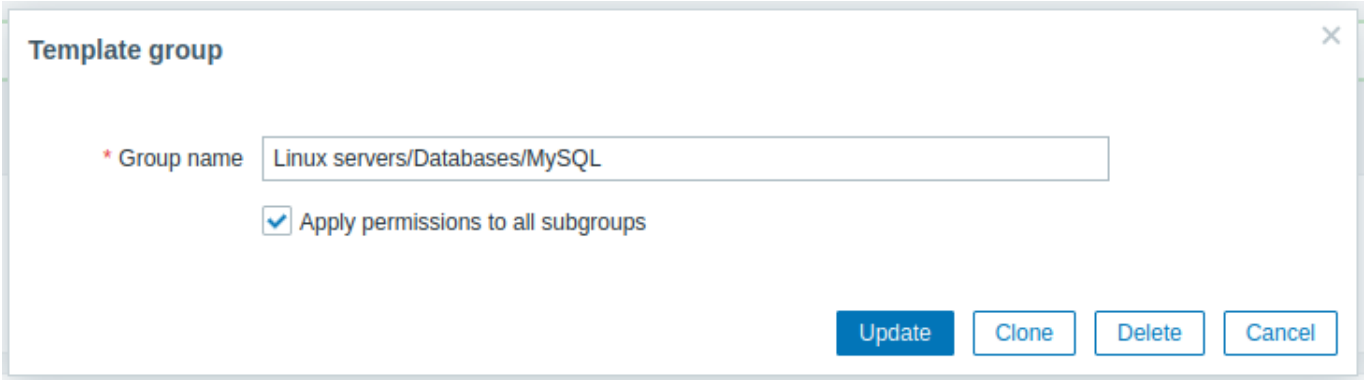
To create a template group in Zabbix frontend, do the following:

- Go to: *Configuration* → *Template groups*
- Click on *Create template group* in the upper right corner of the screen
- Enter the group name in the form



To create a nested template group, use the '/' forward slash separator, for example `Linux servers/Databases/MySQL`. You can create this group even if none of the two parent template groups (`Linux servers/Databases/`) exist. In this case creating these parent template groups is up to the user; they will not be created automatically.
Leading and trailing slashes, several slashes in a row are not allowed. Escaping of '/' is not supported.

Once the group is created, you can click on the group name in the list to edit group name, clone the group or set additional option:



Apply permissions to all subgroups - mark this checkbox and click on *Update* to apply the same level of permissions to all nested template groups. For user groups that may have had differing permissions assigned to nested template groups, the permission level of the parent template group will be enforced on the nested groups. This is a one-time option that is not saved in the database.

Permissions to nested template groups

- When creating a child template group to an existing parent template group, user group permissions to the child are inherited from the parent (for example, when creating `Databases/MySQL` if `Databases` already exists)
- When creating a parent template group to an existing child template group, no permissions to the parent are set (for example, when creating `Databases` if `Databases/MySQL` already exists)

2 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:

- Go to *Configuration* → *Hosts*
- Click on the required host
- 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
- Click on *Add/Update* in the host attributes form

The host will now have all the entities (items, triggers, graphs, etc) of the template.

Attention:
Linking multiple templates to the same host will fail if in those templates there are 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, graphs 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.

If some entity (item, trigger, graph etc.) is not prefixed by the template name, it means that it existed on the host before and was not added by the template.

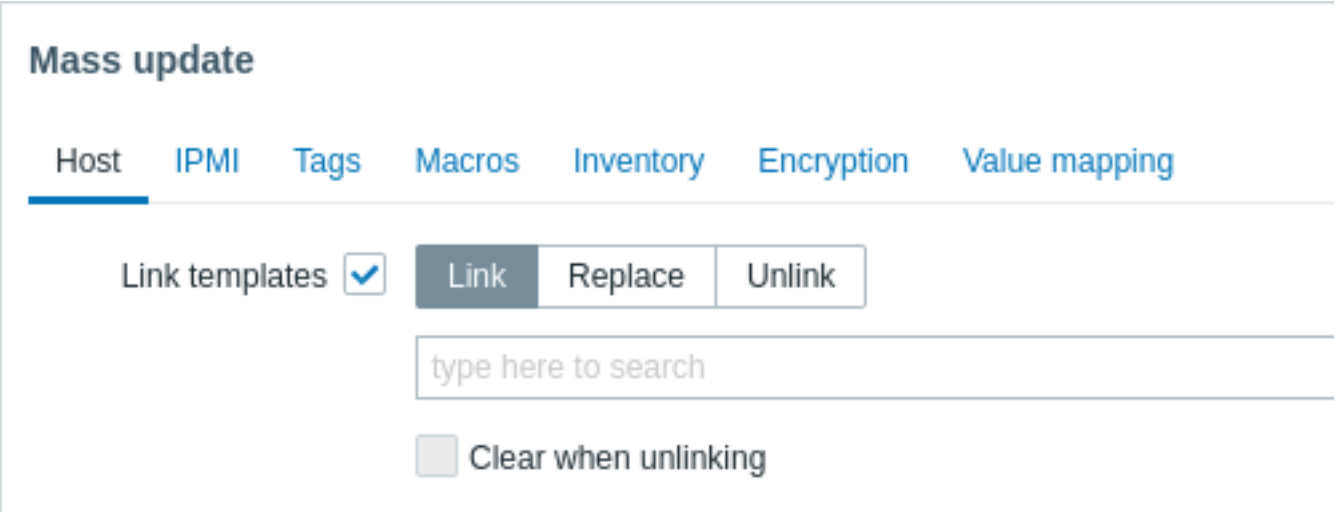
Entity uniqueness criteria

When adding entities (items, triggers, graphs 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 *Configuration* → *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 elements 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 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 one-touch manner on the template level. However, you still can, for example, enable/disable an item on the individual host 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:

- Go to *Configuration* → *Hosts*
- Click on the required host and find the *Templates* field
- Click on *Unlink* or *Unlink and clear* next to the template to unlink
- Click on *Update* in the host attributes form

Choosing the *Unlink* option will simply remove association with the template, while leaving all its entities (items, triggers, graphs etc.) with the host.

Choosing the *Unlink and clear* option will remove both the association with the template and all its entities (items, triggers, graphs etc.).

3 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 (”nest”, parent template) to the host and the host will inherit all entities of the linked templates (”nested”, child templates) automatically. For example, if we link templates T1 and T2 to template T3, we supplement T3 with entities from T1 and T2, and not vice versa. If we link template A to templates B and C, we supplement B and C with entities from A.

Configuring nested templates

To link templates, you need to take an existing template or a new one, and then:

- Open the template configuration form
- Find the *Templates* field
- Click *Select* to open the *Templates* popup window
- In the popup window, choose required templates, then click *Select* to add the templates to the list
- Click *Add* or *Update* in the template configuration form

Thus, all entities of the parent template, as well as all entities of linked templates (such as items, triggers, graphs, etc.) will now appear in the template configuration, except for linked template dashboards, which will, nevertheless, be inherited by hosts.

To unlink any of the linked templates, in the same form use the *Unlink* or *Unlink and clear* buttons and click *Update*.

Choosing the *Unlink* option will simply remove the association with the linked template, while not removing all its entities (items, triggers, graphs, etc.).

Choosing the *Unlink and clear* option will remove both the association with the linked template and all its entities (items, triggers, graphs, etc.).

4 Mass update

Overview

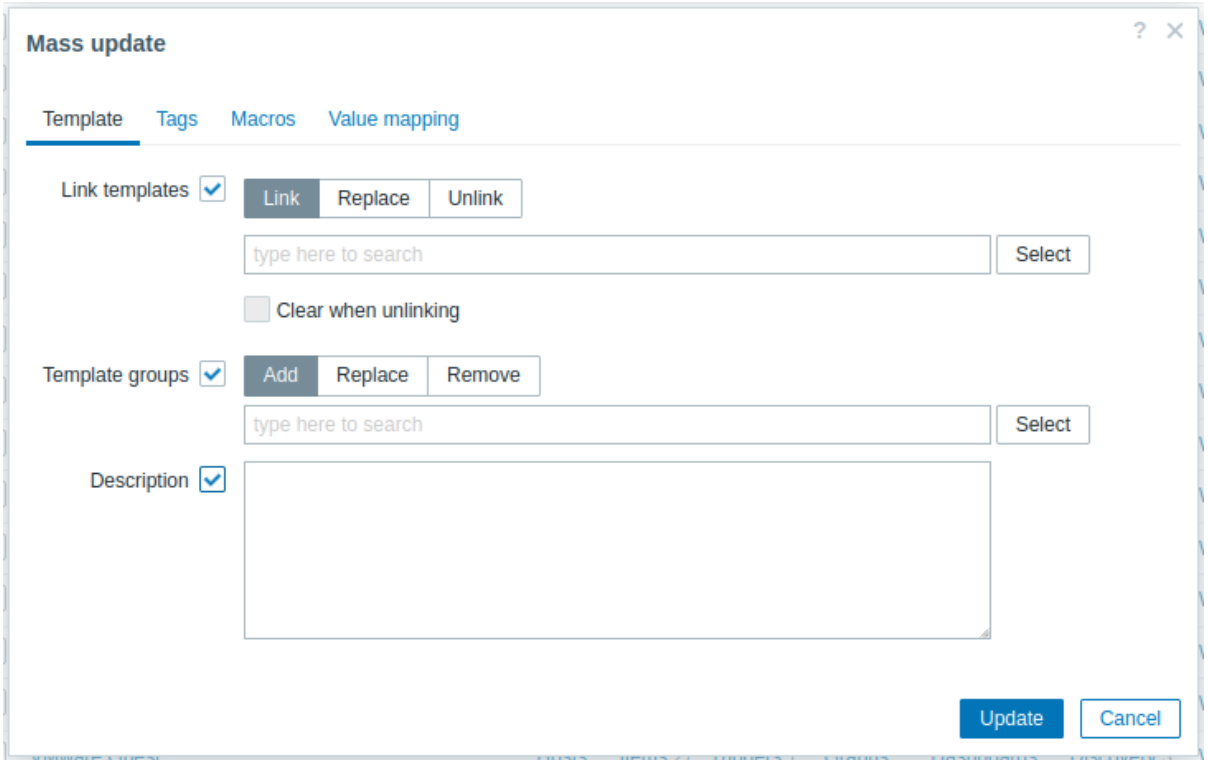
Sometimes you may want to change some attribute for a number of templates at once. Instead of opening each individual template for editing, you may use the mass update function for that.

Using mass update

To mass-update some templates, do the following:

- Mark the checkboxes before the templates you want to update in the template list

- Click on *Mass update* below the list
- Navigate to the tab with required attributes (*Template*, *Tags*, *Macros* 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 templates before
- *Unlink* - specify which templates to unlink

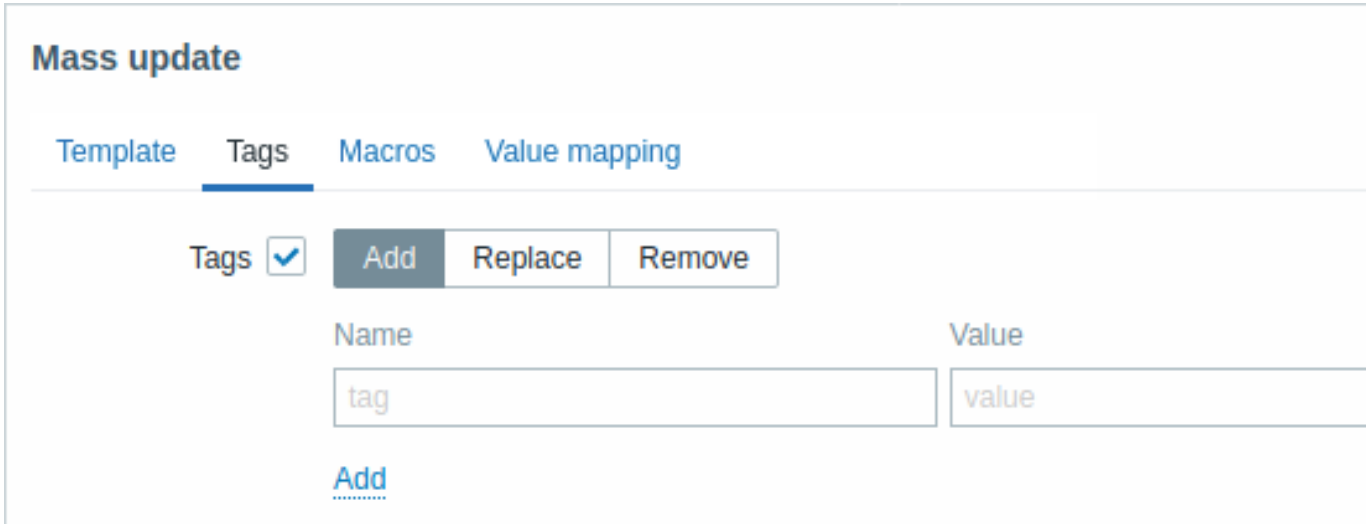
To specify the templates to link/unlink start typing the template name in the auto-complete field until a dropdown appears offering the matching templates. Just scroll down to select the required template.

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.).

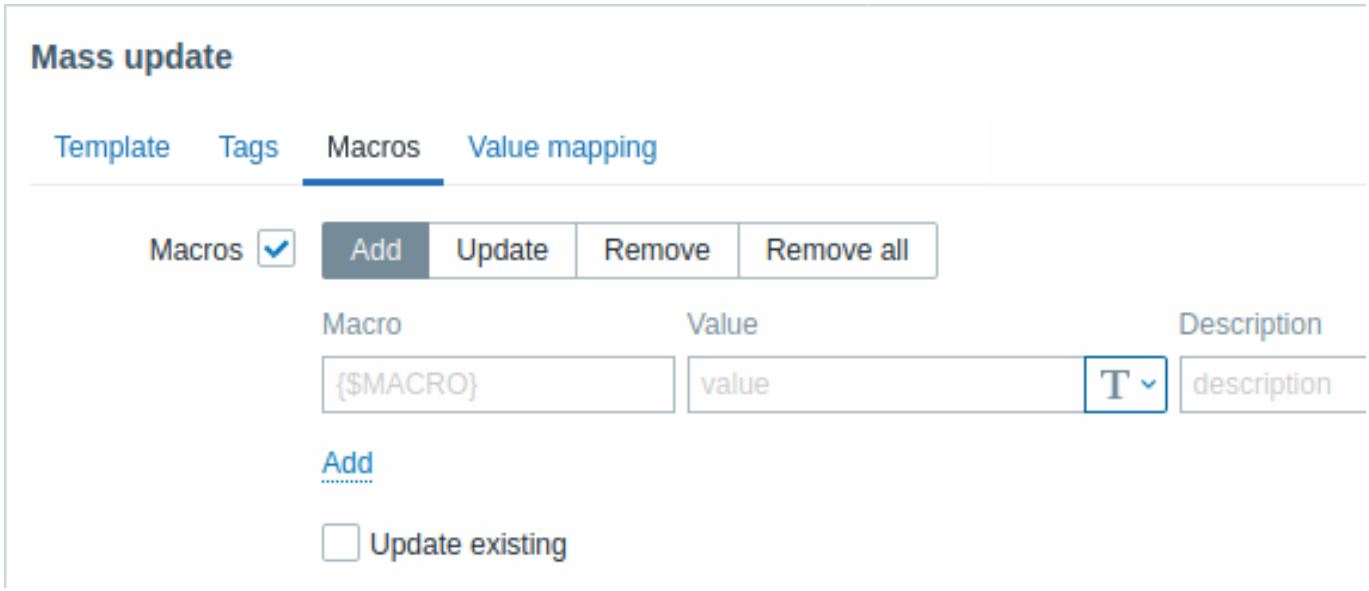
The following options are available when selecting the respective button for **template group** update:

- *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.

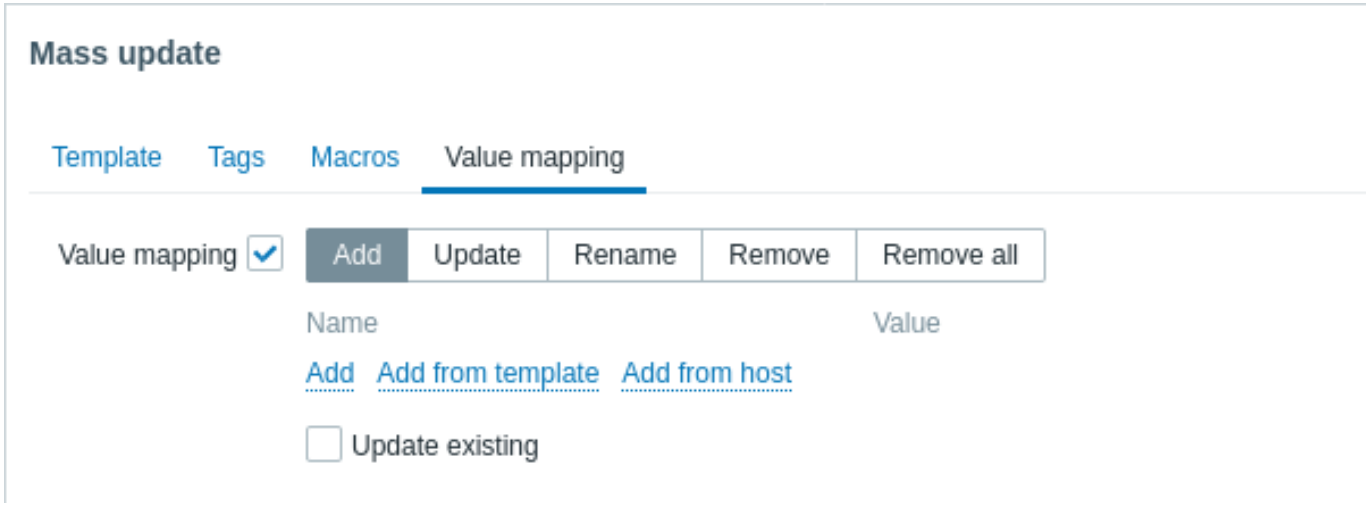


User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags. Note, that tags with the same name, but different values are not considered 'duplicates' and can be added to the same template.



The 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.



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 (since Zabbix 6.2.7). 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:

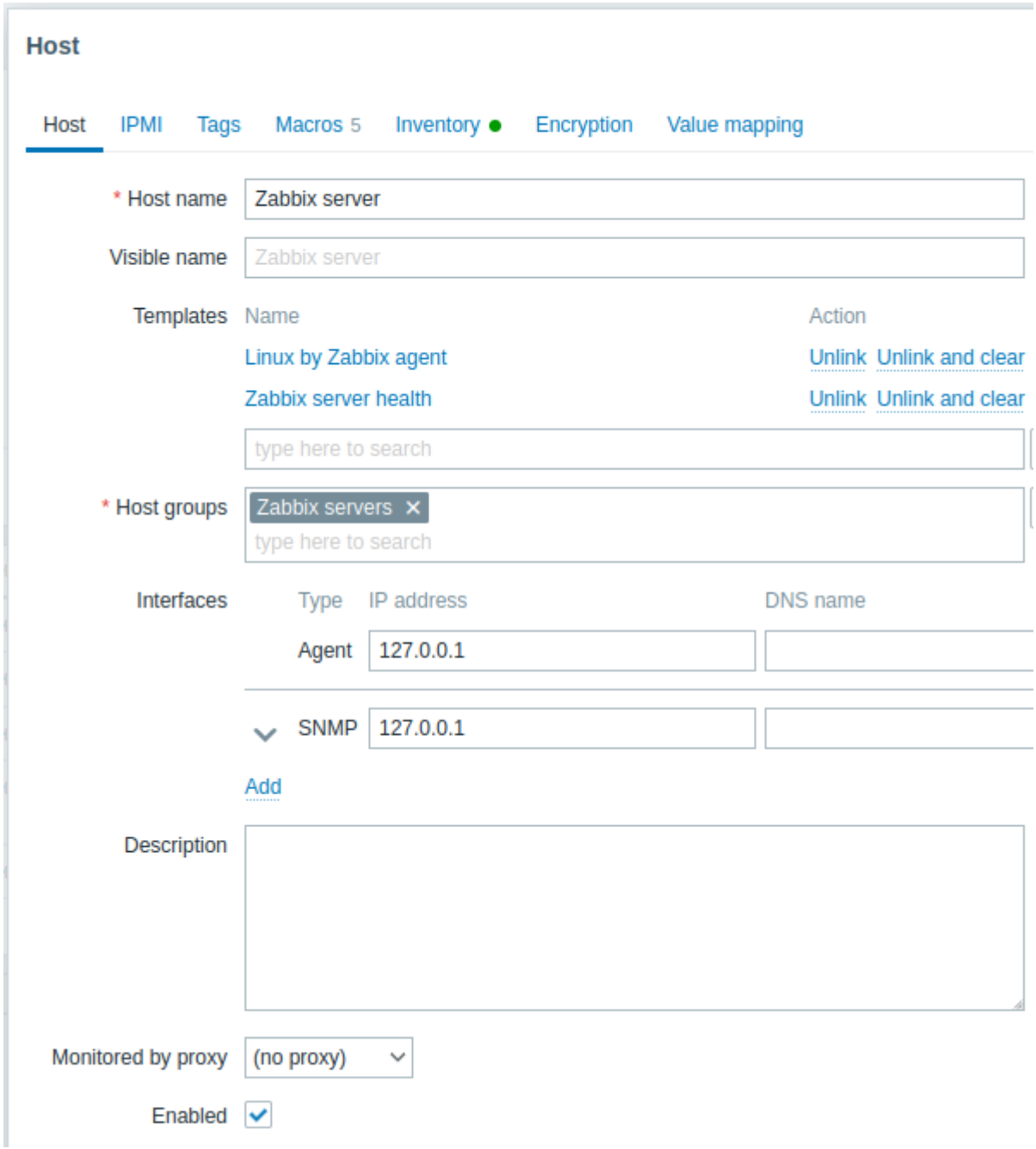
- Go to: *Configuration* → *Hosts* or *Monitoring* → *Hosts*
- Click on *Create host* to the right (or on the host name to edit an existing host)
- Enter parameters of the host in the form

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 (applications, 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.
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.

Parameter	Description
Templates	Link templates to the host. All entities (items, triggers, graphs, etc) will be inherited from the template. 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 popup window. The templates that are selected in the *Link new templates* field 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 items, triggers and graphs *Unlink and clear* - unlink the template and remove all its items, triggers and graphs 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 group 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	Mark the checkbox to make the host active, ready to be monitored. If unchecked, the host is not active, thus not monitored.

The **IPMI** tab contains IPMI management attributes.

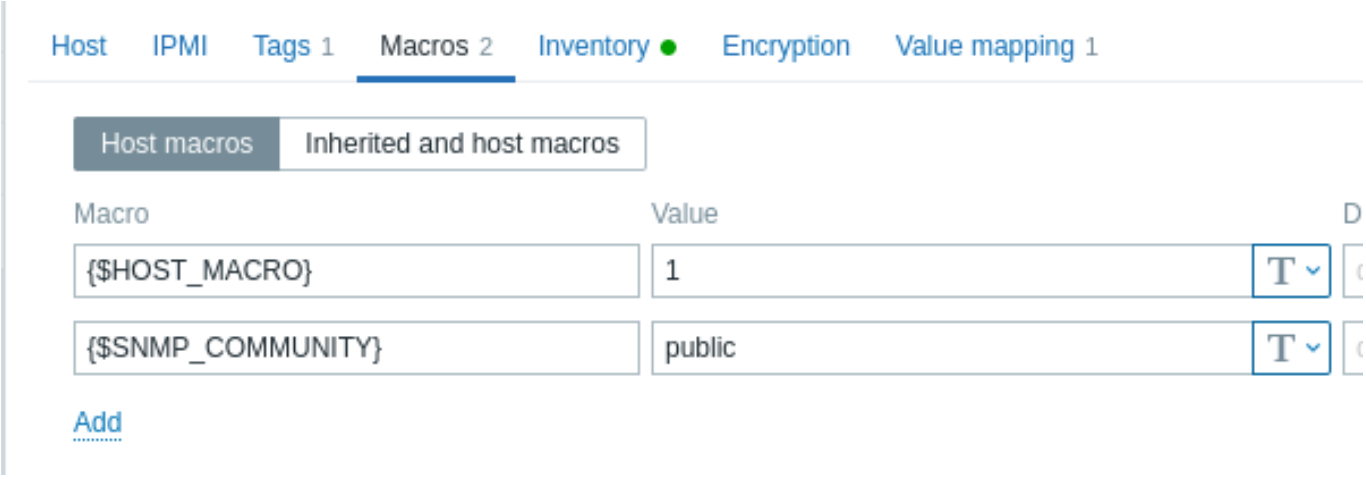
Parameter	Description
Authentication algorithm	Select the authentication algorithm.
Privilege level	Select the privilege level.
Username	User name for authentication. User macros may be used.
Password	Password for authentication. User macros may be used.

The **Tags** tab allows you to define host-level tags. All problems of this host will be tagged with the values entered here.

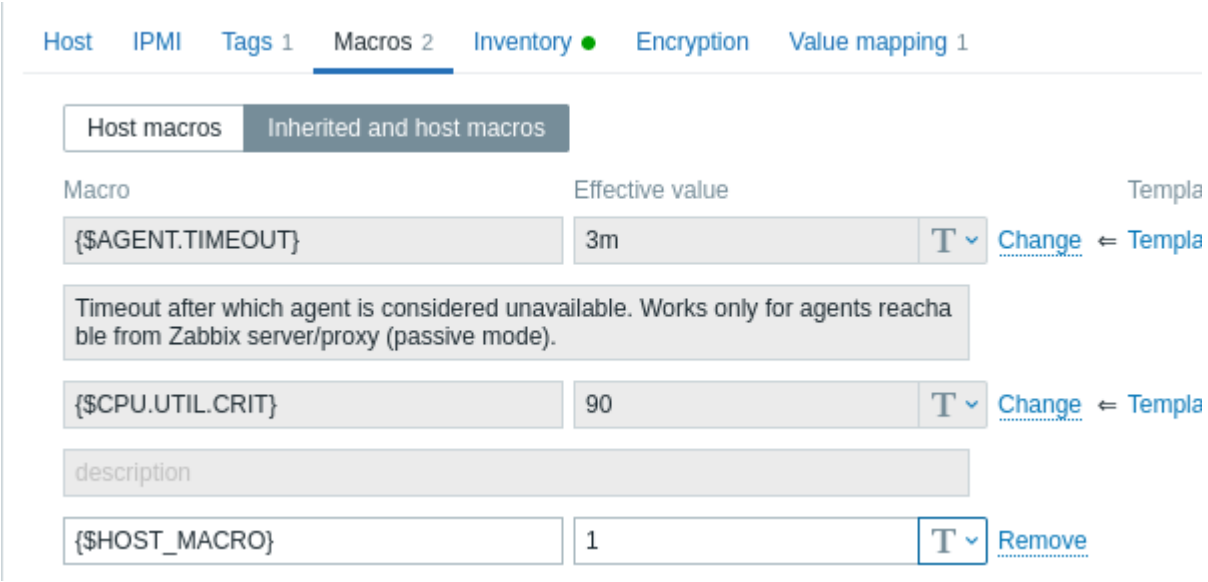


User macros, {INVENTORY.*} macros, {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros are supported in tags.

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.

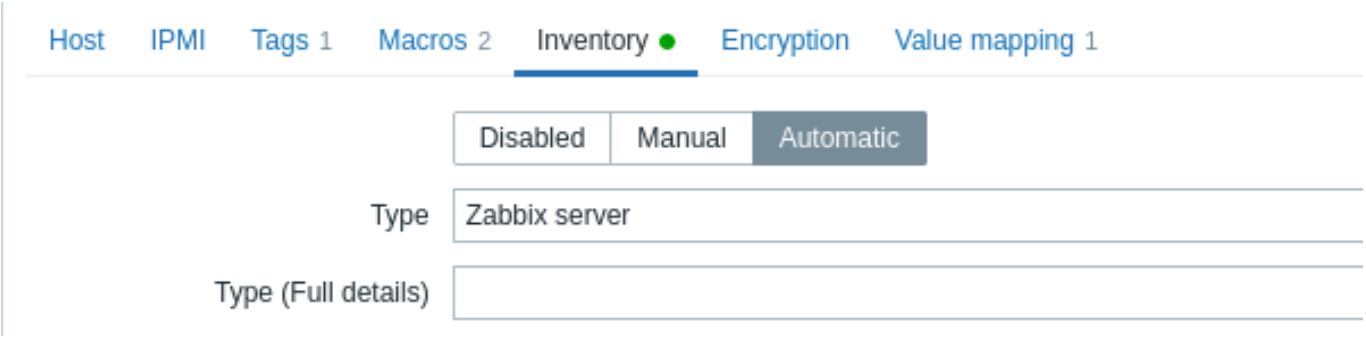


You may also view here template-level and global user macros if you select the *Inherited and host macros* option. That is where all defined user macros for the host are displayed with the value they resolve to as well as their origin.



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 **Host 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.

Parameter	Description
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

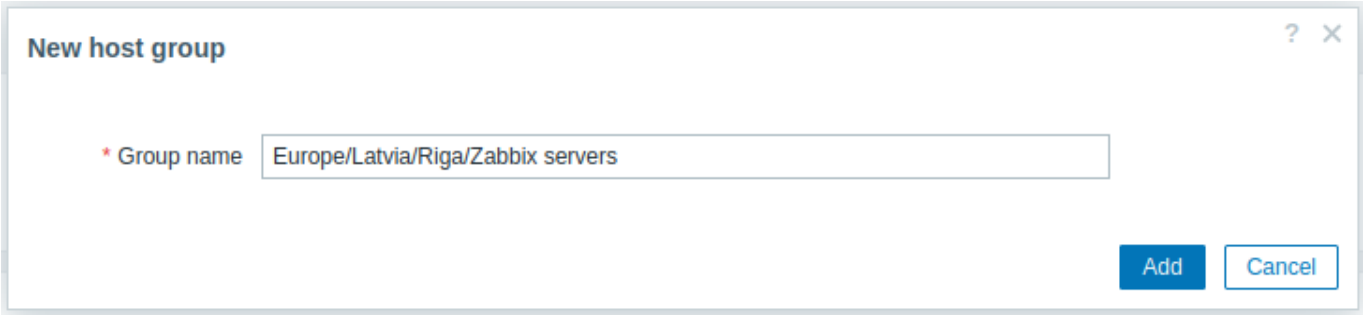
The **Value mapping** tab allows to configure human-friendly representation of item data in value mappings.

Creating a host group

Attention:
Only Super Admin users can create host groups.

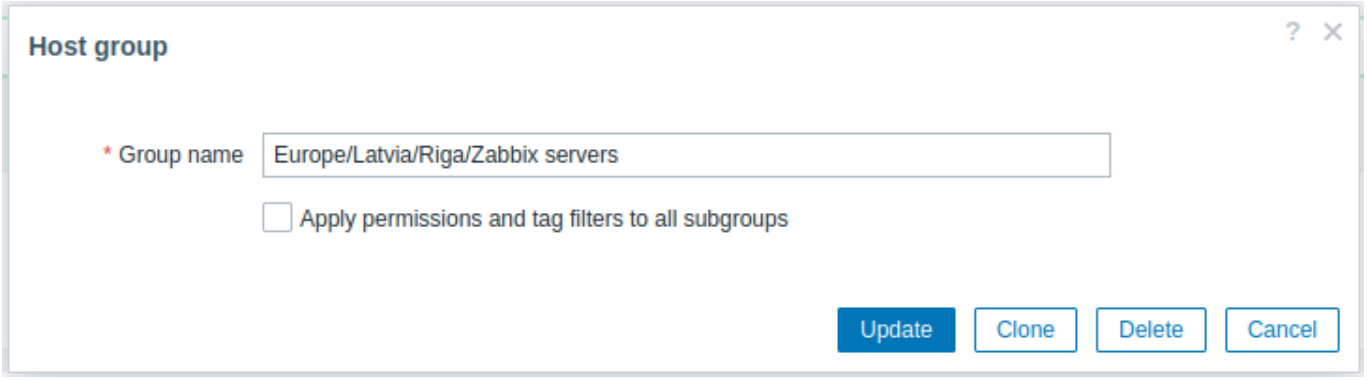
To create a host group in Zabbix frontend, do the following:

- Go to: *Configuration* → *Host groups*
- Click on *Create host group* in the upper right corner of the screen
- Enter the group name in the form



To create a nested host group, use the '/' forward slash separator, for example `Europe/Latvia/Riga/Zabbix servers`. You can create this group even if none of the three parent host groups (`Europe/Latvia/Riga/`) exist. In this case creating these parent host groups is up to the user; they will not be created automatically.
Leading and trailing slashes, several slashes in a row are not allowed. Escaping of '/' is not supported.

Once the group is created, you can click on the group name in the list to edit group name, clone the group or set additional option:



Apply permissions and tag filters to all subgroups - mark this checkbox and click on *Update* to apply the same level of permissions/tag filters to all nested host groups. For user groups that may have had differing permissions assigned to nested host groups, the permission level of the parent host group will be enforced on the nested groups. This is a one-time option that is not saved in the database.

Permissions to nested host groups

- When creating a child host group to an existing parent host group, user group permissions to the child are inherited from the parent (for example, when creating `Riga/Zabbix servers` if `Riga` already exists)
- When creating a parent host group to an existing child host group, no permissions to the parent are set (for example, when creating `Riga` if `Riga/Zabbix servers` already exists)

2 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 *Host 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 *Host 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*.

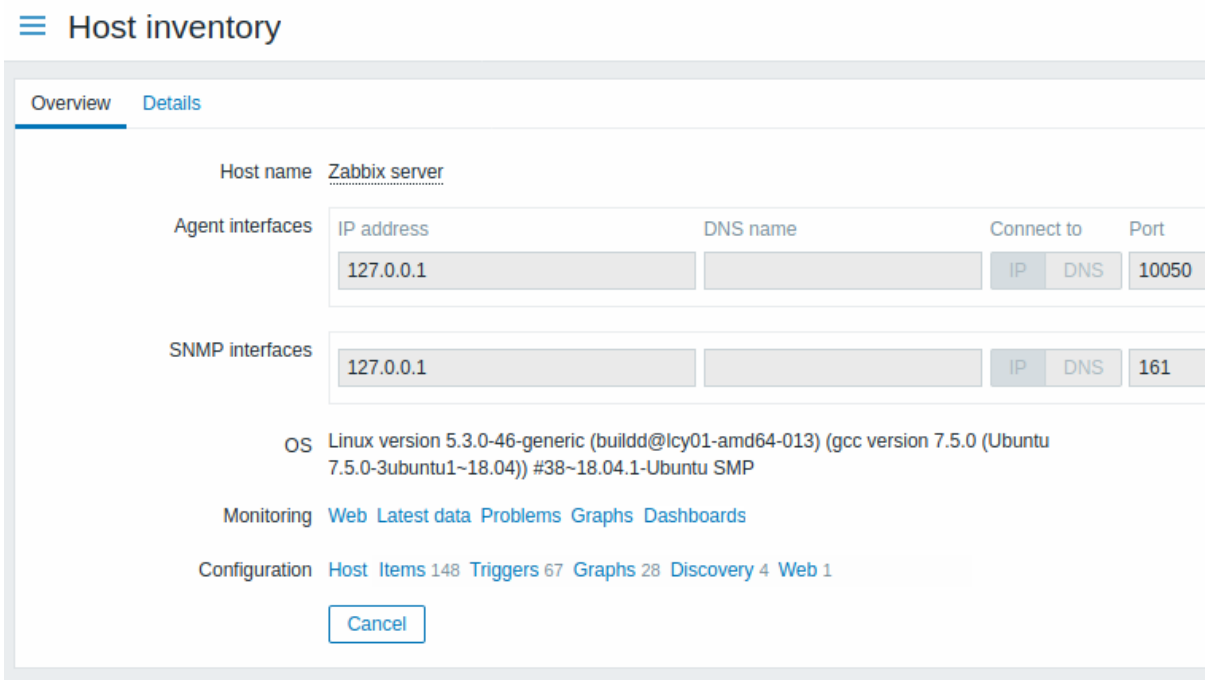
For hosts added by network discovery or autoregistration actions, it is possible to define a *Set host inventory mode* operation selecting manual or automatic mode. This operation overrides the *Default host inventory mode* setting.

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*, *Applications*, *Items*, *Triggers*, *Graphs*, *Discovery*, *Web*. The amount of configured entities is listed in parenthesis 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.

3 Mass update

Overview

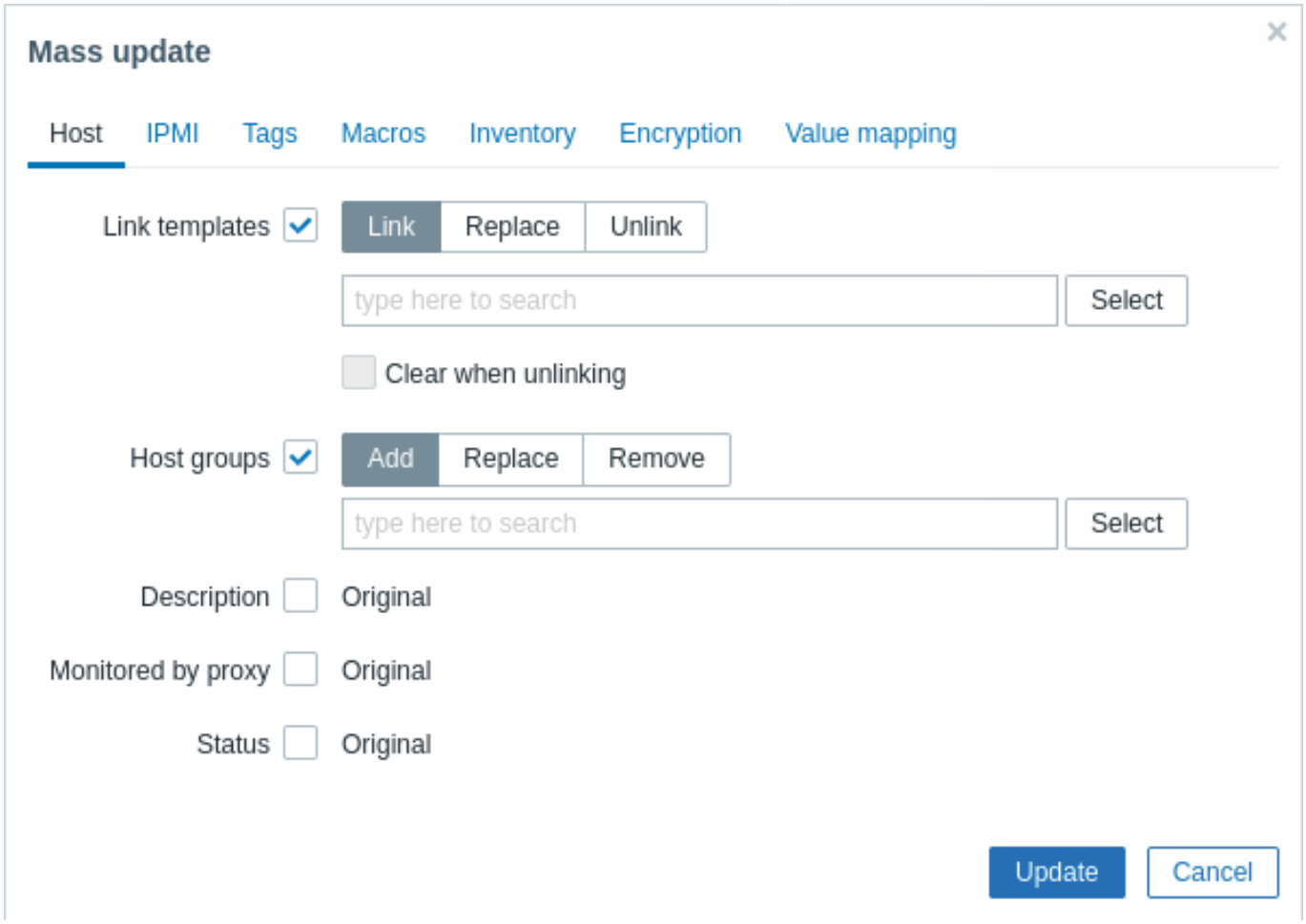
Sometimes you may want to change some attribute for a number of hosts at once. Instead of opening each individual host for editing, you may use the mass update function for that.

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

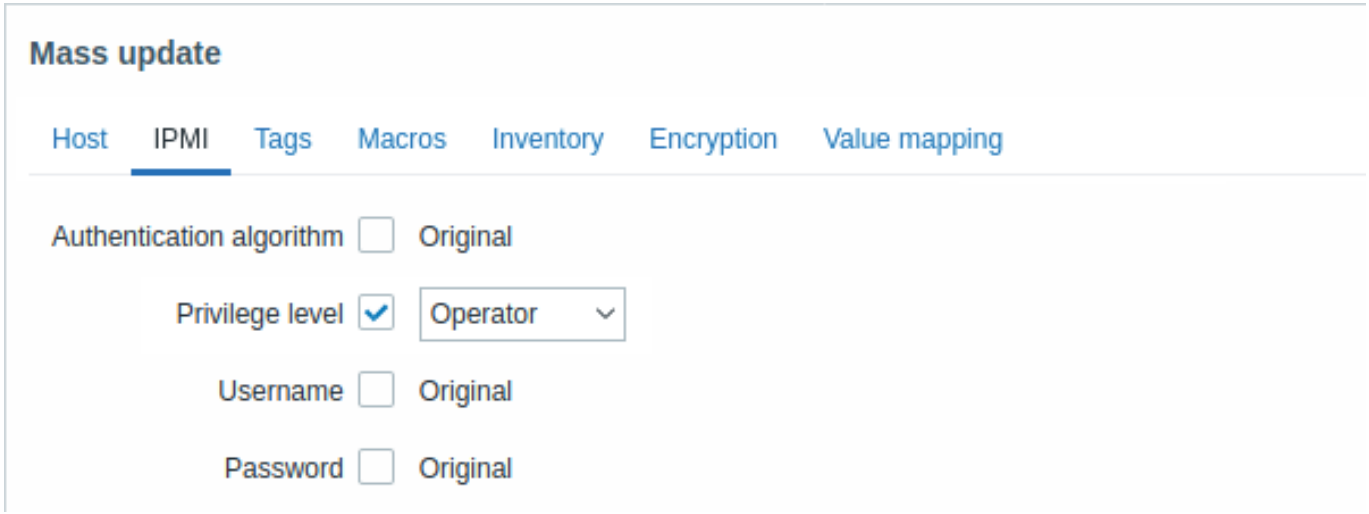
To specify the templates to link/unlink start typing the template name in the auto-complete field until a dropdown appears offering the matching templates. Just scroll down to select the required template.

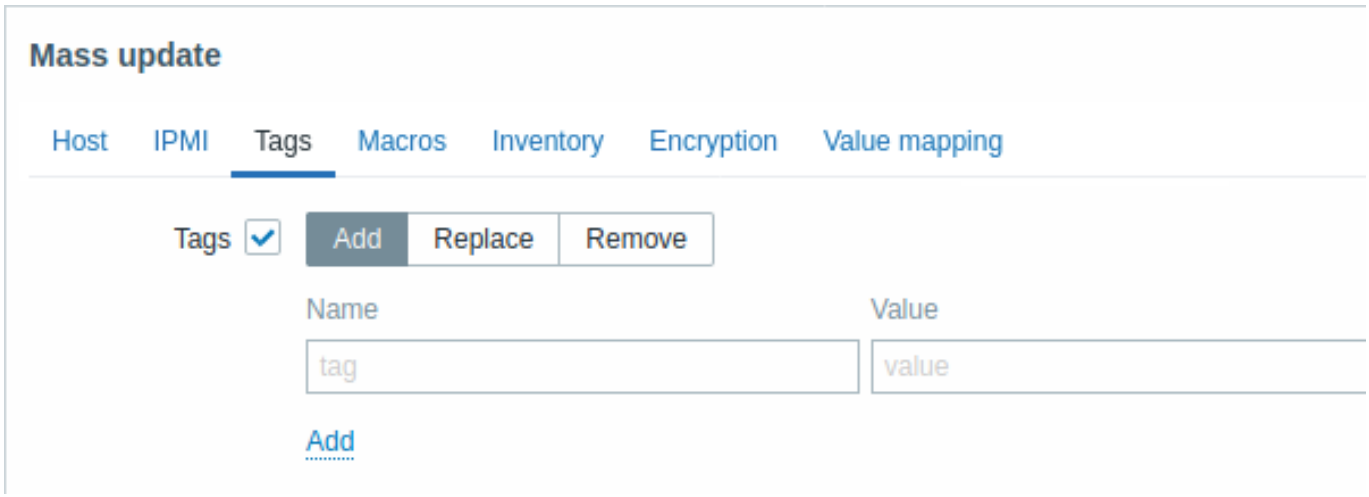
The *Clear when unlinking* option will allow to not only unlink any previously linked templates, but also remove all elements inherited from them (items, triggers, etc.).

The following options are available when selecting the respective button for **host group** update:

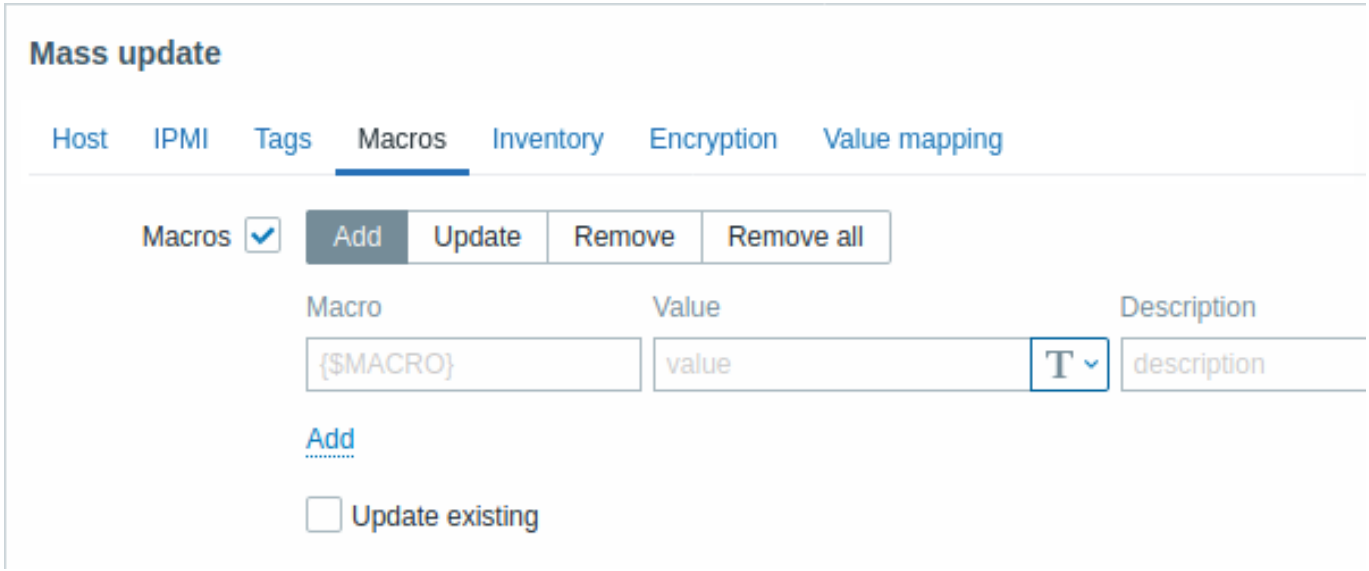
- *Add* - allows to specify additional host groups from the existing ones or enter completely new host groups for the hosts
- *Replace* - will remove the host from any existing host groups and replace them with the one(s) specified in this field (existing or new host groups)
- *Remove* - will remove 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.

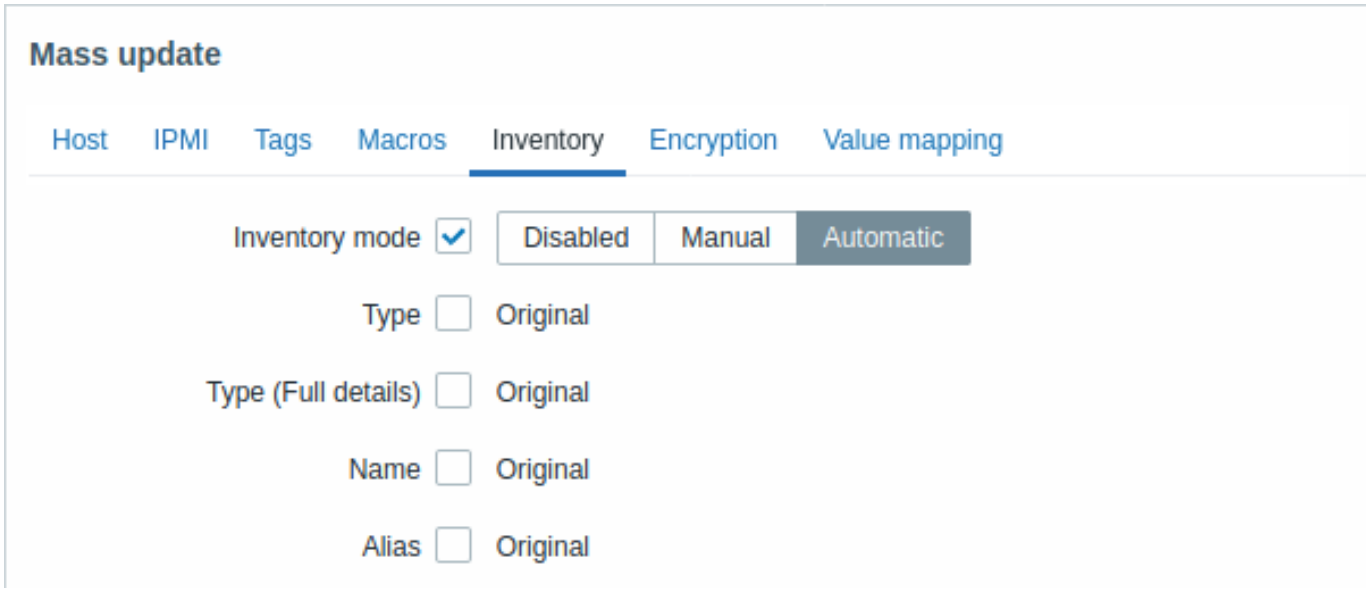


The following options are available when selecting the respective button for macros update:

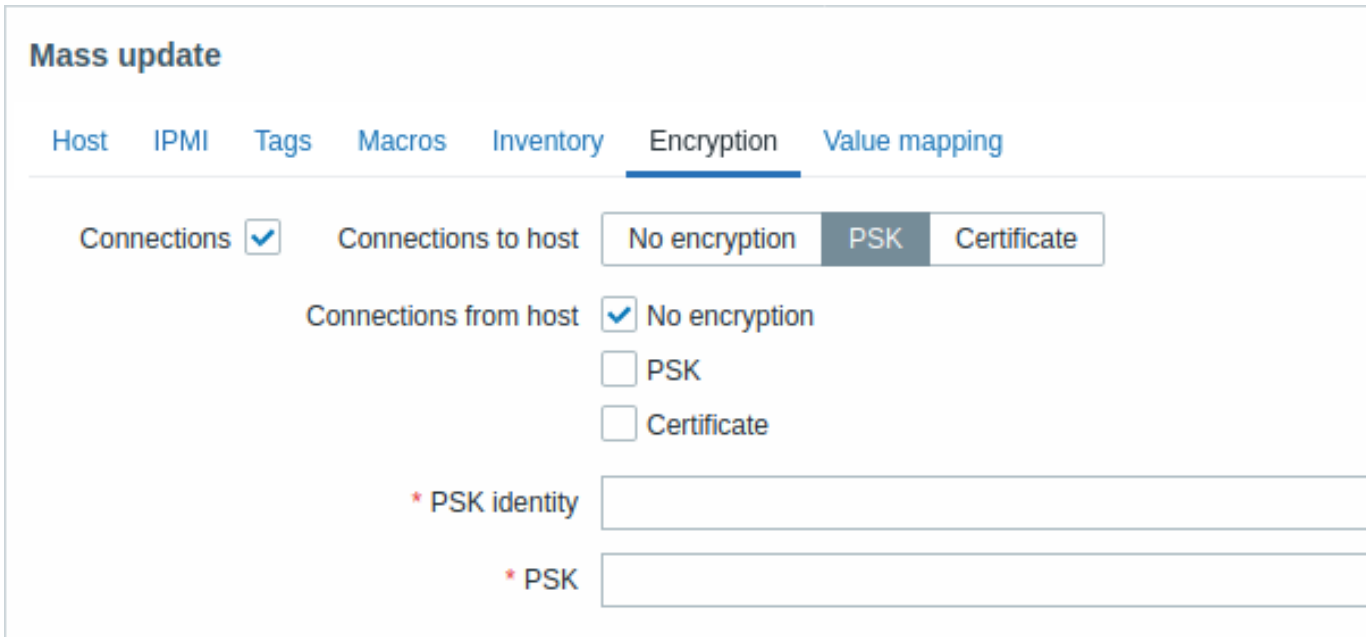
- *Add* - allows to specify additional user macros for the hosts. If *Update existing* checkbox is checked, value, type and description for the specified macro name will be updated. If unchecked, if a macro with that name already exist on the host(s), it will not be updated.
- *Update* - will replace 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

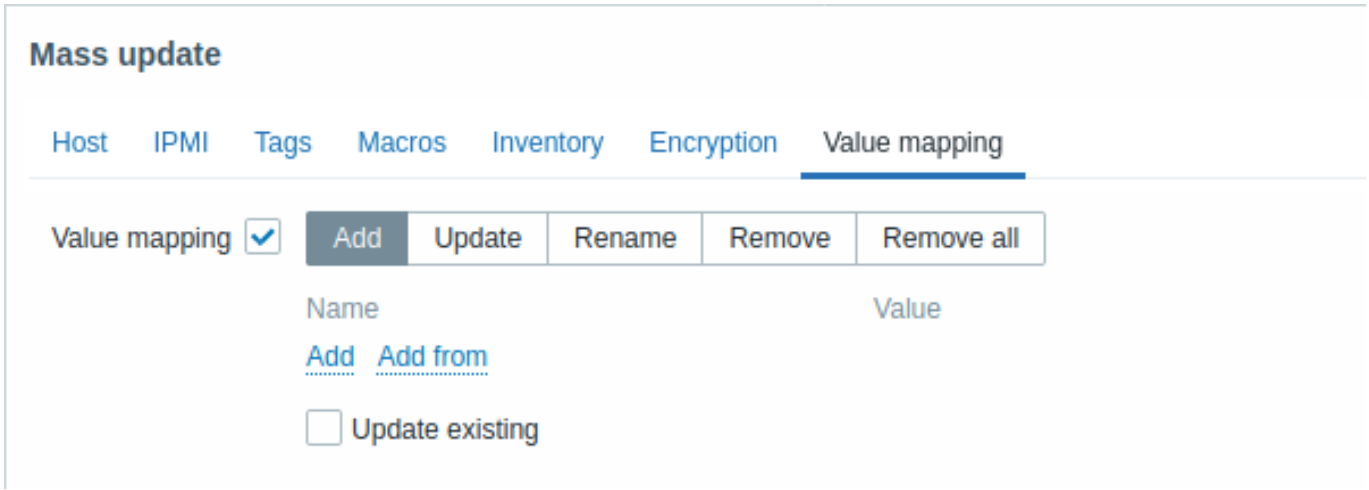
will be removed. If unchecked, only macros specified in the list will be removed.

- *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

Items are the ones that gather data from a host.

Once you have configured a host, you need to add some monitoring items to start getting actual data.

An item is an individual metric. One way of quickly adding many items is to attach one of the predefined templates to a host. For optimized system performance though, you may need to fine-tune the templates to have only as many items and as frequent monitoring as is really necessary.

In an individual item you specify what sort of data will be gathered from the host.

For that purpose you use the item key. Thus an item with the key name **system.cpu.load** will gather data of the processor load, while an item with the key name **net.if.in** will gather incoming traffic information.

To specify further parameters with the key, you include those in square brackets after the key name. Thus, system.cpu.load**[avg5]** will return processor load average for the last 5 minutes, while net.if.in**[eth0]** will show incoming traffic in the interface eth0.

> **Note:**
> For all supported item types and item keys, see individual sections of item types.

Proceed to creating and configuring an item.

1 Creating an item

Overview

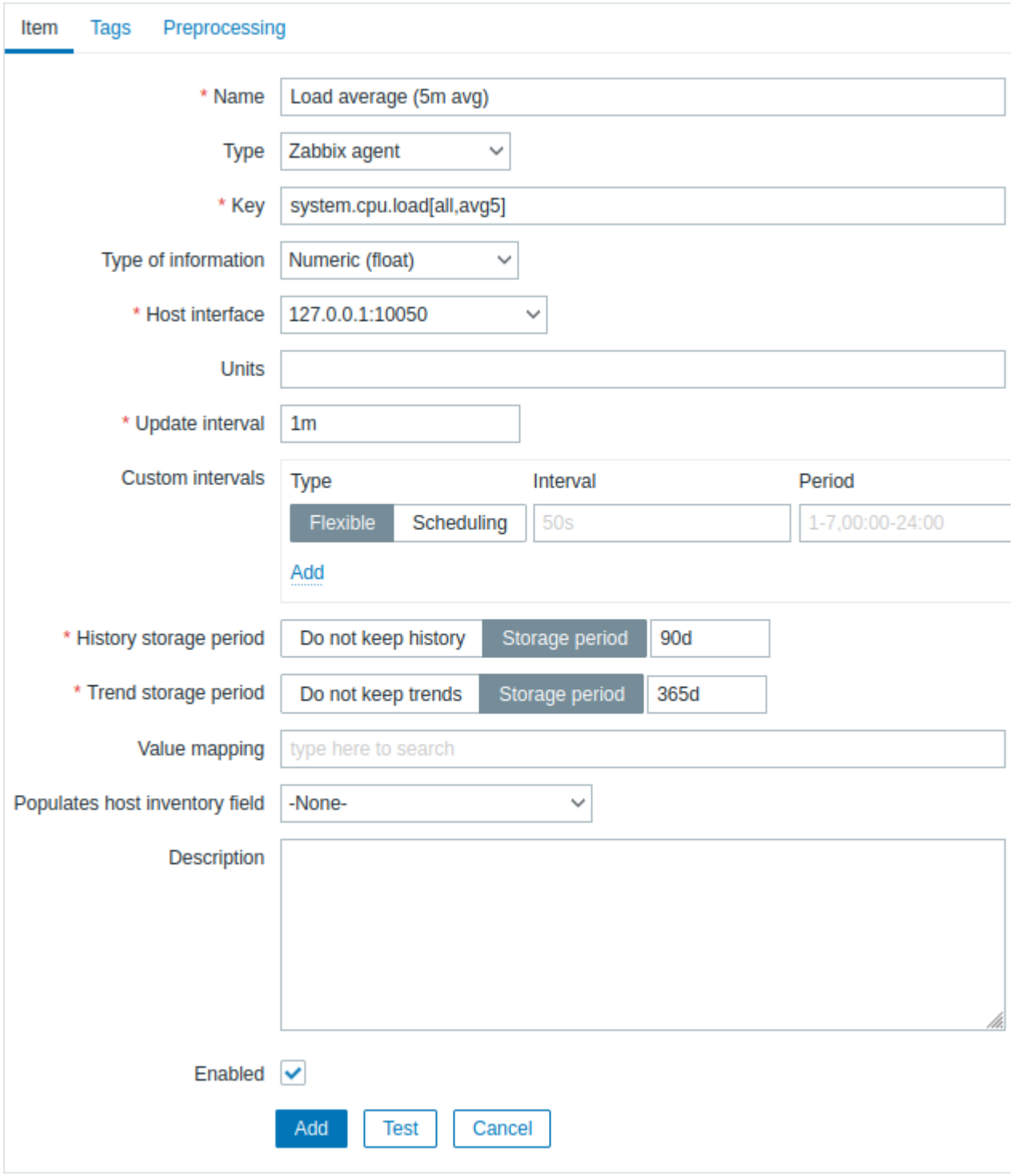
To create an item in Zabbix frontend, do the following:

- Go to: *Configuration* → *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.

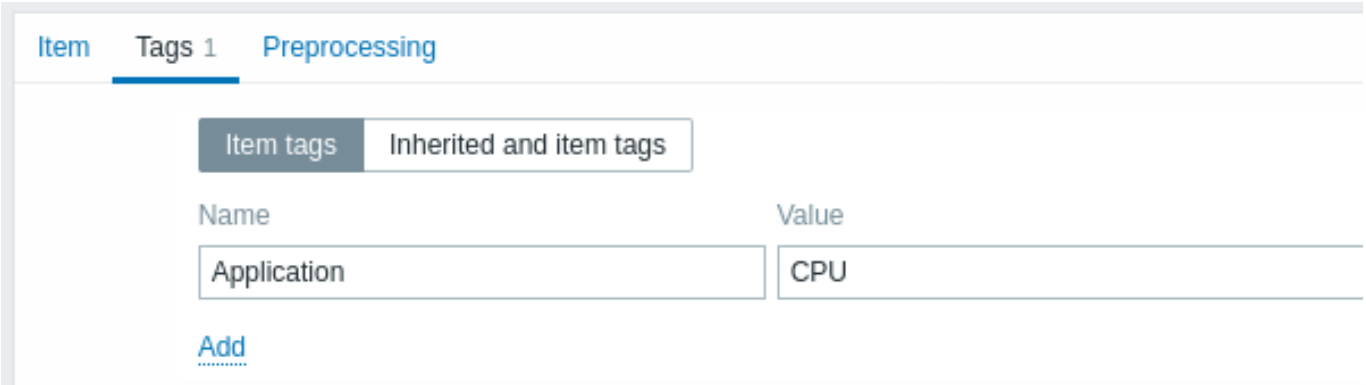
Parameter	Description
Type of information	Type of data as stored in the database after performing conversions, if any. **Numeric (unsigned)** - 64bit unsigned integer **Numeric (float)** - 64bit floating point number This type will allow precision of approximately 15 digits and range from approximately -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 post processing 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*: Not available for Zabbix agent active items.

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* → *General* → *Housekeeper*. 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* → *General* → *Housekeeper*. 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 not be parsed. 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 line. Log time format for this line would be ”ppppp:yyyyMMdd:hhmmss”. Note that ”p” and ”:” chars are just placeholders and can be anything but ”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.
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.

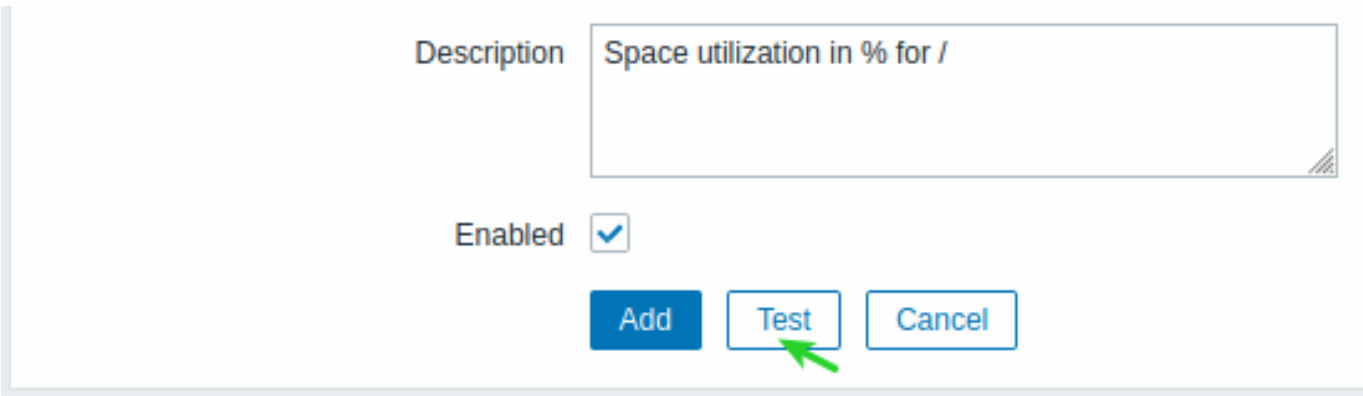
It is possible to test an item and, if configured correctly, get a real value in return. Testing can occur even before an item is saved.

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

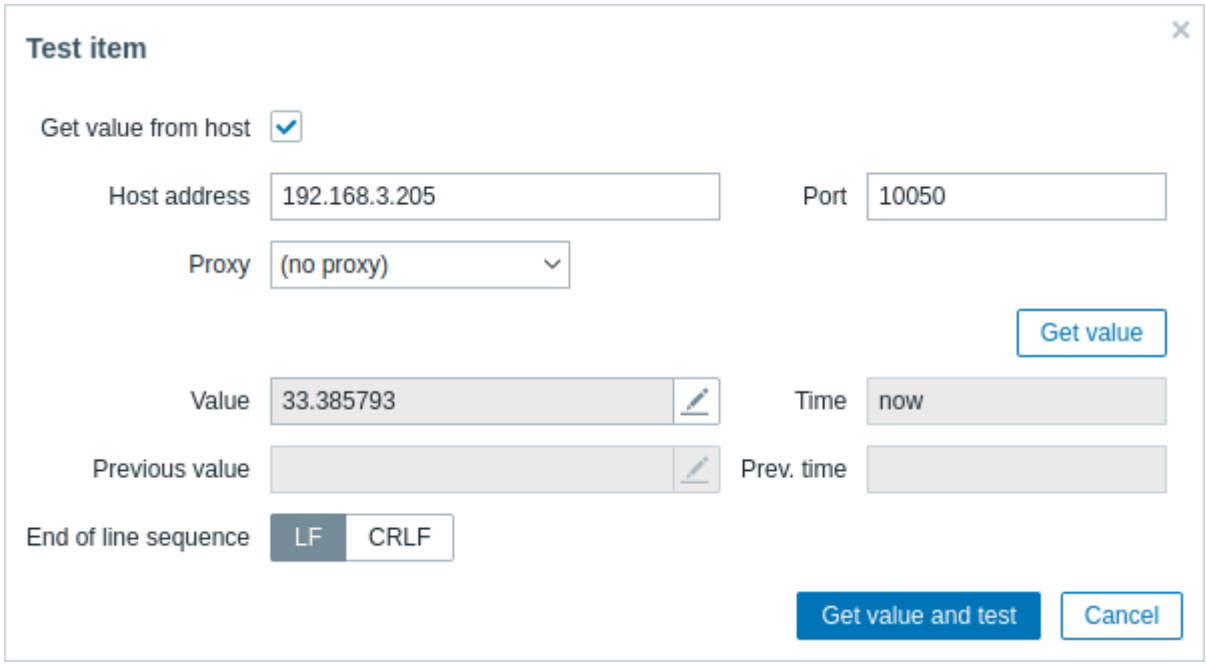
To test an item, click on the *Test* button at the bottom of the item configuration form. Note that the *Test* button will be disabled for items that cannot be tested (like active checks, excluded simple checks).



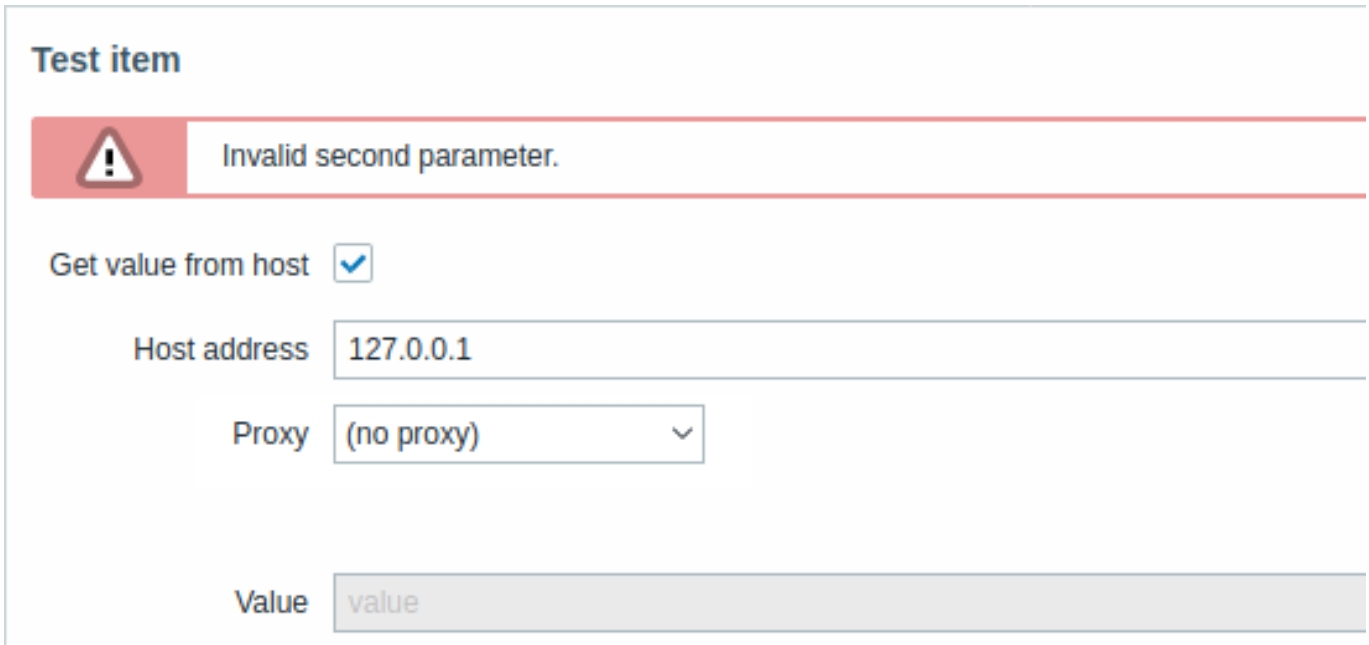
The item testing form has fields for the required host parameters (host address, port, proxy name/no proxy) and item-specific details (such as SNMPv2 community or SNMPv3 security credentials). These fields are context aware:

- 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.

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

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, !%.

Custom script limit

Available custom script length depends on the database used:

Database	Limit in characters	Limit in bytes
MySQL	65535	65535
Oracle Database	2048	4000
PostgreSQL	65535	not limited

Database	Limit in characters	Limit in bytes
SQLite (only Zabbix proxy)	65535	not limited

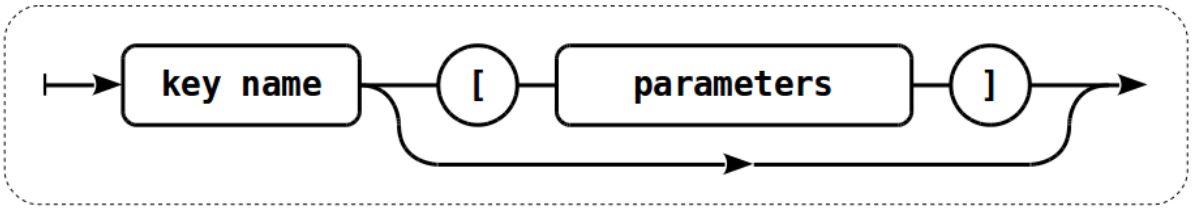
Unsupported items

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.

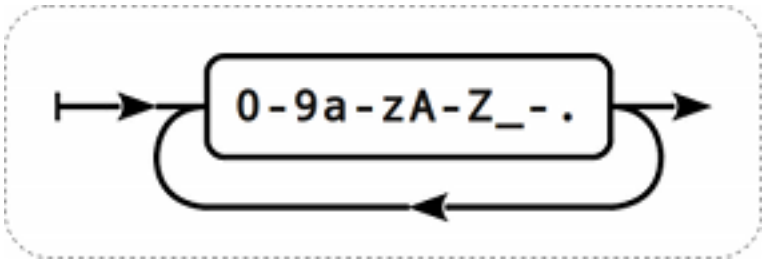
Key name

The key name itself has a limited range of allowed characters, which just follow each other. Allowed characters are:

`0-9a-zA-Z_-.`

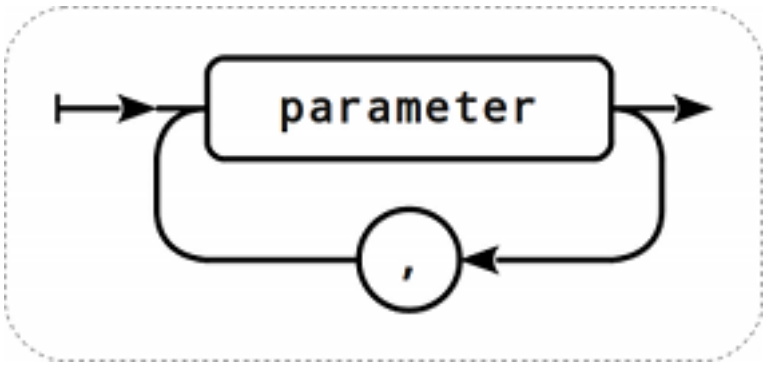
Which means:

- all numbers;
- all lowercase letters;
- all uppercase letters;
- underscore;
- dash;
- dot.

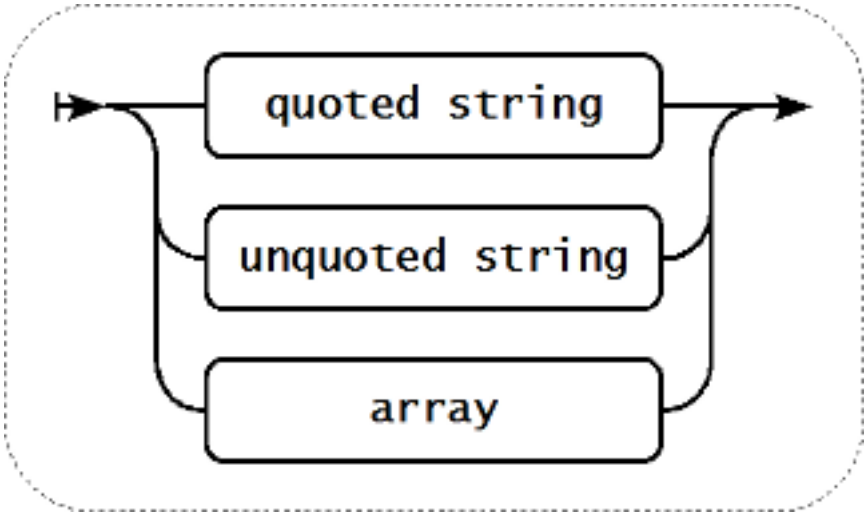


Key parameters

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.



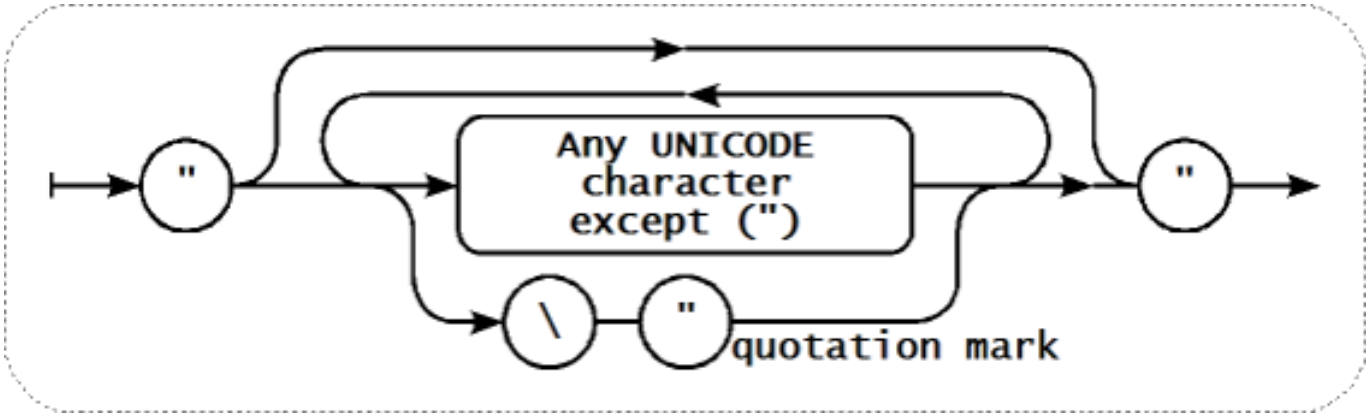
The parameter can also be left empty, thus using the default value. In that case, the appropriate number of commas must be added if any further parameters are specified. For example, item key **icmpping[„200„500]** would specify that the interval between individual pings is 200 milliseconds, timeout - 500 milliseconds, and all other parameters are left at their defaults.

Parameter - quoted string

If the key parameter is a quoted string, any Unicode character is allowed.

If the key parameter string contains comma, this parameter has to be quoted.

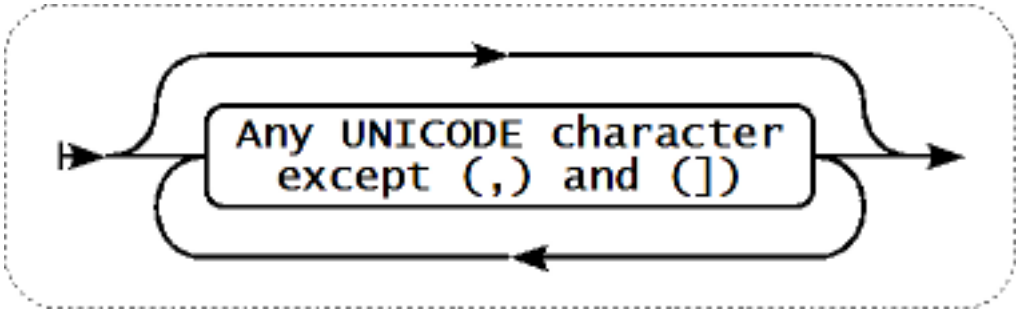
If the key parameter string contains quotation mark, this parameter has to be quoted and each quotation mark which is a part of the parameter string has to be escaped with a backsplash (\) character.



Warning:
To quote item key parameters, use double quotes only. Single quotes are not supported.

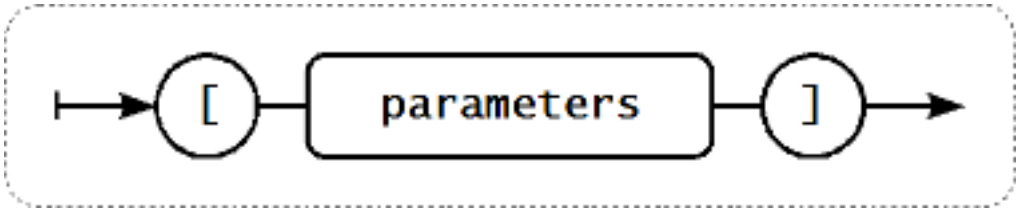
Parameter - unquoted string

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.

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
h9m/30 h10 (*add this as another row in frontend*)	every day at 9:00, 9:30, 10:00

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.

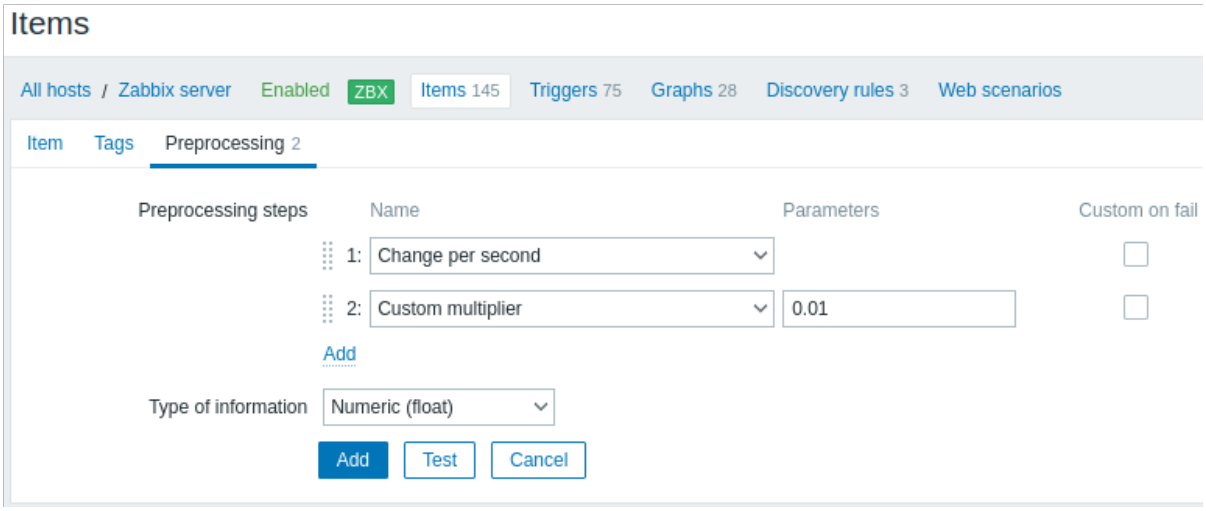
Transformations are executed in the order in which they are defined. Preprocessing is done by Zabbix server or proxy (if items are monitored by proxy).

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 fails, unless custom error handling has been specified using a *Custom on fail* option for supported transformations.

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 item value preprocessing parameters, including JavaScript code.

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, the item will not become unsupported in case of failed preprocessing step and 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	
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, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.

JSON Path	Extract value or fragment from JSON data using JSONPath functionality. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and 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.
XML to JSON	Convert data in XML format to JSON. For more information, see: Serialization rules. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error-handling options: either to discard the value, set a specified value or set a specified error message.
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, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.
Change	

Type	
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, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.
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 extremely useful to get 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, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.
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, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.
Octal to decimal	Convert the value from octal format to decimal. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.
Hexadecimal to decimal	Convert the value from hexadecimal format to decimal. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and 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 a pencil icon. Note that available JavaScript length depends on the database used. For more information, see: Javascript preprocessing.
Validation	

Type	
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, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.
Matches regular expression	Specify a regular expression that a value must match. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and it is possible to specify custom error handling options: either to discard the value, set a specified value, or set a specified error message.
Does not match regular expression	Specify a regular expression that a value must not match. If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of failed preprocessing step and 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 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, the item will not become unsupported in case of failed preprocessing step and 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 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, the item will not become unsupported in case of failed preprocessing step and 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 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, the item will not become unsupported in case of failed preprocessing step and 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 not supported value	Check if there was an error in retrieving item value. Normally that would lead to the item turning 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 (in this case the item will stay supported and the value can be used in triggers) or set a specified error message. Note that for this preprocessing step, the *Custom on fail* checkbox is grayed out and always marked. 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 it is possible for items monitored by Zabbix proxy that very small value differences (less than 0.000001) are correctly not discarded by proxy, but are stored in the history as the same value if the Zabbix server database has not been 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 it is possible for items monitored by Zabbix proxy that very small value differences (less than 0.000001) are correctly not discarded by proxy, but are stored in the history as the same value if the Zabbix server database has not been 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:

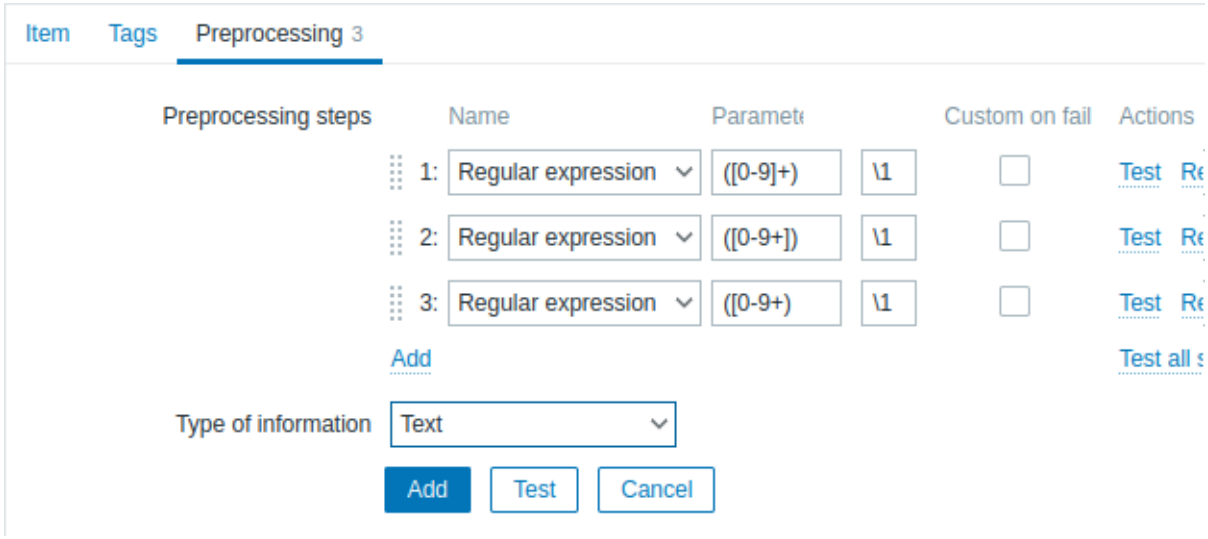
- for *Simple change*, *Change per second* steps - the next value will be ignored because there is no previous value to calculated change from;
- for *Discard unchanged*. *Discard unchanged with heartbeat* steps - the next value will never be discarded, even if it should have been because of discarding 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.

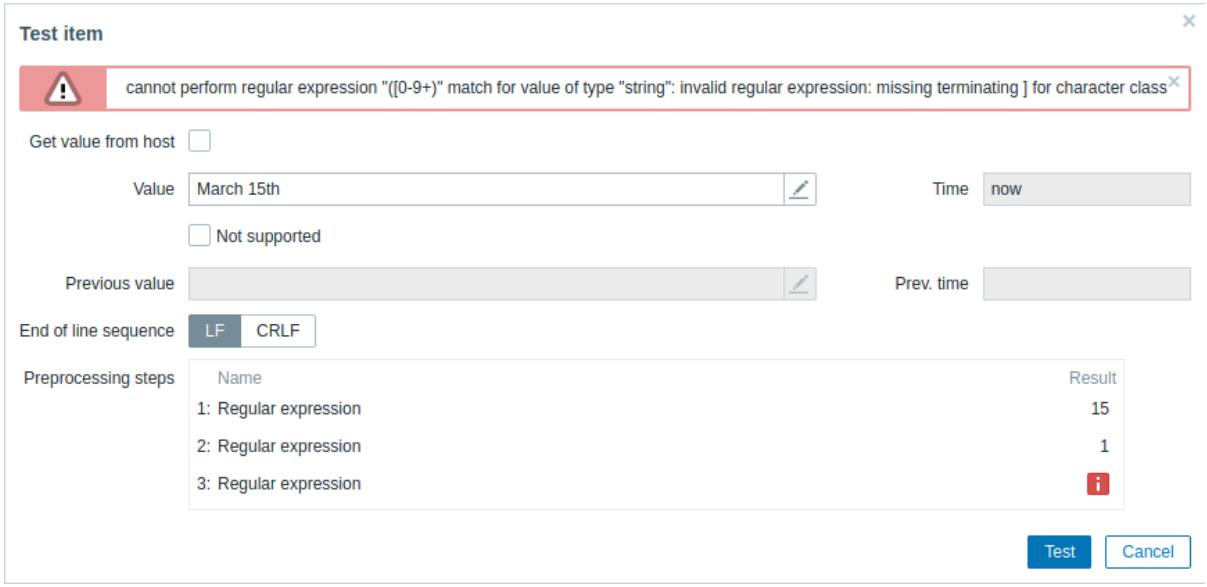


It is possible to test:

- 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. 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)`.

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.

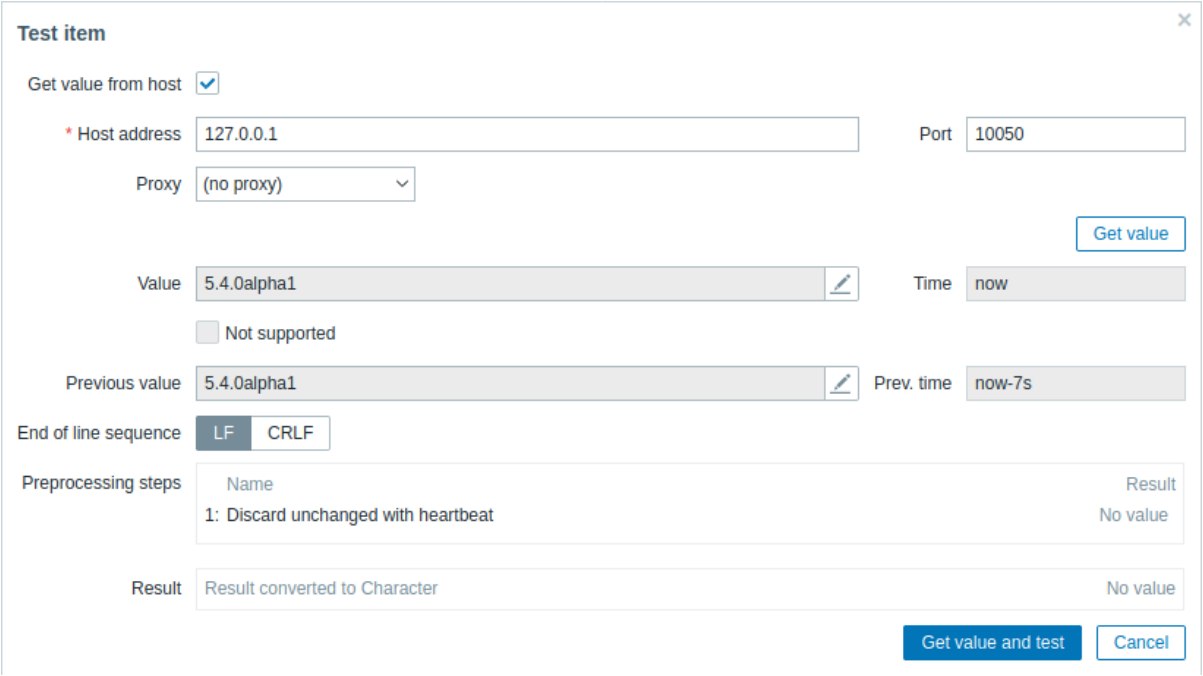
The testing is done by Zabbix server. The frontend sends a corresponding request to the server and waits for the result. The request contains the input value and preprocessing steps (with expanded user macros). For *Change* and *Throttling* steps, an optional previous value and time can be specified. 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

Using a regular expression preprocessing to filter unnecessary events of the VMWare event log.

1. On a working VMWare Hypervisor host check that the event log item `vmware.eventlog[<url>,<mode>]` is present and working properly. Note that the event log item could already be present on the hypervisor if the *Template VM 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.

In the "Preprocessing" tab of the dependent item select the "Matches regular expression" validation option and fill pattern, for example:

```
".* logged in .*" - filters all logging events in the event log
"\bUser\s+\K\S+" -  filter only lines with usernames from the event log
```

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 to discard unmatched value, for example.

Another approach that allows using matching groups and output control is to select "Regular expression" option in the "Preprocessing" tab and fill parameters, for example:

```
pattern: ".*logged in.*", output: "\0" - filters all logging events in the event log
pattern "User (.*?)(?=\ )", output: "\1" - filter only usernames from the event log
```

2 Preprocessing details

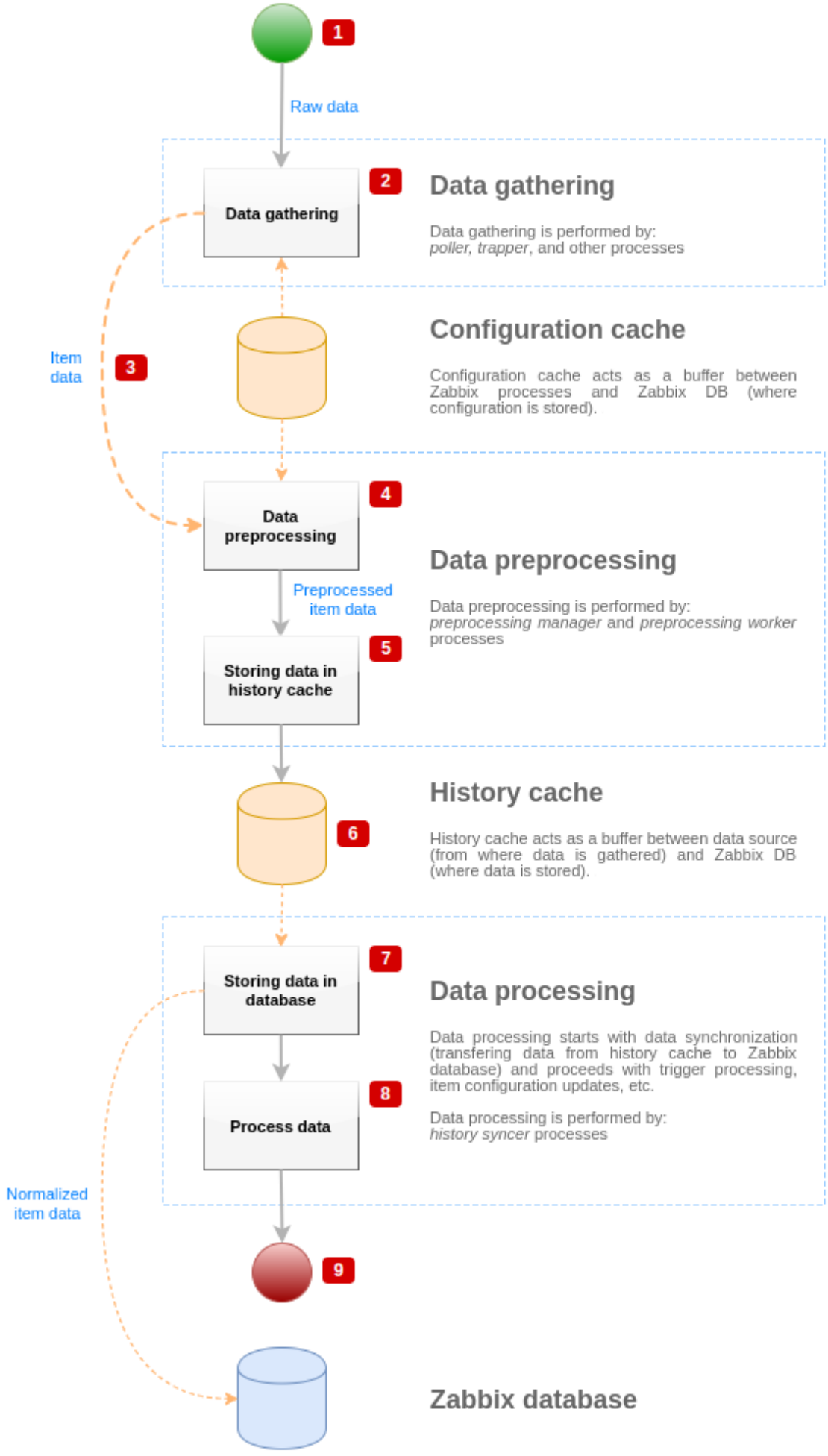
Overview

This section provides item value preprocessing details. Item value preprocessing allows to define and execute transformation rules for the received item values.

Preprocessing is managed by a preprocessing manager process, which was added in Zabbix 3.4, 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. Local data cache of preprocessing manager is not shown either because it doesn't affect data flow directly. The aim of this diagram is to show processes involved in item value processing and the way they interact.

- Data gathering starts with raw data from a data source. At this point, 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 and 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). Raw data is validated, item configuration is retrieved from configuration cache (data is enriched with the configuration data).
- Socket-based IPC mechanism is used to pass data from data gatherers to preprocessing manager. At this point data gatherer continue to gather data without waiting for the response from preprocessing manager.
- Data preprocessing is performed. This includes execution of preprocessing steps and dependent item processing.

Note:
Item can change its state to NOT SUPPORTED while preprocessing is performed if any of preprocessing steps fail.

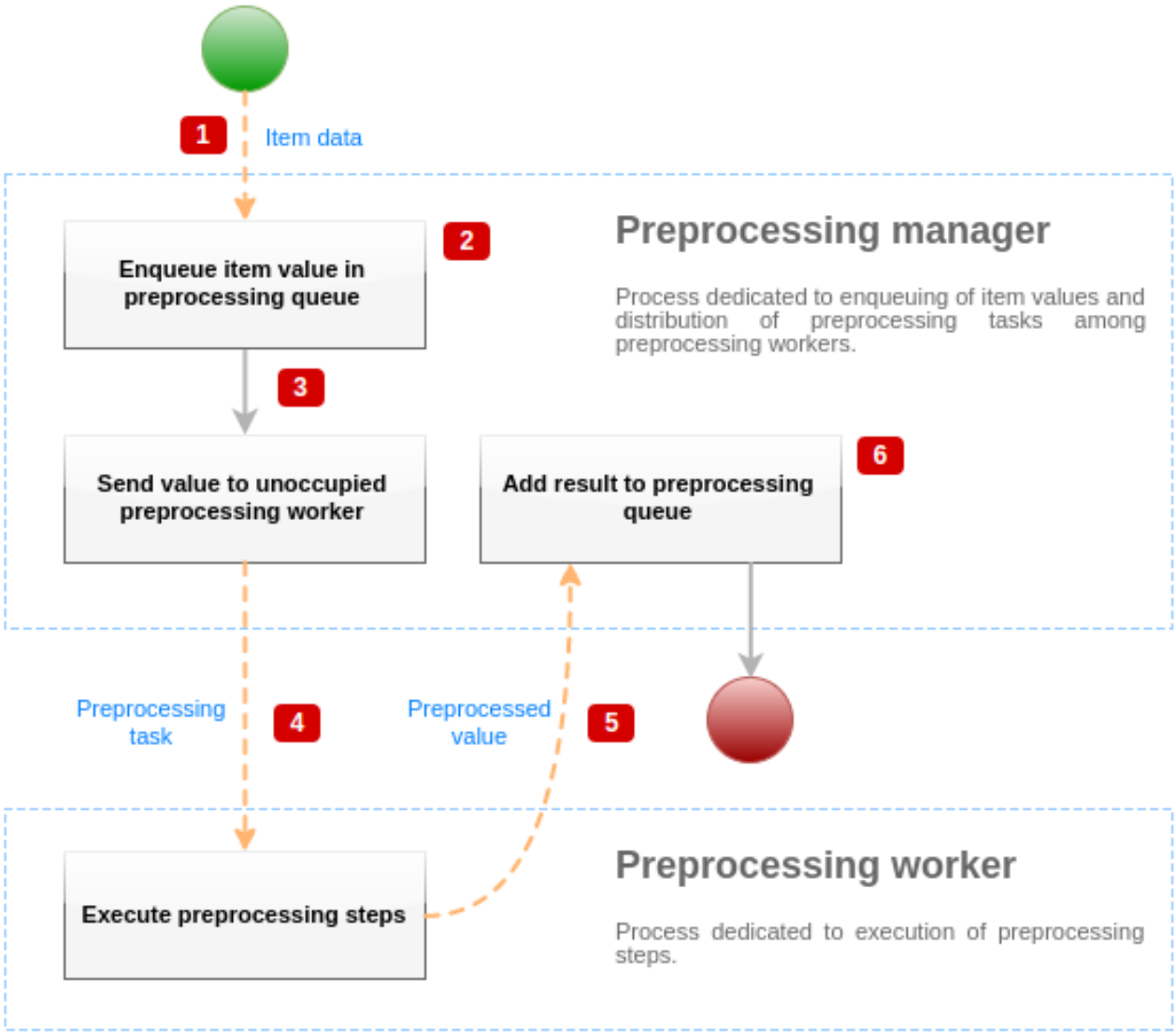
- History data from local data cache of preprocessing manager is being flushed into history cache.
- At this point data flow stops until the next synchronization of history cache (when history syncer process performs data synchronization).
- Synchronization process starts with data normalization storing data in Zabbix database. Data normalization performs conversions to 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). Data is being sent to Zabbix database after normalization is done.

Note:
Item can change its state to NOT SUPPORTED if data normalization fails (for example, when textual value cannot be converted to number).

- Gathered data is being processed - triggers are checked, 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

To visualize the data preprocessing process, we can use the following simplified diagram:



The diagram above shows only processes, objects and main actions related to item value preprocessing in a **simplified** form. The diagram does not show conditional direction changes, error handling or loops. Only one preprocessing worker is shown on this diagram (multiple preprocessing workers can be used in real-life scenarios), only one item value is being processed and we assume that this item requires to execute at least one preprocessing step. The aim of this diagram is to show the idea behind item value preprocessing pipeline.

- Item data and item value is passed to preprocessing manager using socket-based IPC mechanism.
- Item is placed in the preprocessing queue.

Note:
Item can be placed at the end or at the beginning of the preprocessing queue. Zabbix internal items are always placed at the beginning of preprocessing queue, while other item types are enqueued at the end.

- At this point data flow stops until there is at least one unoccupied (that is not executing any tasks) preprocessing worker.
- When preprocessing worker is available, preprocessing task is being sent to it.
- After preprocessing is done (both failed and successful execution of preprocessing steps), preprocessed value is being passed back to preprocessing manager.
- Preprocessing manager converts result to desired format (defined by item value type) and places result in preprocessing queue. If there are dependent items for current item, then dependent items are added to preprocessing queue as well. Dependent items are enqueued in preprocessing queue right after the master item, but only for master items with value set and not in NOT SUPPORTED state.

Value processing pipeline

Item value processing is executed in multiple steps (or phases) by multiple processes. This can cause:

- 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, dependent item receives a value, while master item changes its state to NOT SUPPORTED.

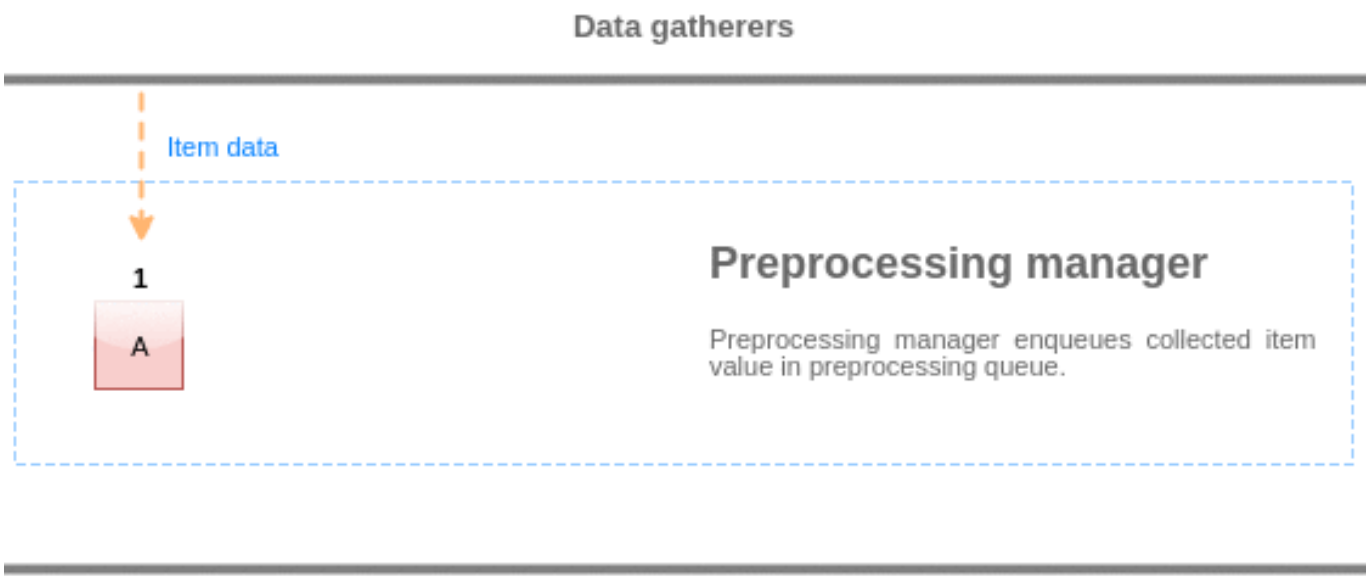
- Dependent item receives value that is not present in master item history. The use case is very similar to the previous one, except for the master item type. For example, if `CHAR` type is used for master item, then master item value will be truncated at the history synchronization phase, while dependent items will receive their value from the initial (not truncated) value of master item.

Preprocessing queue

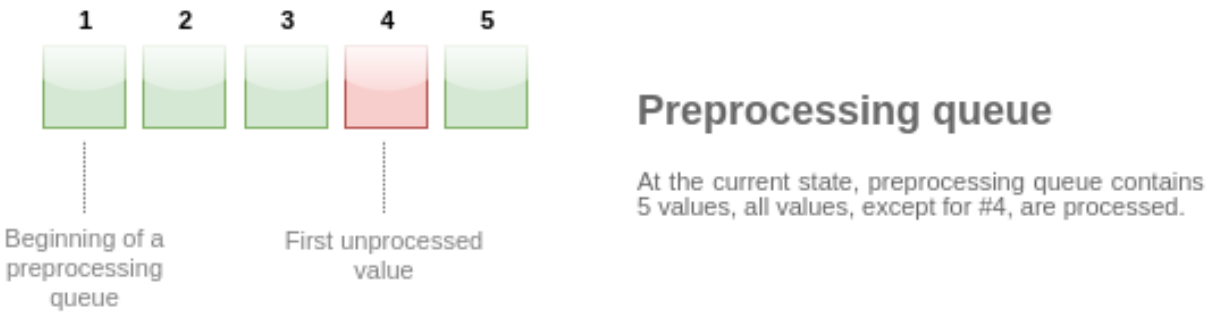
Preprocessing queue is a FIFO data structure that stores values preserving the order in which values are revieved by preprocessing manager. There are multiple exceptions to FIFO logic:

- Internal items are enqueued at the beginning of the queue
- Dependent items are always enqueued after the master item

To visualize the logic of preprocessing queue, we can use the following diagram:



Values from the preprocessing queue are flushed from the beginning of the queue to the first unprocessed value. So, for example, preprocessing manager will flush values 1, 2 and 3, but will not flush value 5 as value 4 is not processed yet:



Only two values will be left in queue (4 and 5) after flushing, values are added into local data cache of preprocessing manager and then values are transferred from local cache into history cache. Preprocessing manager can flush values from local data cache in

single item mode or in bulk mode (used for dependent items and values received in bulk).

Preprocessing workers

Zabbix server configuration file allows users to set count of preprocessing worker processes. StartPreprocessors configuration parameter should be used to set number of pre-forked instances of preprocessing workers. 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), count of data gathering processes, average step count for item preprocessing, etc.

But assuming that there is no heavy preprocessing operations like parsing of large XML / JSON chunks, number of preprocessing workers can match total number of data gatherers. This way, there will mostly (except for the cases when data from 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.

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.
`$[?(@.name == 'Object')].price.first()`	Return the price field of the first object with name 'Object'.
`$[?(@.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.

Segment	Description
`[?(<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\'"
`'<text>'`		"value: '1'"
`<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).
!=	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.

Function	Description	Input	Output
`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.

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”]	

JSONPath	Type	Result	Comments
`$.tags[:3]`	indefinite	[”a”, ”b”, ”c”]	
`$.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	[”Sayings of the Century”, ”Moby Dick”]	
`$.books[1]['author', "title"]`	indefinite	[”Evelyn Waugh”, ”Sword of Honour”]	
`$..id`	indefinite	[1, 2, 3, 4]	
`$.services..price`	indefinite	[5, 154.99, 46, 24.5, 99.49]	
`$.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”]	
`$.books[?(@.author > "Herman Melville")].title`	indefinite	[”Sayings of the Century”, ”The Lord of the Rings”]	
`$.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”]	

JSONPath	Type	Result	Comments

`$..[?(@.id)]`	indefinite	[
		{
		”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
		}
		]	

`$.services..[?(@.price > 50)].description`	indefinite	’[”Printing and assembling book in A5 format”, ”Rebinding torn book”]	
`$..id.length()`	definite	4	
`$.books[?(@.id == 2)].title.first()`	indefinite	Sword of Honour	
`$..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	
`$..price.max()`	definite	154.99	
`$.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[?(@.defintime=="true")].servicegroup`	definite	[1001, 1002]	Text constants must be used in boolean value comparisons.
`$.services[?(@.defintime=="false")].servicegroup`	definite	[1002]	Text constants must be used in boolean value comparisons.
`$.services[?(@.servicegroup=="1002")]~.first()`	definite		

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 64 megabyte heap limit is enforced.

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;
```

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(name, 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://tsupport.zabbix.lan/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(string) - encodes string to base64 string
- atob(base64_string) - decodes base64 string

```
try {
    b64 = btoa("utf8 string");
    utf8 = atob(b64);
}
catch (error) {
    return {'error.name' : error.name, 'error.message' : error.message}
}
```

- md5(string) - calculates the MD5 hash of a string
- sha256(string) - calculates the SHA256 hash of a string
- hmac('<hash type>',key,string) - returns HMAC hash as hex formatted string; MD5 and SHA256 hash types are supported; key and string parameters support binary data. Examples:
  - `hmac('md5',key,string)`
  - `hmac('sha256',key,string)`

5 CSV to JSON preprocessing

Overview

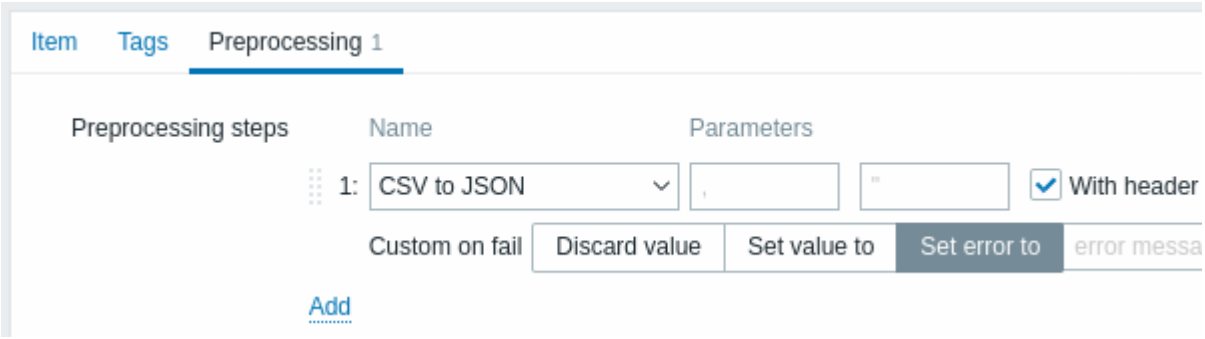
In this preprocessing step it is possible to convert CSV file data into JSON format. It's supported in:

- 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 platforms

Except where specified differently in the item comments, 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.

Supported item keys

The item keys that you can use with Zabbix agent are listed below. The items are grouped in tables by item family.

Parameters without angle brackets are mandatory. Parameters marked with angle brackets **< >** are optional.

Kernel data

Item key			
Description	Return value	Parameters	Comments
kernel.maxfiles			
Maximum number of opened files supported by OS.	Integer		Supported platforms: Linux, FreeBSD, MacOS X, OpenBSD, NetBSD.
kernel.maxproc			
Maximum number of processes supported by OS.	Integer		Supported platforms: Linux 2.6 and later, FreeBSD, Solaris, MacOS X, OpenBSD, NetBSD.
kernel.openfiles			
Return the number of currently open file descriptors.	Integer		Supported platforms: Linux. The item may work on other UNIX-like platforms. This item is supported since Zabbix 6.0.

Log data

See additional information on log monitoring.

Item key			
Description	Return value	Parameters	Comments
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent_dir>]			

Item key			
Monitoring of a log file.	Log	**file** - full path and name of log file **regexp** - regular expression describing the required pattern **encoding** - code page identifier **maxlines** - 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** (since version 2.0)- possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **output** (since version 2.2) - 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** (since version 3.2) - 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** (since version 4.4.7) - 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** (since versions 5.0.18, 5.4.9, only in zabbix_agentd on Unix systems; not supported in Agent2) - absolute pathname of directory where to store persistent files. See also additional notes on persistent files.	See supported platforms. The item must be configured as an active check. If file is missing or permissions do not allow access, 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] *Using `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] → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send only '6080' to server. Because a numeric value is being sent, the "Type of information" for this item can be set to "Numeric (unsigned)" and the value can be used in graphs, triggers etc. *Using `output` parameter for rewriting 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 RECORDS: \3, ERRORS: \4, DURATION: \2"] → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send a modified record "2015-11-13 10:08:26 RECORDS: 6080, ERRORS: 0, DURATION: 6.08" to server.
log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent_dir>]			

Item key			
Count of matched lines in a monitored log file.	Integer	**file** - full path and name of log file **regexp** - regular expression describing the required pattern **encoding** - code page identifier **maxproclines** - maximum number of new lines per second the agent will analyze (cannot exceed 10000). Default value is 10*'MaxLinesPerSecond' in zabbix_agentd.conf. **mode** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **maxdelay** - 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** (since version 4.4.7) - 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** (since versions 5.0.18, 5.4.9, only in zabbix_agentd on Unix systems; not supported in Agent2) - absolute pathname of directory where to store persistent files. See also additional notes on persistent files.	See supported platforms. 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, item turns unsupported. Supported since Zabbix 3.2.0.

logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>,<persistent_dir>]

Item key			
Monitoring of a log file that is rotated.	Log	**file_regexp** - absolute path to file and the file name described by a regular expression. *Note* that only the file name is a regular expression **regexp** - regular expression describing the required content pattern **encoding** - code page identifier **maxlines** - 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** (since version 2.0) - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **output** (since version 2.2) - 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** (since version 3.2) - 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** (since version 4.0; *mtime-reread*, *mtime-noreread* options since 4.4.7) - 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** (since versions 5.0.18, 5.4.9, only in zabbix_agentd on Unix systems; not supported in Agent2) - absolute pathname of directory where to store persistent files. See also additional notes on persistent files.	See supported platforms. 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}$",,,100] → will match a file like "logfile1" (will not match ".logfile1") => logrt["/home/user/^logfile_.*_[0-9]{1,3}$","pattern_to_match","UTF-8",100] → will collect data from files such "logfile_abc_1" or "logfile__001". *Using `output` parameter for extracting a number from log record:* => logrt[/app1/^test.*log$,"task run [0-9.]+ sec, processed ([0-9]+) records, [0-9]+ errors",,,\1] → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send only '6080' to server. Because a numeric value is being sent, the "Type of information" for this item can be set to "Numeric (unsigned)" and the value can be used in graphs, triggers etc. *Using `output` parameter for rewriting log record before sending to server:* => logrt[/app1/^test.*log$,"([0-9 :-]+) task run ([0-9.]+) sec, processed ([0-9]+) records, ([0-9]+) errors",,,"\1 RECORDS: \3, ERRORS: \4, DURATION: \2"] → will match a log record "2015-11-13 10:08:26 task run 6.08 sec, processed 6080 records, 0 errors" and send a modified record "2015-11-13 10:08:26 RECORDS: 6080, ERRORS: 0, DURATION: 6.08" to server.
logrt.count[file_regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>,<persistent_dir>]			

Item key			
Count of matched lines in a monitored log file that is rotated.	Integer	**file_regexp** - absolute path to file and regular expression describing the file name pattern **regexp** - regular expression describing the required content pattern **encoding** - code page identifier **maxproclines** - maximum number of new lines per second the agent will analyze (cannot exceed 10000). Default value is 10*'MaxLinesPerSecond' in zabbix_agentd.conf. **mode** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items). **maxdelay** - 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** (since version 4.0; *mtime-reread*, *mtime-noreread* options since 4.4.7) - 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** (since versions 5.0.18, 5.4.9, only in zabbix_agentd on Unix systems; not supported in Agent2) - absolute pathname of directory where to store persistent files. See also additional notes on persistent files.	See supported platforms. 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. Supported since Zabbix 3.2.0.

Modbus data

Item key			
Description	Return value	Parameters	Comments
modbus.get[endpoint,<slave id>,<function>,<address>,<count>,<type>,<endianness>,<offset>]			

Item key			
Reads Modbus data.	JSON object	**endpoint** - endpoint defined as `protocol://connection_string` **slave id** - slave ID **function** - Modbus function **address** - address of first registry, coil or input **count** - number of records to read **type** - type of data **endianness** - endianness configuration **offset** - number of registers, starting from 'address', the results of which will be discarded. See a detailed description of parameters.	Supported platforms: Linux. Supported since Zabbix 5.2.0.

Network data

Item key			
Description	Return value	Parameters	Comments
net.dns[<ip>,name,<type>,<timeout>,<count>,<protocol>]			
Checks if DNS service is up.	0 - DNS is down (server did not respond or DNS resolution failed) 1 - DNS is up	**ip** - IP address of DNS server (leave empty for the default DNS server, ignored on Windows unless using Zabbix agent 2) **name** - DNS name to query **type** - record type to be queried (default is *SOA*) **timeout** (ignored on Windows unless using Zabbix agent 2) - timeout for the request in seconds (default is 1 second) **count** (ignored on Windows unless using Zabbix agent 2) - number of tries for the request (default is 2) **protocol** (since version 3.0) - the protocol used to perform DNS queries: *udp* (default) or *tcp*	See supported platforms. Example: => net.dns[8.8.8.8,example.com,MX,2,1] 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. Naming before Zabbix 2.0 (still supported): *net.tcp.dns*
net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]			
Performs a DNS query.	Character string with the required type of information	**ip** - IP address of DNS server (leave empty for the default DNS server, ignored on Windows unless using Zabbix agent 2) **name** - DNS name to query **type** - record type to be queried (default is *SOA*) **timeout** (ignored on Windows unless using Zabbix agent 2) - timeout for the request in seconds (default is 1 second) **count** (ignored on Windows unless using Zabbix agent 2) - number of tries for the request (default is 2) **protocol**(since version 3.0) - the protocol used to perform DNS queries: *udp* (default) or *tcp*	See supported platforms. Example: => net.dns.record[8.8.8.8,example.com,MX,2,1] 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. Naming before Zabbix 2.0 (still supported): *net.tcp.dns.query*
net.if.collisions[if]			

Item key			
Number of out-of-window collisions.	Integer	**if** - network interface name	Supported platforms: Linux, FreeBSD, Solaris, AIX, MacOS X, OpenBSD, NetBSD. Root privileges are required on NetBSD.
net.if.discovery			
List of network interfaces. Used for low-level discovery.	JSON object		Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, OpenBSD, NetBSD.
net.if.in[if,<mode>]			
Incoming traffic statistics on network interface.	Integer	**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 transmitted or received by the device driver *multicast* - the number of multicast frames received by the device driver	Supported platforms: Linux, FreeBSD, Solaris[5], HP-UX, AIX, MacOS X, OpenBSD, NetBSD. Root privileges are required on NetBSD. 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,errors] => net.if.in[eth0] You may use this key with the *Change per second* preprocessing step in order to get bytes per second statistics.
net.if.out[if,<mode>]			
Outgoing traffic statistics on network interface.	Integer	**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	Supported platforms: Linux, FreeBSD, Solaris[5], HP-UX, AIX, MacOS X, OpenBSD, NetBSD. Root privileges are required on NetBSD. 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,errors] => net.if.out[eth0] You may use this key with the *Change per second* preprocessing step in order to get bytes per second statistics.
net.if.total[if,<mode>]			

Item key			
Sum of incoming and outgoing traffic statistics on network interface.	Integer	**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 *compressed* - the number of compressed packets transmitted or received by the device driver	Supported platforms: Linux, FreeBSD, Solaris[5], HP-UX, AIX, MacOS X, OpenBSD, NetBSD. Root privileges are required on NetBSD. The *dropped* mode is supported only on Linux, HP-UX. 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,errors] => net.if.total[eth0] You may use this key with the *Change per second* preprocessing step in order to get bytes per second statistics. Note that dropped packets are supported only if both net.if.in and net.if.out work for dropped packets on your platform.
net.tcp.listen[port]			
Checks if this TCP port is in LISTEN state.	0 - it is not in LISTEN state 1 - it is in LISTEN state	**port** - TCP port number	Supported platforms: Linux, FreeBSD, Solaris, MacOS X. Example: => net.tcp.listen[80] On Linux supported since Zabbix 1.8.4 Since Zabbix 3.0.0, on Linux kernels 2.6.14 and above, 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 /proc/net/tcp6 files.
net.tcp.port[<ip>,port]			
Checks if it is possible to make TCP connection to specified port.	0 - cannot connect 1 - can connect	**ip** - IP or DNS name (default is 127.0.0.1) **port** - port number	See supported platforms. Example: => net.tcp.port[,80] → can be used to test availability of web server running on port 80. For simple TCP performance testing use net.tcp.service.perf[tcp,<ip>,<port>] Note that these checks may result in additional messages in system daemon logfiles (SMTP and SSH sessions being logged usually).
net.tcp.service[service,<ip>,<port>]			

Item key			
Checks if service is running and accepting TCP connections.	0 - service is down 1 - service is running	**service** - either of: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	See supported platforms. Example: => net.tcp.service[ftp„45] → can be used to test the availability of FTP server on TCP port 45. Note that 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. Note that the telnet check looks for a login prompt (':' at the end). See also known issues of checking HTTPS service. *https* and *telnet* services are supported since Zabbix 2.0.
net.tcp.service.perf[service,<ip>,<port>]			
Checks performance of TCP service.	0 - service is down seconds - the number of seconds spent while connecting to the service	**service** - either of: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	See supported platforms. Example: => net.tcp.service.perf[ssh] → can be used to test the speed of initial response from SSH server. 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. Note that the telnet check looks for a login prompt (':' at the end). See also known issues of checking HTTPS service. *https* and *telnet* services are supported since Zabbix 2.0.
net.tcp.socket.count[<laddr>,<lport>,<raddr>,<rport>,<state>]			

Item key			
Return the number of TCP sockets that match parameters.	Integer	**laddr** - local IPv4/6 address or CIDR subnet **lport** - local port number or service name **raddr** - remote IPv4/6 address or CIDR subnet **rport** - remote port number or service name **state** - connection state (*established*, *syn_sent*, *syn_recv*, *fin_wait1*, *fin_wait2*, *time_wait*, *close*, *close_wait*, *last_ack*, *listen*, *closing*)	Supported platforms: Linux. Example: => net.tcp.socket.count[,80„,established] → check if local TCP port 80 is in ”established” state This item is supported since Zabbix 6.0.
net.udp.listen[port]			
Checks if this UDP port is in LISTEN state.	0 - it is not in LISTEN state 1 - it is in LISTEN state	**port** - UDP port number	Supported platforms: Linux, FreeBSD, Solaris, MacOS X. Example: => net.udp.listen[68]
net.udp.service[service,<ip>,<port>]			
Checks if service is running and responding to UDP requests.	0 - service is down 1 - service is running	**service** - *ntp* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	See supported platforms. Example: => net.udp.service[ntp„45] → can be used to test the availability of NTP service on UDP port 45. This item is supported since Zabbix 3.0.0, but *ntp* service was available for net.tcp.service[] item in prior versions.
net.udp.service.perf[service,<ip>,<port>]			
Checks performance of UDP service.	0 - service is down seconds - the number of seconds spent waiting for response from the service	**service** - *ntp* (see details) **ip** - IP address (default is 127.0.0.1) **port** - port number (by default standard service port number is used)	See supported platforms. Example: => net.udp.service.perf[ntp] → can be used to test response time from NTP service. This item is supported since Zabbix 3.0.0, but *ntp* service was available for net.tcp.service[] item in prior versions.
net.udp.socket.count[<laddr>,<lport>,<raddr>,<rport>,<state>]			
Return the number of TCP sockets that match parameters.	Integer	**laddr** - local IPv4/6 address or CIDR subnet **lport** - local port number or service name **raddr** - remote IPv4/6 address or CIDR subnet **rport** - remote port number or service name **state** - connection state (*established*, *unconn*)	Supported platforms: Linux. Example: => net.udp.socket.count[„„listening] → check if any UDP socket is in ”listening” state This item is supported since Zabbix 6.0.

Process data

Item key			
Description	Return value	Parameters	Comments
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]			

Item key			
Process CPU utilization percentage.	Float	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **type** - CPU utilization type: *total* (default), *user*, *system* **cmdline** - filter by command line (it is a regular expression) **mode** - data gathering mode: *avg1* (default), *avg5*, *avg15* **zone** - target zone: *current* (default), *all*. This parameter is supported on Solaris only.	Supported platforms: Linux, Solaris[6]. 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 zabbix user The returned value is based on single CPU core utilization percentage. For example 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. *Note* that 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.
proc.get[<name>,<user>,<cmdline>,<mode>]			
List of OS processes and their parameters. Can be used for low-level discovery.	JSON object	**name** - process name (default *all processes*) **user** - 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.	Supported platforms: Linux, FreeBSD, Windows, OpenBSD, NetBSD. 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. Examples: => proc.get[zabbix„,process] → list of all Zabbix processes, returns one entry per PID => proc.get[java„,thread] → list of all Java processes, returns one entry per thread => proc.get[zabbix„,summary] → combined data for Zabbix processes of each type, returns one entry per process name. See also: - Notes on selecting processes with `name` and `cmdline` parameters (Linux-specific). - List of process parameters returned for each mode and OS.
proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]			

Item key			
Memory used by process in bytes. **proc.num**[<name>,<user>,<state>,<cmdline>,<zone>]	Integer - with `mode` as *max*, *min*, *sum* Float - with `mode` as *avg*	**name** - process name (default is *all processes*) **user** - user name (default is *all users*) **mode** - possible values: *avg*, *max*, *min*, *sum* (default) **cmdline** - filter by command line (it is a regular expression) **memtype** - type of memory used by process	Supported platforms: Linux, FreeBSD, Solaris, AIX, Tru64, OpenBSD, NetBSD. The `memtype` parameter is supported only on Linux, FreeBSD, Solaris[6], AIX. Examples: => proc.mem[,root] → memory used by all processes running under the "root" user => proc.mem[zabbix_server,zabbix] → memory used by all zabbix_server processes running under the zabbix user => proc.mem[,oracle,max,oracleZABBIX] → memory used by the most memory-hungry process running under oracle having oracleZABBIX in its command line *Note*: 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.

Item key			
The number of processes.	Integer	**name** - process name (default is *all processes*) **user** - 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** - target zone: *current* (default), *all*. This parameter is supported on Solaris only.	Supported platforms: Linux, FreeBSD, Solaris[6], HP-UX, AIX, Tru64, OpenBSD, NetBSD. The *disk* and *trace* state parameters are supported only on Linux, FreeBSD, OpenBSD, NetBSD. Examples: => proc.num[,mysql] → number of processes running under the mysql user => proc.num[apache2,www-data] → number of apache2 processes running under the www-data user => proc.num[,oracle,sleep,oracleZABBIX] → number of processes in sleep state running under oracle having oracleZABBIX in its command line 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.num[,,,apache2]`), one extra process will be counted, as the agent will count itself. *Note* that 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.

Sensor data

Item key			
Description	Return value	Parameters	Comments
sensor[device,sensor,<mode>]			
Hardware sensor reading.	Float	**device** - device name **sensor** - sensor name **mode** - possible values: *avg*, *max*, *min* (if this parameter is omitted, device and sensor are treated verbatim).	Supported platforms: Linux, OpenBSD. Reads /proc/sys/dev/sensors on Linux 2.4. Example: => sensor[w83781d-i2c-0-2d,temp1] Prior to Zabbix 1.8.4, the *sensor[temp1]* format was used. Reads /sys/class/hwmon on Linux 2.6+. See a more detailed description of sensor item on Linux.

Item key			
			Reads the *hw.sensors* MIB on OpenBSD. Examples: => sensor[cpu0,temp0] → temperature of one CPU => sensor[”cpu[0-2]$”,temp,avg] → average temperature of the first three CPU’s Supported on OpenBSD since Zabbix 1.8.4.

System data

Item key			
Description	Return value	Parameters	Comments
system.boottime			
System boot time.	Integer (Unix timestamp)		Supported platforms: Linux, FreeBSD, Solaris, MacOS X, OpenBSD, NetBSD.
system.cpu.discovery			
List of detected CPUs/CPU cores. Used for low-level discovery.	JSON object		See supported platforms.
system.cpu.intr			
Device interrupts.	Integer		Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD, NetBSD.
system.cpu.load[<cpu>,<mode>]			
CPU load.	Float	**cpu** - possible values: *all* (default), *percpu* (total load divided by online CPU count) **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*	See supported platforms. The *percpu* parameter is not supported on Tru64. Example: => system.cpu.load[,avg5].
system.cpu.num[<type>]			
Number of CPUs.	Integer	**type** - possible values: *online* (default), *max*	Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD. The *max* type parameter is supported only on Linux, FreeBSD, Solaris, MacOS X. Example: => system.cpu.num
system.cpu.switches			
Count of context switches.	Integer		Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD, NetBSD.
system.cpu.util[<cpu>,<type>,<mode>,<logical_or_physical>]			

Item key			
CPU utilization percentage.	Float	**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), *guest_nice* (on Linux kernels 2.6.33 and above). **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15* **logical_or_physical** - possible values: *logical* (default), *physical*. This parameter is supported on AIX only.	Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, Tru64, OpenBSD, NetBSD. 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] Old naming: *system.cpu.idleX, system.cpu.niceX, system.cpu.systemX, system.cpu.userX*
system.hostname[<type>, <transform>]			
System host name.	String	**type** (before version 5.4.7 supported on Windows only) - possible values: *netbios* (default on Windows), *host* (default on Linux), *shorthost* (since version 5.4.7; returns part of the hostname before the first dot, a full string for names without dots). **transform** (since version 5.4.7) - possible values: *none* (default), *lower* (convert to lowercase)	See supported platforms. 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>]			
Chassis information.	String	**info** - one of full (default), model, serial, type or vendor	Supported platforms: Linux. Example: system.hw.chassis[full] Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXXX Desktop] This key depends on the availability of the SMBIOS table. 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. Supported since Zabbix 2.0.
system.hw.cpu[<cpu>,<info>]			

Item key			
CPU information.	String or integer	**cpu** - *<CPU number>* or *all* (default) **info** - possible values: *full* (default), *curfreq*, *maxfreq*, *model* or *vendor*	Supported platforms: Linux. Example: => system.hw.cpu[0,vendor] → AuthenticAMD Gathers info from /proc/cpuinfo and /sys/devices/system/cpu/[cpunum]/cpufreq/cpuinfo_max_ If a CPU number and *curfreq* or *maxfreq* is specified, a numeric value is returned (Hz). Supported since Zabbix 2.0.
system.hw.devices[<type>]			
Listing of PCI or USB devices.	Text	**type** (since version 2.0) - *pci* (default) or *usb*	Supported platforms: Linux. Example: => system.hw.devices[pci] → 00:00.0 Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge [..] Returns the output of either lspci or lsusb utility (executed without any parameters).
system.hw.macaddr[<interface>,<format>]			
Listing of MAC addresses.	String	**interface** - *all* (default) or a regular expression **format** - *full* (default) or *short*	Supported platforms: Linux. Lists MAC addresses of the interfaces whose name matches the given `interface` regular expression (*all* lists for all interfaces). Example: => system.hw.macaddr[”eth0$”,full] → [eth0] 00:11:22:33:44:55 If `format` is specified as *short*, interface names and identical MAC addresses are not listed. Supported since Zabbix 2.0.
system.localtime[<type>]			
System time.	Integer - with `type` as *utc* String - with `type` as *local*	**type** - possible values: *utc* - (default) the time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds. *local* - the time in the ’yyyy-mm-dd,hh:mm:ss.nnn,+hh:mm’ format	See supported platforms. Must be used as a passive check only. Example: => system.localtime[local] → create an item using this key and then use it to display host time in the *Clock* dashboard widget.
system.run[command,<mode>]			

Item key			
Run specified command on the host.	Text result of the command 1 - with `mode` as *nowait* (regardless of command result)	**command** - command for execution **mode** - possible values: *wait* - wait end of execution (default), *nowait* - do not wait	See supported platforms. Up to 512KB of data can be returned, including trailing whitespace that is truncated. To be processed correctly, the output of the command must be text. Example: => system.run[ls -l /] → detailed file list of root directory. *Note*: system.run items are disabled by default. Learn how to enable them. The return value of the item is standard output together with standard error produced by command. The exit code is not checked. Empty result is allowed starting with Zabbix 2.4.0. See also: Command execution.
system.stat[resource,<type>]			

Item key			
System statistics.	Integer or float	**ent** - 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)	
system.sw.arch			
Software architecture information.	String		See supported platforms. Example: => system.sw.arch → i686 Info is acquired from uname() function. Supported since Zabbix 2.0.

Item key			
system.sw.os[<info>]			
Operating system information.	String	**info** - possible values: *full* (default), *short* or *name*	Supported platforms: Linux. Example: => system.sw.os[short]→ Ubuntu 2.6.35-28.50-generic 2.6.35.11 Info is acquired from (note that not all files and options are present in all distributions): /proc/version (*full*) /proc/version_signature (*short*) PRETTY_NAME parameter from /etc/os-release on systems supporting it, or /etc/issue.net (*name*) Supported since Zabbix 2.0.
system.sw.packages[<regexp>,<manager>,<format>]			
Listing of installed packages.	Text	**regexp** - *all* (default) or a regular expression **manager** - *all* (default) or a package manager **format** - *full* (default) or *short*	Supported platforms: Linux. Lists (alphabetically) installed packages whose name matches the given `package` regular expression (*all* lists them all). Example: => system.sw.packages[mini,dpkg,short] → python-minimal, python2.6-minimal, ubuntu-minimal 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. Supported since Zabbix 2.0.
system.swap.in[<device>,<type>]			
Swap in (from device into memory) statistics.	Integer	**device** - specify 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), *pages* (pages swapped in, default on Linux). Note that *pages* will only work if device was not specified.	Supported platforms: Linux, FreeBSD, OpenBSD. The *sectors* type parameter is supported only on Linux. Example: => system.swap.in[,pages] The source of this information is: /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4) /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)

Item key			
system.swap.out[<device>,<type>]			
Swap out (from memory onto device) statistics.	Integer	**device** - specify 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), *pages* (pages swapped out, default on Linux). Note that *pages* will only work if device was not specified.	Supported platforms: Linux, FreeBSD, OpenBSD. The *sectors* type parameter is supported only on Linux. Example: => system.swap.out[,pages] The source of this information is: /proc/swaps, /proc/partitions, /proc/stat (Linux 2.4) /proc/swaps, /proc/diskstats, /proc/vmstat (Linux 2.6)
system.swap.size[<device>,<type>]			
Swap space size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**device** - specify 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), *used* (used swap space) Note that *pfree*, *pused* are not supported on Windows if swap size is 0.	Supported platforms: Linux, FreeBSD, Solaris, AIX, Tru64, OpenBSD. Example: => system.swap.size[,pfree] → free swap space percentage If *device* is not specified Zabbix agent will only take into account swap devices (files), physical memory will be ignored. For example, on Solaris systems *swap -s* command includes a portion of physical memory and swap devices (unlike *swap -l*).
system.uname			
Identification of the system.	String		See supported platforms. Example of returned value (Unix): FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386 On Unix since Zabbix 2.2.0 the value for this item is obtained with uname() system call. Previously it was obtained by invoking ”uname -a”. The value of this item might differ from the output of ”uname -a” and does not include additional information that ”uname -a” prints based on other sources. Note that on Windows the item returns OS architecture, whereas on Unix it returns CPU architecture.
system.uptime			
System uptime in seconds.	Integer		Supported platforms: Linux, FreeBSD, Solaris, AIX, MacOS X, OpenBSD, NetBSD. Support on Tru64 is unknown. In item configuration, use **s** or **uptime** units to get readable values.
system.users.num			

Item key			
Number of users logged in.	Integer		See supported platforms. **who** command is used on the agent side to obtain the value.

Virtual file system data

Item key			
Description	Return value	Parameters	Comments
vfs.dev.discovery			
List of block devices and their type. Used for low-level discovery.	JSON object		Supported platforms: Linux. Supported since Zabbix 4.4.0.
vfs.dev.read[<device>,<type>,<mode>]			
Disk read statistics.	Integer - with `type` in *sectors*, *operations*, *bytes* Float - with `type` in *sps*, *ops*, *bps* *Note*: if using an update interval of three hours or more[2], will always return '0'	**device** - disk device (default is *all* [3]) **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* Note that 'type' parameter support and defaults depend on the platform. *sps*, *ops*, *bps* stand for: sectors, operations, bytes per second, respectively. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*. This parameter is supported only with `type` in: sps, ops, bps.	Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD. The *sectors* and *sps* type parameters are supported only on Linux. The *ops* type parameter is supported only on Linux, 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. Default values of 'type' parameter for different OSes: AIX - operations FreeBSD - bps Linux - sps OpenBSD - operations Solaris - bytes Example: => vfs.dev.read[,operations] *sps*, *ops* and *bps* on supported platforms is limited to 1024 devices (1023 individual and one for *all*).
vfs.dev.write[<device>,<type>,<mode>]			

Item key			
Disk write statistics.	Integer - with `type` in *sectors*, *operations*, *bytes* Float - with `type` in *sps*, *ops*, *bps* *Note*: if using an update interval of three hours or more[2], will always return '0'	**device** - disk device (default is *all* [3]) **type** - possible values: *sectors*, *operations*, *bytes*, *sps*, *ops*, *bps* Note that 'type' parameter support and defaults depend on the platform. *sps*, *ops*, *bps* stand for: sectors, operations, bytes per second, respectively. **mode** - possible values: *avg1* (one-minute average, default), *avg5*, *avg15*. This parameter is supported only with `type` in: sps, ops, bps.	Supported platforms: Linux, FreeBSD, Solaris, AIX, OpenBSD. The *sectors* and *sps* type parameters are supported only on Linux. The *ops* type parameter is supported only on Linux, 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. Default values of 'type' parameter for different OSes: AIX - operations FreeBSD - bps Linux - sps OpenBSD - operations Solaris - bytes Example: => vfs.dev.write[,operations] *sps*, *ops* and *bps* on supported platforms is limited to 1024 devices (1023 individual and one for *all*).
vfs.dir.count[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_ag			

Item key			
Directory entry count.	Integer	**dir** - absolute path to directory **regex_incl** - regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value) **regex_excl** - 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** - maximum depth of subdirectories to traverse. **-1** (default) - unlimited, **0** - no descending into subdirectories. **min_size** - minimum size (in bytes) for file to be counted. Smaller files will not be counted. Memory suffixes can be used. **max_size** - maximum size (in bytes) for file to be counted. Larger files will not be counted. Memory suffixes can be used. **min_age** - minimum age (in seconds) of directory entry to be counted. More recent entries will not be counted. Time suffixes can be used. **max_age** - 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** - regular expression describing the name pattern of the directory to exclude. All content of the directory will be excluded (in contrast to regex_excl)	See supported platforms. Environment variables, e.g. %APP_HOME%, $HOME and %TEMP% are not supported. Pseudo-directories "." and ".." are never counted. Symbolic links are never followed for directory traversal. Both `regex_incl` and `regex_excl` are being applied to files and directories when calculating entry size, but are ignored when picking subdirectories to traverse (if regex_incl is "(?i)^.+\.zip$" and max_depth is not set, then all subdirectories will be traversed, but only files of type zip will be counted). Execution time is limited by the default 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 number of devices in /dev (Linux) Supported since Zabbix 4.0.0.
vfs.dir.get[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_age>			

Item key			
Directory entry list.	JSON	**dir** - absolute path to directory **regex_incl** - regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value) **regex_excl** - 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** - maximum depth of subdirectories to traverse. **-1** (default) - unlimited, **0** - no descending into subdirectories. **min_size** - minimum size (in bytes) for file to be listed. Smaller files will not be listed. Memory suffixes can be used. **max_size** - maximum size (in bytes) for file to be listed. Larger files will not be counted. Memory suffixes can be used. **min_age** - minimum age (in seconds) of directory entry to be listed. More recent entries will not be listed. Time suffixes can be used. **max_age** - 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** - regular expression describing the name pattern of the directory to exclude. All content of the directory will be excluded (in contrast to regex_excl)	See supported platforms. Environment variables, e.g. %APP_HOME%, $HOME and %TEMP% are not supported. Pseudo-directories "." and ".." are never listed. Symbolic links are never followed for directory traversal. Both `regex_incl` and `regex_excl` are being applied to files and directories when calculating entry size, but are ignored when picking subdirectories to traverse (if regex_incl is "(?i)^.+\.zip$" and max_depth is not set, then all subdirectories will be traversed, but only files of type zip will be listed). Execution time is limited by the default 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 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 device list in /dev (Linux) Supported since Zabbix 6.0.0.
vfs.dir.size[dir,<regex_incl>,<regex_excl>,<mode>,<max_depth>,<regex_excl_dir>]			

Item key			
Directory size (in bytes).	Integer	**dir** - absolute path to directory **regex_incl** - regular expression describing the name pattern of the entity (file, directory, symbolic link) to include; include all if empty (default value) **regex_excl** - 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 du command, vfs.dir.size item takes hidden files in account when calculating directory size (acts as `du -sb .[^.]* *` within dir). **max_depth** - maximum depth of subdirectories to traverse. **-1** (default) - unlimited, **0** - no descending into subdirectories. **regex_excl_dir** - regular expression describing the name pattern of the directory to exclude. All content of the directory will be excluded (in contrast to regex_excl)	Supported platforms: Linux. The item may work on other UNIX-like platforms. Only directories with at least read permission for *zabbix* user are calculated. 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. Examples: ⇒ vfs.dir.size[/tmp,log] - calculates size of all files in /tmp which contain 'log' ⇒ vfs.dir.size[/tmp,log,^.+\.old$] - calculates size of all files in /tmp which contain 'log', excluding files containing '.old' The file size limit depends on large file support. Supported since Zabbix 3.4.0.
vfs.file.cksum[file,<mode>]			
File checksum, calculated by the UNIX cksum algorithm.	Integer - with `mode` as *crc32* String - with `mode` as *md5*, *sha256*	**file** - full path to file **mode** - *crc32* (default), *md5*, *sha256*	See #supported platforms. Example: => vfs.file.cksum[/etc/passwd] Example of returned values (crc32/md5/sha256 respectively): 675436101 9845acf68b73991eb7fd7ee0ded23c44 ae67546e4aac995e5c921042d0cf0f1f7147703aa42bfbfb The file size limit depends on large file support. The `mode` parameter is supported since Zabbix 6.0.
vfs.file.contents[file,<encoding>]			
Retrieving contents of a file.	Text	**file** - full path to file **encoding** - code page identifier	See supported platforms. Returns an empty string if the file is empty or contains LF/CR characters only. Byte order mark (BOM) is excluded from the output. Example: => vfs.file.contents[/etc/passwd] This item is limited to files no larger than 64 Kbytes. Supported since Zabbix 2.0.
vfs.file.exists[file,<types_incl>,<types_excl>]			

Item key			
Checks if file exists.	0 - not found 1 - file of the specified type exists	**file** - full path to file **types_incl** - 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** - list of file types to exclude, see types_incl for possible values (by default no types are excluded)	See supported platforms. 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. Examples: => vfs.file.exists[/tmp/application.pid] => vfs.file.exists[/tmp/application.pid,"file,dir,sym"] => vfs.file.exists[/tmp/application_dir,dir] The file size limit depends on large file support.
vfs.file.get[file]			
Return information about a file.	JSON object	**file** - full path to file	See supported platforms. Supported file types on UNIX-like systems: regular file, directory, symbolic link, socket, block device, character device, FIFO Example: => vfs.file.get[/etc/passwd] → return a JSON with information about the /etc/passwd file (type, user, permissions, SID, uid etc) Supported since Zabbix 6.0.
vfs.file.md5sum[file]			
MD5 checksum of file.	Character string (MD5 hash of the file)	**file** - full path to file	See supported platforms. Example: => vfs.file.md5sum[/usr/local/etc/zabbix_agentd.conf] Example of returned value: b5052decb577e0fffd622d6ddc017e82 The file size limit (64 MB) for this item was removed in version 1.8.6. The file size limit depends on large file support.
vfs.file.owner[file,<ownertype>,<resulttype>]			
Retrieve owner of a file.	Character string	**file** - 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	See supported platforms. Example: => vfs.file.owner[/tmp/zabbix_server.log] → return file owner of /tmp/zabbix_server.log => vfs.file.owner[/tmp/zabbix_server.log„id] → return file owner ID of /tmp/zabbix_server.log Supported since Zabbix 6.0.
vfs.file.permissions[file]			

Item key			
Return a 4-digit string containing the octal number with Unix permissions.	String	**file** - full path to the file	Supported platforms: Linux. The item may work on other UNIX-like platforms. Example: => vfs.file.permissions[/etc/passwd] → return permissions of /etc/passwd, for example, '0644' Supported since Zabbix 6.0.
vfs.file.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]			
Find string in a file.	The line containing the matched string, or as specified by the optional `output` parameter	**file** - full path to file **regexp** - regular expression describing the required pattern **encoding** - code page identifier **start line** - the number of first line to search (first line of file by default). **end line** - the number of 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).	See supported platforms. Only the first matching line is returned. An empty string is returned if no line matched the expression. Byte order mark (BOM) is excluded from the output. Content extraction using the `output` parameter takes place on the agent. The `start line`, `end line` and `output` parameters are supported from version 2.2. 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 string in a file.	0 - match not found 1 - found	**file** - full path to file **regexp** - regular expression describing the required pattern **encoding** - code page identifier **start line** - the number of first line to search (first line of file by default). **end line** - the number of last line to search (last line of file by default).	See supported platforms. Byte order mark (BOM) is ignored. The `start line` and `end line` parameters are supported from version 2.2. Example: => vfs.file.regmatch[/var/log/app.log,error]
vfs.file.size[file,<mode>]			

Item key			
File size (in bytes).	Integer	**file** - full path to file **mode** - possible values: *bytes* (default) or *lines* (empty lines are counted, too)	See supported platforms. The file must have read permissions for user *zabbix*. Example: => vfs.file.size[/var/log/syslog] The file size limit depends on large file support. The `mode` parameter is supported since Zabbix 6.0.
vfs.file.time[file,<mode>]			
File time information.	Integer (Unix timestamp)	**file** - full path to the file **mode** - possible values: *modify* (default) - last time of modifying file content, *access* - last time of reading file, *change* - last time of changing file properties	See supported platforms. Example: => vfs.file.time[/etc/passwd,modify] The file size limit depends on large file support.
vfs.fs.discovery			
List of mounted filesystems and their types. Used for low-level discovery.	JSON object		Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD.
vfs.fs.get			
List of mounted filesystems, their types, disk space and inode statistics. Can be used for low-level discovery.	JSON object		Supported platforms: Linux, FreeBSD, Solaris, HP-UX, AIX, MacOS X, OpenBSD, NetBSD. Since Zabbix 6.2.5, this item is capable of reporting file systems with the inode count equal to zero, which can be the case for file systems with dynamic inodes (e.g. btrfs).
vfs.fs.inode[fs,<mode>]			
Number or percentage of inodes.	Integer - for number Float - for percentage	**fs** - filesystem **mode** - possible values: *total* (default), *free*, *used*, *pfree* (free, percentage), *pused* (used, percentage)	See supported platforms. Since Zabbix 6.2.5, this item will not become unsupported in pfree/pused modes if the inode count equals to zero, which can be the case for file systems with dynamic inodes (e.g. btrfs). Instead the pfree/pused values for such file systems will be reported as ”100” and ”0” respectively. Example: => vfs.fs.inode[/,pfree]
vfs.fs.size[fs,<mode>]			

Item key			
Disk space in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**fs** - filesystem **mode** - possible values: *total* (default), *free*, *used*, *pfree* (free, percentage), *pused* (used, percentage)	See supported platforms. In case of a mounted volume, disk space for local file system is returned. Example: => vfs.fs.size[/tmp,free] Reserved space of a file system is taken into account and not included when using the *free* mode.

Virtual memory data

Item key			
Description	Return value	Parameters	Comments
vm.memory.size[<mode>]			
Memory size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**mode** - possible values: *total* (default), *active*, *anon*, *buffers*, *cached*, *exec*, *file*, *free*, *inactive*, *pinned*, *shared*, *slab*, *wired*, *used*, *pused* (used, percentage), *available*, *pavailable* (available, percentage) The `mode` parameter support is platform-specific (see item comments). See also additional details for this parameter.	See supported platforms. 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. 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*.

Web monitoring data

Item key			
Description	Return value	Parameters	Comments
web.page.get[host,<path>,<port>]			

Item key			
Get content of web page.	Web page source as text (including headers)	**host** - hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[4]. Missing scheme will be treated as *http*. If 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** - path to HTML document (default is /) **port** - port number (default is 80 for HTTP)	See supported platforms. This item turns unsupported if the resource specified in `host` does not exist or is unavailable. `host` can be 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>]			
Loading time of full web page (in seconds).	Float	**host** - hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[4]. Missing scheme will be treated as *http*. If 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** - path to HTML document (default is /) **port** - port number (default is 80 for HTTP)	See supported platforms. This item turns unsupported if the resource specified in `host` does not exist or is unavailable. `host` can be 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>]			

Item key			
Find string on a web page.	The matched string, or as specified by the optional `output` parameter	**host** - hostname or URL (as `scheme://host:port/path`, where only *host* is mandatory). Allowed URL schemes: *http*, *https*[4]. Missing scheme will be treated as *http*. If 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** - path to HTML document (default is /) **port** - port number (default is 80 for HTTP) **regexp** - regular expression describing the required pattern **length** - 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).	See supported platforms. This item turns unsupported if the resource specified in `host` does not exist or is unavailable. `host` can be 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. The `output` parameter is supported from version 2.2. Example: => web.page.regexp[www.example.com,index.php,80,OK,2] => web.page.regexp[https://www.example.com,,,OK,2]

Zabbix metrics

Item key			
Description	Return value	Parameters	Comments
agent.hostmetadata			
Agent host metadata.	String		See supported platforms. Returns the value of HostMetadata or HostMetadataItem parameters, or empty string if none are defined. Supported since Zabbix 6.0.
agent.hostname			
Agent host name.	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			
Agent availability check.	Nothing - unavailable 1 - available		See supported platforms. Use the **nodata()** trigger function to check for host unavailability.
agent.variant			

Item key			
Variant of Zabbix agent (Zabbix agent or Zabbix agent 2). **agent.version**	Integer		See supported platforms. Example of returned value: 1 - Zabbix agent 2 - Zabbix agent 2
Version of Zabbix agent.	String		See supported platforms. Example of returned value: 6.0.3
zabbix.stats[<ip>,<port>]			
Return a set of Zabbix server or proxy internal metrics remotely.	JSON object	**ip** - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1) **port** - port of server/proxy to be remotely queried (default is 10051)	See supported platforms. Note that 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>]			
Return number of monitored items in the queue which are delayed on Zabbix server or proxy remotely.	JSON object	**ip** - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1) **port** - 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)	See supported platforms. 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, sbd 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.

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.

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:
  - **eventlog[]**
  - **net.if.list**
  - **perf_counter[]**
  - **perf_counter_en[]**
  - **perf_instance.discovery[]**
  - **proc_info[]**
  - **registry.data[]**
  - **registry.get[]**
  - **service.discovery**
  - **service.info[]**
  - **services**
  - **vm.vmemory.size[]**
  - **wmi.get[]**
  - **wmi.getall[]**

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.

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 in the UNIX Zabbix agent item group
- The item key signature includes only those parameters that are supported on Windows
- Windows-relevant item comments are included

Item key	Comments
agent.hostmetadata	
agent.hostname	

Item key	Comments
agent.ping	
agent.variant	
agent.version	
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>]	This item is not supported for Windows Event Log. The `persistent_dir` parameter is not supported on Windows.
log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>]	This item is not supported for Windows Event Log. The `persistent_dir` parameter is not supported on Windows.
logrt[file_regexp,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>]	This item is not supported for Windows Event Log. The `persistent_dir` parameter is not supported on Windows.
logrt.count[file_regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>]	This item is not supported for Windows Event Log. The `persistent_dir` parameter is not supported on Windows.
modbus.get[endpoint,<slave id>,<function>,<address>,<count>,<type>,<endianness>,<offset>]	
net.dns[<ip>,name,<type>,<timeout>,<count>]	The `ip`, `timeout` and `count` parameters are ignored on Windows.
net.dns.record[<ip>,name,<type>,<timeout>,<count>]	The `ip`, `timeout` and `count` parameters are ignored on Windows.
net.if.discovery	Some Windows versions (for example, Server 2008) might require the latest updates installed to support non-ASCII characters in interface names.
net.if.in[if,<mode>]	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[if,<mode>]	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[if,<mode>]	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[port]	
net.tcp.port[<ip>,port]	
net.tcp.service[service,<ip>,<port>]	Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2.
net.tcp.service.perf[service,<ip>,<port>]	Checking of LDAP and HTTPS on Windows is only supported by Zabbix agent 2.
net.tcp.socket.count[<laddr>,<lport>,<state>] *	*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[service,<ip>,<port>]	
net.udp.service.perf[service,<ip>,<port>]	
net.udp.socket.count[<laddr>,<lport>,<state>] *	*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[<name>,<user>]	On Windows, only the `name` and `user` parameters are supported.
system.cpu.discovery	
system.cpu.load[<cpu>,<mode>]	
system.cpu.num[<type>]	
system.cpu.util[<cpu>,<type>,<mode>]	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.

Item key	Comments
system.hostname[<type>, <transform>]	The value is acquired by either GetComputerName() (for **netbios**) or gethostname() (for **host**) functions on Windows. Examples of returned values: => system.hostname → WIN-SERV2008-I6 => system.hostname[host] → Win-Serv2008-I6LonG => system.hostname[host,lower] → win-serv2008-i6long See also a more detailed description.
system.localtime[<type>]	
system.run[command,<mode>]	
system.sw.arch	
system.swap.size[<device>,<type>]	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.
system.uname	Example of returned value: Windows ZABBIX-WIN 6.0.6001 Microsoft® Windows Server® 2008 Standard Service Pack 1 x86 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	
vfs.dir.count[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_age>]	On Windows directory symlinks are skipped and hard links are counted only once. Example: ⇒ vfs.dir.count["C:\Users\ADMINI~1\AppData\Local\Temp"] - monitors the number of files in temporary directory
vfs.dir.get[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_age>]	On Windows directory symlinks are skipped and hard links are counted only once. Example: ⇒ vfs.dir.get["C:\Users\ADMINI~1\AppData\Local\Temp"] - retrieves the file list in temporary directory
vfs.dir.size[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_age>]	On Windows any symlink is skipped and hard links are taken into account only once.
vfs.file.cksum[file,<mode>]	
vfs.file.contents[file,<encoding>]	
vfs.file.exists[file,<types_incl>,<types_excl>]	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[file]	Supported file types on Windows: regular file, directory, symbolic link
vfs.file.md5sum[file]	
vfs.file.owner[file,<ownertype>,<resulttype>]	
vfs.file.regexp[\file,regexp,<encoding>,<start line>,<end line>,<output>]	
vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]	
vfs.file.size[file,<mode>]	
vfs.file.time[file,<mode>]	On Windows XP vfs.file.time[file,change] may be equal to vfs.file.time[file,access].
vfs.fs.discovery	The {#FSLABEL} macro is supported on Windows since Zabbix 6.0.
vfs.fs.get	The {#FSLABEL} macro is supported on Windows since Zabbix 6.0.
vfs.fs.size[fs,<mode>]	
vm.memory.size[<mode>]	
web.page.get[host,<path>,<port>]	
web.page.perf[host,<path>,<port>]	
web.page.regexp[host,<path>,<port>,regexp,<length>,<output>]	

Item key	Comments
zabbix.stats[<ip>,<port>] **zabbix.stats**[<ip>,<port>,queue,<from>,<to>]	

Windows-specific items

The table provides details on the item keys that are supported **only** by the Windows Zabbix agent.

Item key			
Description	Return value	Parameters	Comments
eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>,<mode>]			
Event log monitoring.	Log	**name** - name of event log **regexp** - regular expression describing the required pattern **severity** - regular expression describing severity (case-insensitive) This parameter accepts the following values: ”Information”, ”Warning”, ”Error”, ”Critical”, ”Verbose” (since Zabbix 2.2.0 running on Windows Vista or newer) **source** - regular expression describing source identifier (case-insensitive; regular expression is supported since Zabbix 2.2.0) **eventid** - regular expression describing the event identifier(s) **maxlines** - 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.win.conf **mode** - possible values: *all* (default), *skip* - skip processing of older data (affects only newly created items).	The item must be configured as an active check. 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 a *Result is TRUE* type, the expression itself being `^1$\|^70$`). *Note* that the agent is unable to send in events from the ”Forwarded events” log. The `mode` parameter is supported since Zabbix 2.0.0. ”Windows Eventing 6.0” is supported since Zabbix 2.2.0. Note that 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.
net.if.list Network interface list (includes interface type, status, IPv4 address, description).	Text		Supported since Zabbix agent version 1.8.1. Multi-byte interface names supported since Zabbix agent version 1.8.6. Disabled interfaces are not listed. Note that 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>]			

Item key			
Value of any Windows performance counter.	Integer, float, string or text (depending on the request)	**counter** - path to the counter **interval** - last N seconds for storing the average value. The `interval` must be between 1 and 900 seconds (included) and the default value is 1.	Performance Monitor can be used to obtain list of available counters. Until version 1.6 this parameter will return correct value only for counters that require just one sample (like \System\Threads). It will not work as expected for counters that require more than one sample - like CPU utilization. Since 1.6, `interval` is used, so the check returns an average value for last "interval" seconds every time. See also: Windows performance counters.
perf_counter_en[counter,<interval>]			
Value of any Windows performance counter in English.	Integer, float, string or text (depending on the request)	**counter** - path to the counter in English **interval** - last N seconds for storing the average value. The `interval` must be between 1 and 900 seconds (included) and the default value is 1.	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\Windo NT\CurrentVersion\Perflib\009`. Supported since Zabbix agent versions 4.0.13 and 4.2.7.
perf_instance.discovery[object]			
List of object instances of Windows performance counters. Used for low-level discovery.	JSON object	**object** - object name (localized)	Supported since Zabbix agent version 5.0.1.
perf_instance_en.discovery[object]			
List of object instances of Windows performance counters, discovered using object names in English. Used for low-level discovery.	JSON object	**object** - object name (in English)	Supported since Zabbix agent version 5.0.1.
proc_info[process,<attribute>,<type>]			

Item key			
Various information about specific process(es).	Float	**process** - process name **attribute** - requested process attribute **type** - representation type (meaningful when more than one process with the same name exists)	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> Examples: => proc_info[iexplore.exe,wkset,sum] - to get the amount of physical memory taken by all Internet Explorer processes => proc_info[iexplore.exe,pf,avg] - to get the average number of page faults for Internet Explorer processes Note that on a 64-bit system, a 64-bit Zabbix agent is required for this item to work correctly. Note: *io_**, *gdiobj* and *userobj* attributes are available only on Windows 2000 and later versions of Windows, not on Windows NT 4.0.
registry.data[key,<value name>]			

Item key			
Return data for the specified value name in the Windows Registry key.	Integer, string or text (depending on the value type)	**key** - registry key including the root key; root abbreviations (e.g. HKLM) are allowed **value name** - registry value name in the key (empty string "" by default). The default value is returned if the value name is not supplied.	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\W Error Reporting"] - return the data of the default value of this key => registry.data["HKLM\SOFTWARE\Microsoft\Windows\Windows Error Reporting","EnableZip"] - return the data of the value named "Enable Zip" in this key This key is supported since Zabbix 6.2.0.
registry.get[key,<mode>,<name regexp>]			
List of Windows Registry values or keys located at given key.	JSON	**key** - 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), *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\CurrentVer - return the data of the values named "DisplayName" or "DisplayValue" in this key. The JSON will include details of the key, last subkey, value name, value type and value data. => registry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVer - return the data of the all values in this key. The JSON will include details of the key, last subkey, value name, value type and value data. => registry.get[HKLM\SOFTWARE\Microsoft\Windows\CurrentVer - return all subkeys of this key. The JSON will include details of the key and last subkey. This key is supported since Zabbix 6.2.0.
service.discovery			
List of Windows services. Used for low-level discovery.	JSON object		Supported since Zabbix agent version 3.0.
service.info[service,<param>]			

Item key			
Information about a service.	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	**service** - a real service name or its display name as seen in MMC Services snap-in **param** - *state* (default), *displayname*, *path*, *user*, *startup* or *description*	Examples: => service.info[SNMPTRAP] - state of the SNMPTRAP service => service.info[SNMP Trap] - state of the same service, but with display name specified => service.info[EventLog,startup] - startup type of the EventLog service Items service.info[service,state] and service.info[service] will return the same information. Note that only with `param` as *state* this item returns a value for non-existing services (255). This item is supported since Zabbix 3.0.0. It should be used instead of the deprecated service_state[service] item.
services[<type>,<state>,<exclude>]			
Listing of services.	0 - if empty Text - list of services separated by a newline	**type** - *all* (default), *automatic*, *manual* or *disabled* **state** - *all* (default), *stopped*, *started*, *start_pending*, *stop_pending*, *running*, *continue_pending*, *pause_pending* or *paused* **exclude** - services to exclude from the result. Excluded services should be listed in double quotes, separated by comma, without spaces.	Examples: => services[,started] - list of started services => services[automatic, stopped] - list of stopped services, that should be run => services[automatic, stopped, ”service1,service2,service3”] - list of stopped services, that should be run, excluding services with names service1, service2 and service3 The `exclude` parameter is supported since Zabbix 1.8.1.
vm.vmemory.size[<type>]			

Item key			
Virtual memory size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**type** - possible values: *available* (available virtual memory), *pavailable* (available virtual memory, in percent), *pused* (used virtual memory, in percent), *total* (total virtual memory, default), *used* (used virtual memory)	Example: => vm.vmemory.size[pavailable] → available virtual memory, in percentage 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. This key is supported since Zabbix 3.0.7 and 3.2.3.
wmi.get[<namespace>,<query>]			
Execute WMI query and return the first selected object.	Integer, float, string or text (depending on the request)	**namespace** - WMI namespace **query** - 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 status of the first physical disk This key is supported since Zabbix 2.2.0.
wmi.getall[<namespace>,<query>]			
Execute WMI query and return the whole response. Can be used for low-level discovery.	JSON object	**namespace** - WMI namespace **query** - WMI query	WMI queries are performed with WQL. Example: => wmi.getall[root\cimv2,select * from Win32_DiskDrive where Name like '%PHYSICALDRIVE%'] - returns status information of physical disks JSONPath preprocessing can be used to point to more specific values in the returned JSON. This key is supported since Zabbix 4.4.0.

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 that name by going to 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 the 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)
- *Show value*: select the *Windows service state* value mapping

Two value maps are available *Windows service state* and *Windows service startup type* to map a numerical value to a text representation in the Frontend.

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.

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.

See also: Plugins supplied out-of-the-box

Note:
Parameters without angle brackets are mandatory. Parameters marked with angle brackets **< >** are optional.

Ceph

Key

Description	Return value	Parameters	Comments
ceph.df.details[connString,<user>,<apikey>]			
Cluster's data usage and distribution among pools.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.	
ceph.osd.stats[connString,<user>,<apikey>]			
Aggregated and per OSD statistics.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.	
ceph.osd.discovery[connString,<user>,<apikey>]			
List of discovered OSDs. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.	
ceph.osd.dump[connString,<user>,<apikey>]			
Usage thresholds and statuses of OSDs.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.	
ceph.ping[connString,<user>,<apikey>]			
Tests whether a connection to Ceph can be established.	0 - connection is broken (if there is any error presented including AUTH and configuration issues) 1 - connection is successful.	**connString** - URI or session name. **user, password** - Ceph login credentials.	

Key		
ceph.pool.discovery[connString,<user>,<apikey>]		
List of discovered pools. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.
ceph.status[connString,<user>,<apikey>]		
Overall cluster's status.	JSON object	**connString** - URI or session name. **user, password** - Ceph login credentials.

Docker

Key			
Description	**Return value**	**Parameters**	**Comments**
docker.container_info[<ID>,<info>]			
Low-level information about a container.	An output of the Container-Inspect API call serialized as JSON	**ID** - ID or name of the container. **info** - the amount of information returned. Supported values: *short* (default) or *full*.	The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.container_stats[<ID>]			
Container resource usage statistics.	An output of the Container-Stats API call and CPU usage percentage serialized as JSON	**ID** - ID or name of the container.	The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.containers			
A list of containers.	An output of the ContainerList API call serialized as JSON	-	The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.containers.discovery[<options>]			
A list of containers. Used for low-level discovery.	JSON object	**options** - specifies whether all or only running containers should be discovered. Supported values: *true* - return all containers; *false* - return only running containers (default).	The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.data_usage			
Information about current data usage.	An output of the System-DataUsage API call serialized as JSON	-	The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.images			
A list of images.	An output of the ImageList API call serialized as JSON	-	The Agent2 user ('zabbix') must be added to the 'docker' group for sufficient privileges. Otherwise the check will fail.
docker.images.discovery			

Key			
A list of images. Used for low-level discovery. docker.info	JSON object	-	The Agent2 user (‘zabbix’) must be added to the ‘docker’ group for sufficient privileges. Otherwise the check will fail.
System information.	An output of the SystemInfo API call serialized as JSON	-	The Agent2 user (‘zabbix’) must be added to the ‘docker’ group for sufficient privileges. Otherwise the check will fail.
docker.ping Test if a Docker daemon is alive or not.	1 - connection is alive 0 - connection is broken	-	The Agent2 user (‘zabbix’) must be added to the ‘docker’ group for sufficient privileges. Otherwise the check will fail.

Memcached

Key			
Description	**Return value**	**Parameters**	**Comments**
memcached.ping[connString,<user>,<password>]			
Test if a connection is alive or not.	1 - connection is alive 0 - connection is broken (if there is any error presented including AUTH and configuration issues)	**connString** - URI or session name.	
memcached.stats[connString,<user>,<password>,<type>]			
Gets the output of the STATS command.	JSON - output is serialized as JSON	**connString** - URI or session name. **user, password** - Memcached login credentials. **type** - stat type to be returned: *items*, *sizes*, *slabs* or *settings* (empty by default, returns general statistics).	

MongoDB

Key			
Description	**Return value**	**Parameters**	**Comments**
mongodb.collection.stats[connString,<user>,<password>,<database>,collection]			
Returns a variety of storage statistics for a given collection.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials. **database** - database name (default: admin). **collection** — collection name.	
mongodb.collections.discovery[connString,<user>,<password>]			

Key		
Returns a list of discovered collections. Used for low-level discovery. mongodb.collections.usage[connString,<user>,<password>]	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.
Returns usage statistics for collections. mongodb.connpool.stats[connString,<user>,<password>]	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.
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[connString,<user>,<password>,<database>]	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.
Returns statistics reflecting a given database system state. mongodb.db.discovery[connString,<user>,<password>]	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials. **database** - database name (default: admin).
Returns a list of discovered databases. Used for low-level discovery. mongodb.jumbo_chunks.count[connString,<user>,<password>]	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.
Returns count of jumbo chunks. mongodb.oplog.stats[connString,<user>,<password>]	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.
Returns a status of the replica set, using data polled from the oplog. mongodb.ping[connString,<user>,<password>]	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.

Key		
Tests if a connection is alive or not.	1 - connection is alive 0 - connection is broken (if there is any error presented including AUTH and configuration issues).	**connString** - URI or session name. **user, password** - MongoDB login credentials.
mongodb.rs.config[connString,<user>,<password>]		
Returns a current configuration of the replica set.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.
mongodb.rs.status[connString,<user>,<password>]		
Returns a replica set status from the point of view of the member where the method is run.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.
mongodb.server.status[connString,<user>,<password>]		
Returns database state.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.
mongodb.sh.discovery[connString,<user>,<password>]		
Returns a list of discovered shards present in the cluster.	JSON object	**connString** - URI or session name. **user, password** - MongoDB login credentials.

MQTT

Key			
Description	**Return value**	**Parameters**	**Comments**
mqtt.get[<broker_url>,topic,<username>,<password>]			
Subscribes to a specific topic or topics (with wildcards) of the provided broker and waits for publications.	Depending on topic content. If wildcards are used, returns topic content as JSON.	**broker_url** - MQTT broker URL (if empty, *localhost* with port 1883 is used). **topic** - MQTT topic (mandatory). Wildcards (+,#) are supported. **username,password** - authentication credentials (if required)	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.

MySQL

Key

Description	Return value	Parameters	Comments
mysql.db.discovery[connString,<username>,<password>]			
List of MySQL databases. Used for low-level discovery.	Result of the "show databases" SQL query in LLD JSON format.	**connString** - URI or session name. **username, password** - MySQL login credentials.	
mysql.db.size[connString,<username>,<password>,dbName]			
Database size in bytes.	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.	**connString** - URI or session name. **username, password** - MySQL login credentials. **dbName** - Database name.	
mysql.get_status_variables[connString,<username>,<password>]			
Values of global status variables.	Result of the "show global status" SQL query in JSON format.	**connString** - URI or session name. **username, password** - MySQL login credentials.	
mysql.ping[connString,<username>,<password>]			
Test if a connection is alive or not.	1 - connection is alive 0 - connection is broken (if there is any error presented including AUTH and configuration issues).	**connString** - URI or session name. **username, password** - MySQL login credentials.	
mysql.replication.discovery[connString,<username>,<password>]			
List of MySQL replications. Used for low-level discovery.	Result of the "show slave status" SQL query in LLD JSON format.	**connString** - URI or session name. **username, password** - MySQL login credentials.	
mysql.replication.get_slave_status[connString,<username>,<password>,<masterHost>]			
Replication status.	Result of the "show slave status" SQL query in JSON format.	**connString** - URI or session name. **username, password** - MySQL login credentials. **masterHost** - Replication master host name.	
mysql.version[connString,<username>,<password>]			
MySQL version.	String with MySQL instance version.	**connString** - URI or session name. **username, password** - MySQL login credentials.	

Key			
Description	**Return value**	**Parameters**	**Comments**
oracle.diskgroups.stats[connString,<user>,<password>,<service>,<diskgroup>]			
Automatic Storage Management (ASM) disk groups statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **diskgroup** - name of the ASM disk group to query.	
oracle.diskgroups.discovery[connString,<user>,<password>,<service>]			
List of ASM disk groups. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	
oracle.archive.info[connString,<user>,<password>,<service>,<destination>]			
Archive logs statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **destination** - name of the destination to query.	
oracle.cdb.info[connString,<user>,<password>,<service>,<database>]			
Container Databases (CDBs) information.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **database** - name of the database to query.	
oracle.custom.query[connString,<user>,<password>,<service>,queryName,<args...>]			
Result of a custom query.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **queryName** — name of a custom query (must be equal to a name of an sql file without an extension). **args...** — one or several comma-separated arguments to pass to a query.	
oracle.datafiles.stats[connString,<user>,<password>,<service>]			
Data files statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	
oracle.db.discovery[connString,<user>,<password>,<service>]			
List of databases. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	
oracle.fra.stats[connString,<user>,<password>,<service>]			
Fast Recovery Area (FRA) statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	
oracle.instance.info[connString,<user>,<password>,<service>]			
Instance statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	
oracle.pdb.info[connString,<user>,<password>,<service>,<database>]			

Key		
Pluggable Databases (PDBs) information.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **database** - name of the database to query.
oracle.pdb.discovery[connString,<user>,<password>,<service>]		
List of PDBs. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.
oracle.pga.stats[connString,<user>,<password>,<service>]		
Program Global Area (PGA) statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.
oracle.ping[connString,<user>,<password>,<service>]		
Tests whether a connection to Oracle can be established.	0 - connection is broken (if there is any error presented including AUTH and configuration issues) 1 - connection is successful.	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.
oracle.proc.stats[connString,<user>,<password>,<service>]		
Processes statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.
oracle.redolog.info[connString,<user>,<password>,<service>]		
Log file information from the control file.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.
oracle.sga.stats[connString,<user>,<password>,<service>]		
System Global Area (SGA) statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.
oracle.sessions.stats[connString,<user>,<password>,<service>,<lockMaxTime>]		
Sessions statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **lockMaxTime** - 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>]		
A set of system metric values.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **duration** - capturing interval (in seconds) of system metric values. Possible values: *60* — long duration (default), *15* — short duration.
oracle.sys.params[connString,<user>,<password>,<service>]		
A set of system parameter values.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.

Key			
oracle.ts.stats[connString,<user>,<password>,<service>,<tablespace>,<type>]			
Tablespaces statistics.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - 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** - type of the tablespace to query. Default (if `tablespace` is set): ”PERMANENT”.	
oracle.ts.discovery[connString,<user>,<password>,<service>]			
List of tablespaces. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name.	
oracle.user.info[connString,<user>,<password>,<service>,<username>]			
List of tablespaces. Used for low-level discovery.	JSON object	**connString** - URI or session name. **user, password** - Oracle login credentials. **service** - Oracle service name. **username** - a username, for which the information is needed. Lowercase usernames are not supported. Default: current user.	

PostgreSQL

Key			
Description	**Return value**	**Parameters**	**Comments**
pgsql.autovacuum.count[uri,<username>,<password>,<dbName>]			
The number of autovacuum workers.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	
pgsql.archive[uri,<username>,<password>,<dbName>]			
Information about archived files.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	Returned data are processed by dependent items: **pgsql.archive.count_archived_files** - the number of WAL files that have been successfully archived. **pgsql.archive.failed_trying_to_archive** - the number of failed attempts for archiving WAL files. **pgsql.archive.count_files_to_archive** - the number of files to archive. **pgsql.archive.size_files_to_archive** -the size of files to archive.
pgsql.bgwriter[uri,<username>,<password>,<dbName>]			

Key			
Combined number of checkpoints for the database cluster, broken down by checkpoint type.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	Returned data are processed by dependent items: **pgsql.bgwriter.buffers_alloc** - the number of buffers allocated. **pgsql.bgwriter.buffers_backend** -the number of buffers written directly by a backend. **pgsql.bgwriter.maxwritten_clean** - the number of times the background writer stopped a cleaning scan, because it had written too many buffers. **pgsql.bgwriter.buffers_backend_fsync** -the number of times a backend had to execute its own fsync call instead of the background writer. **pgsql.bgwriter.buffers_clean** - the number of buffers written by the background writer. **pgsql.bgwriter.buffers_checkpoint** - the number of buffers written during checkpoints. **pgsql.bgwriter.checkpoints_timed** - the number of scheduled checkpoints that have been performed. **pgsql.bgwriter.checkpoints_req** - the number of requested checkpoints that have been performed. **pgsql.bgwriter.checkpoint_write_time** - the total amount of time spent in the portion of checkpoint processing where files are written to disk, in milliseconds. **pgsql.bgwriter.sync_time** - the total amount of time spent in the portion of checkpoint processing where files are synchronized with disk.
pgsql.cache.hit[uri,<username>,<password>,<dbName>]			
PostgreSQL buffer cache hit rate.	Float	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	
pgsql.connections[uri,<username>,<password>,<dbName>]			

Key			
Connections by type.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	Returned data are processed by dependent items: **pgsql.connections.active** - the backend is executing a query. **pgsql.connections.fastpath_function_call** -the backend is executing a fast-path function. **pgsql.connections.idle** - the backend is waiting for a new client command. **pgsql.connections.idle_in_transaction** - the backend is in a transaction, but is not currently executing a query. **pgsql.connections.prepared** - the number of prepared connections. **pgsql.connections.total** - the total number of connections. **pgsql.connections.total_pct** - percantange of total connections in respect to 'max_connections' setting of the PostgreSQL server. **pgsql.connections.waiting** - number of connections in a query. **pgsql.connections.idle_in_transaction_aborted** - the backend is in a transaction, but is not currently executing a query and one of the statements in the transaction caused an error.
pgsql.custom.query[uri,<username>,<password>,queryName[,args...]]			
Returns result of a custom query.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **queryName** - name of a custom query, must match SQL file name without an extension. **args**(optional) - arguments to pass to a query.	
pgsql.dbstat[uri,<username>,<password>,dbName]			

Key			
Collects statistics per database. Used for low-level discovery.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	Returned data are processed by dependent items: **pgsql.dbstat.numbackends["{#DBNAME}"]** - time spent reading data file blocks by backends in this database, in milliseconds. **pgsql.dbstat.sum.blk_read_time["{#DBNAME}"]** - time spent reading data file blocks by backends in this database, in milliseconds. **pgsql.dbstat.sum.blk_write_time["{#DBNAME}"]** - time spent writing data file blocks by backends in this database, in milliseconds. **pgsql.dbstat.sum.checksum_failures["{#DBNAME}** - the number of data page checksum failures detected (or on a shared object), or NULL if data checksums are not enabled.(PostgreSQL version 12 only) **pgsql.dbstat.blks_read.rate["{#DBNAME}"]** - the number of disk blocks read in this database. **pgsql.dbstat.deadlocks.rate["{#DBNAME}"]** - the number of deadlocks detected in this database. **pgsql.dbstat.blks_hit.rate["{#DBNAME}"]** - the number of times disk blocks were found already in the buffer cache, so that a read was not necessary (this only includes hits in the PostgreSQL Pro buffer cache, not the operating system's file system cache). **pgsql.dbstat.xact_rollback.rate["{#DBNAME}"]** - the number of transactions in this database that have been rolled back. **pgsql.dbstat.xact_commit.rate["{#DBNAME}"]** - the number of transactions in this database that have been committed. **pgsql.dbstat.tup_updated.rate["{#DBNAME}"]** - the number of rows updated by queries in this database. **pgsql.dbstat.tup_returned.rate["{#DBNAME}"]** - the number of rows returned by queries in this database. **pgsql.dbstat.tup_inserted.rate["{#DBNAME}"]** - the number of rows inserted by queries in this database. **pgsql.dbstat.tup_fetched.rate["{#DBNAME}"]** - the number of rows fetched by queries in this database. **pgsql.dbstat.tup_deleted.rate["{#DBNAME}"]** - the number of rows deleted by queries in this database. **pgsql.dbstat.conflicts.rate["{#DBNAME}"]** - the number of queries canceled due to conflicts with recovery in this database (the conflicts occur only on standby servers). **pgsql.dbstat.temp_files.rate["{#DBNAME}"]** - the number of temporary files created by queries in this database. All temporary files are counted, regardless of the log_temp_files settings and reasons for which the temporary file was created (e.g., sorting or hashing). **pgsql.dbstat.temp_bytes.rate["{#DBNAME}"]** - the total amount of data written to temporary files by queries in this

Key
pgsql.dbstat.sum[uri,<username>,<password>,<dbName>]

Key			
Summarized data for all databases in a cluster.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	Returned data are processed by the dependent items: **pgsql.dbstat.numbackends** - the number of backends currently connected to this database. **pgsql.dbstat.sum.blk_read_time** - time spent reading data file blocks by backends in this database, in milliseconds. **pgsql.dbstat.sum.blk_write_time** - time spent writing data file blocks by backends in this database, in milliseconds. **pgsql.dbstat.sum.checksum_failures** - the number of data page checksum failures detected (or on a shared object), or NULL if data checksums are not enabled (PostgreSQL version 12 only). **pgsql.dbstat.sum.xact_commit** - the number of transactions in this database that have been committed. **pgsql.dbstat.sum.conflicts** - database statistics about query cancels due to conflict with recovery on standby servers. **pgsql.dbstat.sum.deadlocks** - the number of deadlocks detected in this database. **pgsql.dbstat.sum.blks_read** - the number of disk blocks read in this database. **pgsql.dbstat.sum.blks_hit** - the number of times disk blocks were found already in the buffer cache, so a read was not necessary (only hits in the PostgreSQL Pro buffer cache are included). **pgsql.dbstat.sum.temp_bytes** - the total amount of data written to temporary files by queries in this database. Includes data from all temporary files, regardless of the log_temp_files settings and reasons for which the temporary file was created (e.g., sorting or hashing). **pgsql.dbstat.sum.temp_files** - the number of temporary files created by queries in this database. All temporary files are counted, regardless of the log_temp_files settings and reasons for which the temporary file was created (e.g., sorting or hashing). **pgsql.dbstat.sum.xact_rollback** - the number of rolled-back transactions in this database. **pgsql.dbstat.sum.tup_deleted** - the number of rows deleted by queries in this database. **pgsql.dbstat.sum.tup_fetched** - the number of rows fetched by queries in this database. **pgsql.dbstat.sum.tup_inserted** - the number of rows inserted by queries in this database. **pgsql.dbstat.sum.tup_returned** - the number of rows returned by queries in this database. **pgsql.dbstat.sum.tup_updated** - the number of rows updated by queries in this database.

Key			
pgsql.db.age[uri,<username>,<password>,dbName]			
Age of the oldest FrozenXID of the database. Used for low-level discovery.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	
pgsql.db.bloating_tables[uri,<username>,<password>,<dbName>]			
The number of bloating tables per database. Used for low-level discovery.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	
pgsql.db.discovery[uri,<username>,<password>,<dbName>]			
List of the PostgreSQL databases. Used for low-level discovery.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	
pgsql.db.size[uri,<username>,<password>,dbName]			
Database size in bytes. Used for low-level discovery.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	
pgsql.locks[uri,<username>,<password>,<dbName>]			
Information about granted locks per database. Used for low-level discovery.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	Returned data are processed by dependent items: **pgsql.locks.shareupdateexclusive["{#DBNAME}"]** - the number of share update exclusive locks. **pgsql.locks.accessexclusive["{#DBNAME}"]** - the number of access exclusive locks. **pgsql.locks.accessshare["{#DBNAME}"]** - the number of access share locks. **pgsql.locks.exclusive["{#DBNAME}"]** - the number of exclusive locks. **pgsql.locks.rowexclusive["{#DBNAME}"]** - the number of row exclusive locks. **pgsql.locks.rowshare["{#DBNAME}"]** - the number of row share locks. **pgsql.locks.share["{#DBNAME}"]** - the number of shared locks. **pgsql.locks.sharerowexclusive["{#DBNAME}"]** - the number of share row exclusive locks.
pgsql.oldest.xid[uri,<username>,<password>,<dbName>]			
Age of the oldest XID.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	
pgsql.ping[uri,<username>,<password>,<dbName>]			

Key			
Tests whether a connection is alive or not.	1 - connection is alive 0 - connection is broken (if there is any error presented including AUTH and configuration issues).	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	
pgsql.queries[uri,<username>,<password>,<dbName>,timePeriod]			
Queries metrics by execution time.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name. **timePeriod** - execution time limit for count of slow queries (must be a positive integer).	Returned data are processed by dependent items: **pgsql.queries.mro.time_max["{#DBNAME}"]** - max maintenance query time. **pgsql.queries.query.time_max["{#DBNAME}"]** - max query time. **pgsql.queries.tx.time_max["{#DBNAME}"]** - max transaction query time. **pgsql.queries.mro.slow_count["{#DBNAME}"]** - slow maintenance query count. **pgsql.queries.query.slow_count["{#DBNAME}"]** - slow query count. **pgsql.queries.tx.slow_count["{#DBNAME}"]** - slow transaction query count. **pgsql.queries.mro.time_sum["{#DBNAME}"]** - sum maintenance query time. **pgsql.queries.query.time_sum["{#DBNAME}"]** - sum query time. **pgsql.queries.tx.time_sum["{#DBNAME}"]** - sum transaction query time. This item is supported since Zabbix 6.0.3
pgsql.replication.count[uri,<username>,<password>]			
The number of standby servers.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	
pgsql.replication.process[uri,<username>,<password>]			
Flush lag, write lag and replay lag per each sender process.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	
pgsql.replication.process.discovery[uri,<username>,<password>]			
Replication process name discovery.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	
pgsql.replication.recovery_role[uri,<username>,<password>]			
Recovery status.	0 - master mode 1 - recovery is still in progress (standby mode)	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	
pgsql.replication.status[uri,<username>,<password>]			

Key			
The status of replication.	0 - streaming is down 1 - streaming is up 2 - master mode	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	
pgsql.replication_lag.b[uri,<username>,<password>]			
Replication lag in bytes.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	
pgsql.replication_lag.sec[uri,<username>,<password>]			
Replication lag in seconds.	Integer	**uri** - URI or session name. **username, password** - PostgreSQL credentials.	
pgsql.uptime[uri,<username>,<password>,<dbName>]			
PostgreSQL uptime in milliseconds.	Float	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	
pgsql.wal.stat[uri,<username>,<password>,<dbName>]			
WAL statistics.	JSON object	**uri** - URI or session name. **username, password** - PostgreSQL credentials. **dbName** - Database name.	Returned data are processed by dependent items: **pgsql.wal.count** — the number of WAL files. **pgsql.wal.write** - the WAL lsn used (in bytes).

Redis

Key			
Description	**Return value**	**Parameters**	**Comments**
redis.config[connString,<password>,<pattern>]			
Gets the configuration parameters of a Redis instance that match the pattern.	JSON - if a glob-style pattern was used single value - if a pattern did not contain any wildcard character	**connString** - URI or session name. **password** - Redis password. **pattern** - glob-style pattern (* by default).	
redis.info[connString,<password>,<section>]			
Gets the output of the INFO command.	JSON - output is serialized as JSON	**connString** - URI or session name. **password** - Redis password. **section** - section of information (*default* by default).	
redis.ping[connString,<password>]			

Key		
Test if a connection is alive or not.	1 - connection is alive 0 - connection is broken (if there is any error presented including AUTH and configuration issues)	**connString** - URI or session name. **password** - Redis password.
redis.slowlog.count[connString,<password>]		
The number of slow log entries since Redis was started.	Integer	**connString** - URI or session name. **password** - Redis password.

S.M.A.R.T.

Key			
Description	**Return value**	**Parameters**	**Comments**
smart.attribute.discovery			
Returns a list of S.M.A.R.T. device attributes.	JSON object		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.	JSON object		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, {#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.	JSON object	**path** (since Zabbix 6.0.4) - disk path, the {#PATH} macro may be used as a value **raid_type** (since Zabbix 6.0.4) - RAID type, the {#RAID} macro may be used as a value	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

Key

Description	Return value	Parameters	Comments
systemd.unit.get[unit name,<interface>]			
Returns all properties of a systemd unit.	JSON object	**unit name** - unit name (you may want to use the {#UNIT.NAME} macro in item prototype to discover the name) **interface** - unit interface type, possible values: *Unit* (default), *Service*, *Socket*, *Device*, *Mount*, *Automount*, *Swap*, *Target*, *Path*	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":"active"}`
systemd.unit.info[unit name,<property>,<interface>]			
Systemd unit information.	String	**unit name** - 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** - unit interface type (e.g. Unit (default), Socket, Service)	This item allows to retrieve a specific property from specific type of interface as described in dbus API. This item is supported on Linux platform only. Examples: => systemd.unit.info[”{#UNIT.NAME}”] - collect active state (active, reloading, inactive, failed, activating, deactivating) info on discovered systemd units => sys-temd.unit.info[”{#UNIT.NAME}”,LoadState] - collect load state info on discovered systemd units => systemd.unit.info[mysqld.service,Id] - retrieve service technical name (mysqld.service) => sys-temd.unit.info[mysqld.service,Description] - retrieve service description (MySQL Server) => sys-temd.unit.info[mysqld.service,ActiveEnterTimestamp] - retrieve the last time the service entered the active state (1562565036283903) => sys-temd.unit.info[dbus.socket,NConnections,Socket] - collect the number of connections from this socket unit
systemd.unit.discovery[<type>]			
List of systemd units and their details. Used for low-level discovery.	JSON object	**type** - possible values: *all*, *automount*, *device*, *mount*, *path*, *service* (default), *socket*, *swap*, *target*	This item is supported on Linux platform only.

Web certificate

Key

Description	Return value	Parameters	Comments
web.certificate.get[hostname,<port>,<address>]			

Key			
Validates certificates and returns certificate details.	JSON object	**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** - 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.	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 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.

SNMP checks are performed over the UDP protocol only.

Zabbix server and proxy daemons query SNMP devices for multiple values in a single request. This affects all kinds of SNMP items (regular SNMP items, SNMP items with dynamic indexes, and SNMP low-level discovery) and should make SNMP processing much more efficient. See the bulk processing section for technical details on how it works internally. Bulk requests can also be disabled for devices that cannot handle them properly using the "Use bulk requests" setting for each interface.

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 bulk 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 bulk 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::ifInOctets.3` string from this line:

```
IF-MIB::ifInOctets.3 = Counter32: 3409739121
```

You may now use the **snmpget** command to find out the numeric OID for 'IF-MIB::ifInOctets.3':

```
shell> snmpget -v 2c -c public -On <host IP> IF-MIB::ifInOctets.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:

```
.1.3.6.1.2.1.2.2.1.10.3 = Counter32: 3472126941
```

Again, the last number in the OID is the port number.

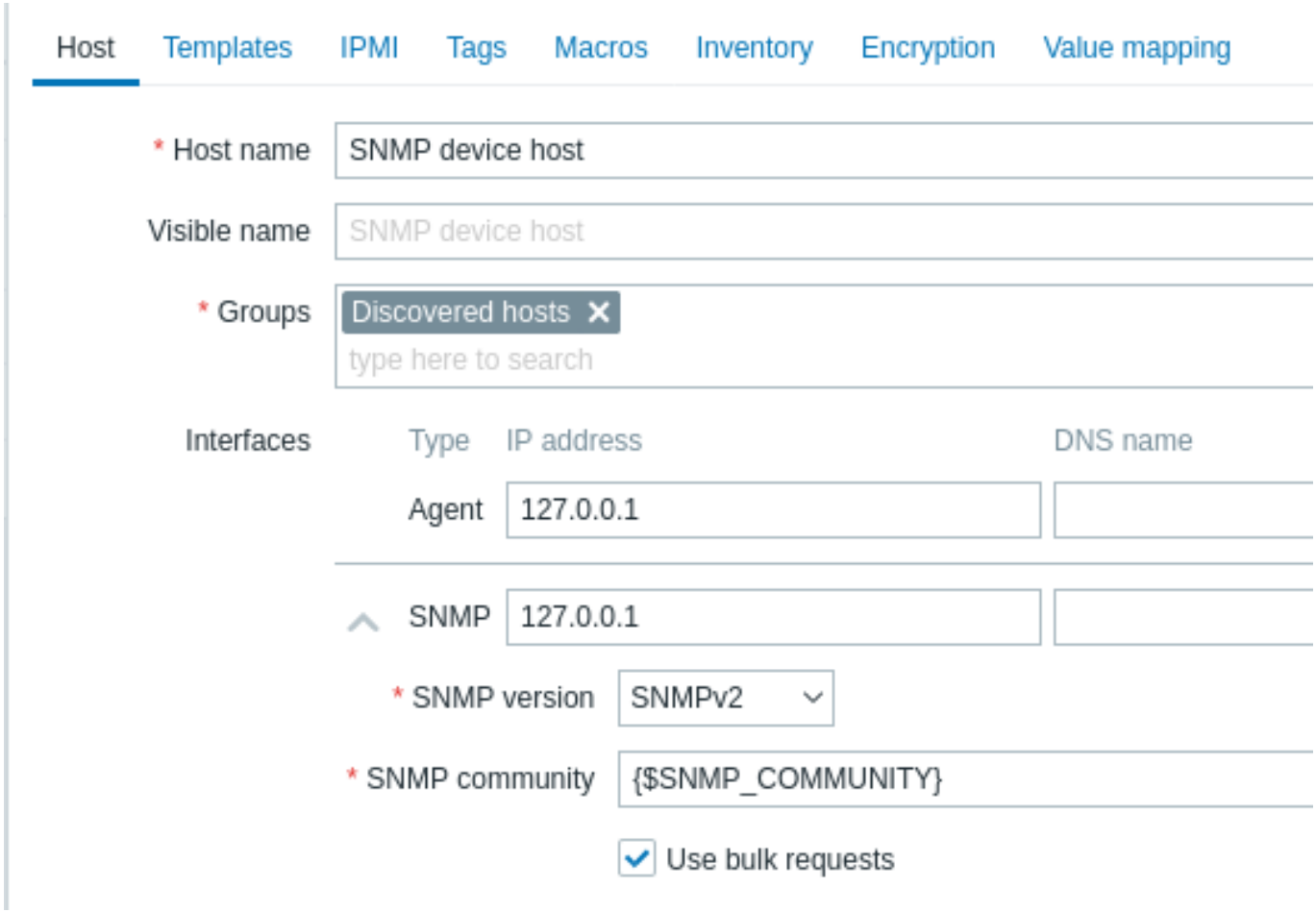
Note:
3COM seem to use port numbers in the hundreds, e.g. port 1 = port 101, port 3 = port 103, but Cisco use regular numbers, e.g. port 3 = 3.

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 "Counter32", which internally corresponds to ASN_COUNTER 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 (since 2.2.8, 2.4.3). 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)
- Leave the *Use bulk requests* checkbox marked to allow bulk processing of SNMP 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).
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;
- (since Zabbix 6.2.7) 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.

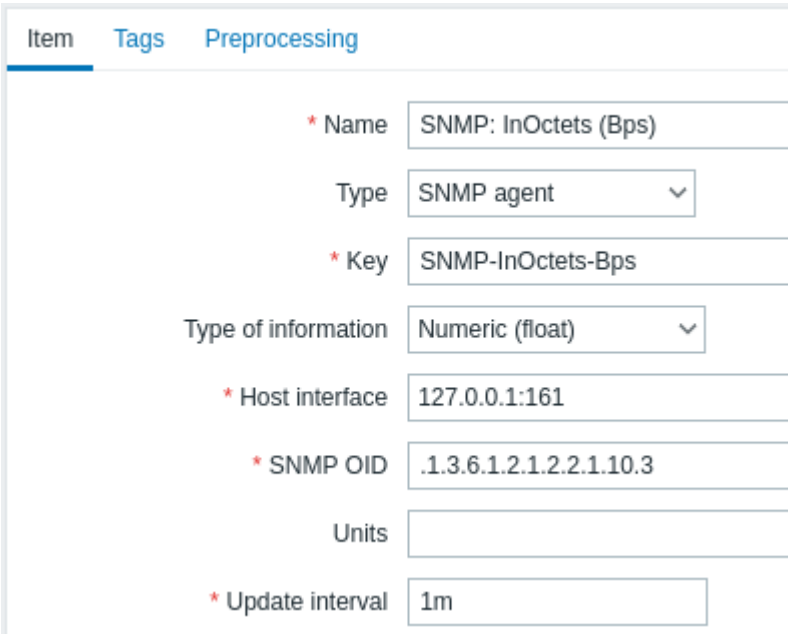
You can use one of the provided SNMP templates (*Template SNMP Device* and others) that will automatically add a set of items. However, the template may not be compatible with the host. Click on *Add* to save the host.

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*. In the new item form:

- Enter the item name
- Change the 'Type' field to 'SNMP agent'
- Enter the 'Key' as something meaningful, e.g. SNMP-InOctets-Bps
- Make sure the 'Host interface' field has your switch/router in it
- Enter the textual or numeric OID that you retrieved earlier into the 'SNMP OID' field, for example: .1.3.6.1.2.1.2.2.1.10.3
- Set the 'Type of information' to *Numeric (float)*
- Enter an 'Update interval' and 'History storage' period if you want them to be different from the default
- In the *Preprocessing* tab, add a *Change per second* step (important, otherwise you will get cumulative values from the SNMP device instead of the latest change). Choose a custom multiplier if you want one.



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
```

Monitoring of SNMP parameters is possible if --with-net-snmp flag was specified while configuring Zabbix sources.

Example 2

Monitoring of uptime:

Parameter	Description
OID	MIB::sysUpTime.0
Key	router.uptime
Value type	Float
Units	uptime
Multiplier	0.01

Internal workings of bulk processing

Starting from 2.2.3 Zabbix server and proxy 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 bulk 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 bulk requests properly for other reasons and the heuristic described above does not work, since Zabbix 2.4 there is a "Use bulk requests" setting for each interface that allows to disable bulk 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.
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 a Management Information Base. MIB files allow you to use textual representation of the OID (Object Identifier).

For example,

`ifHCOutOctets`

is textual representation of OID

`1.3.6.1.2.1.31.1.1.1.10`

You can use either, when monitoring SNMP devices with Zabbix, but if you feel more comfortable when using textual representation you have to install MIB files.

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 the mib files should be enabled by default. On Debian-based systems you have to edit 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.:

```
# service zabbix-server restart
```

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 *Configuration* → *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		

Key		
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 = [DATE TIME FORMAT]
```

Warning:
The "net-snmp-perl" package has been removed in RHEL/CentOS 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/6.2/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 line 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 0x800000000102030
```

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 10:54:432022-08-05 10:54:41

20220805.105441 UDP: [127.0.0.1]:44262->[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, get a copy of IPMI 2.0 specifications at http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-specifications.html (At the time of writing the newest document was http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/second-gen-interface-spec-v2.pdf)

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 simple checks

List of supported simple checks:

See also:

- VMware monitoring item keys

Key			
Description	**Return value**	**Parameters**	**Comments**
icmpping[<target>,<packets>,<interval>,<size>,<timeout>]			
Host accessibility by ICMP ping.	0 - ICMP ping fails 1 - ICMP ping successful	**target** - host IP or DNS name **packets** - number of packets **interval** - time between successive packets in milliseconds **size** - packet size in bytes **timeout** - timeout in milliseconds	Example: => icmpping[,4] → if at least one packet of the four is returned, the item will return 1. See also: table of default values.
icmppingloss[<target>,<packets>,<interval>,<size>,<timeout>]			
Percentage of lost packets.	Float.	**target** - host IP or DNS name **packets** - number of packets **interval** - time between successive packets in milliseconds **size** - packet size in bytes **timeout** - timeout in milliseconds	See also: table of default values.
icmppingsec[<target>,<packets>,<interval>,<size>,<timeout>,<mode>]			
ICMP ping response time (in seconds).	Float.	**target** - host IP or DNS name **packets** - number of packets **interval** - time between successive packets in milliseconds **size** - packet size in bytes **timeout** - timeout in milliseconds **mode** - possible values: *min*, *max*, *avg* (default)	Packets which are lost or timed out are not used in the calculation. If 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: table of default values.
net.tcp.service[service,<ip>,<port>]			
Checks if service is running and accepting TCP connections.	0 - service is down 1 - service is running	**service** - possible values: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address or DNS name (by default host IP/DNS is used) **port** - port number (by default standard service port number is used).	Example: => net.tcp.service[ftp„45] → can be used to test the availability of FTP server on TCP port 45. 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. *https* and *telnet* services are supported since Zabbix 2.0.
net.tcp.service.perf[service,<ip>,<port>]			

Key			
Checks performance of TCP service.	Float. 0.000000 - service is down seconds - the number of seconds spent while connecting to the service	**service** - possible values: *ssh*, *ldap*, *smtp*, *ftp*, *http*, *pop*, *nntp*, *imap*, *tcp*, *https*, *telnet* (see details) **ip** - IP address or DNS name (by default, host IP/DNS is used) **port** - port number (by default standard service port number is used).	Example: => net.tcp.service.perf[ssh] → can be used to test the speed of initial response from SSH server. 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.perf[tcp,<ip>,port] for checks like these. *https* and *telnet* services are supported since Zabbix 2.0. Called tcp_perf before Zabbix 2.0.
net.udp.service[service,<ip>,<port>]			
Checks if service is running and responding to UDP requests.	0 - service is down 1 - service is running	**service** - possible values: *ntp* (see details) **ip** - IP address or DNS name (by default host IP/DNS is used) **port** - port number (by default standard service port number is used).	Example: => net.udp.service[ntp„45] → can be used to test the availability of NTP service on UDP port 45. This item is supported since Zabbix 3.0, but *ntp* service was available for net.tcp.service[] item in prior versions.
net.udp.service.perf[service,<ip>,<port>]			
Checks performance of UDP service.	Float. 0.000000 - service is down seconds - the number of seconds spent waiting for response from the service	**service** - possible values: *ntp* (see details) **ip** - IP address or DNS name (by default, host IP/DNS is used) **port** - port number (by default standard service port number is used).	Example: => net.udp.service.perf[ntp] → can be used to test response time from NTP service. This item is supported since Zabbix 3.0, but *ntp* service was available for net.tcp.service[] item in prior versions.

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 external utility **fping** for processing of ICMP pings.

The utility is not part of Zabbix distribution and has to be additionally installed. If the utility is missing, has wrong permissions or its location does not match the location set in the Zabbix server/proxy configuration file ('FpingLocation' parameter), ICMP pings (**icmpping**, **icmppingloss**, **icmppingsec**) will not be processed.

See also: known issues

fping must be executable by the user Zabbix daemons run as and setuid root. Run these commands as user **root** in order to set up correct permissions:

```
shell> chown root:zabbix /usr/sbin/fping
shell> chmod 4710 /usr/sbin/fping
```

After performing the two commands above check ownership of the **fping** executable. In some cases the ownership can be reset by executing the chmod command.

Also check, if user zabbix belongs to group zabbix by running:

```
shell> groups zabbix
```

and if it's not add by issuing:

```
shell> usermod -a -G zabbix zabbix
```

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 to a target	-C		3	1	10000
interval	milliseconds	time to wait between successive packets	-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** - inherited from *-p* flag, but not more than 2000		50	unlimited

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). Those 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.

Warning:
Warning: fping defaults can differ depending on platform and version - if in doubt, check fping documentation.

Zabbix writes IP addresses to be checked by any of three *icmpping** keys to a temporary file, which is then passed to **fping**. If items have different key parameters, only 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.

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

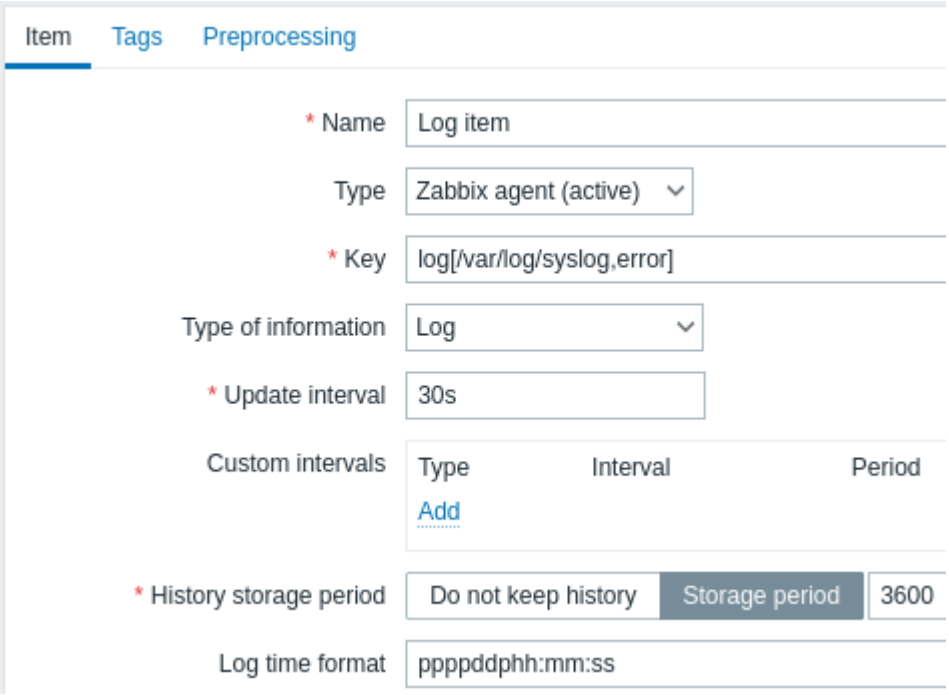
Verify agent parameters

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. If left blank the timestamp will not be parsed. Supported placeholders: * **y**: *Year (0001-9999)* * **M**: *Month (01-12)* * **d**: *Day (01-31)* * **h**: *Hour (00-23)* * **m**: *Minute (00-59)* * **s**: *Second (00-59)* 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 line. Log time format for this line would be "ppppp:yyyyMMdd:hhmmss". Note that "p" and ":" chars are just placeholders and can be anything but "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 receives a non-zero size counter from server, but the logrt[] or logrt.count[] item has not found and does not 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.
- 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.

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.

7 Calculated items

Overview

With calculated items it is possible to create calculations based on the values of other items.

Calculations may use both:

- single values of individual items

- complex filters to select multiple items for aggregations (see aggregate calculations for details)

Thus, 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)*.

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.

Calculation definition should be entered in the **Formula** field. There is virtually 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

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 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

- Parameters without angle brackets are constants - for example, 'host' and 'available' in `zabbix[host,<type>,available]`. Use them in the item key *as is*.
- Values for items and item parameters that are "not supported on proxy" can only be gathered if the host is monitored by server. And vice versa, values "not supported on server" can only be gathered if the host is monitored by proxy.

Key	Description	Return value	Comments
zabbix[boottime]	Startup time of Zabbix server or Zabbix proxy process in seconds.	Integer	
zabbix[cluster,discovery,nodes]	Discover high availability cluster nodes.	JSON	This item can be used in low-level discovery.
zabbix[host,,items]			

Key		
Number of enabled items (supported and not supported) on the host. **zabbix**[host,,items_unsupported]	Integer	This item is supported since Zabbix 3.0.0.
Number of enabled unsupported items on the host. **zabbix**[host,,maintenance]	Integer	This item is supported since Zabbix 3.0.0.*
Current maintenance status of a host. **zabbix**[host,active_agent,available]	0 - host in normal state, 1 - host in maintenance with data collection, 2 - host in maintenance without data collection.	This item is always processed by Zabbix server regardless of 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.
Availability of active agent checks on the host. **zabbix**[host,discovery,interfaces]	0 - unknown, 1 - available, 2 - not available.	

Key		
Details of all configured interfaces of the host in Zabbix frontend. **zabbix**[host,<type>,available]	JSON object.	This item can be used in low-level discovery. This item is supported since Zabbix 3.4.0. *(not supported on proxy)*
Availability of the main interface of a particular type of checks on the host. **zabbix**[hosts]	0 - not available, 1 - available, 2 - unknown.	Valid **types** are: *agent*, *snmp*, *ipmi*, *jmx* The item value is calculated according to configuration parameters regarding host unreachability/unavailability. This item is supported since Zabbix 2.0.0.
Number of monitored hosts. **zabbix**[items]	Integer	
Number of enabled items (supported and not supported). **zabbix**[items_unsupported]	Integer	

Key		
Number of not supported items.	Integer	
zabbix[java,„<param>]		
Information about Zabbix Java gateway.	If <param> is *ping*, "1" is returned. Can be used to check Java gateway availability using nodata() trigger function. If <param> is *version*, version of Java gateway is returned. Example: "2.0.0".	Valid values for **param** are: *ping*, *version* Second parameter must be empty and is reserved for future use.
zabbix[lld_queue]		
Count of values enqueued in the low-level discovery processing queue.	Integer	This item can be used to monitor the low-level discovery processing queue length. This item is supported since Zabbix 4.2.0.
zabbix[preprocessing_queue]		
Count of values enqueued in the preprocessing queue.	Integer	This item can be used to monitor the preprocessing queue length. This item is supported since Zabbix 3.4.0.
zabbix[process,<type>,<mode>,<state>]		

Key		
Time a particular Zabbix process or a group of processes (identified by <type> and <mode>) spent in <state> in percentage. It is calculated for the last minute only. If <mode> is Zabbix process number that is not running (for example, with 5 pollers running <mode>	Percentage of time. Float	Supported **types** of server processes: *alert manager*, *alert syncer*, *alerter*, *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*, *snmp trapper*, *task manager*, *timer*, *trapper*, *unreachable poller*, *vmware collector* Supported **types** of proxy processes: *availability manager*, *configuration syncer*, *data sender*, *discoverer*, *heartbeat sender*, *history syncer*, *housekeeper*, *http poller*, *icmp pinger*, *ipmi manager*, *ipmi poller*, *java poller*, *odbs poller*, *poller*, *preprocessing manager*, *preprocessing worker*, *self-monitoring*, *snmp trapper*, *task manager*, *trapper*, *unreachable poller*, *vmware collector* Valid **modes** are: *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. Valid **states** are: *busy* - process is in busy state, for example, processing request (default). *idle* - process is in idle state doing nothing. Examples: => zabbix[process,poller,avg,busy] → average time of poller processes spent doing something during the last minute => zabbix[process,”icmp pinger”,max,busy] → maximum time spent doing something by any ICMP pinger process during the last minute => zabbix[process,”history syncer”,2,busy] → time spent doing something by history syncer number 2 during the last minute => zabbix[process,trapper,count] → amount of currently running trapper processes

Key		
zabbix[proxy,<name>,<param>]		
Information about Zabbix proxy.	Integer	**name**: proxy name Valid values for **param** are: *lastaccess* - timestamp of last heart beat message received from proxy *delay* - how long collected values are unsent, calculated as ”proxy delay” (difference between the current proxy time and the timestamp of the oldest unsent value on proxy) + (”current server time” - ”proxy lastaccess”) Example: => zabbix[proxy,”Germany”,lastaccess] `fuzzytime()` function can be used to check availability of proxies. This item is always processed by Zabbix server regardless of host location (on server or proxy).
zabbix[proxy_history]		
Number of values in the proxy history table waiting to be sent to the server.	Integer	*(not supported on server)*
zabbix[queue,<from>,<to>]		

Key		
Number of monitored items in the queue which are delayed at least by <from> seconds but less than by <to> seconds.	Integer	**from** - default: 6 seconds **to** - default: infinity Time-unit symbols (s,m,h,d,w) are supported for these parameters.
zabbix[rcache,<cache>,<mode>]		
Availability statistics of Zabbix configuration cache.	Integer (for size); float (for percentage).	**cache**: *buffer* Valid **modes** are: *total* - total size of buffer *free* - size of free buffer *pfree* - percentage of free buffer *used* - size of used buffer *pused* - percentage of used buffer *pused* mode is supported since Zabbix 4.0.0.
zabbix[requiredperformance]		
Required performance of Zabbix server or Zabbix proxy, in new values per second expected.	Float	Approximately correlates with "Required server performance, new values per second" in *Reports* → *System information*.

Key		
zabbix[stats,<ip>,<port>]		
Remote Zabbix server or proxy internal metrics.	JSON object.	**ip** - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1) **port** - port of server/proxy to be remotely queried (default is 10051) Note that 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. Supported since 4.2.0.
zabbix[stats,<ip>,<port>,queue,<from>,<to>]		
Remote Zabbix server or proxy internal queue metrics (see zabbix[queue,<from>,<to>]).	JSON object.	**ip** - IP/DNS/network mask list of servers/proxies to be remotely queried (default is 127.0.0.1) **port** - 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) Note that the stats request will only be accepted from the addresses listed in the 'StatsAllowedIP' server/proxy parameter on the target instance. Supported since 4.2.0.
zabbix[tcache,cache,<parameter>]		
Effectiveness statistics of the Zabbix trend function cache.	Integer (for size); float (for percentage).	Valid **parameters** are: *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. Supported since 5.4.0. *(not supported on proxy)*
zabbix[triggers]		

Key			
Number of enabled triggers in Zabbix database, with all items enabled on enabled hosts.	Integer	*(not supported on proxy)*	
zabbix[uptime]			
Uptime of Zabbix server or Zabbix proxy process in seconds.		Integer	
zabbix[vcache,buffer,<mode>]			
Availability statistics of Zabbix value cache.		Integer (for size); float (for percentage).	Valid **modes** are: *total* - total size of buffer *free* - size of free buffer *pfree* - percentage of free buffer *used* - size of used buffer *pused* - percentage of used buffer *(not supported on proxy)*
zabbix[vcache,cache,<parameter>]			
Effectiveness statistics of Zabbix value cache.		Integer With the *mode* parameter: 0 - normal mode, 1 - low memory mode	Valid **parameter** values are: *requests* - total number of requests *hits* - number of cache hits (history values taken from the cache) *misses* - number of cache misses (history values taken from the database) *mode* - value cache operating mode This item is supported since Zabbix 2.2.0 and the *mode* parameter since Zabbix 3.0.0. *(not supported on proxy)* 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.

Key				
zabbix[version]				
Version of Zabbix server or proxy.			String	This item is supported since Zabbix 5.0.0. Example of return value: 5.0.0beta1
zabbix[vmware,buffer,<mode>]				
Availability statistics of Zabbix vmware cache.			Integer (for size); float (for percentage).	Valid **modes** are: *total* - total size of buffer *free* - size of free buffer *pfree* - percentage of free buffer *used* - size of used buffer *pused* - percentage of used buffer
zabbix[wcache,<cache>,<mode>]				
Statistics and availability of Zabbix write cache.				Specifying <cache> is mandatory.
	Cache values	**Mode**		
	all *(default)*	Total number of values processed by Zabbix server or Zabbix proxy, except unsupported items.	Integer	Counter. You may use this key with the *Change per second* preprocessing step in order to get values per second statistics.
	float	Number of processed float values.	Integer	Counter.
	uint	Number of processed unsigned integer values.	Integer	Counter.
	str	Number of processed character/string values.	Integer	Counter.
	log	Number of processed log values.	Integer	Counter.

Key				
	text	Number of processed text values.	Integer	Counter.
	not supported	Number of times item processing resulted in item becoming unsupported or keeping that state.	Integer	Counter.
history	pfree *(default)*	Percentage of free history buffer.	Float	History cache is used to store item values. A low number indicates performance problems on the database side.
	free	Size of free history buffer.	Integer	
	total	Total size of history buffer.	Integer	
	used	Size of used history buffer.	Integer	
	pused	Percentage of used history buffer.	Float	*pused* mode is supported since Zabbix 4.0.0.
index	pfree *(default)*	Percentage of free history index buffer.	Float	History index cache is used to index values stored in history cache. *Index* cache is supported since Zabbix 3.0.0.
	free	Size of free history index history buffer.	Integer	
	total	Total size of history index history buffer.	Integer	
	used	Size of used history index history buffer.	Integer	
	pused	Percentage of used history index buffer.	Float	*pused* mode is supported since Zabbix 4.0.0.

Key				
trend[pfree]	Percentage of free trend cache.	Float	Trend cache stores aggregate for the current hour for all items that receive data.	
(default)			*(not supported on proxy)*	
free	Size of free trend buffer.	Integer	*(not supported on proxy)*	
total	Total size of trend buffer.	Integer	*(not supported on proxy)*	
used	Size of used trend buffer.	Integer	*(not supported on proxy)*	
pused	Percentage of used trend buffer.	Float	*(not supported on proxy)*	
			pused mode is supported since Zabbix 4.0.0.	

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 (libssh2 or libssh). See also: Requirements.

Attention:
Only libssh is supported starting with RHEL/CentOS 8.

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 libssh2/libssh 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 full path to a folder where public and private keys will be located:

```
SSHKeyLocation=/home/zabbix/.ssh
```

Save the file and restart zabbix_server afterwards.

/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 a ssh-keygen command inside the home directory.

Usually installation packages of zabbix-server from different OS distributions create *zabbix* user account with a home directory in not very well-known places (as for system accounts), e. g. */var/lib/zabbix*.

Before starting to generate the keys, an approach to reallocate the home directory to a better known place (intuitively expected) could be considered. This will correspond with the *SSHKeyLocation* Zabbix server configuration parameter mentioned above.

These steps can be skipped if *zabbix* account has been added manually according to the installation section because in this case most likely the home directory is already located at */home/zabbix*.

To change the setting for the *zabbix* user account all working processes which are using it have to be stopped:

```
# service zabbix-agent stop
# service zabbix-server stop
```

To change the home directory location with an attempt to move it (if it exists) a command should be executed:

```
# usermod -m -d /home/zabbix zabbix
```

It's absolutely possible that a home directory did not exist in the old place (in the CentOS for example), so it should be created at the new place. 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 now can be started again:

```
# service zabbix-agent start
# service zabbix-server start
```

Now steps to generate public and private keys can be performed by a command:

```
##### 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):
##### <enter the created directory '/home/zabbix/.ssh'>
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
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: public and private keys (*id_rsa.pub* and *id_rsa* respectively) 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 command the **public** key file can be installed on a remote host *10.10.10.10* so that then SSH checks can be performed with a *root* account:

```
##### 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:
Now try logging into the machine, with "ssh 'root@10.10.10.10'", and check in:
    .ssh/authorized_keys
to make sure we haven't added extra keys that you weren't expecting.
```

Now it's possible to check the SSH login using the default private key (*/home/zabbix/.ssh/id_rsa*) for *zabbix* user account:

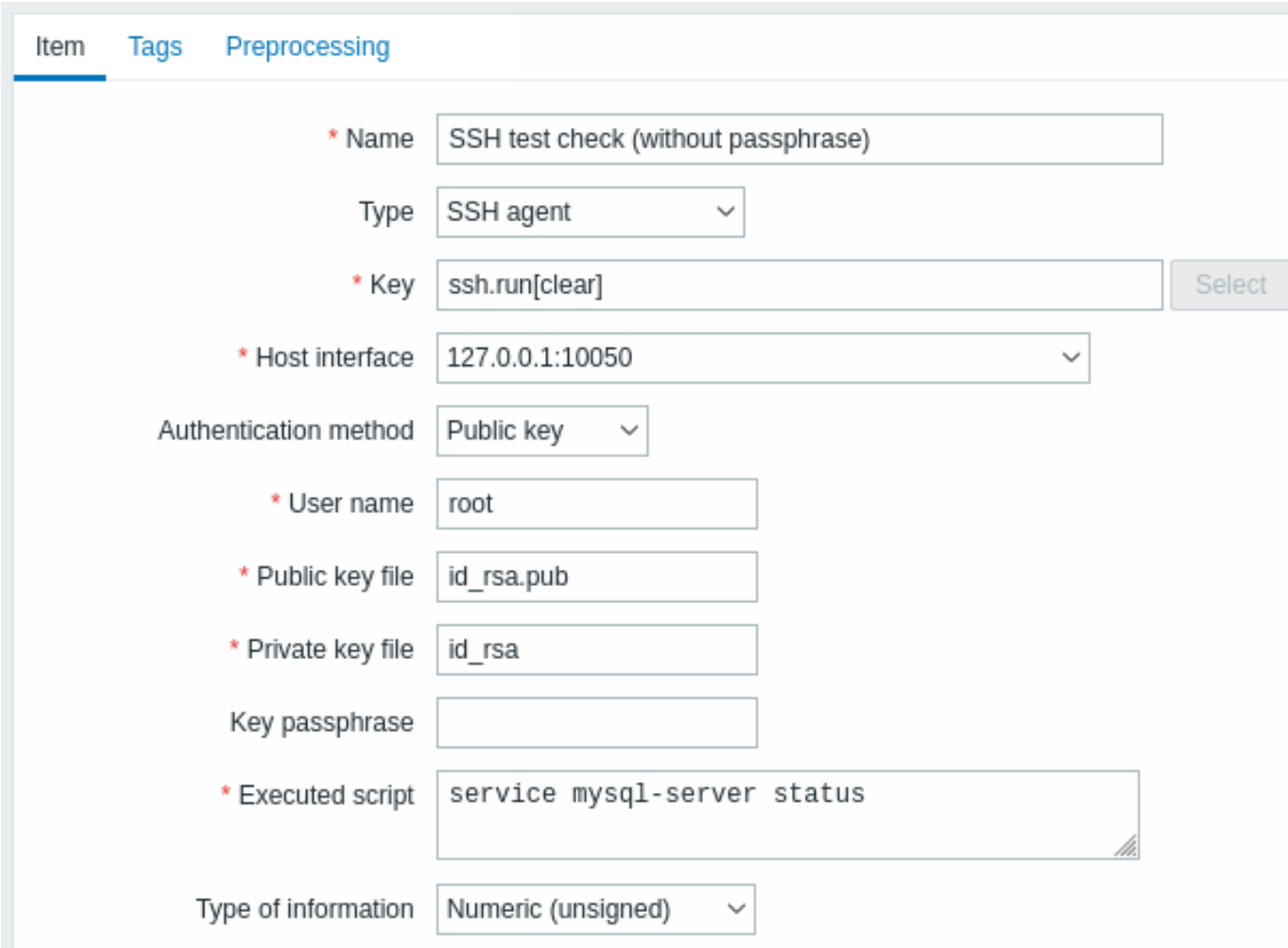
```
# sudo -u zabbix ssh root@10.10.10.10
```

If the login is successful, then the configuration part in the shell is finished and 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 also will 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.	
Key	Unique (per host) item key in format **ssh.run[<unique short description>,<ip>,<port>,<encoding>]**	<unique short description> is required and should be unique for all SSH items per host Default port is 22, not the port specified in the interface to which this item is assigned
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

Parameter	Description	Comments
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	Examples: *date +%s* *service mysql-server status* *ps auxww | grep httpd | wc -l*

Attention:
libssh2 library may truncate executable scripts to ~32kB.

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 will look in the directory defined as the location for external scripts (parameter 'ExternalScripts' in Zabbix server configuration file) and execute the command. The command will be executed as the user Zabbix server runs as, so any access permissions or environment variables should be handled in a wrapper script, if necessary, and permissions on the command should 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, running many scripts can decrease Zabbix performance a lot.

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 will execute:

```
check_oracle.sh '-h' '192.168.1.4'
```

External check result

The return value of the check is standard output together with standard error (the full output with trimmed trailing whitespace is returned since Zabbix 2.0).

Attention:
A text (character, log or text type of information) item will not become unsupported in case of standard error output.

In case the requested script is not found or Zabbix server has no permissions to execute it, the item will become unsupported and corresponding error message will be set. In case of a timeout, the item will be marked as unsupported as well, an according error message will be displayed and the forked process for the script will be killed.

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:

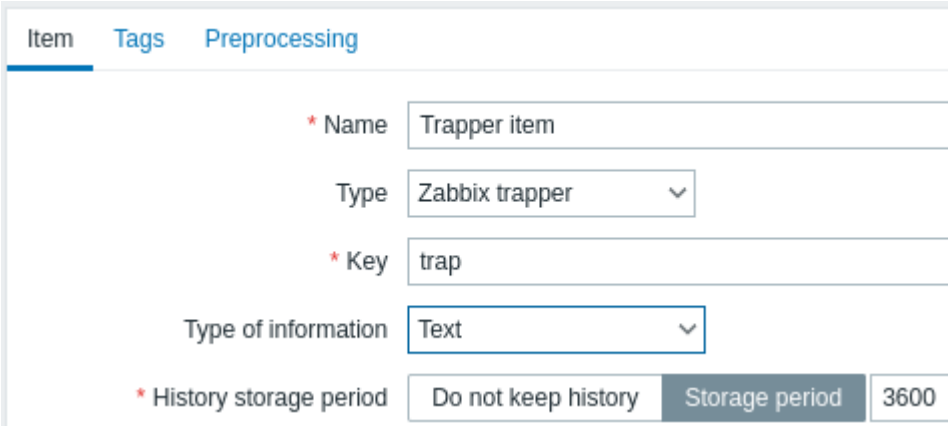
- have a trapper item set up in Zabbix
- send in the data into Zabbix

Configuration

Item configuration

To configure a trapper item:

- Go to: *Configuration* → *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 hostnames. 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: Server=127.0.0.1, 192.168.1.0/24, 192.168.3.1-255, 192.168.1-10.1-255, ::1,2001:db8::/32, zabbix.domain 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)

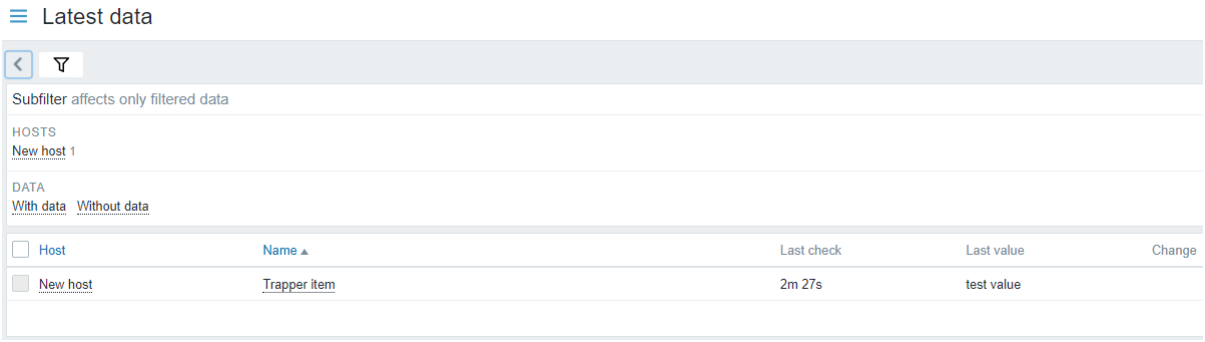
-k - to specify the key of the item we just defined

-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.

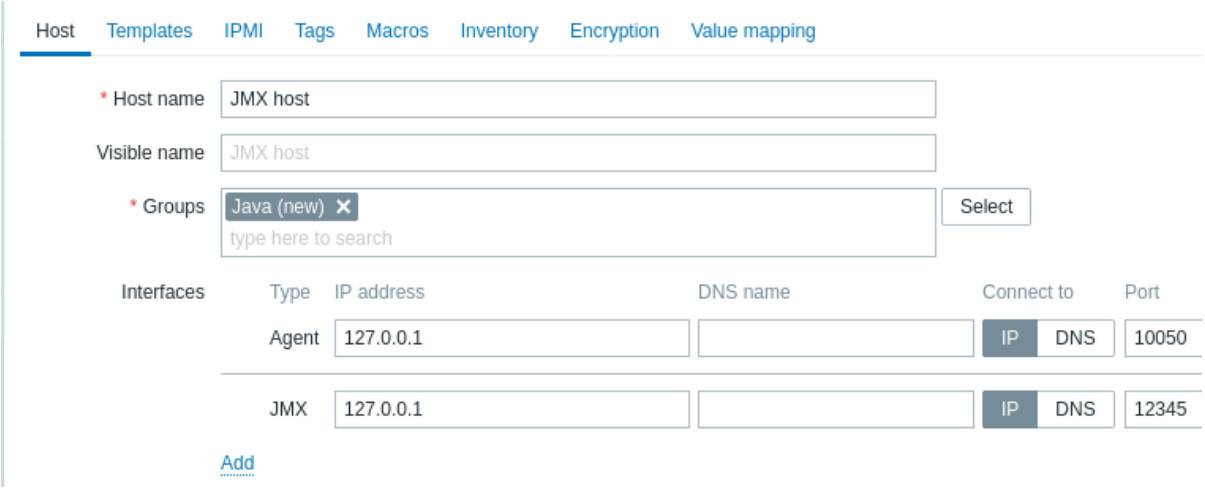
See Monitoring and Management Using JMX for a detailed description.

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.

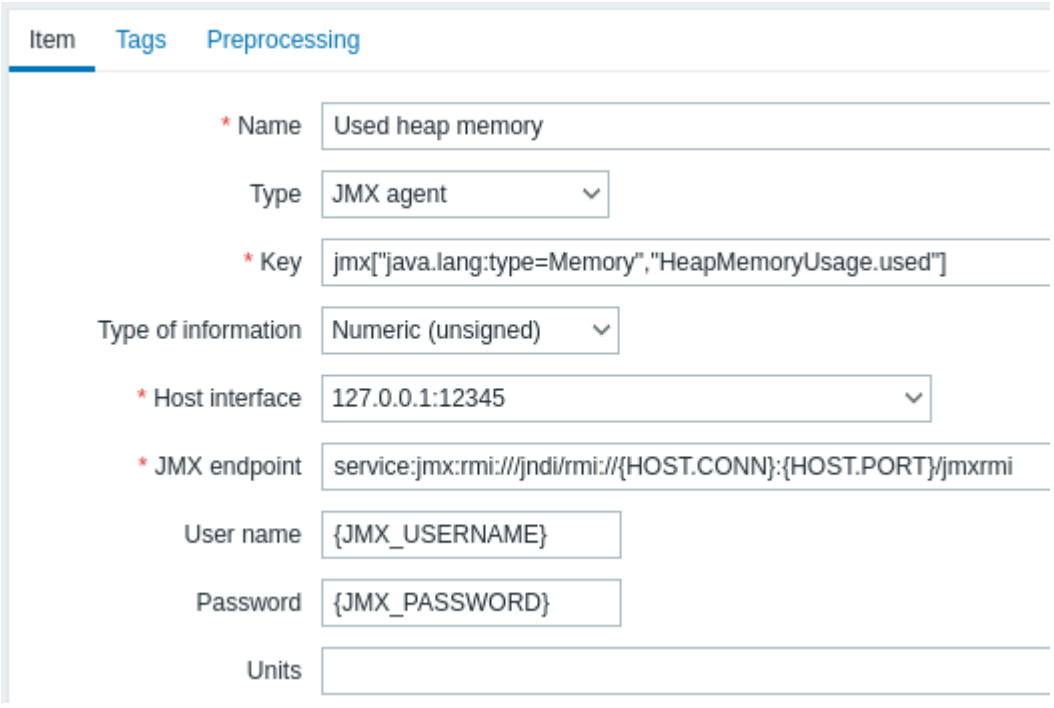


All mandatory input fields are marked with a red asterisk.

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 an object name is "com.example:Type=Hello", attribute name is "weight" and probably the returned value type should 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
```

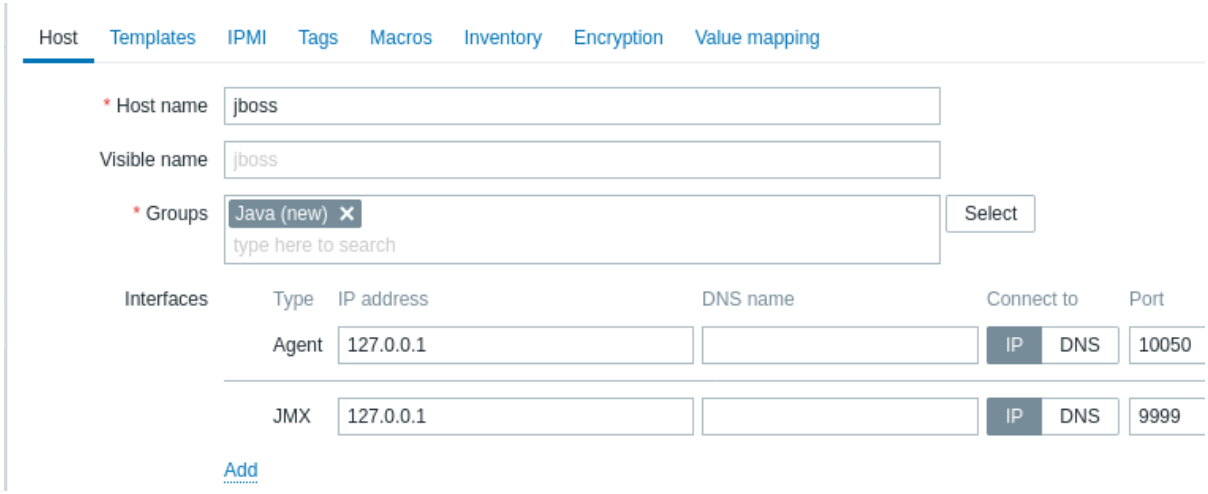
And in the zabbix_java/settings.sh configuration file (or zabbix_java_gateway.conf):

```
START_POLLERS=5
```

Check that JBoss listens to its standard management port:

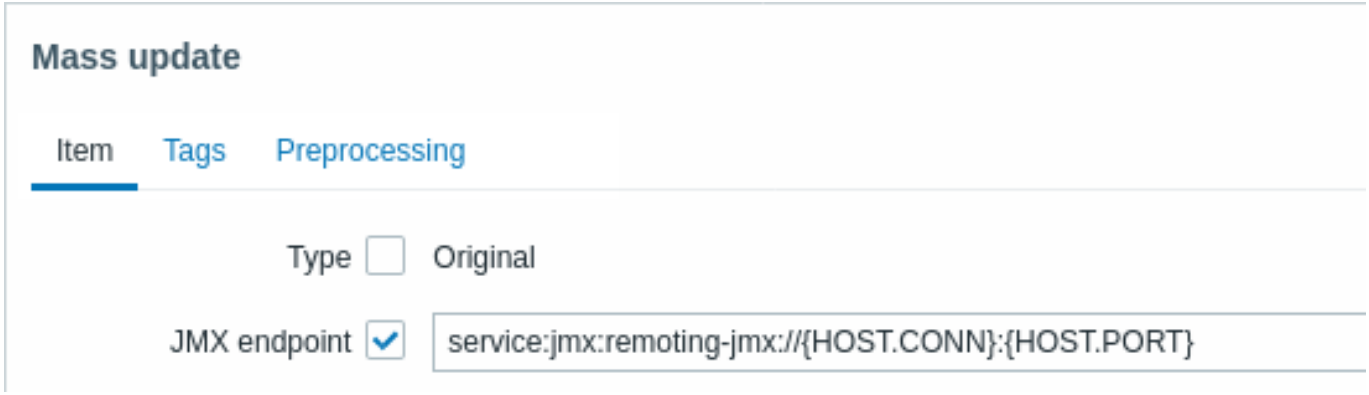
```
$ netstat -natp | grep 9999
tcp        0      0 127.0.0.1:9999          0.0.0.0:*               LISTEN      10148/java
```

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</pre>
```

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/local
```

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 RedHat/Fedora based systems using the *yum* package manager:

```
shell> dnf -y install unixODBC unixODBC-devel
```

Installing unixODBC on SUSE based systems using the *zypper* package manager:

```
# zypper in unixODBC-devel
```

> **Note:**
> The 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 RedHat/Fedora based systems using the *yum* package manager:

```
shell> dnf install mysql-connector-odbc
```

Installing MySQL database driver on SUSE based systems using the *zypper* package manager:

```
zypper in MyODBC-unixODBC
```

Configuring unixODBC

ODBC configuration is done by editing the **odbcinst.ini** and **odbc.ini** files. To verify the configuration file location, type:

```
shell> 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.
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):

```
shell> isql test
+---------------------------------------+
| Connected!                            |
|                                       |
| sql-statement                         |
| help [tablename]                      |
| quit                                  |
|                                       |
+---------------------------------------+
SQL>
```

Compiling Zabbix with ODBC support

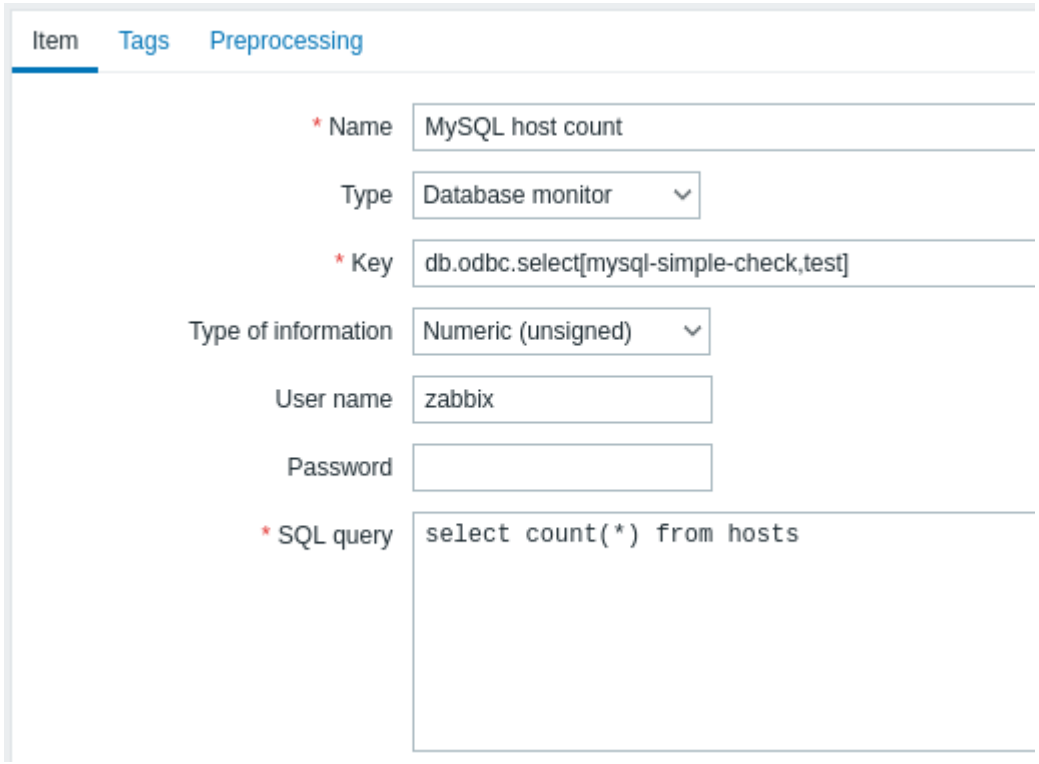
To enable ODBC support, Zabbix should be compiled with the following flag:

```
  --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>` If a password contains semicolon, it should be wrapped in curly brackets, for example: *Password*: `{P?;)*word}` (if an actual password is `P?;)*word`) The password will be appended to connection string after the username as: `UID=<username>;PWD={P?;)*word}` To test the resulting string, run: `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:

```
Cannot execute ODBC query: [SQL_ERROR]:[42601][7][ERROR: syntax error at or near ";"; Error while executin

                                                Native error code              Native error message
                                          SQLState
                Zabbix message     ODBC return code
```

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/CentOS**:

# dnf install mysql-connector-odbc

***Debian/Ubuntu**:
```

Please refer to MySQL 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 **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/CentOS**:

```
# 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 = Yes
# Send tobackend 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/CentOS**:

```
# dnf -y install freetds unixODBC
```

- **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.

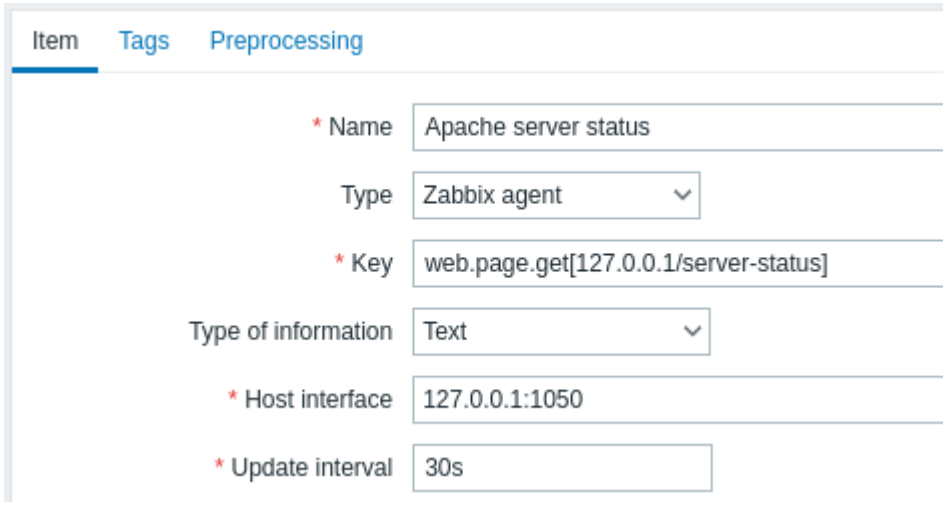
Limitations

- 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:

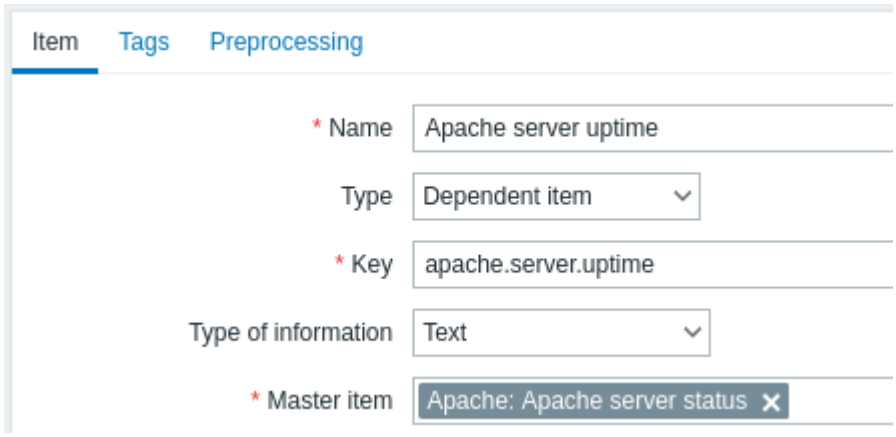
- Go to: *Configuration* → *Hosts*
- Click on *Items* in the row of the host
- Click on *Create item*
- Enter parameters of the item in the form



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**.

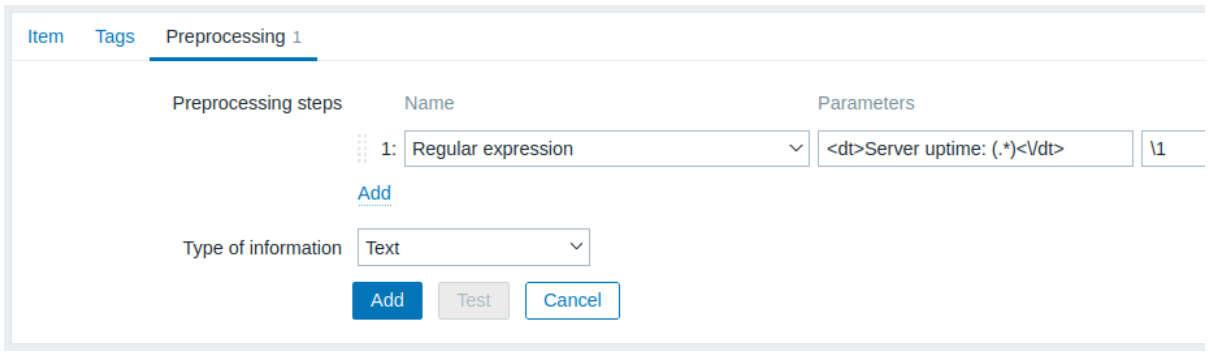


All mandatory input fields are marked with a red asterisk.

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.

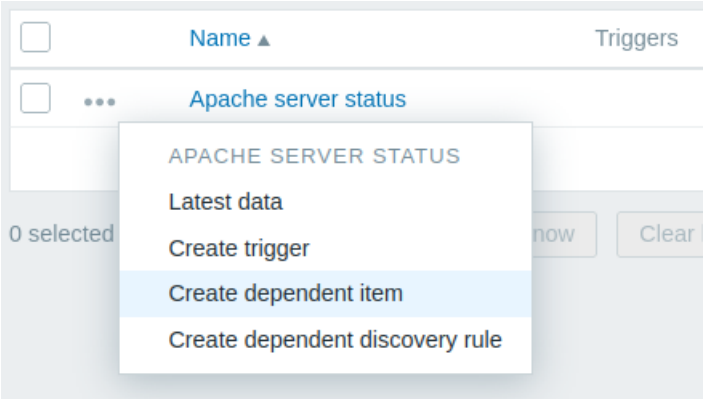
You may use item value preprocessing to extract the required part of the master item value.



Without preprocessing, the dependent item value will be exactly the same as the master item value.

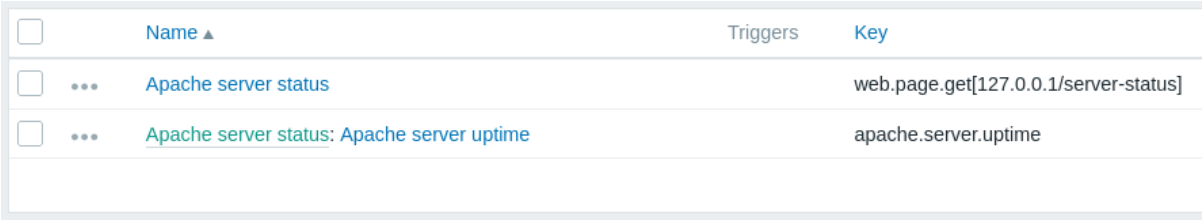
Click on *Add* to save the dependent item.

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).

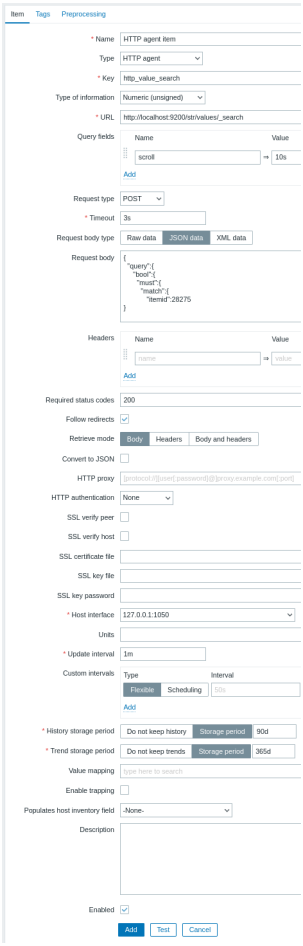
See also known issues for when using HTTPS protocol.

> **Attention:**
> Zabbix server/proxy must be initially configured with cURL (libcurl) support.

Configuration

To configure an HTTP item:

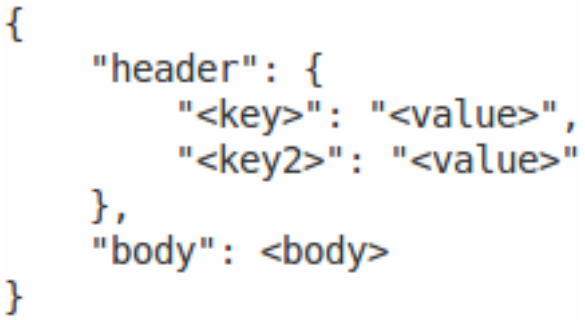
- Go to: *Configuration* → *Hosts*
- Click on *Items* in the row of the host
- Click on *Create item*
- Enter parameters of the item in the form



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 hostnames. 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: Server=127.0.0.1, 192.168.1.0/24, 192.168.3.1-255, 192.168.1-10.1-255, ::1,2001:db8::/32, zabbix.domain 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:

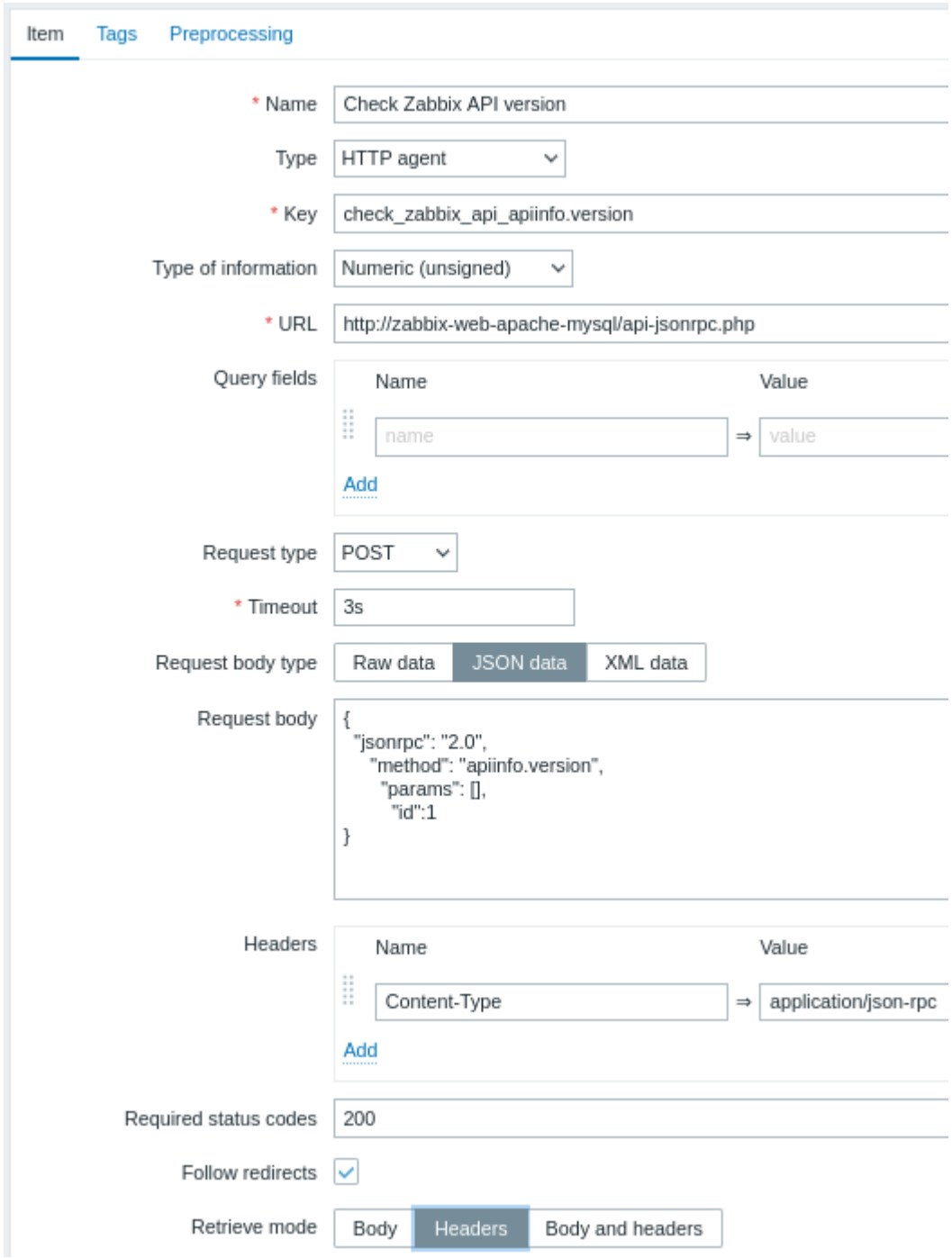

```
{
    "_scroll_id": "DnF1ZXJ5VGhlbkZldGNoBQAAAAAAAAAkFllRMlZBWS1UU1pxTmdEeGVwQjRBTFEAAAAAAAAAAJRZZUTJWQVktVFN
    "took": 18,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 1,
        "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

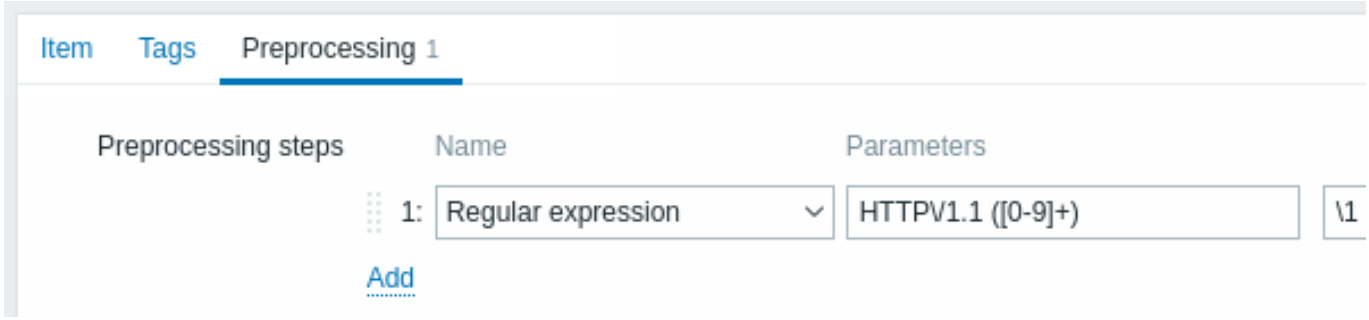
Checking if Zabbix API is alive, using apiinfo.version.

- Item configuration:

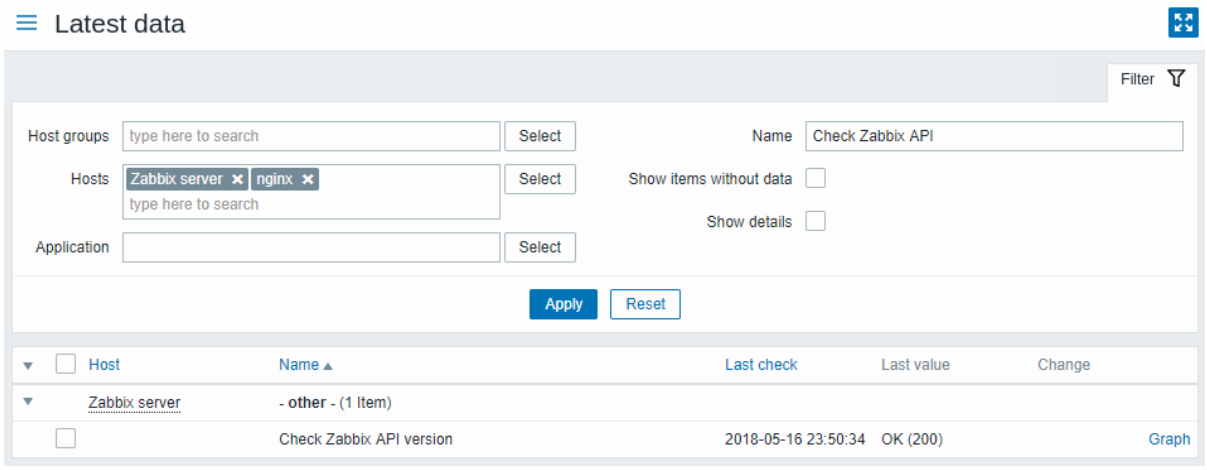


Note the use of the POST method with JSON data, setting request headers and asking to return headers only:

- Item value preprocessing with regular expression to get HTTP code:



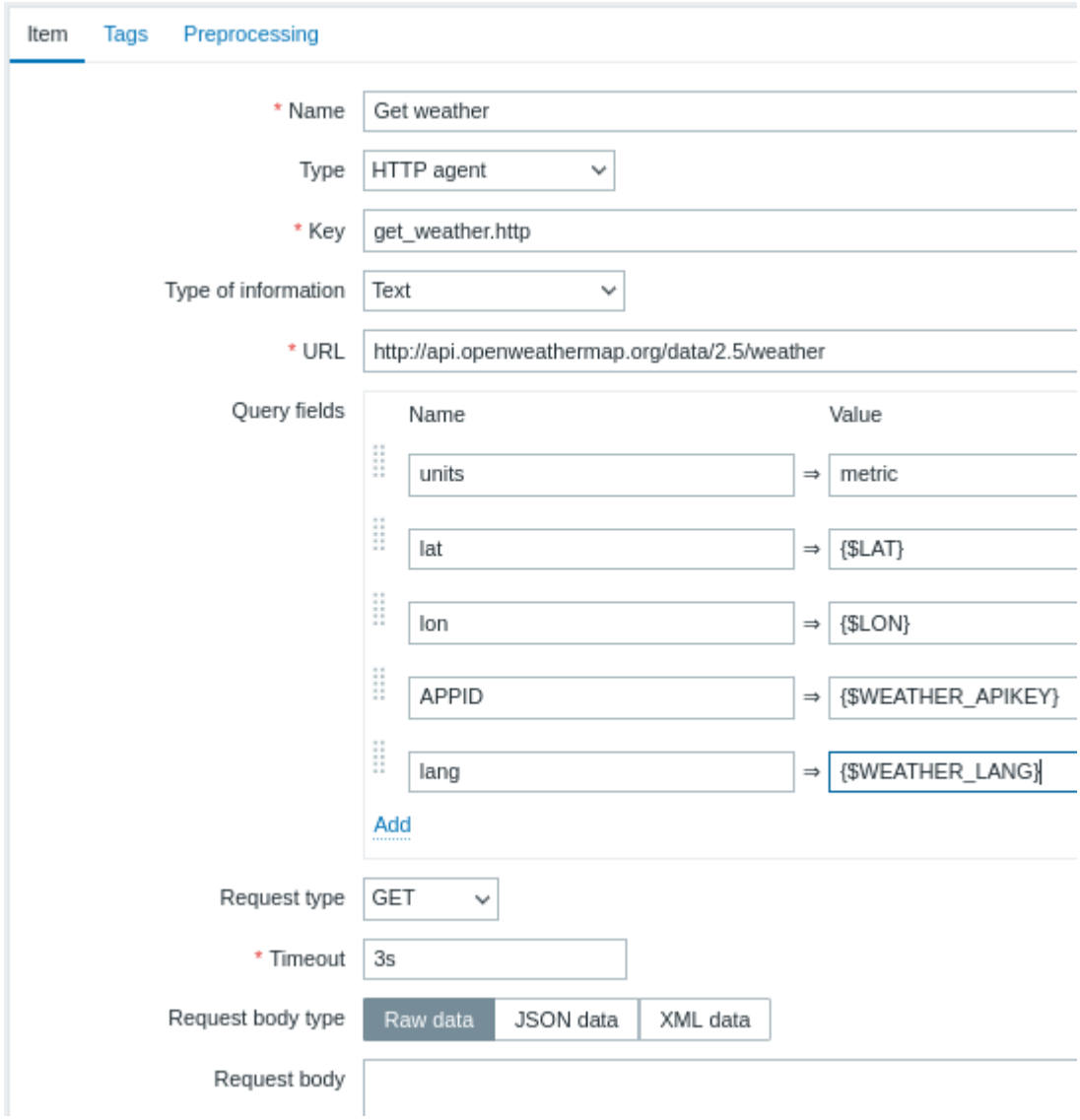
- Checking the result in *Latest data*:



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.

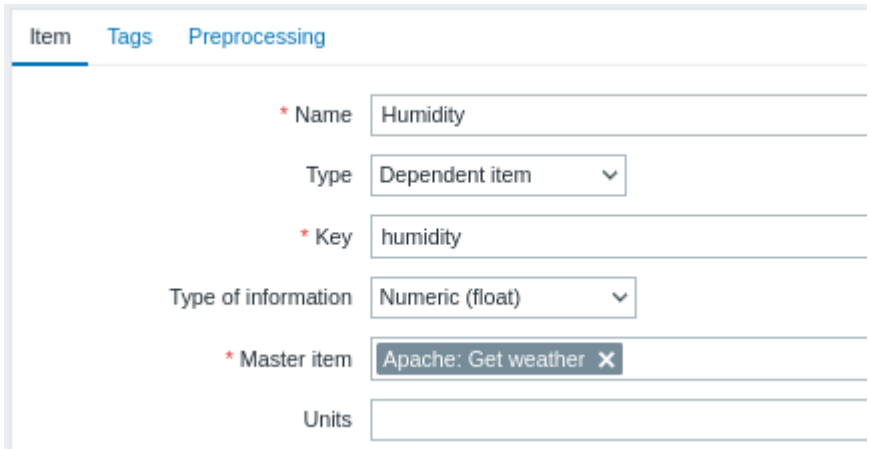
Sample JSON returned in response to HTTP agent:

```
{
    "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
    }
}
```

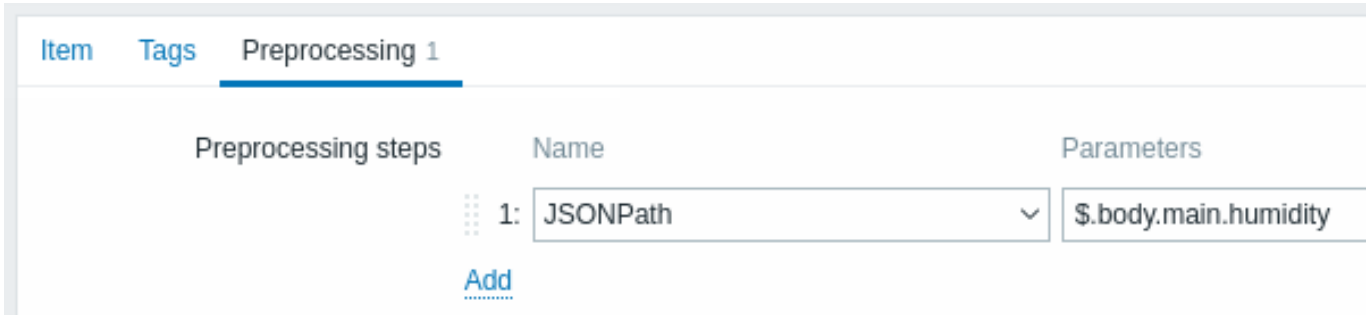

The next task is to configure dependent items that extract data from the JSON.

- Configure a sample dependent item for humidity:

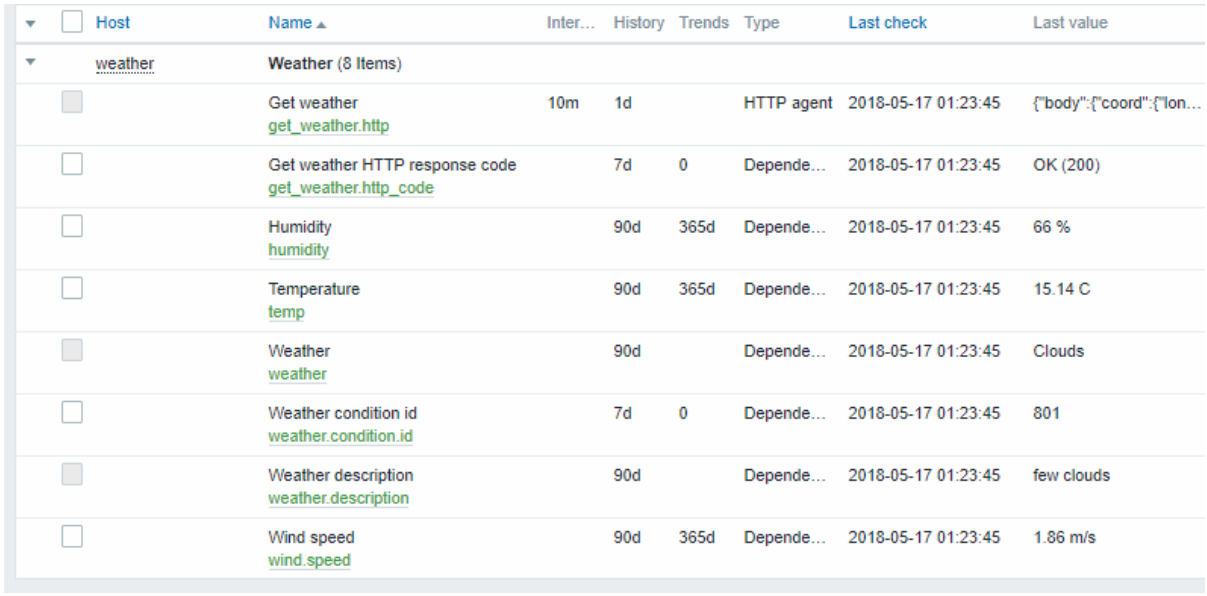


Other weather metrics such as 'Temperature' are added in the same manner.

- Sample dependent item value preprocessing with JSONPath:



- Check the result of weather data in *Latest data*:

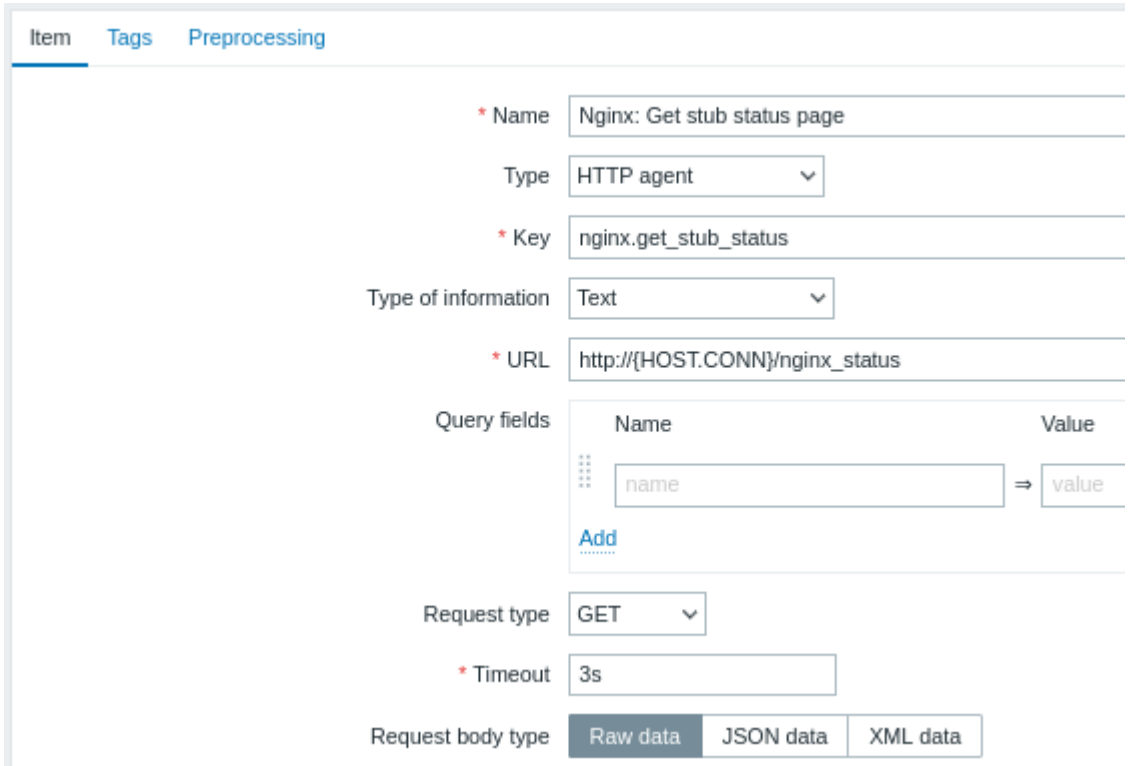


Host	Name ▲	Inter...	History	Trends	Type	Last check	Last value
weather	**Weather** (8 Items)						
	Get weather get_weather.http	10m	1d		HTTP agent	2018-05-17 01:23:45	{"body":{"coord":{"lon...
	Get weather HTTP response code get_weather.http_code		7d	0	Depende...	2018-05-17 01:23:45	OK (200)
	Humidity 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:

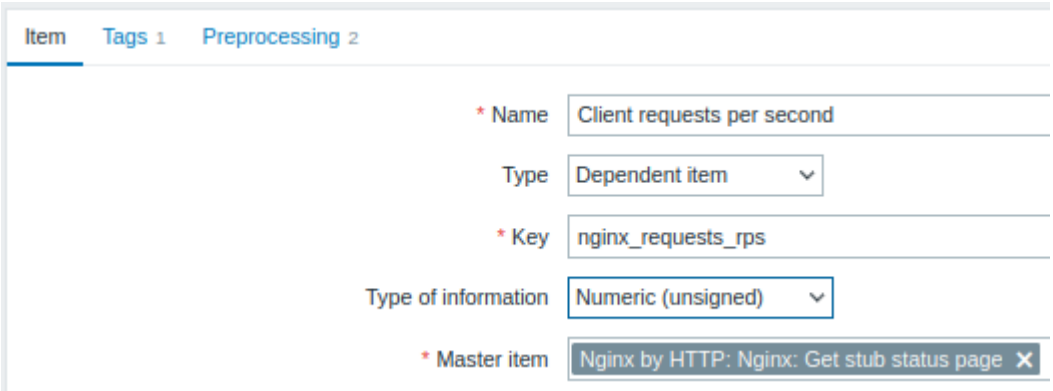


Sample Nginx stub status output:

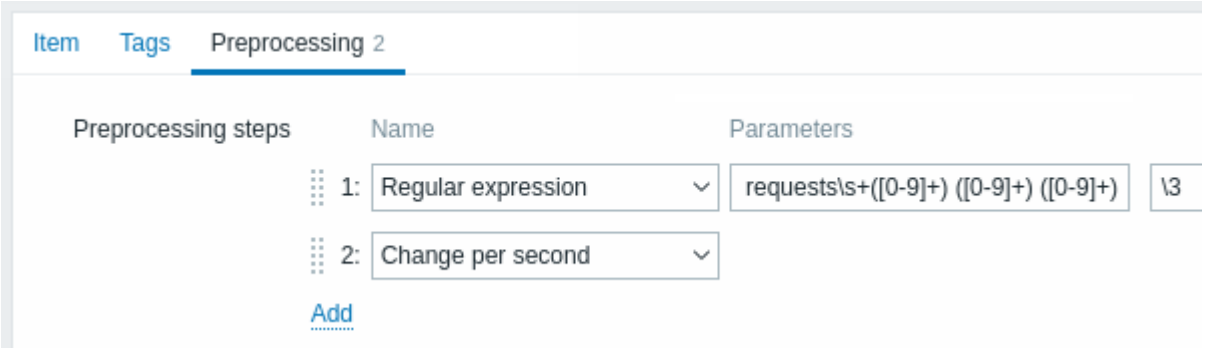
```
Active connections: 1 Active connections:
server accepts handled requests
 52 52 52
Reading: 0 Writing: 1 Waiting: 0
```

The next task is to configure dependent items that extract data.

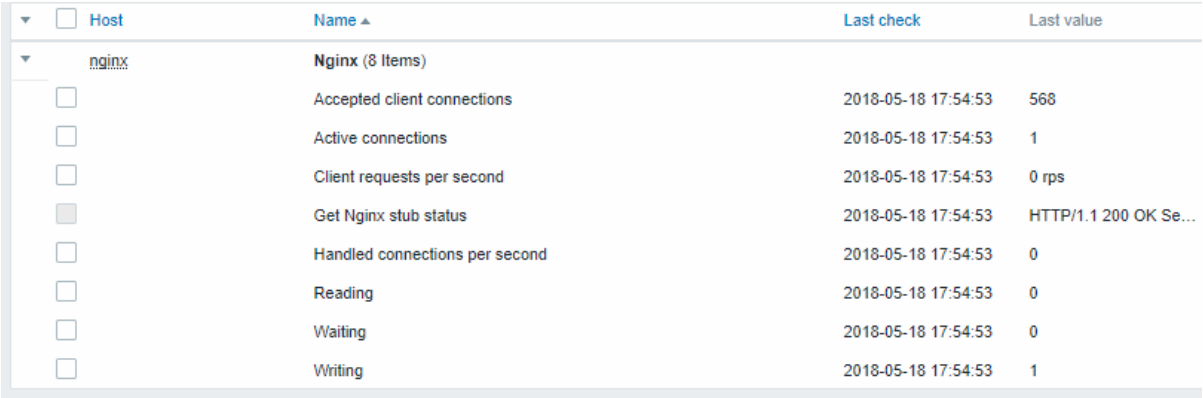
- Configure a sample dependent item for requests per second:



- 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

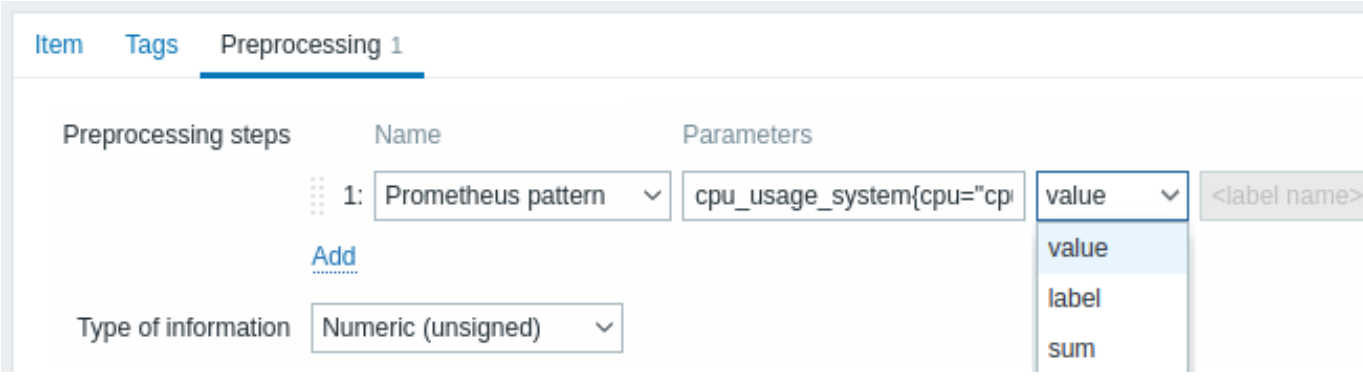
Bulk processing is supported for dependent items. To enable caching and indexing, the *Prometheus pattern* preprocessing must be the **first** preprocessing step. When *Prometheus pattern* is first preprocessing step then the parsed Prometheus data is cached and indexed by the first `<label>==<value>` 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

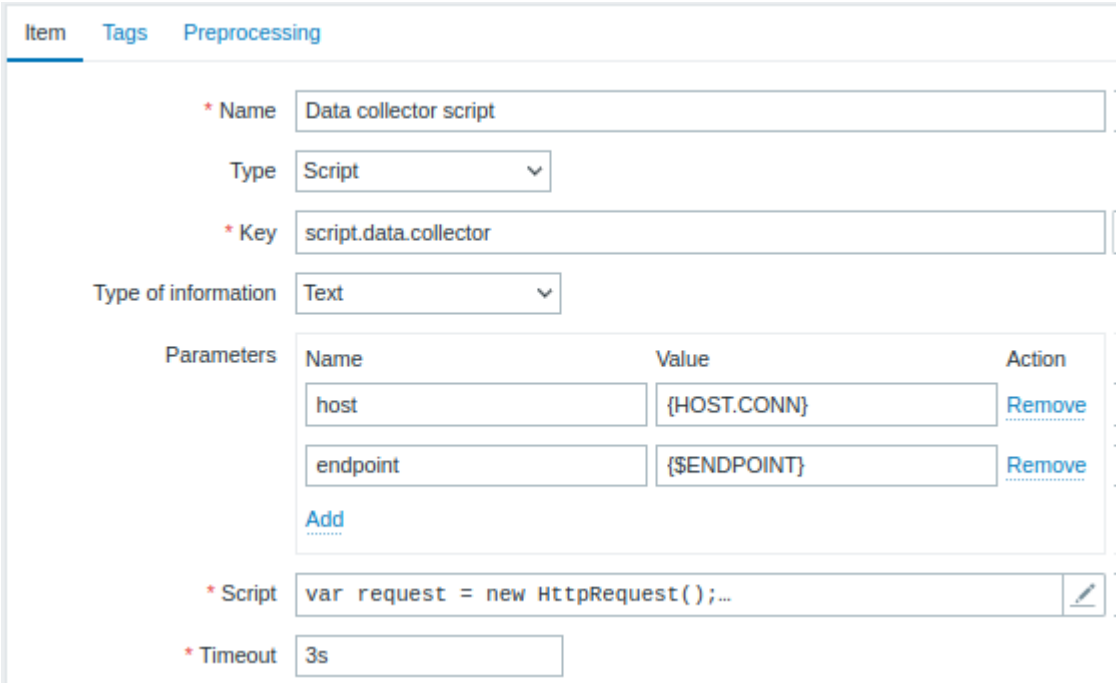
Script items can be used to collect data by executing a user-defined JavaScript code with the ability to retrieve data over HTTP/HTTPS. In addition to the script, an optional list of parameters (pairs of name and value) and timeout can be specified.

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*:

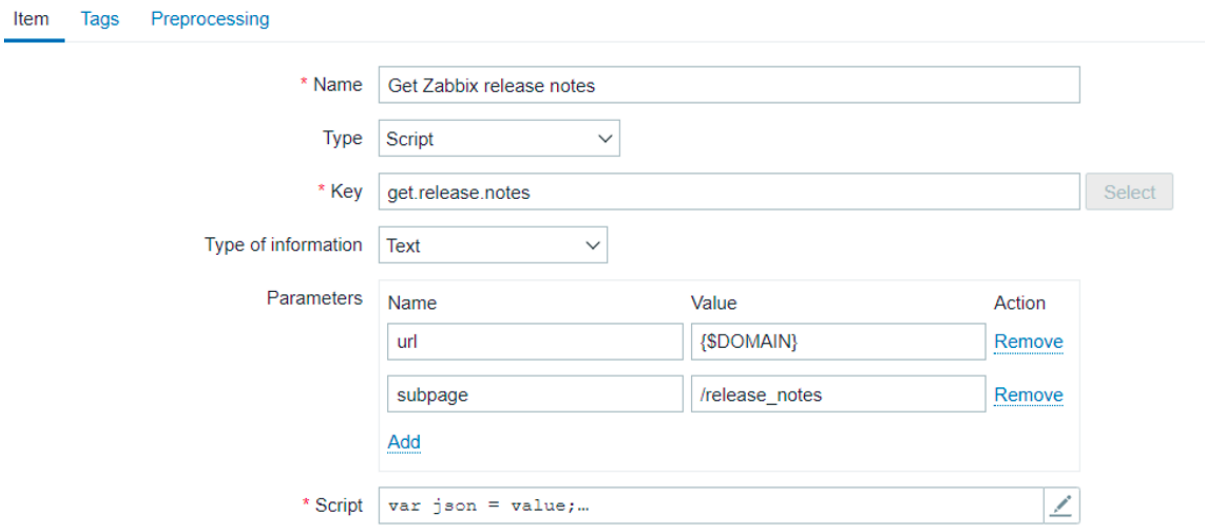
- Create an item with type "Script".
- In the *Script* field, enter:

```
var request = new HttpRequest();
return request.get("https://www.example.com/release_notes");
```

Data collection with parameters

Get 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 a 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. Up to 512KB of data can be returned before item preprocessing steps. Note, however, that the text value that can be eventually stored in database is limited to 64KB on MySQL (see info on other databases in the table).

/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 standard output together with standard error.

Attention:
A text (character, log or text type of information) item will not become unsupported in case of standard error output.

User parameters that return text (character, log, text type of information) can return whitespace. In case of 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 Loadable modules

1 Overview

Loadable modules offer a performance-minded option for extending Zabbix functionality.

There already are ways of extending Zabbix functionality by way of:

- user parameters (agent metrics)
- external checks (agent-less monitoring)
- `system.run[]` Zabbix agent item.

They 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).

2 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.

2.1 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.

2.2 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.

2.3 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.

2.4 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.

2.5 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.

3 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. It will fail if a module is missing, in case of bad permissions or if a shared library is not a Zabbix module.

4 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.

5 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;
}
```

```c
/******************************************************************************
 *                                                                            *
 * 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]`

6 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.

7 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.

8 Mass update

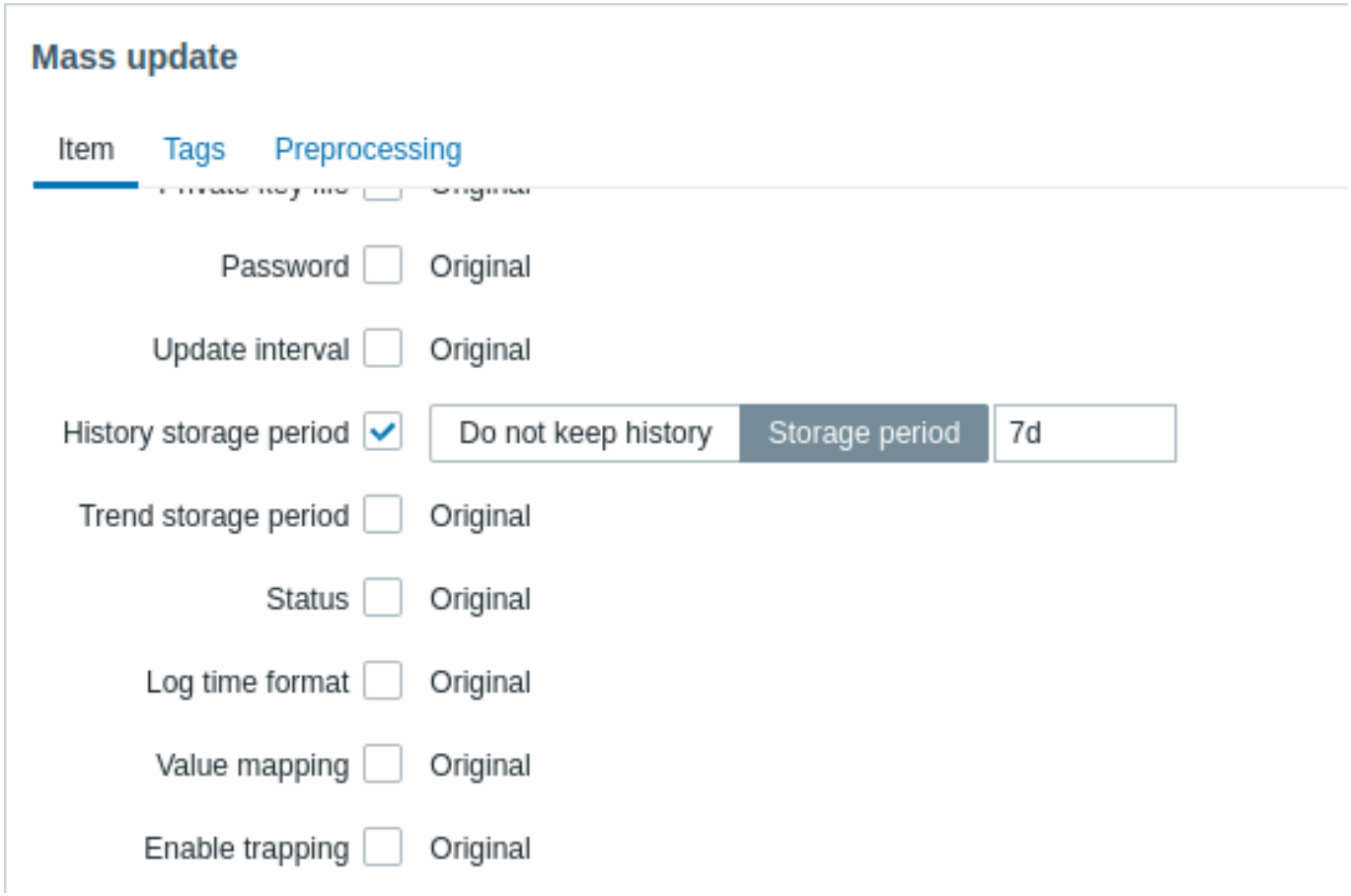
Overview

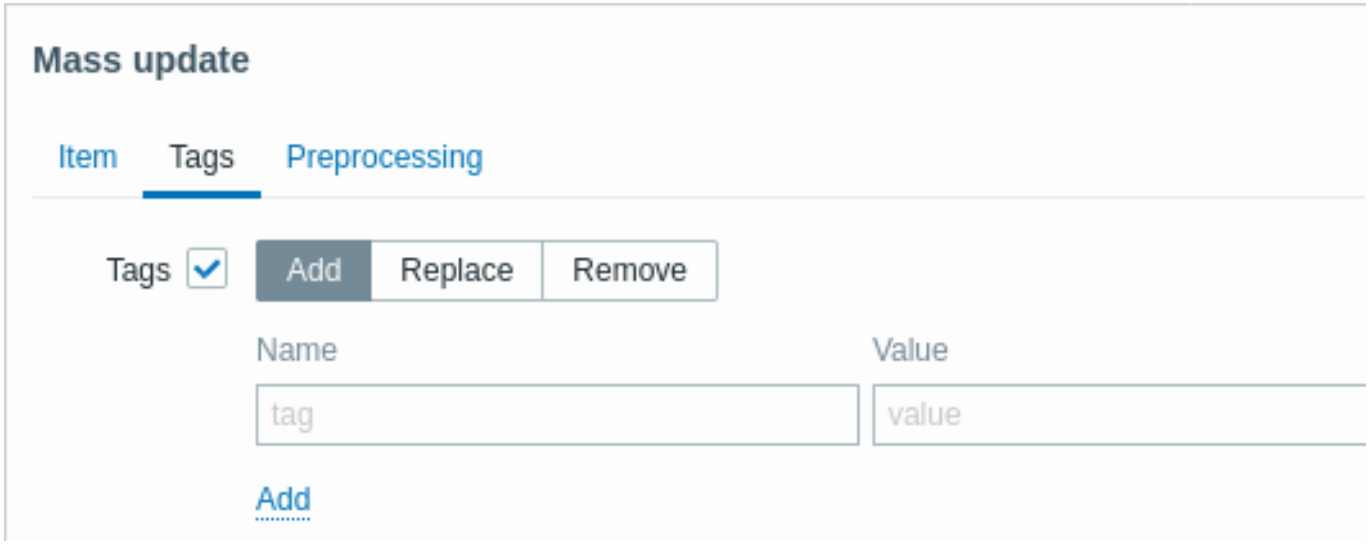
Sometimes you may want to change some attribute for a number of items at once. Instead of opening each individual item for editing, you may use the mass update function for that.

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

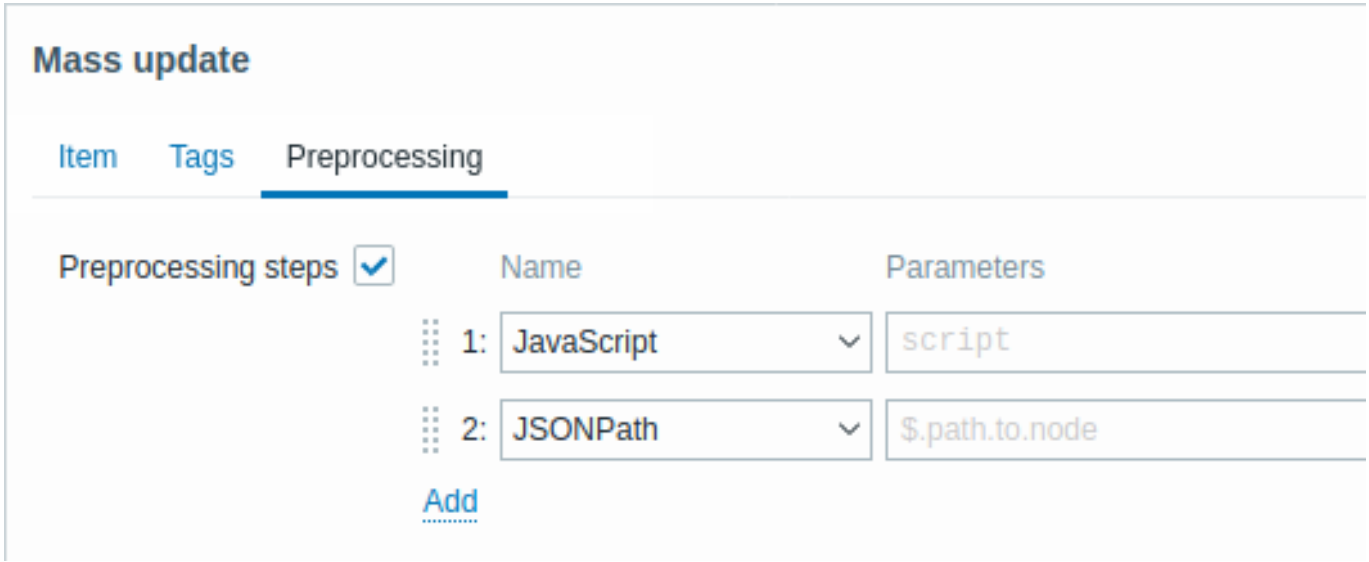




The *Tags* option allows to:

- *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.



When done, click on *Update*.

9 Value mapping

Overview

For a more "human" representation of received values, you can use value maps that contain the mapping between numeric/string values and string representations.

Value mappings can be used in both the Zabbix frontend and notifications sent by media types.

For example, an item which has value '0' or '1' can use value mapping to represent the values in a human-readable form:

- '0' => 'Not Available'
- '1' => 'Available'

Or, a backup related value map could be:

- 'F' => 'Full'
- 'D' => 'Differential'
- 'I' => 'Incremental'

In another example, value ranges for voltage may be mapped:

- '<=209' => 'Low'
- '210-230' => 'OK'
- '>=231' => 'High'

Value mappings are defined on template or host level. Once defined they become available for all items of the respective template or host. There is no value map inheritance - a template item on a host still uses the value map from the template; linking a template with value maps to the host does not make the host inherit the value maps.

When configuring items you can use a value map to "humanize" the way an item value will be displayed. To do that, you refer to the name of a previously defined value map in the *Value mapping* field.

Note:
Value mapping can be used with items having *Numeric (unsigned)*, *Numeric (float)* and *Character* type of information.

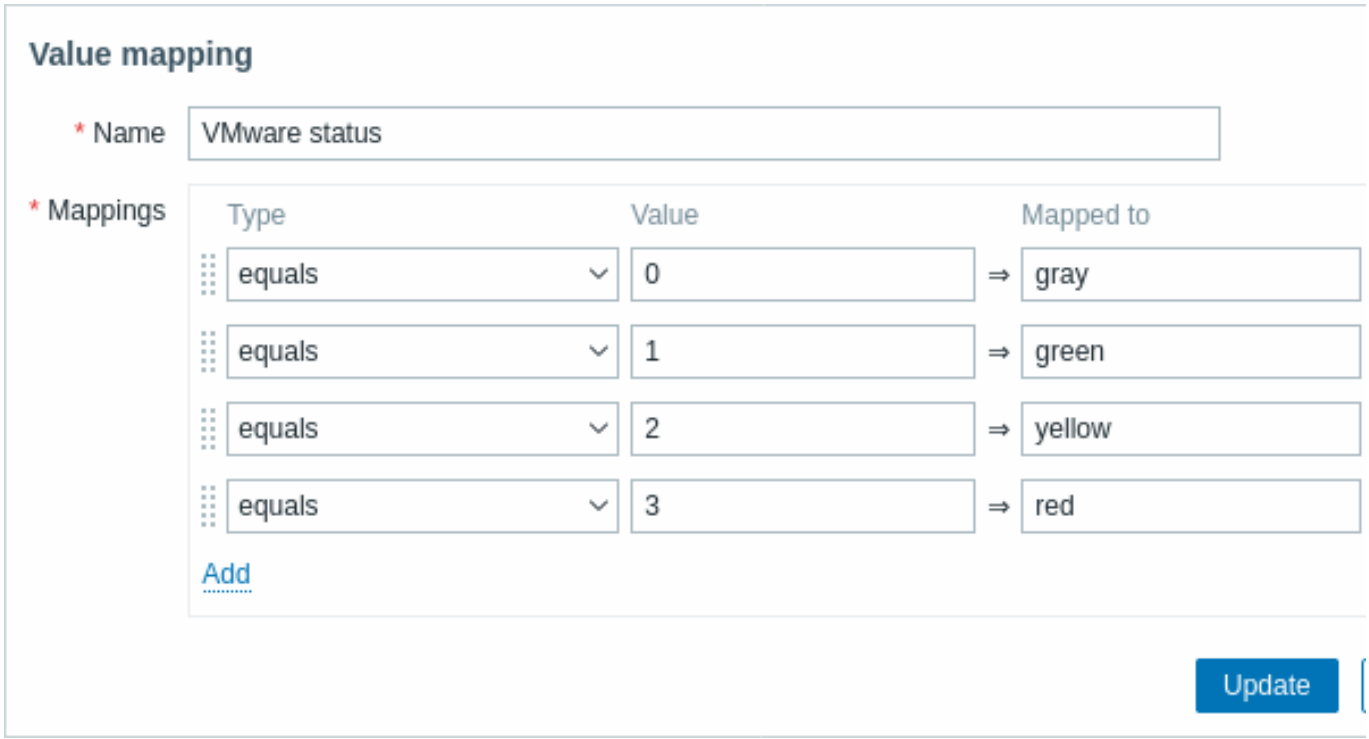
Value mappings can be exported/imported with the respective template or host.

Value mappings can be mass updated. Both host and template mass update forms have a *Value mapping* tab for mass updating value maps.

Configuration

To define a value map:

- Open a host or template configuration form
- Go to the *Value mapping* tab
- Click on *Add* to add a new map
- Click on the name of an existing map to edit it



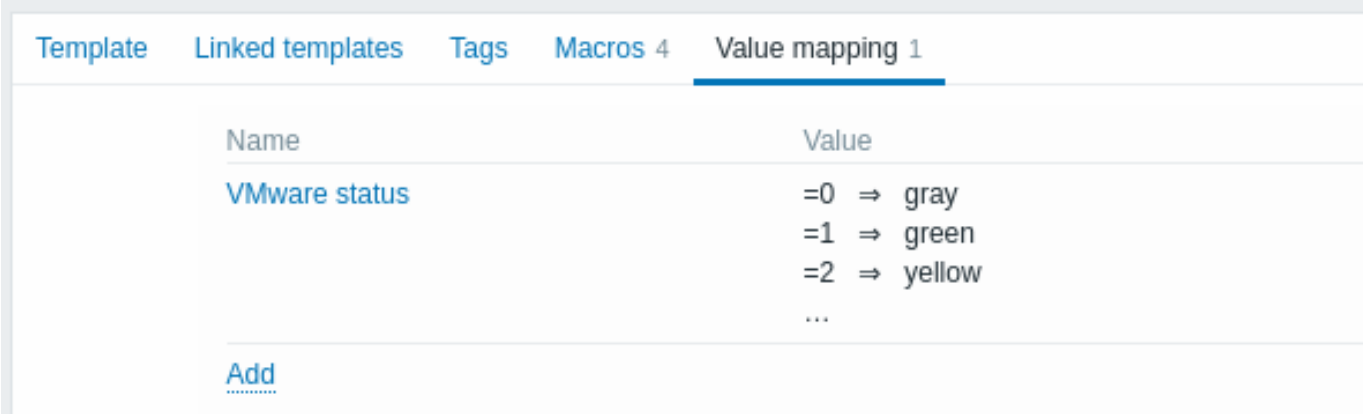
Parameters of a value map:

Parameter	Description
Name	Unique name of a set of value mappings.
Mappings	Individual mapping rules for mapping numeric/string values to string representations. Mapping is applied according to the order of mapping rules. It is possible to reorder mappings by dragging. Only numeric value types are supported for mapping ranges (*is greater than or equals*, *is less than or equals*, *in range* mapping types).

Parameter	Description
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 (e.g. 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
Value	Incoming value. Depending on the mapping type, may also contain a range or regular expression.
Mapped to	String representation for the incoming value.

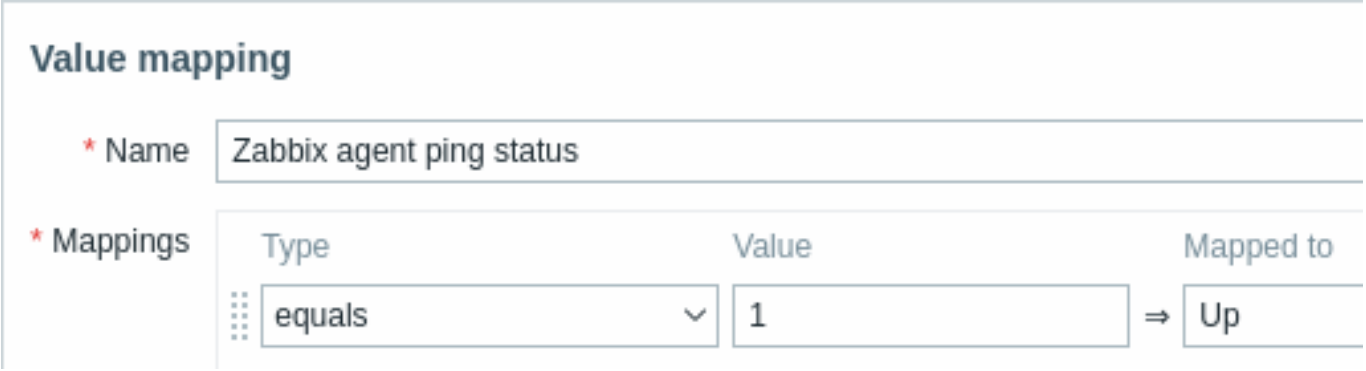
All mandatory input fields are marked with a red asterisk.

When the value map is displayed in the list, only the first three mappings of it are visible, while three dots indicate that more mappings exist.



How this works

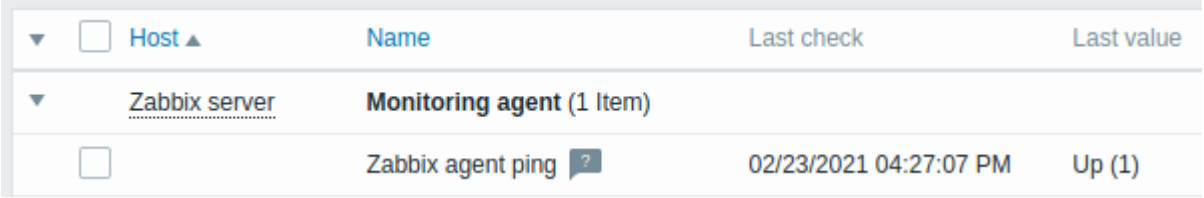
For example, one of the predefined agent items 'Zabbix agent ping' uses a template-level value map called 'Zabbix agent ping status' to display its values.



In the item configuration form you can see a reference to this value map in the *Value mapping* field:



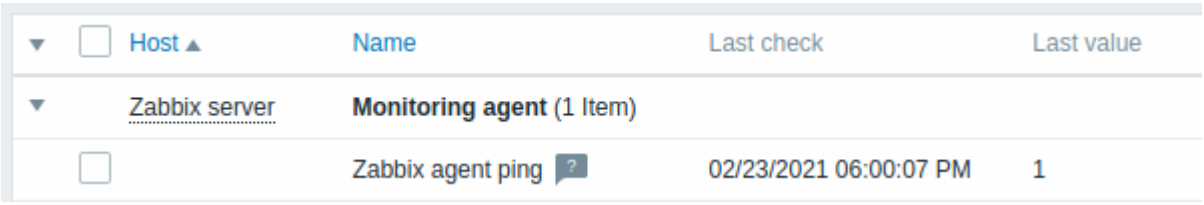
So in *Monitoring* → *Latest data* the mapping is put to use to display 'Up' (with the raw value in parentheses).



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 separately (displayed in parentheses).

Note:
A value being displayed in a human-readable form is also easier to understand when receiving notifications.

Without a predefined value map you would only get this:



So in this case you would either have to guess what the '1' stands for or do a search of documentation to find out.

10 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:

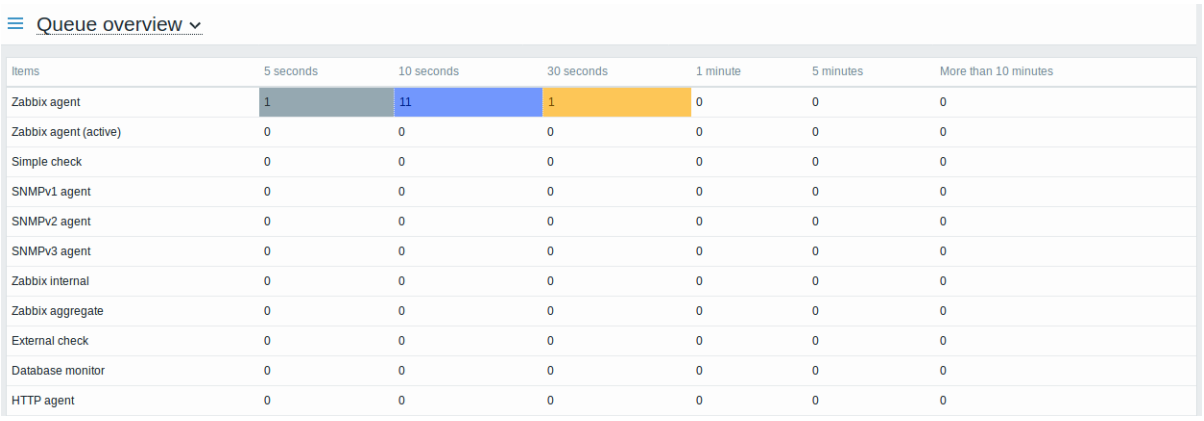
- log, logrt and event log active Zabbix agent items
- SNMP trap items
- trapper items
- web monitoring items
- dependent items

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.

Reading the queue

To read the queue, go to *Administration* → *Queue*.



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.

Queue overview

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.

11 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.

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.

12 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:

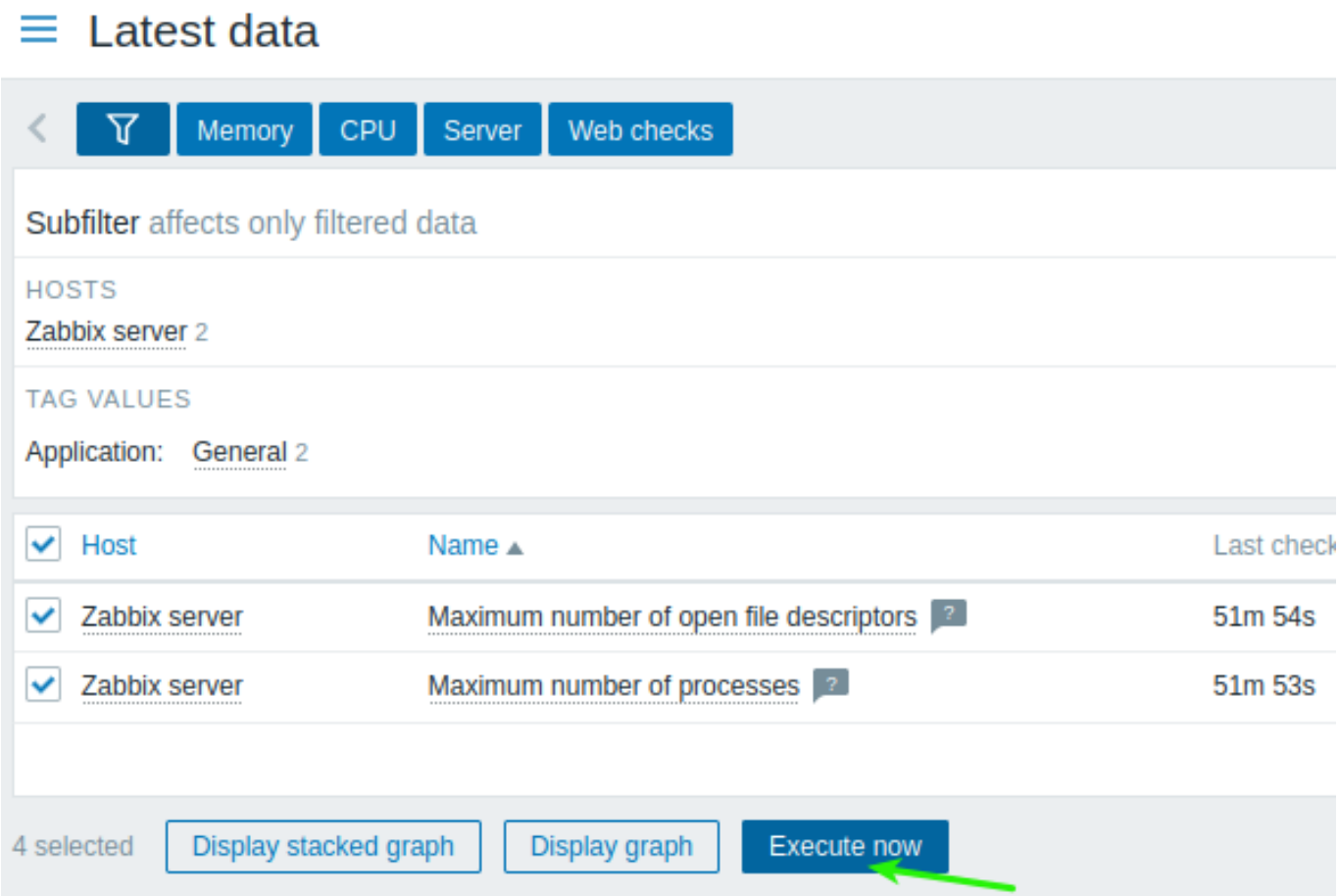
- Zabbix agent (passive)
- SNMPv1/v2/v3 agent
- IPMI agent
- Simple check
- Zabbix internal
- External check
- Database monitor
- JMX agent
- SSH agent
- Telnet
- Calculated
- HTTP agent
- Dependent item
- Script

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:

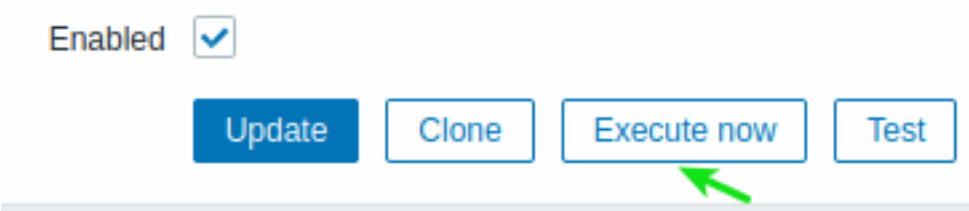
- click on *Execute now* for selected items in the list of latest data:



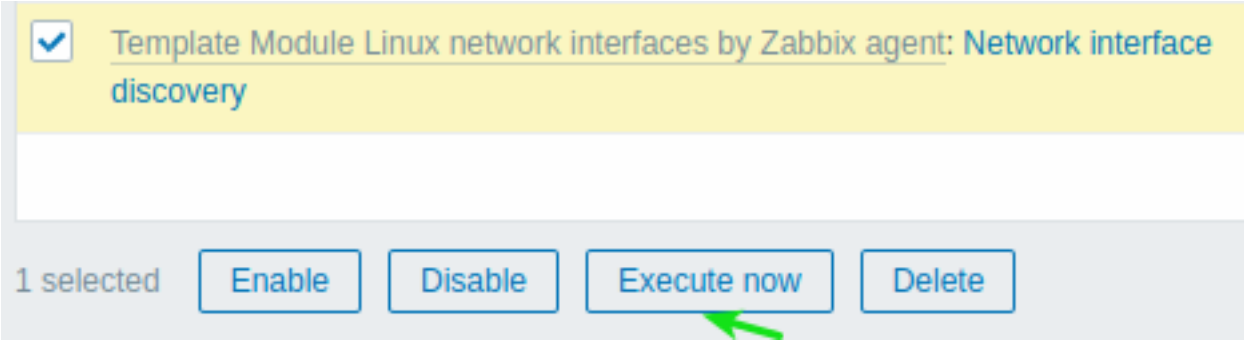
Several items can be selected and "executed now" at once.

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.

13 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 *Administration → 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.

14 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, such as user parameters (agent metrics), external checks (agent-less monitoring), and `system.run[]` Zabbix agent item.

The following features are specific to Zabbix 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 about writing your own plugins, see Plugin development guidelines. 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.

Each plugin parameter should have the following structure:

Plugins.<PluginName>.<Parameter>=<Value>

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.

Named sessions

Named sessions represent an additional level of plugin parameters and can be used to define 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 password separately. In item keys, the first parameter can be either a connString or a Uri. If the first key parameter matches a session name specified in the configuration file, the check will be executed using named session parameters. If the first key parameter doesn't match any session name, it will be treated as a Uri.

Note, that:

- when providing a connString (session name) in key parameters, key parameters for username and password must be empty;
- passing embedded URI credentials is not supported, consider using named sessions instead;
- in case an authentication parameter is not specified for the named session, a hardcoded default value will be used.

The list of available named session parameters depends on the plugin, see individual plugin configuration files for details.

Example: Monitoring of two instances "MySQL1" and "MySQL2" can be configured in the following way:

```
Plugins.Mysql.Sessions.MySQL1.Uri=tcp://127.0.0.1:3306
Plugins.Mysql.Sessions.MySQL1.User=<UsernameForMySQL1>
Plugins.Mysql.Sessions.MySQL1.Password=<PasswordForMySQL1>
Plugins.Mysql.Sessions.MySQL2.Uri=tcp://127.0.0.1:3307
Plugins.Mysql.Sessions.MySQL2.User=<UsernameForMySQL2>
Plugins.Mysql.Sessions.MySQL2.Password=<PasswordForMySQL2>
```

Now, these names may be used as connStrings in keys instead of URIs:

```
mysql.ping[MySQL1]
mysql.ping[MySQL2]
```

Hardcoded defaults

If a parameter required for authentication is not provided in an item key or in the named session parameters, the plugin will use a hardcoded default value.

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	
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	

Plugin name	Description	Supported item keys	Comments
MySQL	Monitoring of MySQL and its forks.	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, use named sessions and specify the TLS parameters for the named session in the agent configuration file. Currently, TLS parameters cannot be passed as item key parameters.
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.
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	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
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	

Plugin name	Description	Supported item keys	Comments
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.
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	This plugin is loadable since Zabbix 6.0.6 (built-in previously). To configure encrypted connections to the database, use named sessions and specify the TLS parameters for the named session in the agent configuration file. Supported in plugin versions 1.2.1, 6.2.7 and newer[1]. Currently, TLS parameters cannot be passed as item key parameters. See also MongoDB plugin configuration file.
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.wal.stat	This plugin is loadable since Zabbix 6.2.4 (built-in previously). To configure encrypted connections to the database, use named sessions and specify the TLS parameters for the named session in the agent configuration file. Currently, TLS parameters cannot be passed as item key parameters.

See also: Building loadable plugins.

Footnotes

[1] - Since Zabbix 6.2.7, loadable plugins started using the same versioning system as Zabbix itself. As a result, MongoDB plugin version has changed from 1.2.1 to 6.2.7.

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-6.2.7.tar.gz
```

Make sure to replace "zabbix-agent2-plugin-postgresql-6.2.7.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 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 state to PROBLEM.

A trigger may have the following states:

State	Description
OK	This is a normal trigger state.
Problem	Something has happened. For example, the processor load is too high.
Unknown	The trigger value cannot be calculated. See Unknown state.

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 state

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:

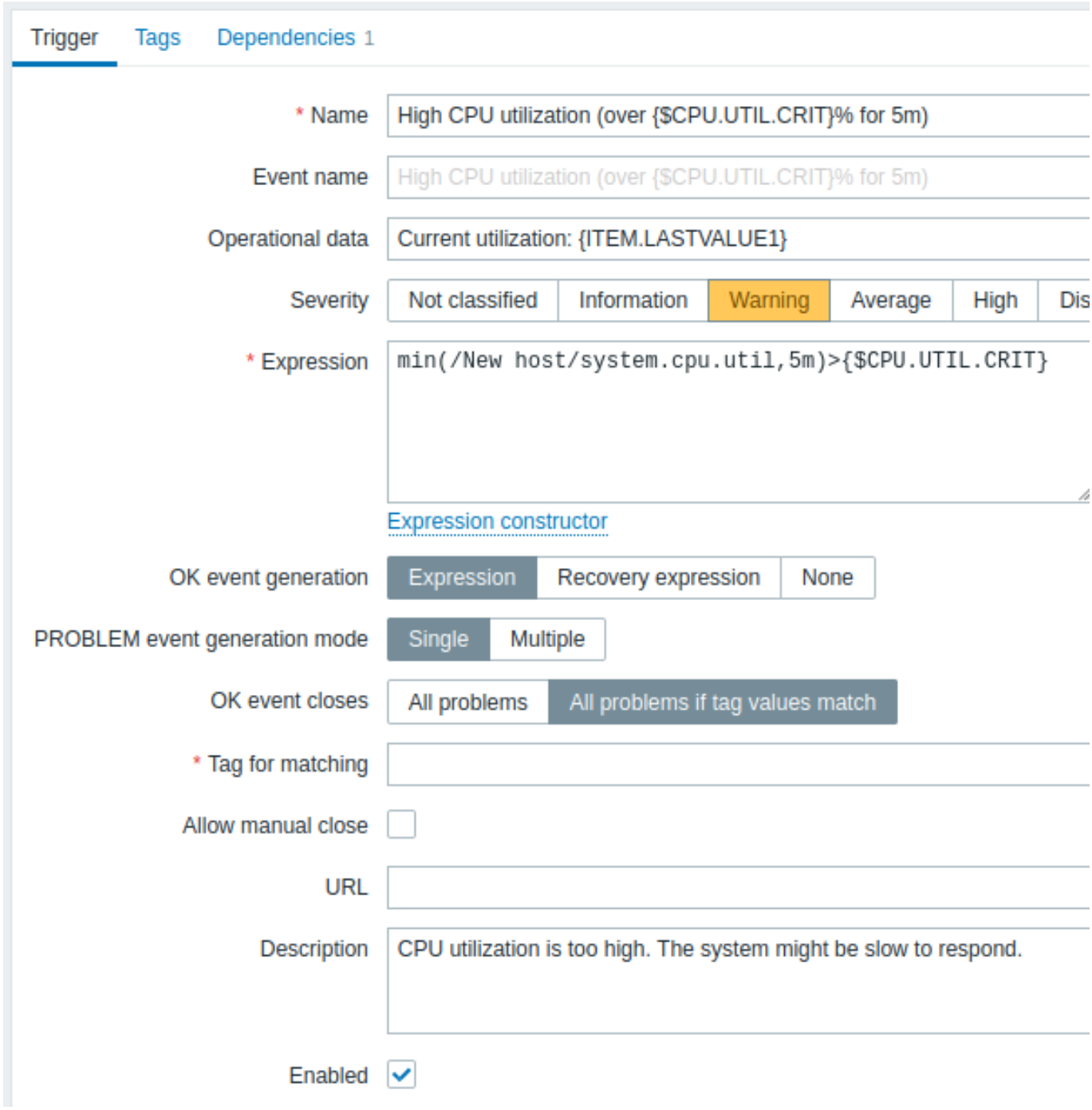
- Go to: *Configuration* → *Hosts*
- Click on *Triggers* in the row of the host

- 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

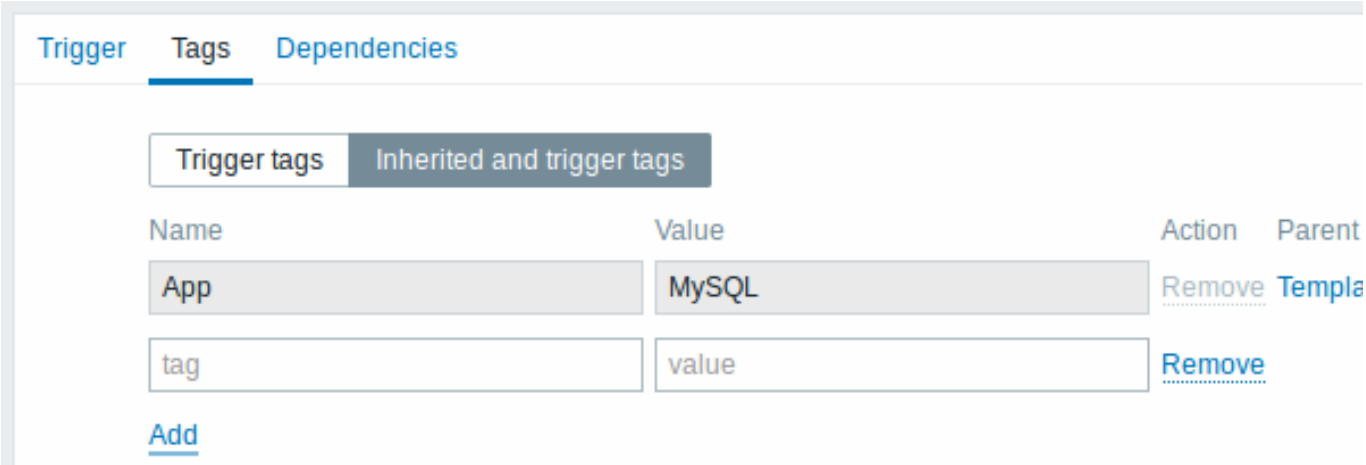
The **Trigger** tab contains all the essential trigger attributes.



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.
URL	If not empty, the URL entered here is available as a link in several frontend locations, e.g. when clicking on the problem name in *Monitoring* → *Problems* (*URL* option in the *Trigger* menu) and *Problems* dashboard widget. The same set of macros is supported as in the trigger name, plus {EVENT.ID}, {HOST.ID} and {TRIGGER.ID}. Note that 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 macro 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.

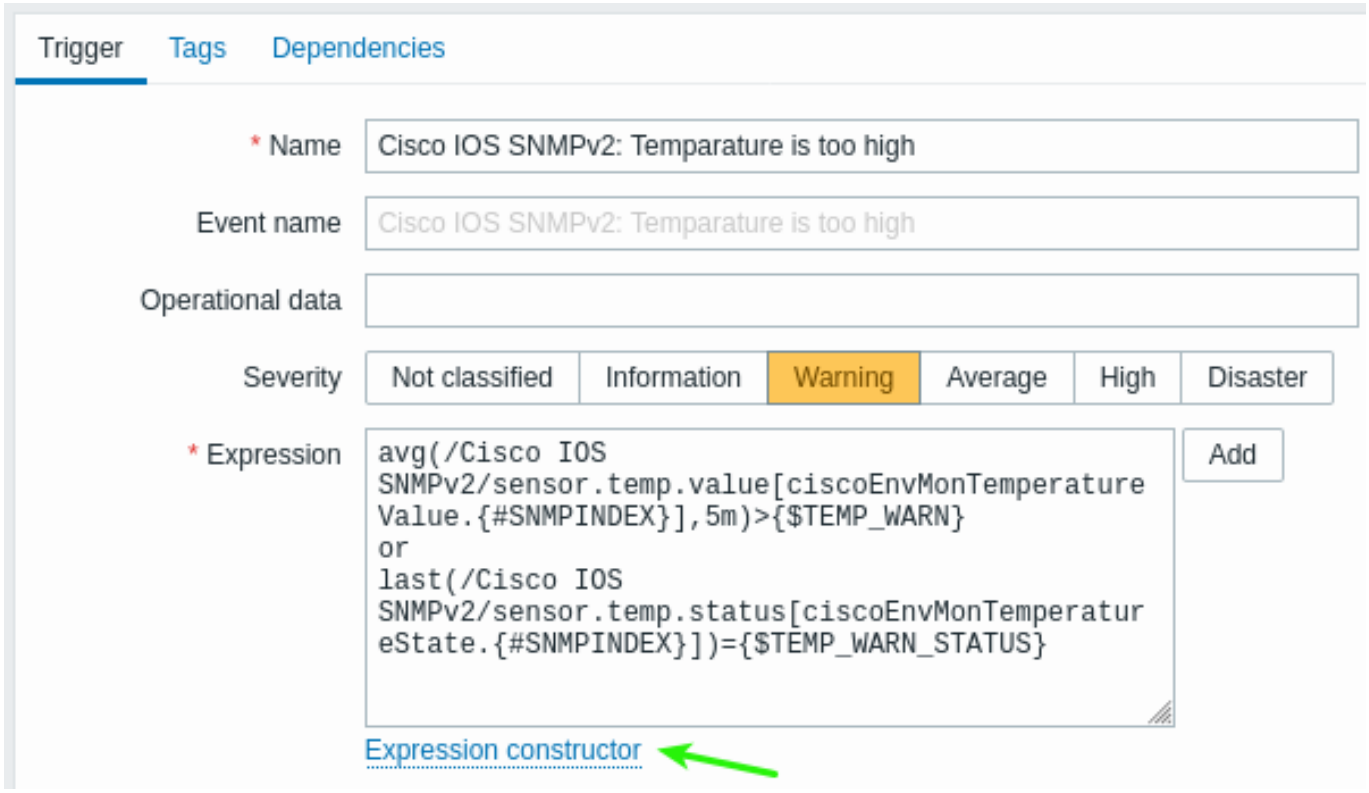
Testing expressions

It is possible to test the configured trigger expression as to what the expression result would be depending on the received value.

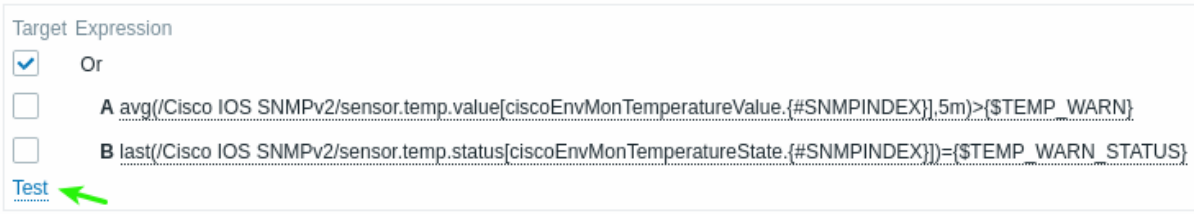
The following expression from an official template is taken as an example:

```
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}
```

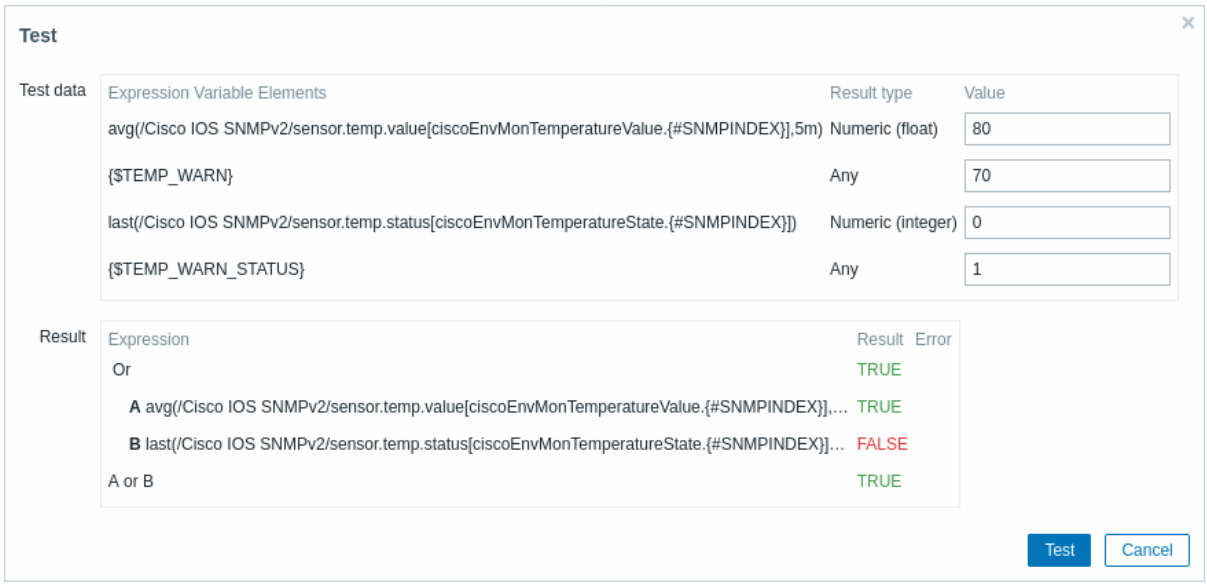
To test the expression, click on *Expression constructor* under the expression field.



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 hashtag, 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 hashtag 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.

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()>000000 and time()<060000
```

The trigger may change its state to true 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 true 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 expression 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 over 5.

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 some 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 for 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 state as long as the trigger it depends on is in '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.

Actions on dependent triggers will not be executed if the trigger they depend on:

- 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 dependent trigger
- is disabled, has disabled item or disabled item host

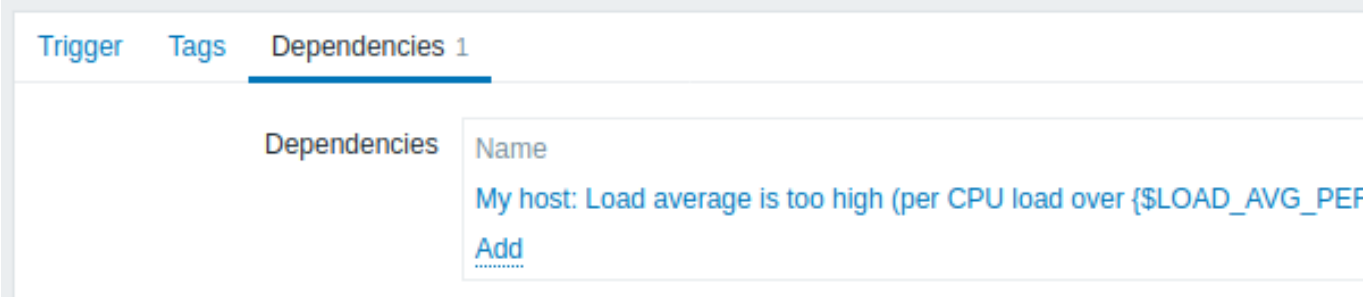
Note that "secondary" (dependent) trigger in the above-mentioned cases will not be immediately updated. While parent trigger is in PROBLEM state, it's dependents may report values, which we cannot trust. Thus, dependent trigger will only be re-evaluated, and change it's state, only after parent trigger is in OK state and we have received trusty metrics.

Also:

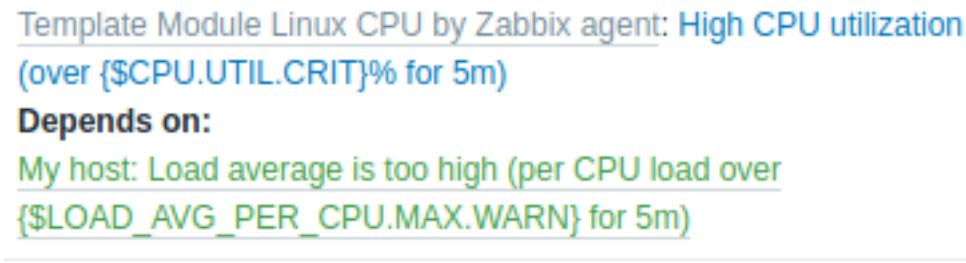
- Trigger dependency may be added from any host trigger to any other host trigger, as long as it wouldn't result in a circular dependency.
- Trigger dependency may be added from a template to a template. If a trigger from template A depends on a trigger from template B, template A may only be linked 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 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 trigger was depending on. This allows to, for example, have a template where some triggers depend on router (host) triggers. All hosts linked to this template will depend on that specific router.
- Trigger dependency from a host trigger to a template trigger may not be added.
- 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. Host trigger prototype cannot depend on a trigger from a template.

Configuration

To define a dependency, open the Dependencies tab in a trigger configuration form. Click on *Add* in the 'Dependencies' block and select one or more triggers that our trigger will depend on.



Click *Update*. Now the trigger has an indication of its dependency in the list.



Example of several dependencies

For example, a Host is behind a Router2 and the Router2 is behind a Router1.

```
Zabbix - Router1 - Router2 - Host
```

If Router1 is down, then obviously Host and Router2 are also unreachable yet we don't want to receive three notifications about Host, Router1 and Router2 all being down.

So in this case we define two dependencies:

```
'Host is down' trigger depends on 'Router2 is down' trigger
'Router2 is down' trigger depends on 'Router1 is down' trigger
```

Before changing the status of the 'Host is down' trigger, Zabbix will check for corresponding trigger dependencies. If found, and one of those triggers is in 'Problem' state, then the trigger status will not be changed and thus actions will not be executed and notifications will not be sent.

Zabbix performs this check recursively. If Router1 or Router2 is unreachable, the Host trigger won't be updated.

4 Trigger severity

Trigger severity defines how important a trigger is. Zabbix supports the following trigger severities:

SEVERITY	DEFINITION	COLOR
Not classified	Unknown severity.	Gray
Information	For information purposes.	Light blue
Warning	Be warned.	Yellow
Average	Average problem.	Orange
High	Something important has happened.	Light red
Disaster	Disaster. Financial losses, etc.	Red

The 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 - high severity, email - other.
- limiting actions by conditions against trigger severities

It is possible to customize trigger severity names and colors.

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

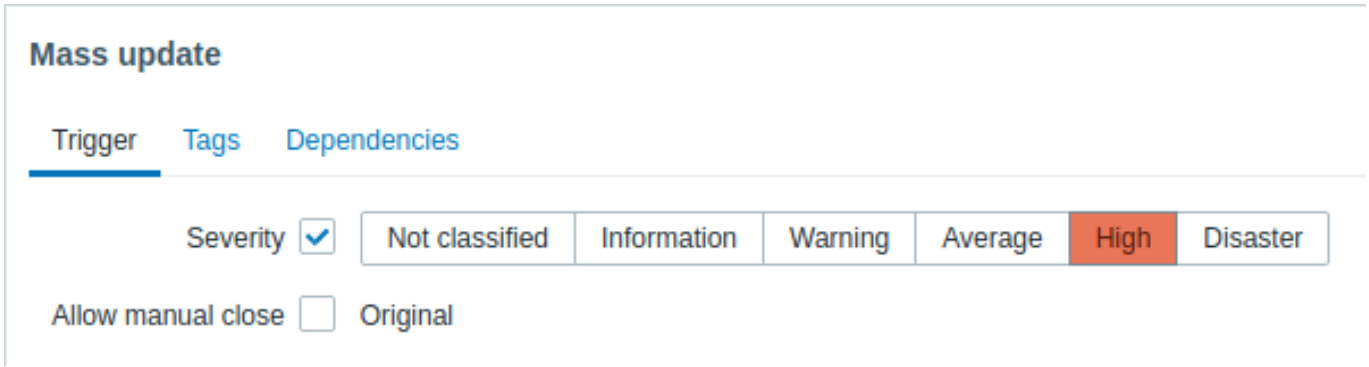
Overview

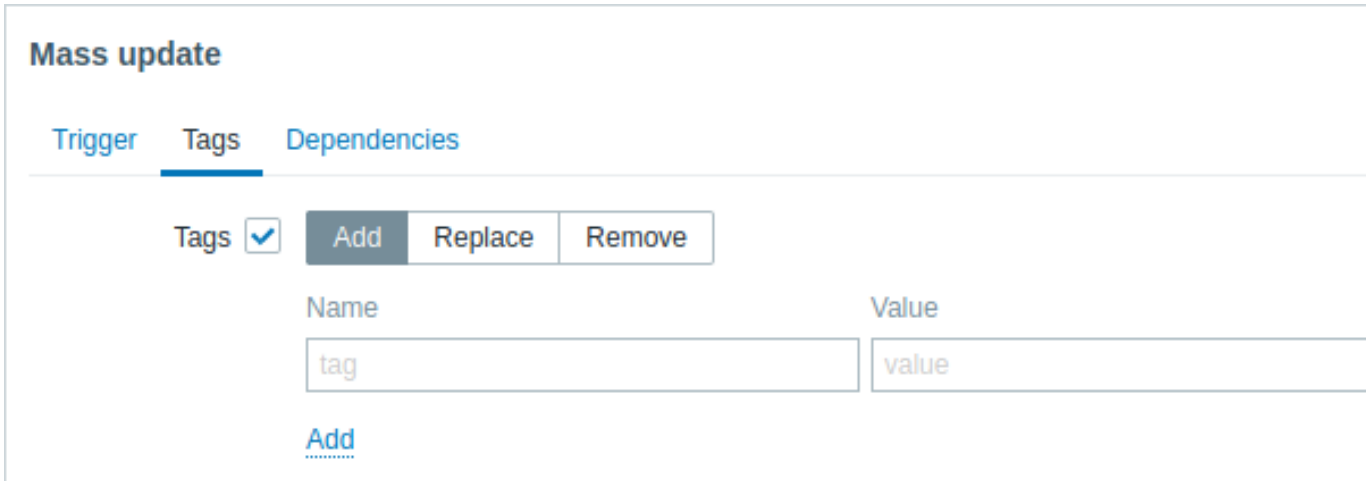
With mass update you may change some attribute for a number of triggers at once, saving you the need to open each individual trigger for editing.

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

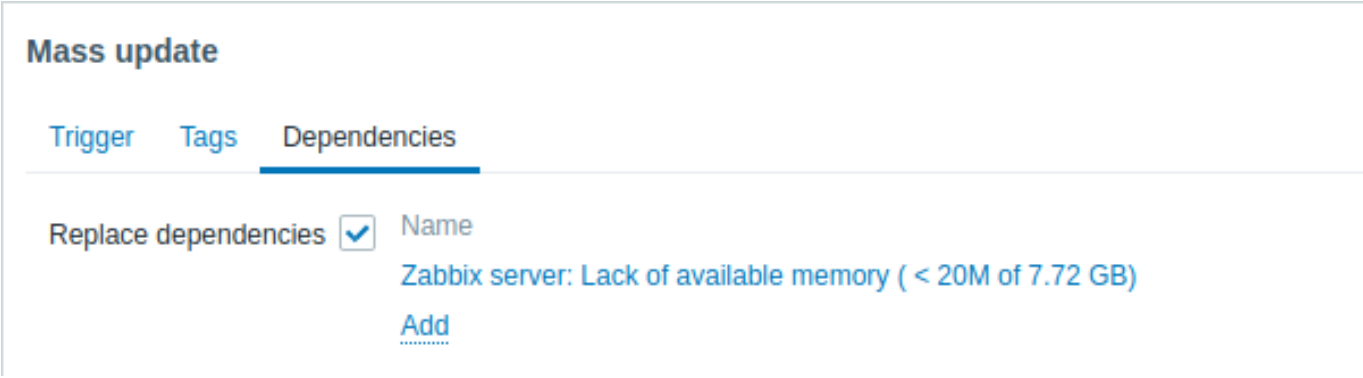




The following options are available when selecting the respective button for tag update:

- *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.

1 Functions

Two things one needs to know is how to define a problem state 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 after ”time to act” is expected to be in a problem state. 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.

2 Parameters

Both functions use almost the same set of parameters. Use the list of supported functions for reference.

2.1 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.

2.2 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.

2.3 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.[a]

[a] According to specification these are voltages on chip pins and generally speaking may need scaling.

2.4 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[1]	$x = a_0 + a_1*t + a_2*t^2 + ... + a_n*t^n$
exponential	$x = a*exp(b*t)$
logarithmic	$x = a + b*log(t)$
power	$x = a*t^b$

2.5 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

3 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.

4 Potential errors

Functions return -1 in such situations:

- specified evaluation period contains no data;
- result of mathematical operation is not defined[2];
- numerical complications (unfortunately, for some sets of input data range and precision of double-precision floating-point format become insufficient)[3].

Note:
No warnings or errors are flagged if chosen fit poorly describes provided data or there is just too few data for accurate prediction.

5 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 [4]:

```
timeleft(/host/vfs.fs.size[/,free],1h,0)<1h and timeleft(/host/vfs.fs.size[/,free],1h,0)<>-1
```

[1] `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.

[2] 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.

[3] 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).

[4] 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[/,free],1h,0)<1h and ({TRIGGER.VALUE}=0 and timeleft(/host/vfs.fs.size[/,free],1h,0)<>-1 or {TRIGGER.VALUE}=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.

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 e-mail 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 auto-discovery 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 Configuration → 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:

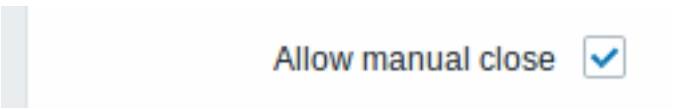
- When new data arrive for any item included in the trigger expression (note that the values discarded by a throttling preprocessing step are not considered as received and will not cause trigger expression to be re-evaluated);
- When time-based functions are used in the expression. Complete time-based function list can be found on Triggers page.

Configuration

Two steps are required to close a problem manually.

Trigger configuration

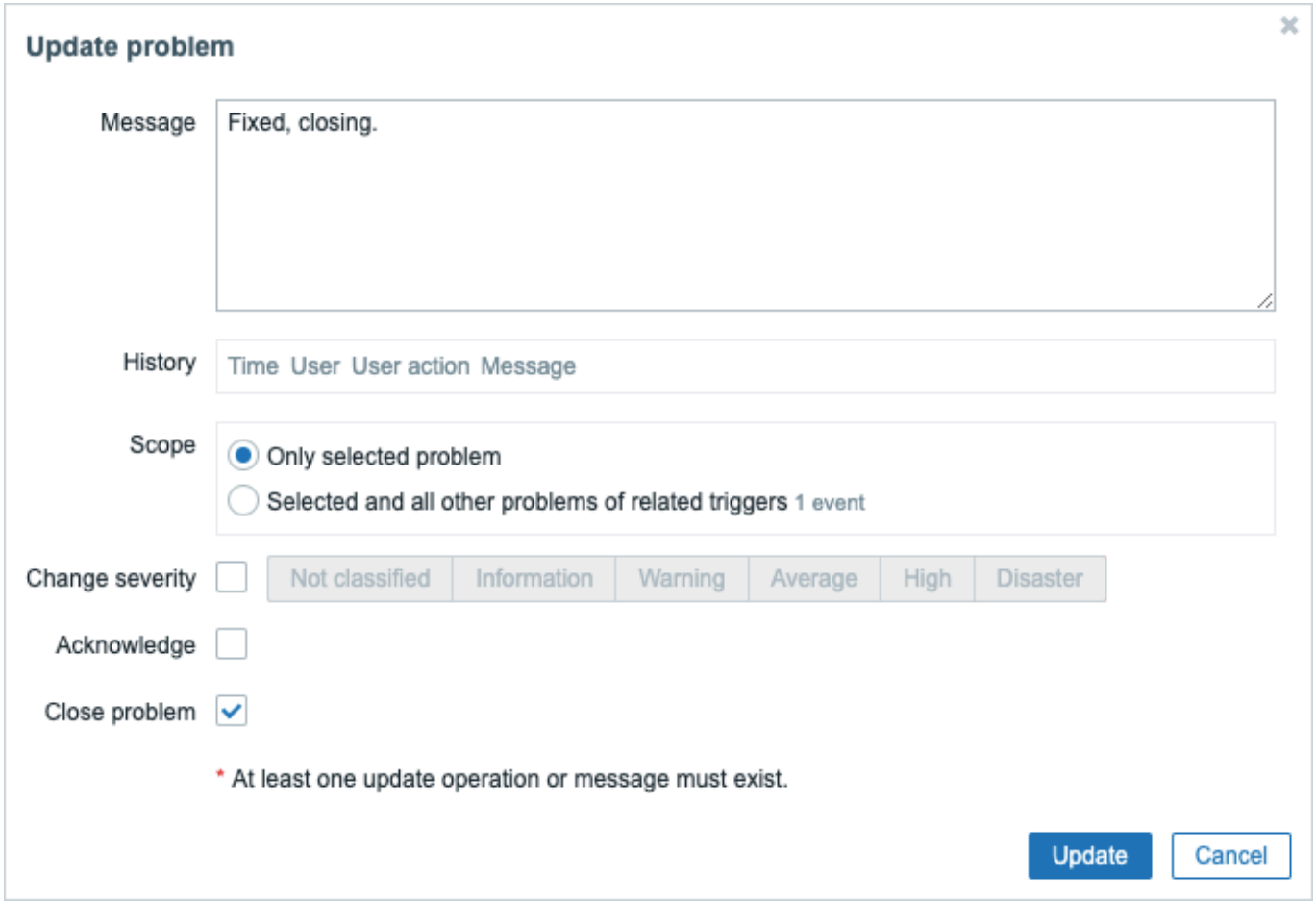
In trigger configuration, enable the *Allow manual close* option.



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 *Multiple Problem Event Generation* enabled. 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
```

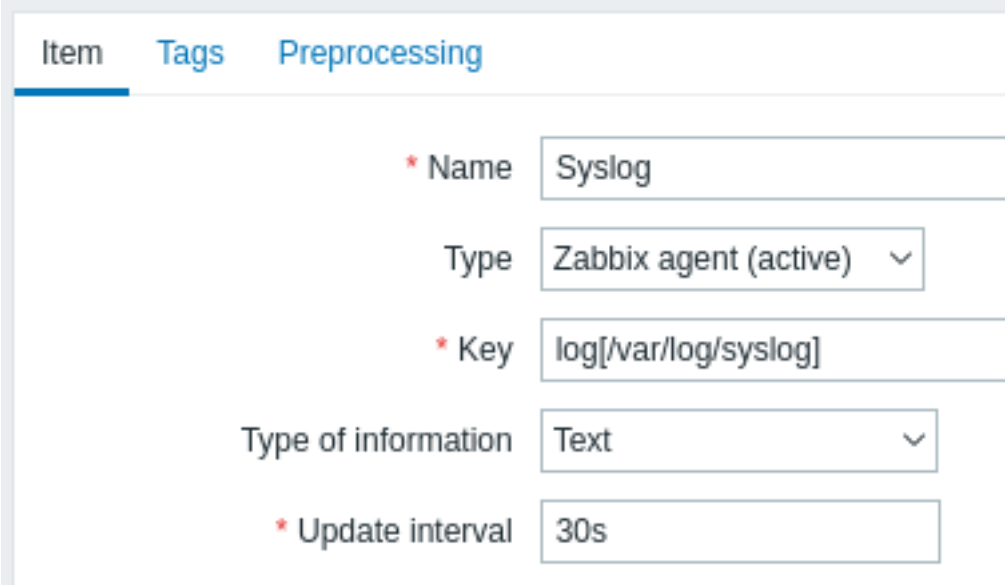
To do this you need to tag these related events as, for example, ”Application 1” and ”Application 2”. That can be done by applying a regular expression to the log line to extract the tag value. Then, when events are created, they are tagged ”Application 1” and ”Application 2” respectively and problem 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’.

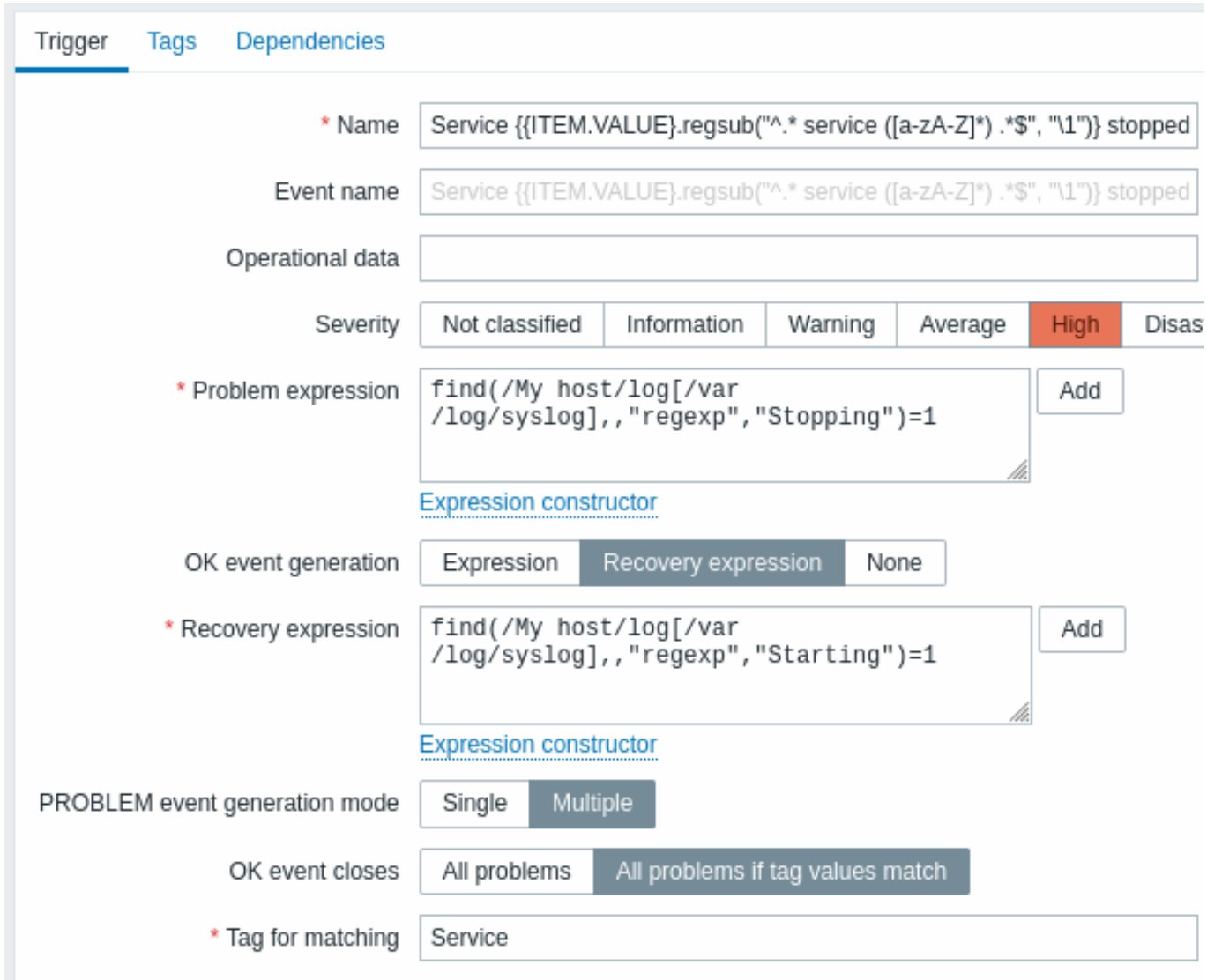
Then define a recovery expression. The following recovery expression will resolve all problems if a log line is found containing the string ”Starting”:

```
find(/My host/log[/var/log/syslog],,"regexp","Starting")=1
```

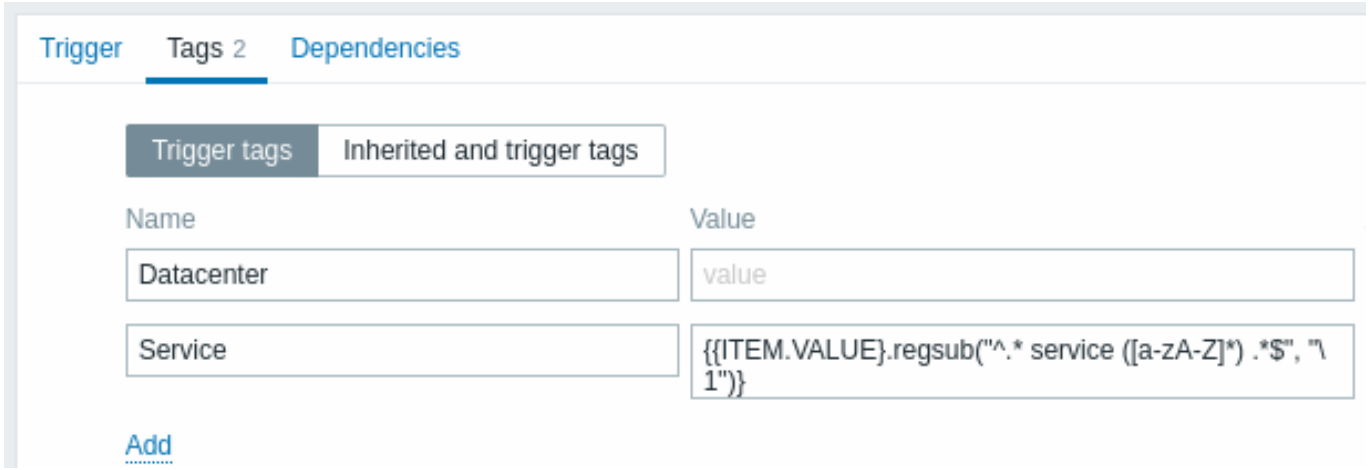
Since we do not want that it’s important to make sure somehow that the corresponding root problems are closed, not just all problems. That’s where tagging can help.

Problems and resolutions can be matched by specifying a tag in the trigger configuration. The following settings have to be made:

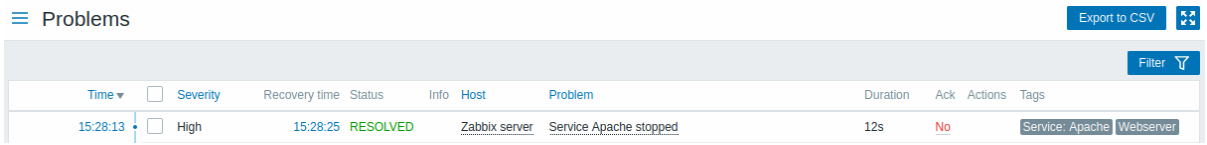
- *Problem event generation mode*: Multiple
- *OK event closes*: All problems if tag values match
- Enter the name of the tag for event matching



- 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);

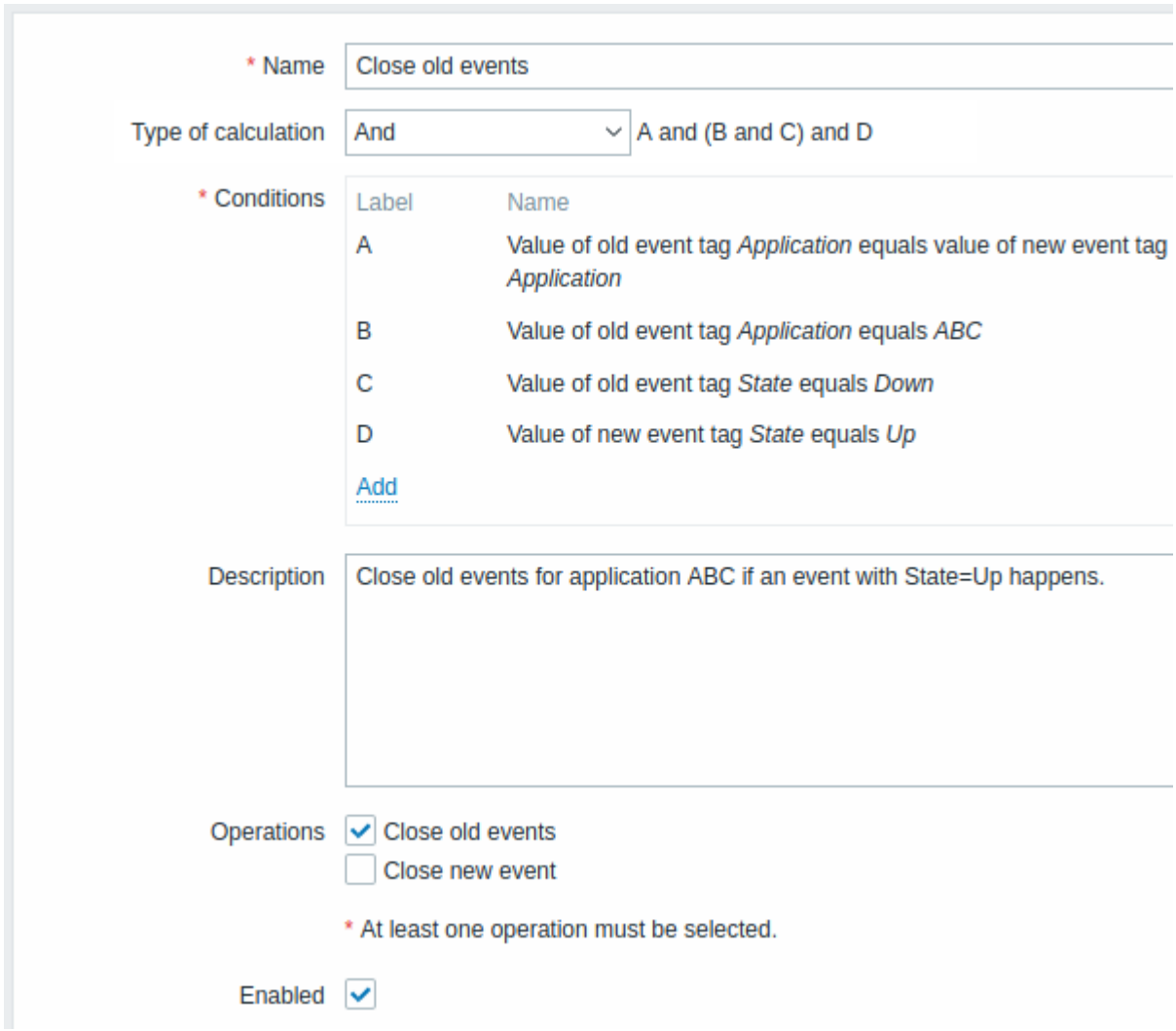
- Avoid using common tag names that may end up being used by different correlation configurations;
- Keep the number of correlation rules limited to the ones you really need.

See also: known issues.

Configuration

To configure event correlation rules globally:

- Go to *Configuration* → *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

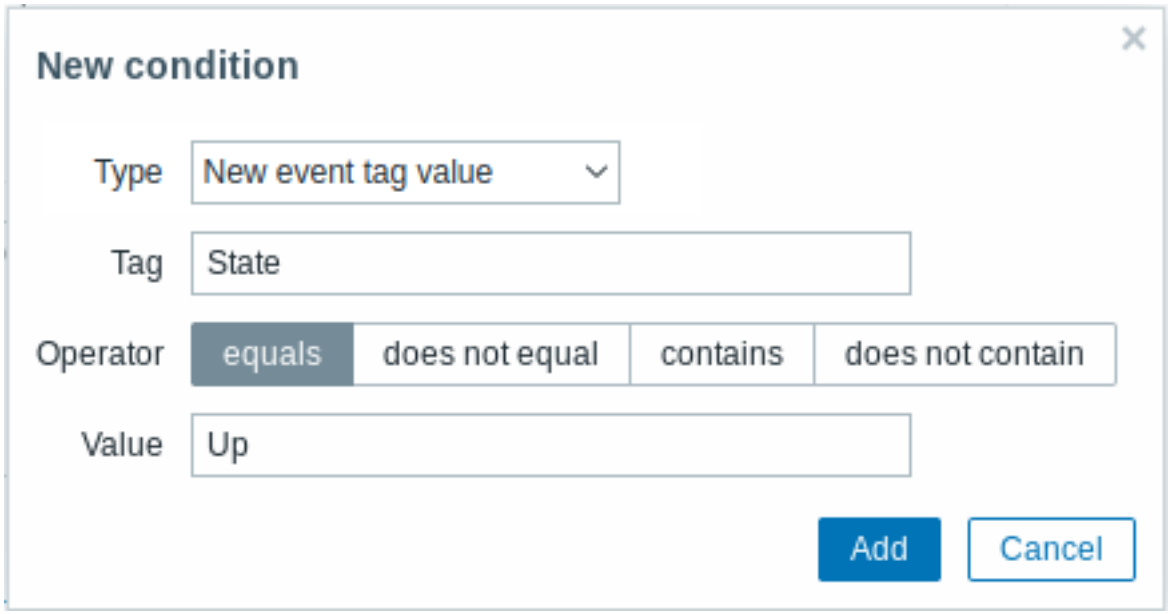


All mandatory input fields are marked with a red asterisk.

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.

Parameter	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

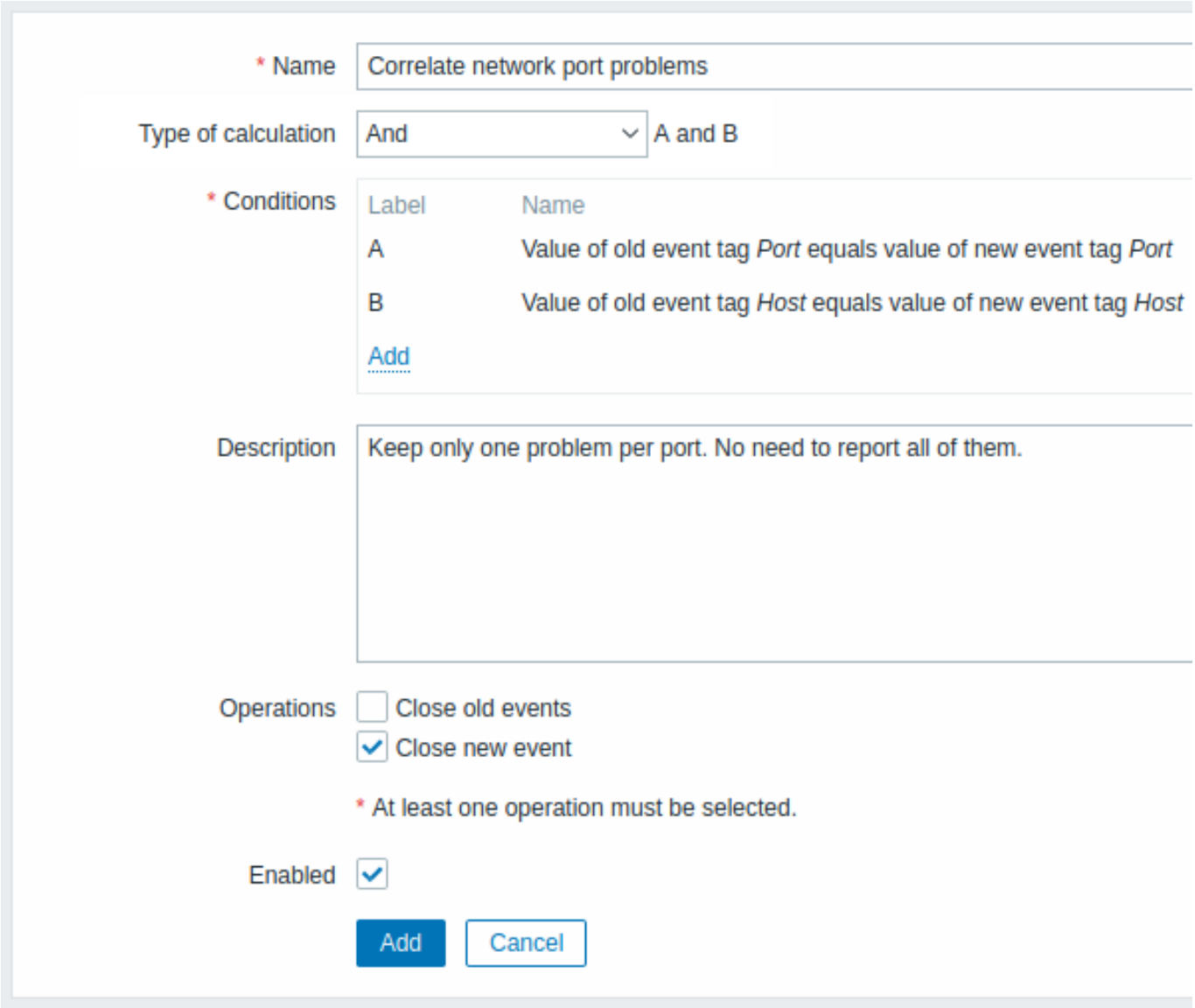
Warning:
Because misconfiguration is possible, when similar event tags may be created for **unrelated** problems, please review the cases outlined below!

- 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.

Examples

Example 1

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

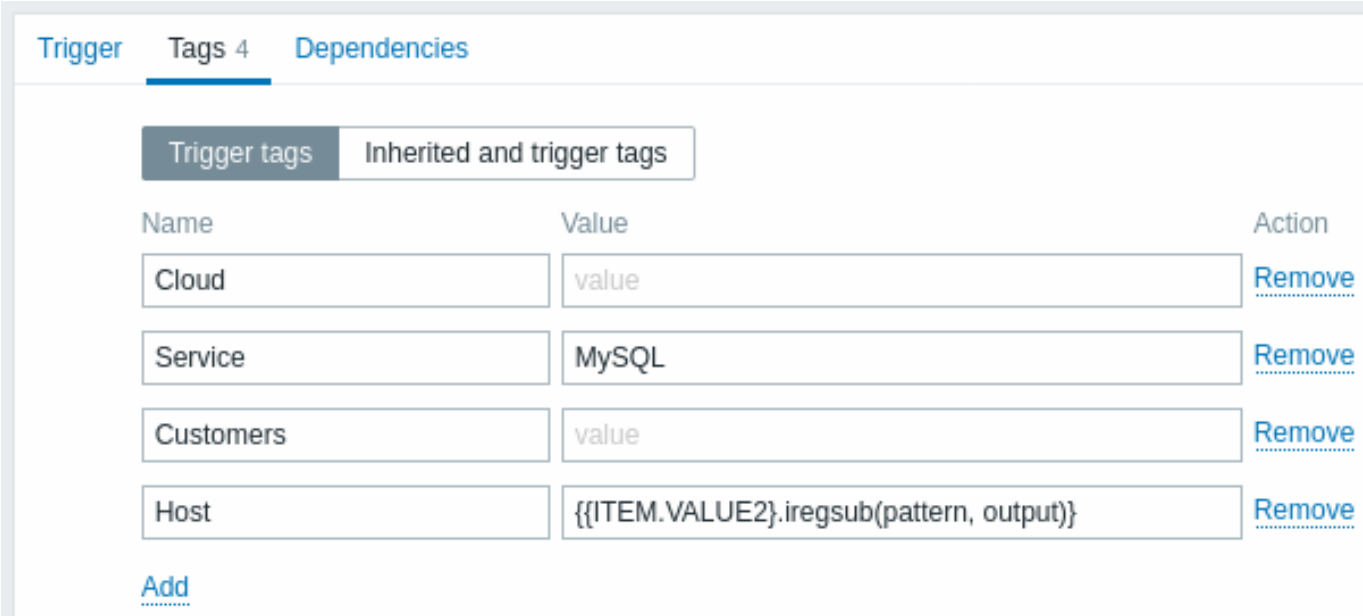
Some use cases for this functionality are as follows:

1. Mark trigger events in the frontend
    - Define tags on trigger level;
    - See how all trigger problems are marked with these tags in *Monitoring* → *Problems*.
2. Mark all template-inherited problems
    - Define a tag on template level, for example 'App=MySQL';
    - See how those host problems that are created by triggers from this template are marked with these tags in *Monitoring* → *Problems*.
3. Mark all host problems
    - Define a tag on host level, for example 'Service=JIRA';
    - See how all problems of the host triggers are marked with these tags in *Monitoring* → *Problems*
4. Group related items
    - Define a tag on item level, for example 'MySQL';
    - See all items tagged as 'MySQL' in *Latest data* by using the tag filter
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 trigger configuration, have multiple problem event generation mode;
    - In 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 on 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 item value.
8. Identify problems better in notifications
    - Define tags on 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 on the template level
    - Define tags on 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.

Configuration

Tags can be entered in a dedicated tab, for example, in trigger configuration:



Macro support

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 macro context is supported for the tag name/value. User macro 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)}
```

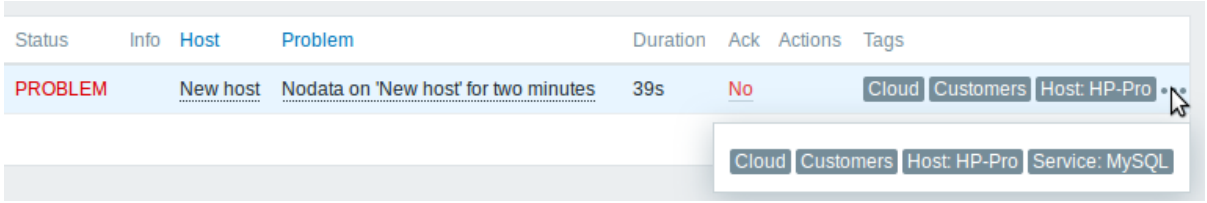
Tag name and value will be cut to 255 characters if their length exceeds 255 characters after macro resolution.

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*
- *Monitoring* → *Dashboard* → *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

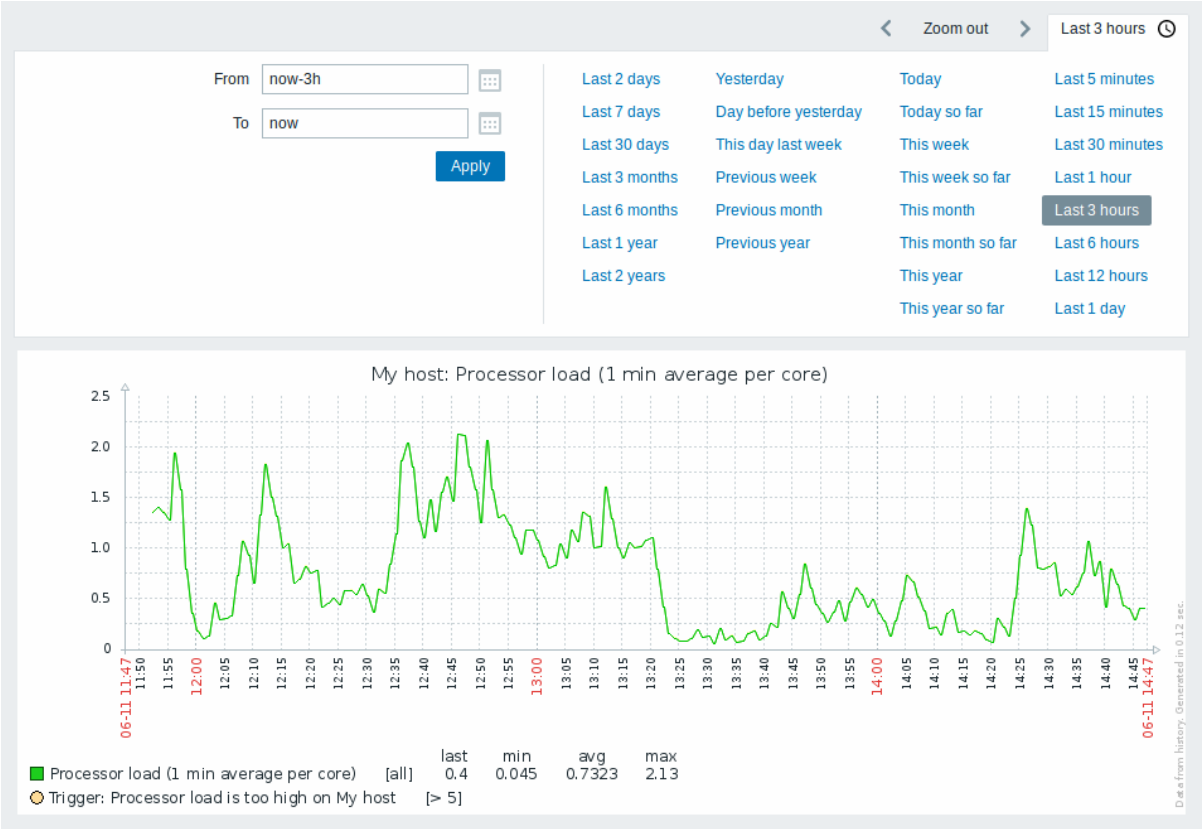
1 Simple graphs

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

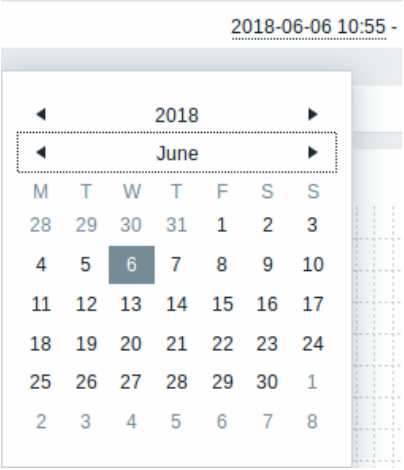
Precision	From	To
M	First day of the month 00:00:00	Last day of the month 23:59:59
y	1st of January of the year 00:00:00	31st of December of the year 23:59:59

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.

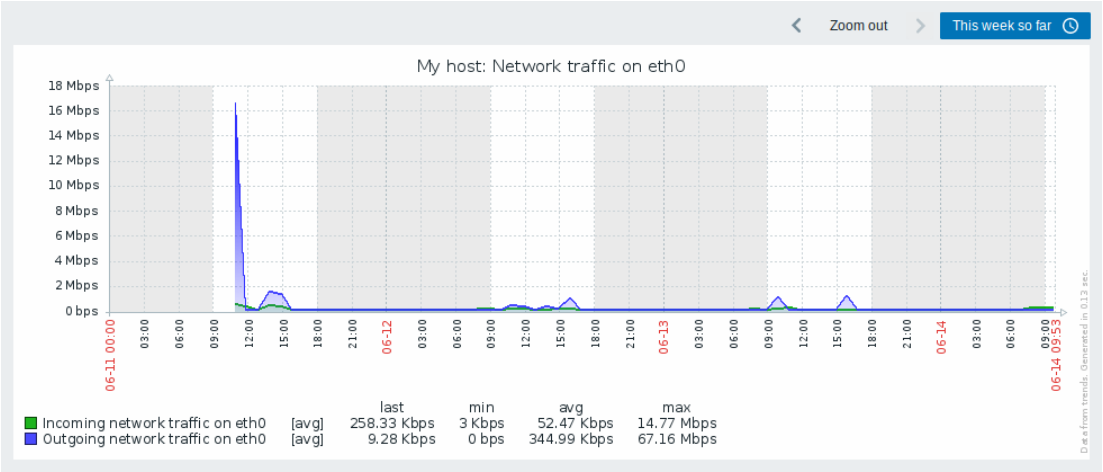
In case no time value is specified or field is left blank, time value will be set to ”00:00:00”. This doesn’t apply to today’s date selection: in that case time will be set to current value.

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).

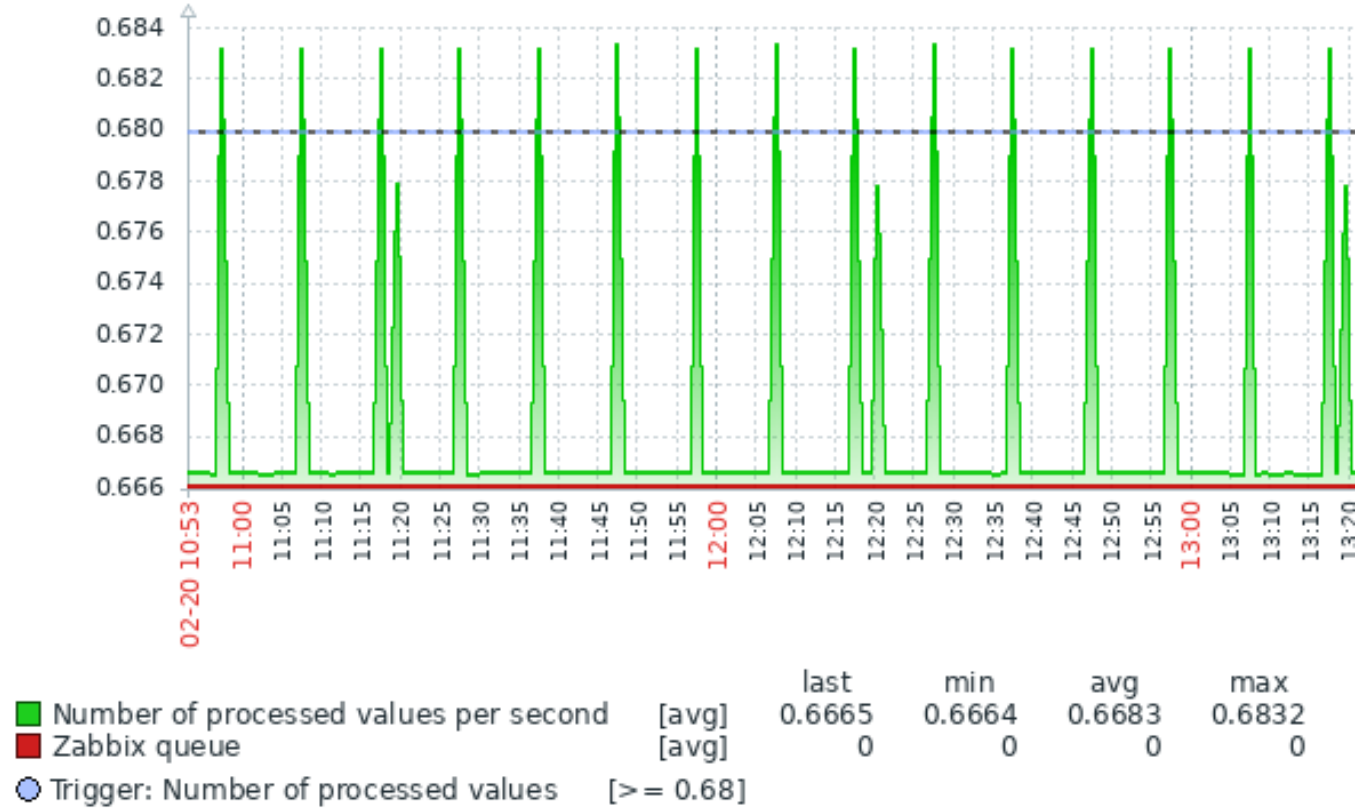


Working time is always displayed in simple graphs, whereas displaying it in custom graphs is a user preference.

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.

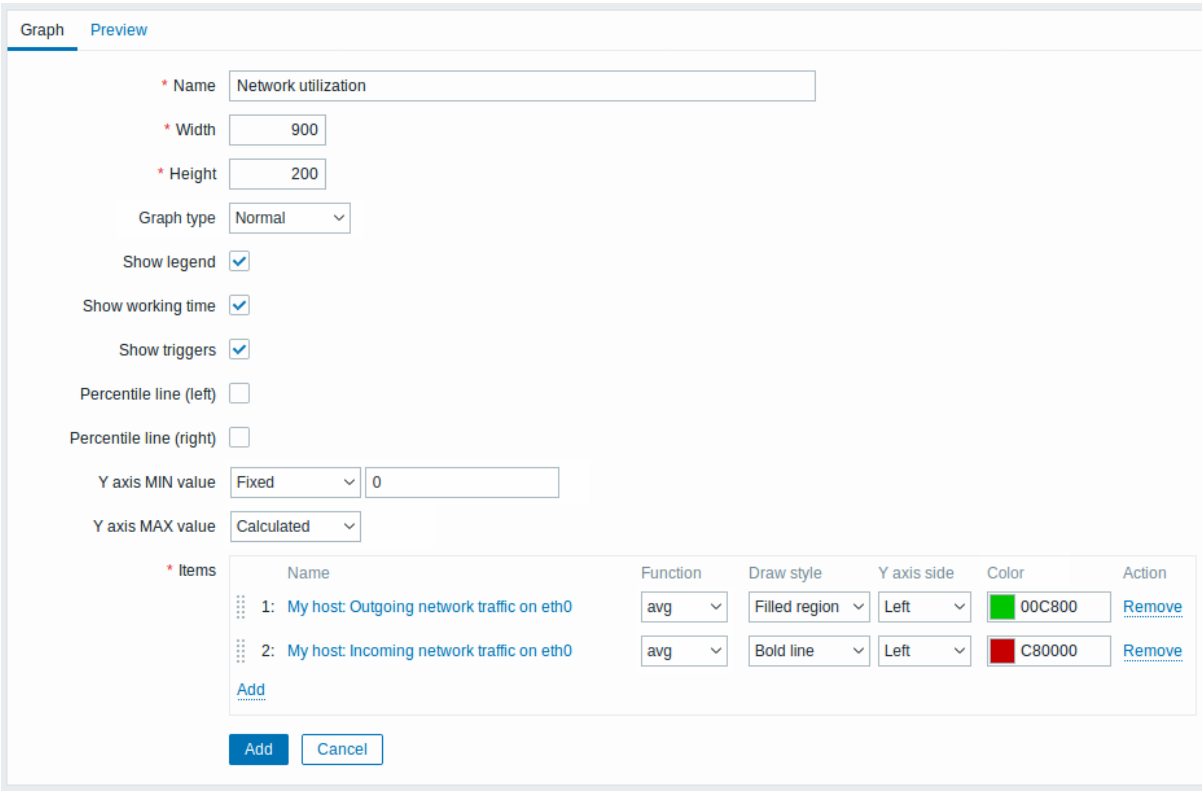
Custom graphs are configured manually.

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 *Configuration → 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.

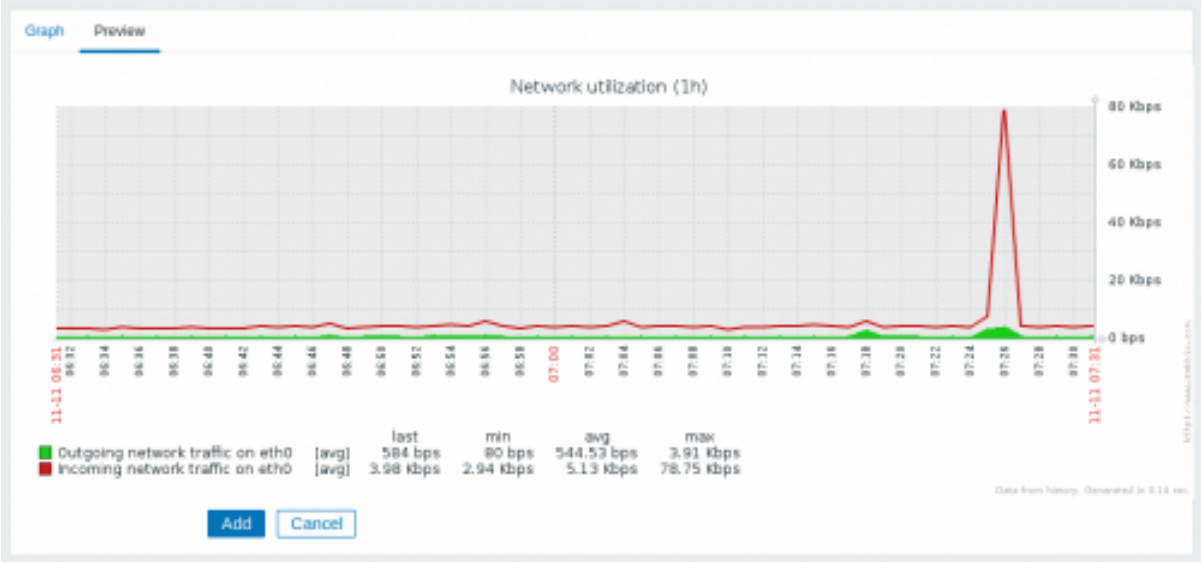
Graph attributes:

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. 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. 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. Not available for pie and exploded pie graphs. **Item** - last value of the selected item will be the minimum value
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. Not available for pie and exploded pie graphs. **Item** - last value of the selected item will be the maximum value
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 by the arrow 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

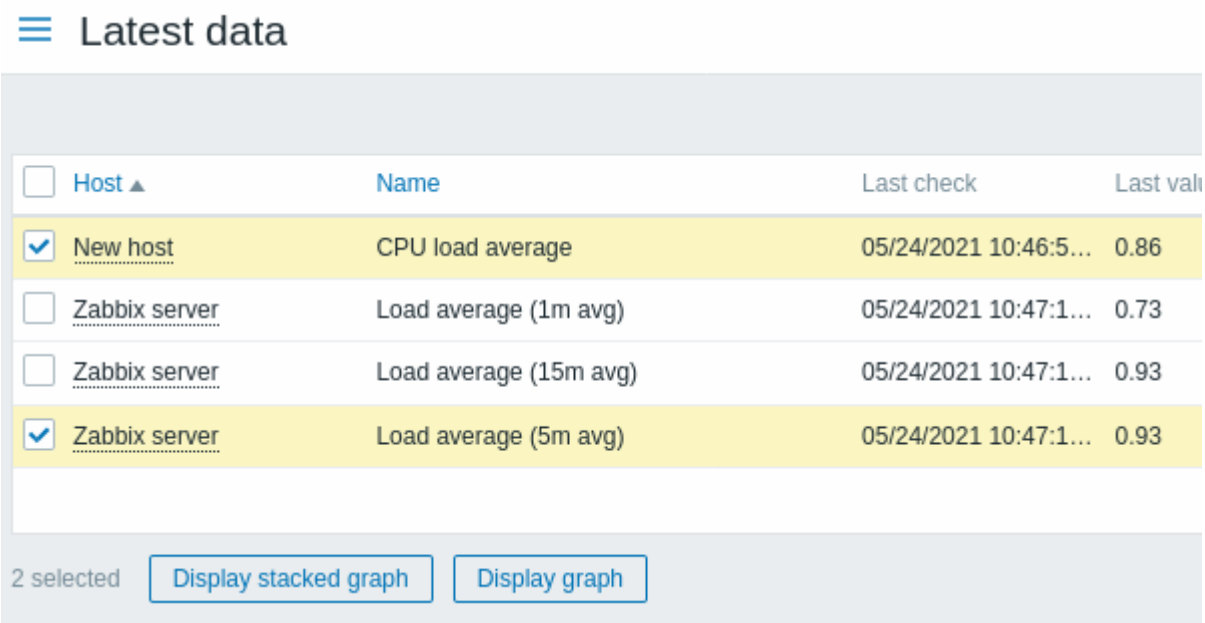
While a simple graph is great for accessing data of one item and custom graphs offer customization options, none of the two allow to quickly create a comparison graph for multiple items with little effort and no maintenance.

To address this issue, since Zabbix 2.4 it is possible to create ad-hoc graphs for several items in a very quick way.

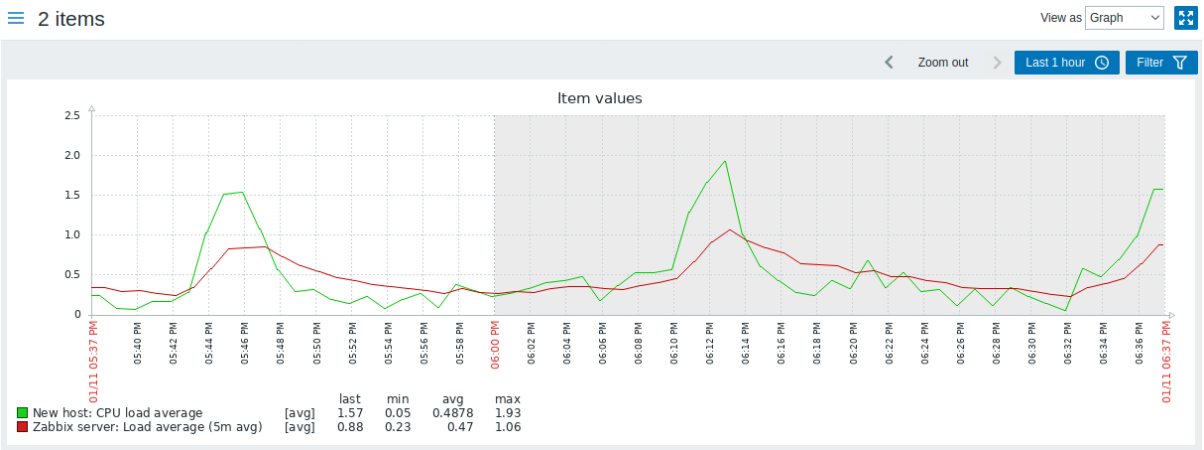
Configuration

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

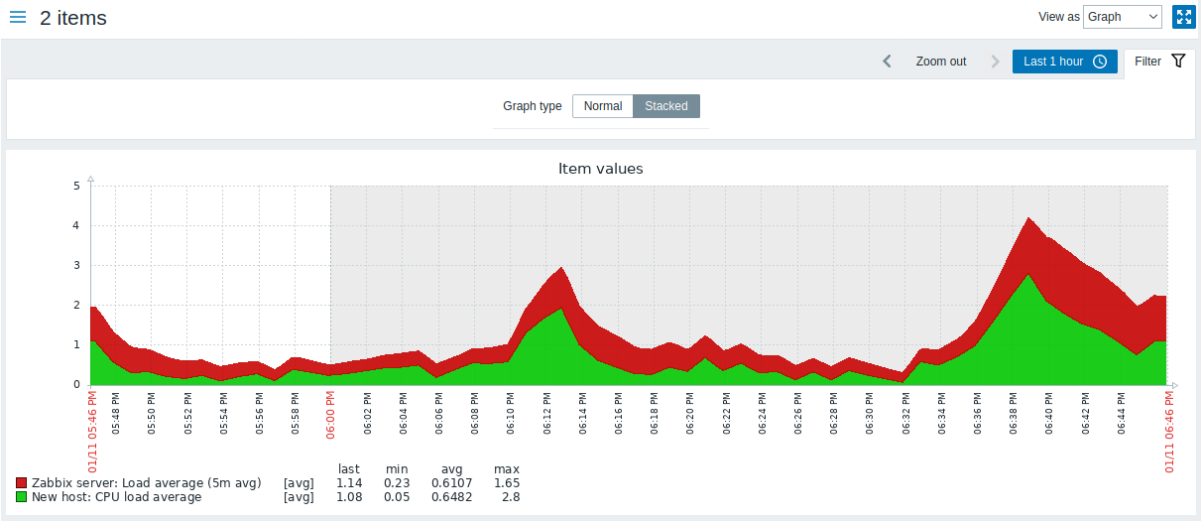


Your graph is created instantly:



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

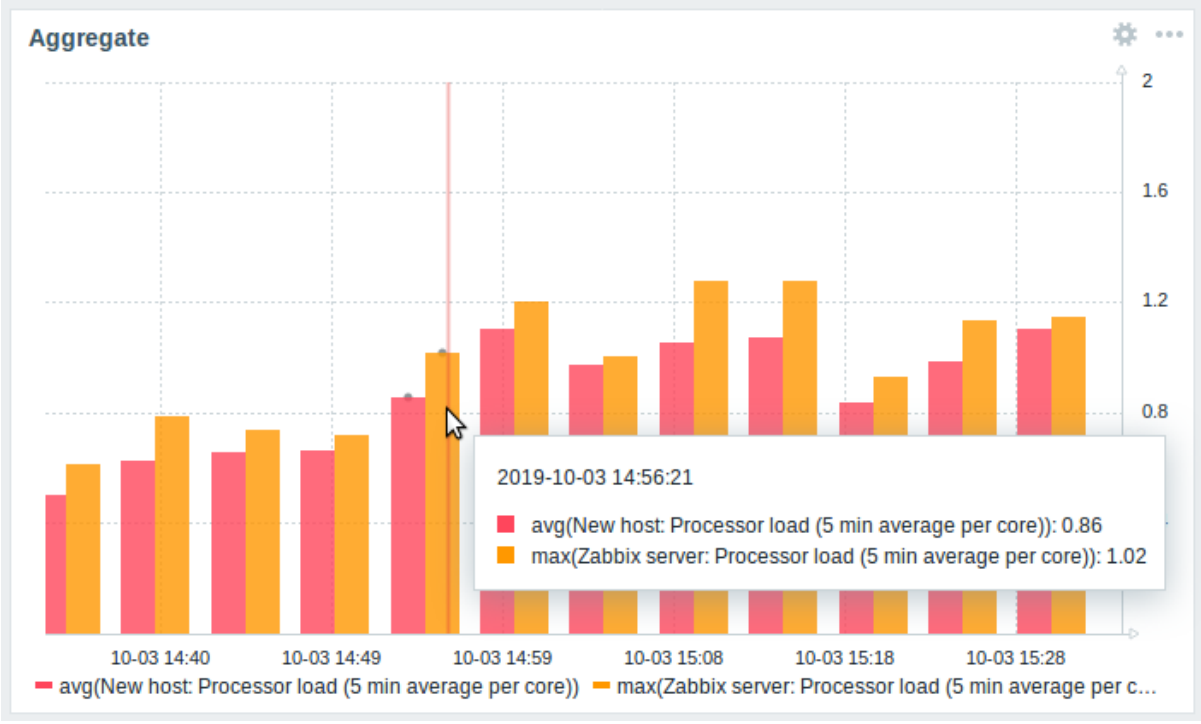
Overview

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)

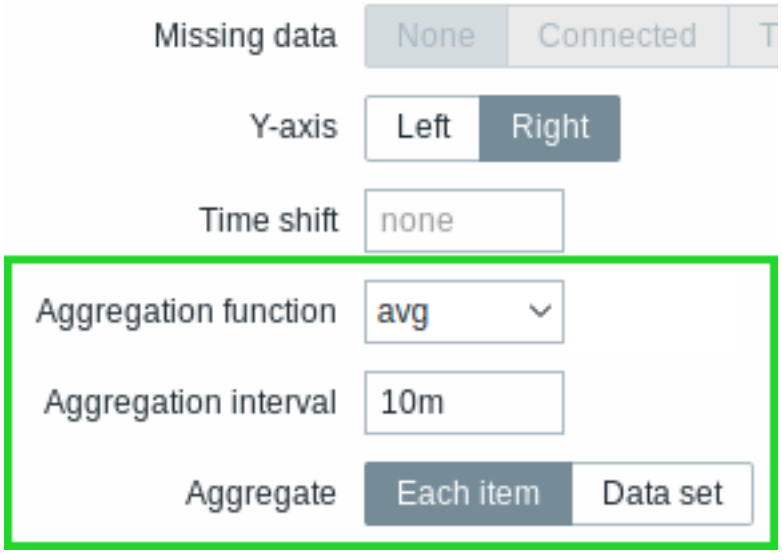
The most exciting use of data aggregation is the possibility to create nice side-by-side comparisons of data for some period:



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. 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.

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.

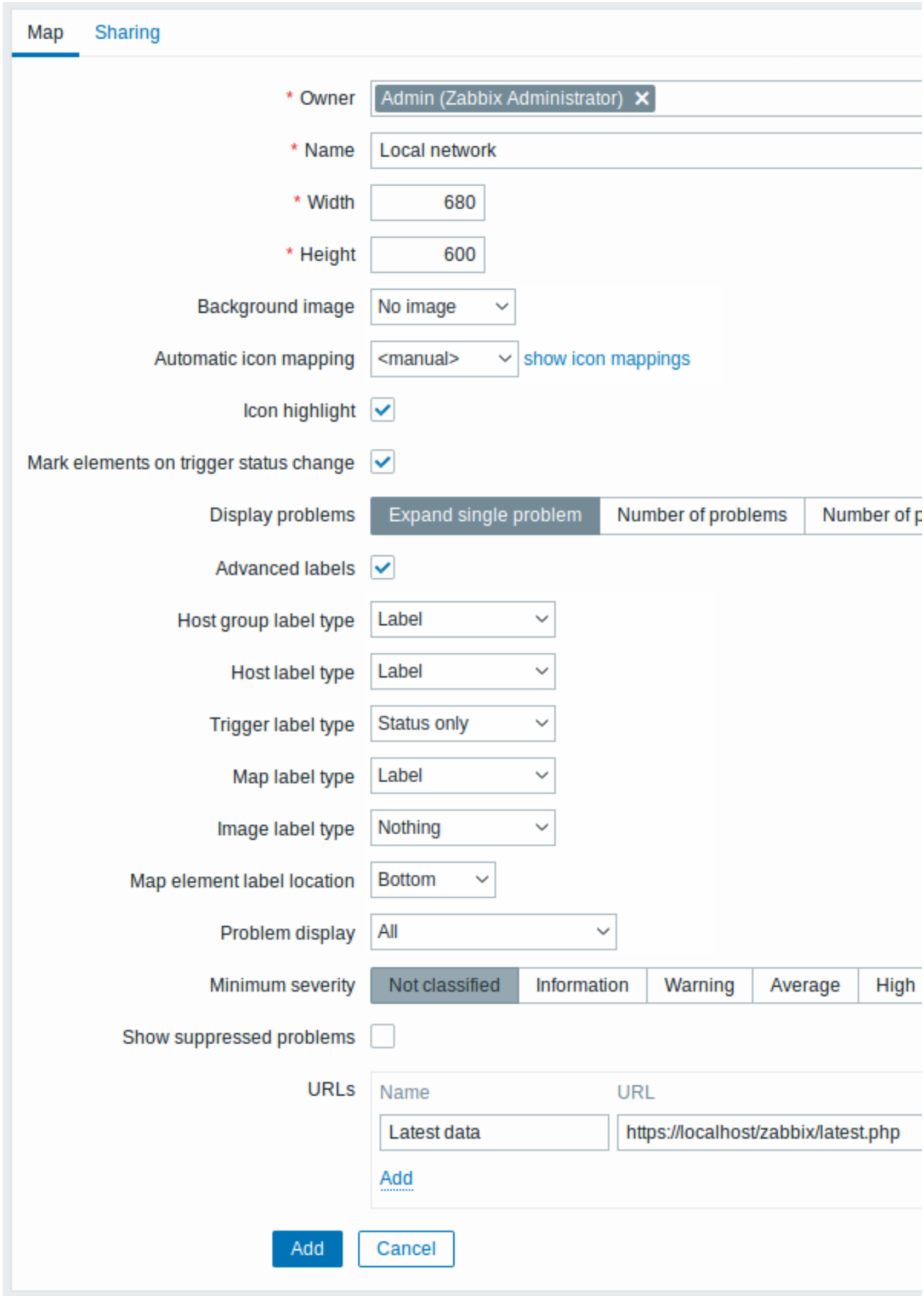
Creating a map

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:



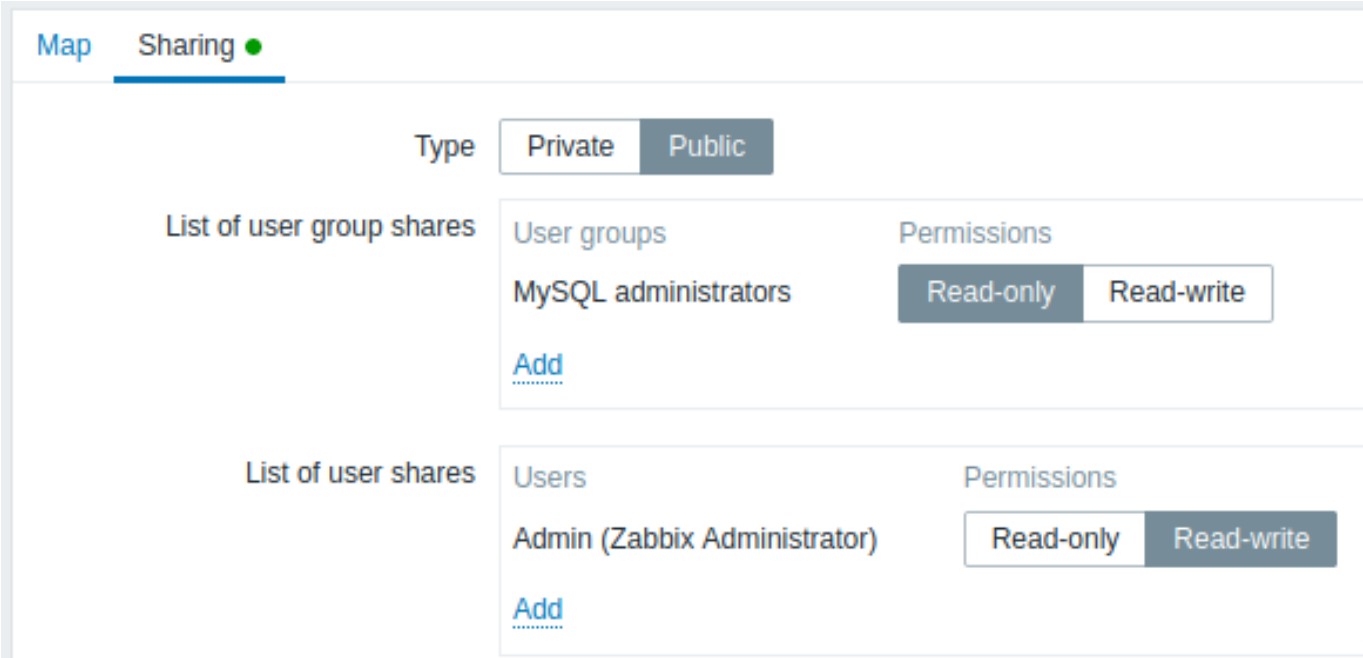
All mandatory input fields are marked with a red asterisk.

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.

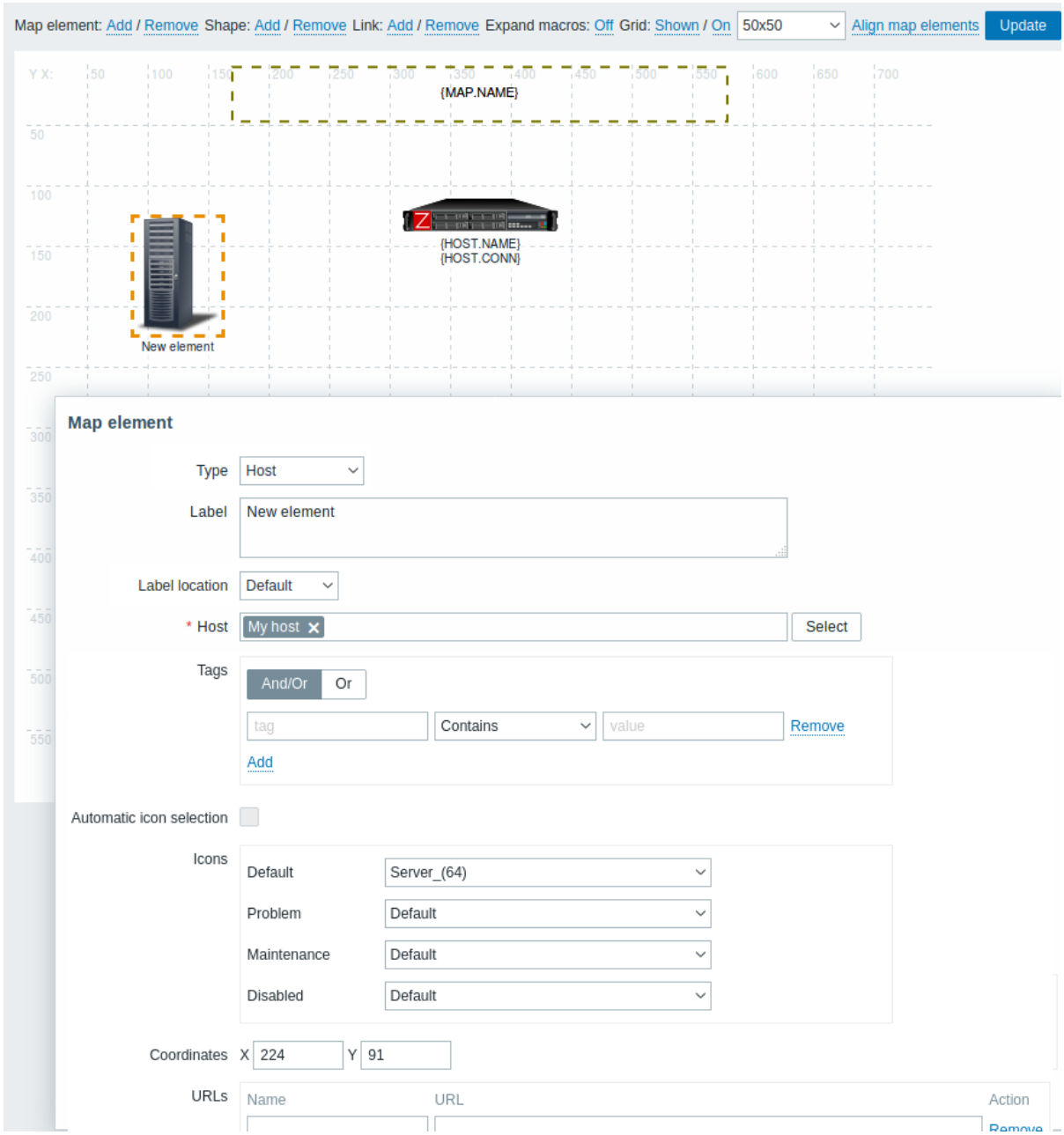
When you click on *Add* to save this map, you have created an empty map with a name, dimensions, and certain preferences. Now you need to add some elements. For that, click on *Constructor* in the map list to open the editable area.

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 with the *Align map elements* button.)

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'.
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'.

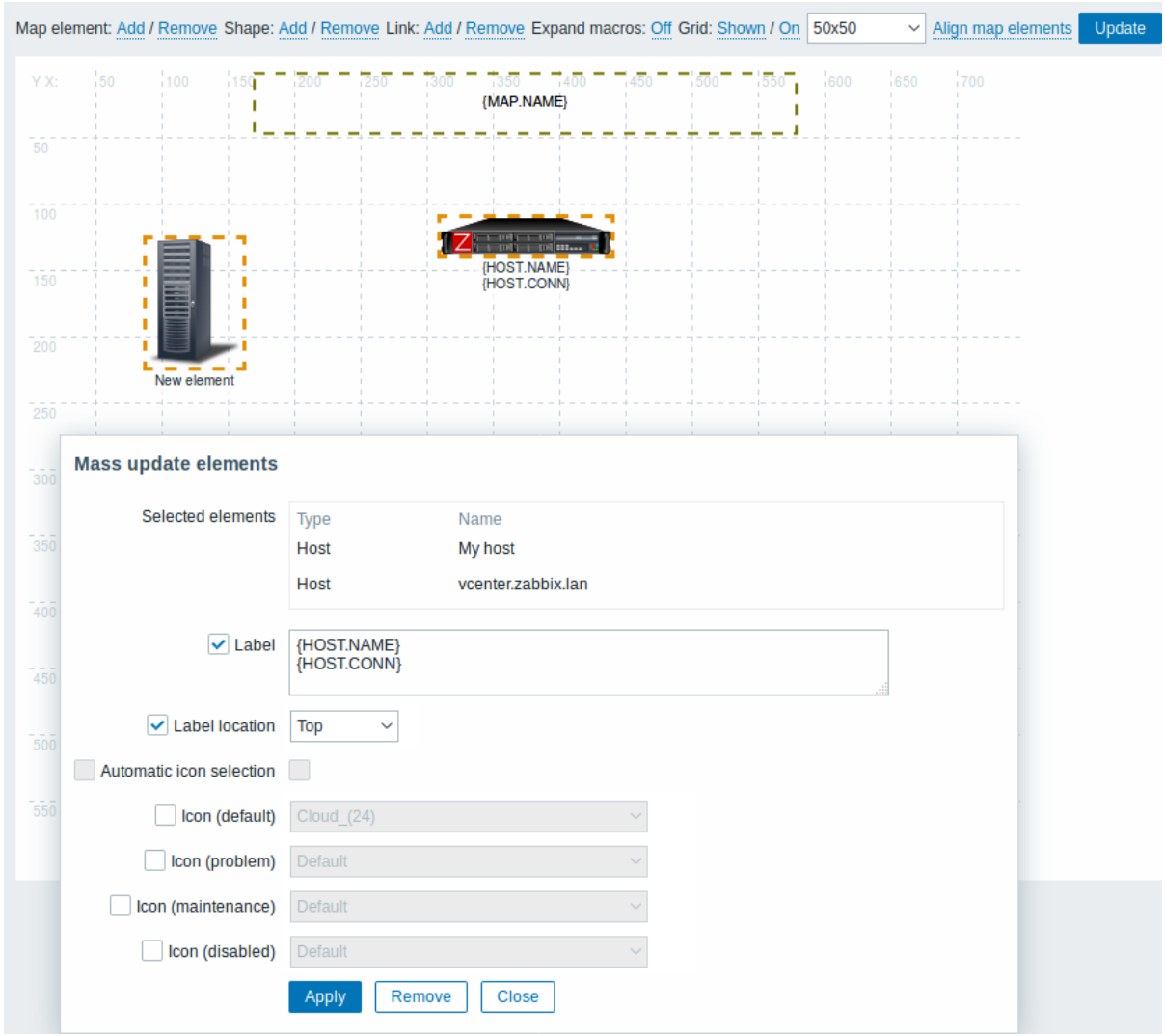
Attention:
Added elements are not automatically saved. If you navigate away from the page, all changes may be lost.
Therefore it is a good idea to click on the **Update** button in the top right corner. Once clicked, the changes are saved regardless of what you choose in the following popup.
Selected 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.

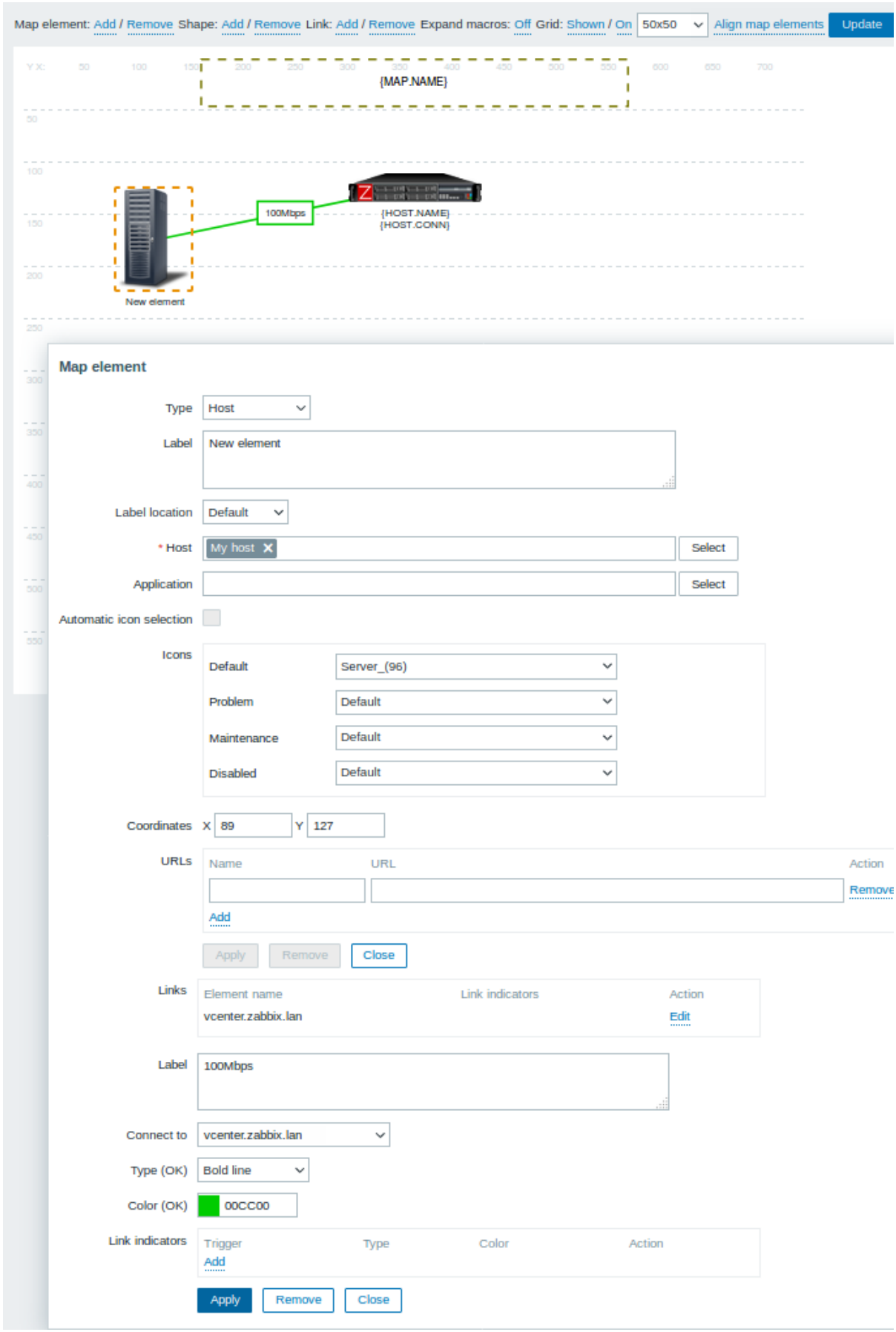
Once you select more than one element, the element property form shifts to the mass-update mode so you can change attributes of selected elements in one go. To do so, mark the attribute using the checkbox and enter a new value for it. You may use macros here (for example, {HOST.NAME} for the element 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:

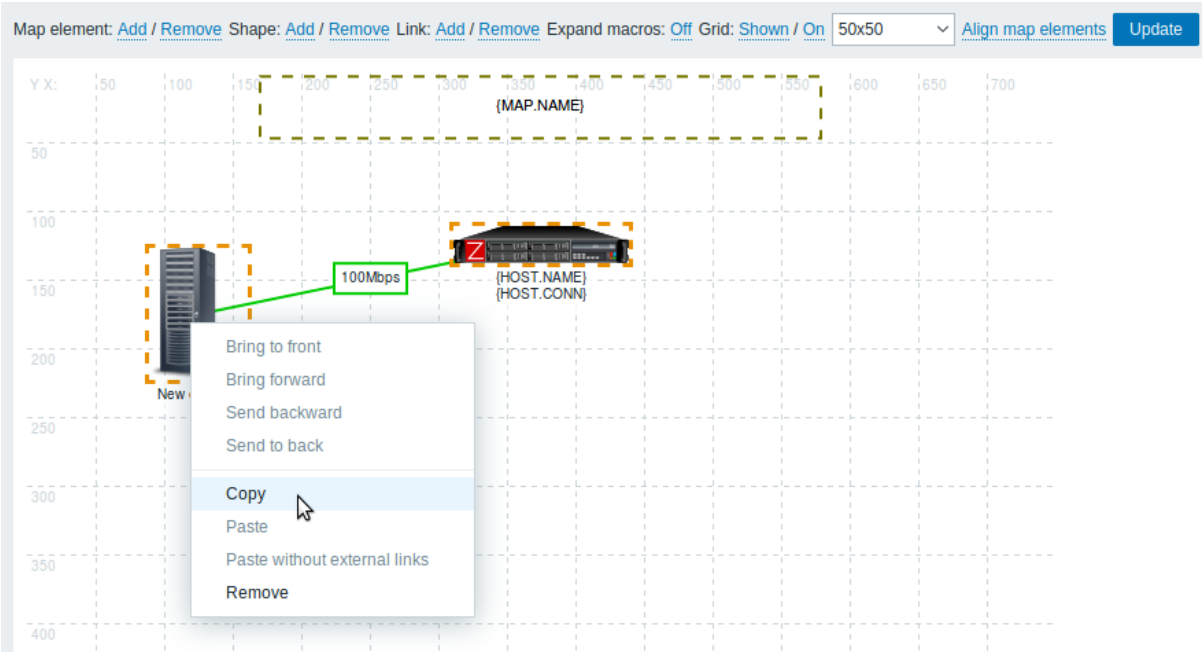
Parameter	Description
Label	Label that will be rendered on top of the link. Expression macros are supported in this field, but only with `avg`, `last`, `min` and `max` functions, with time as parameter (for example, `{?avg(/host/key,1h)}`).

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.

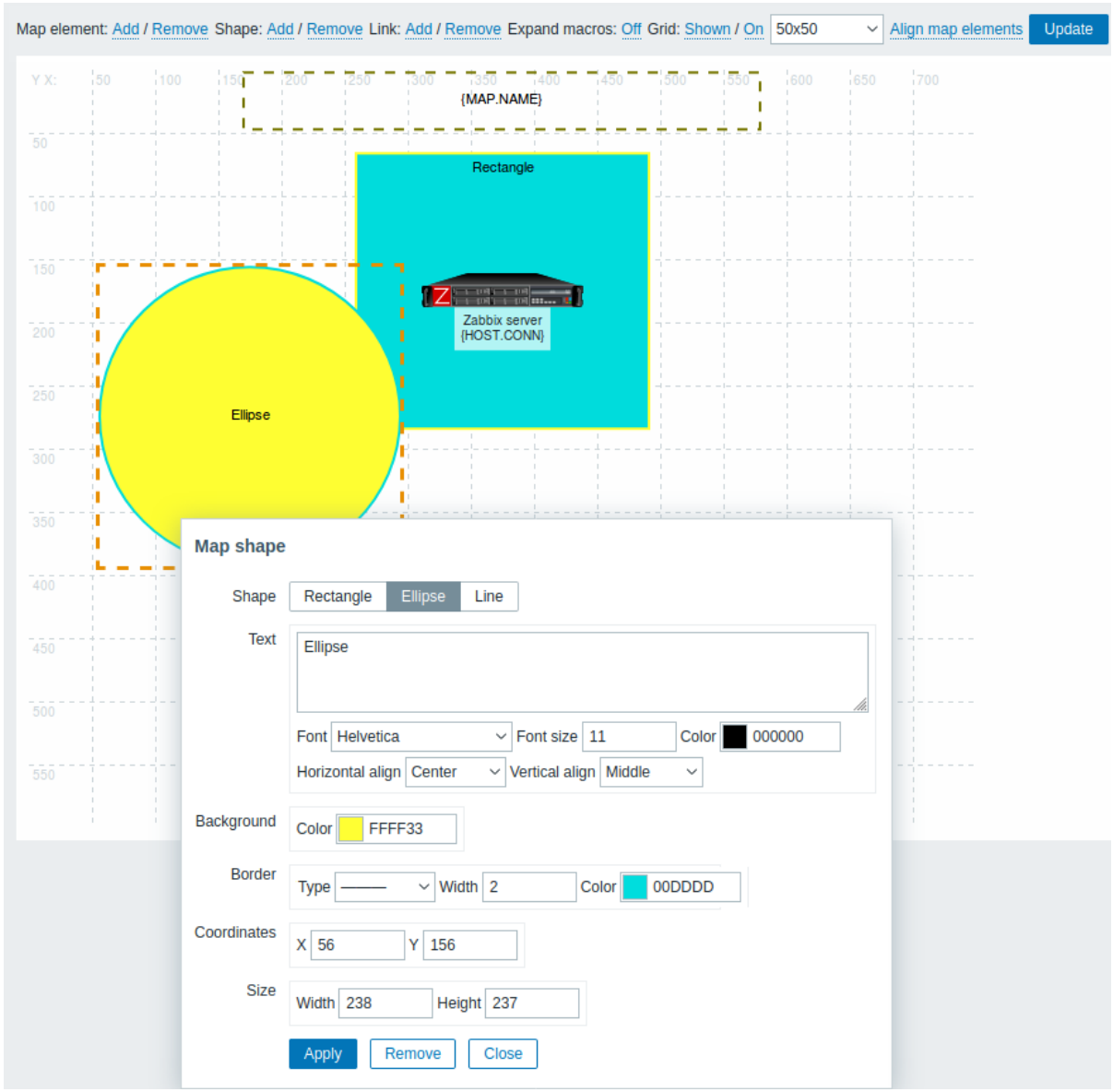
Copy-pasting works within the same browser window. Keyboard shortcuts are not supported.

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.

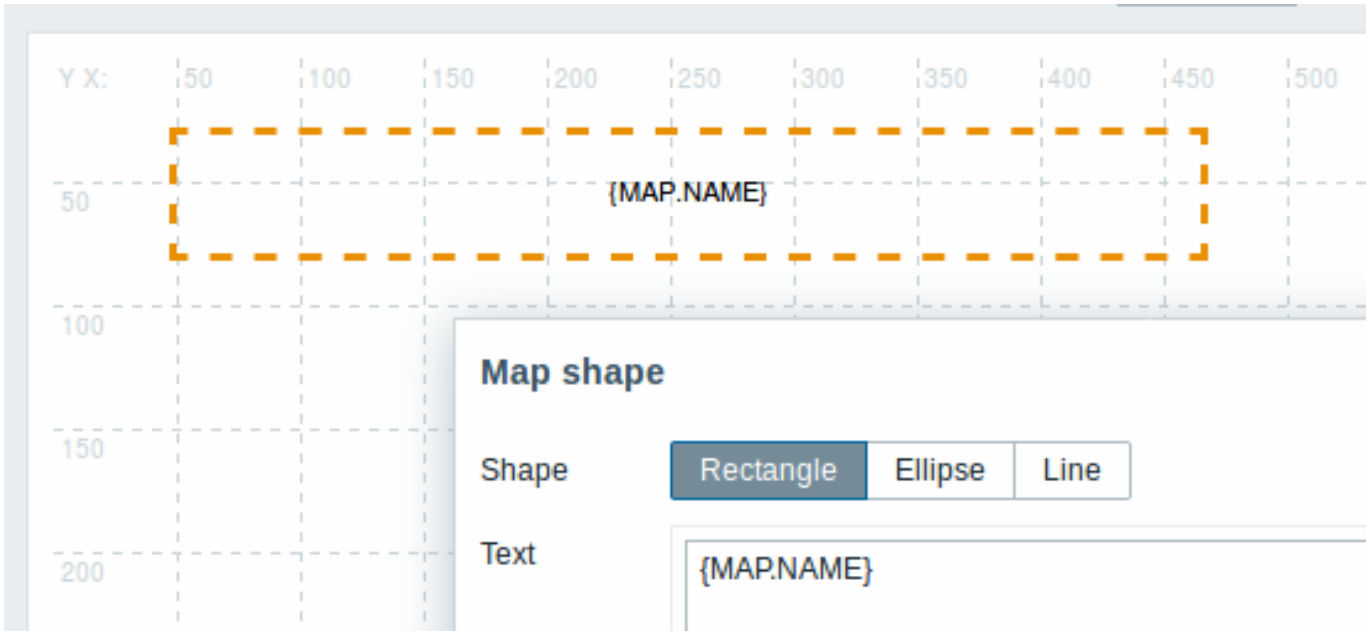
A new shape is added with default colors. By clicking on the shape, a form is displayed and you can customize the way a shape looks, add text, etc.



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.

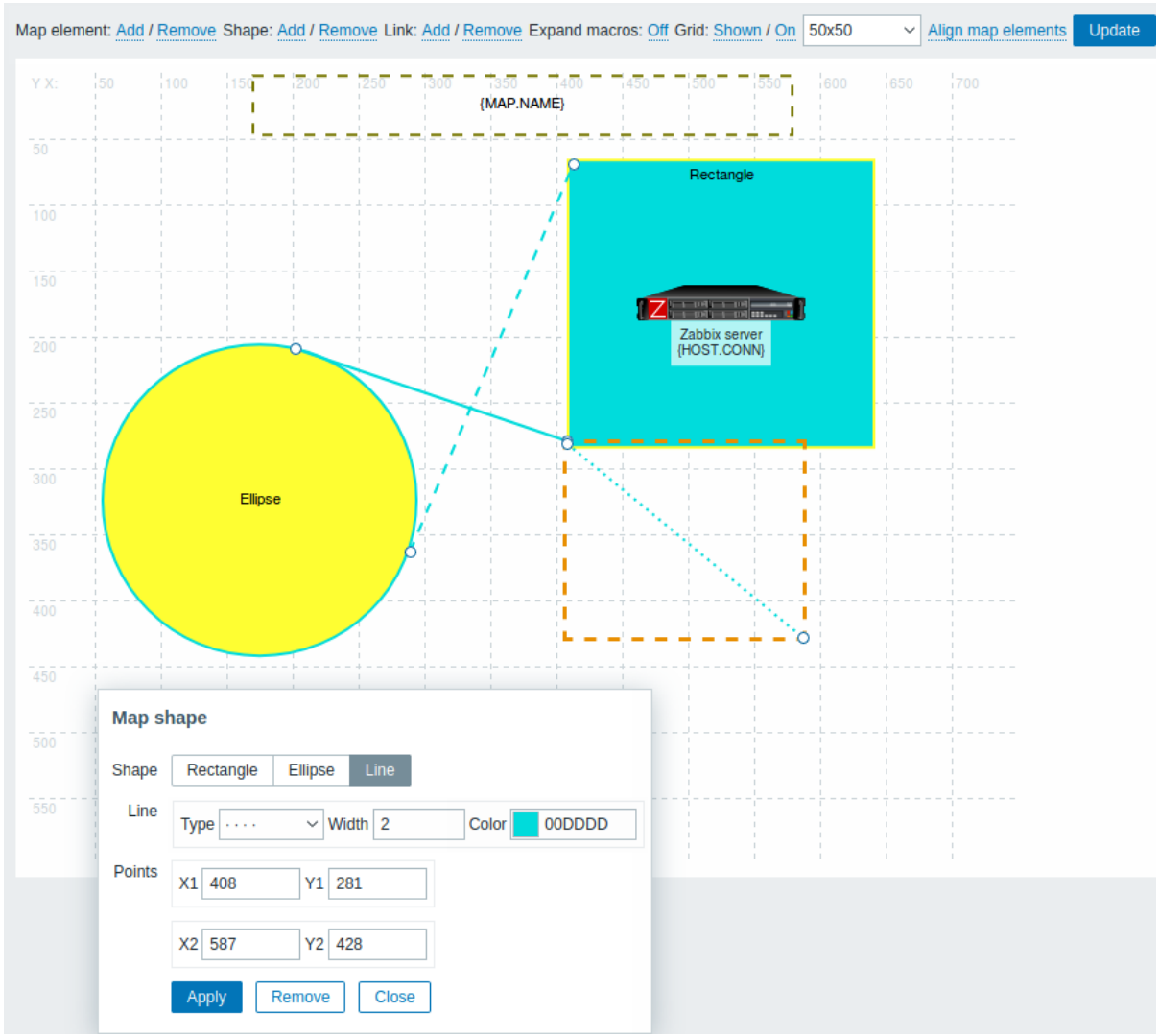
If hyperlinks are used in the text, they become clickable when viewing the map.

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

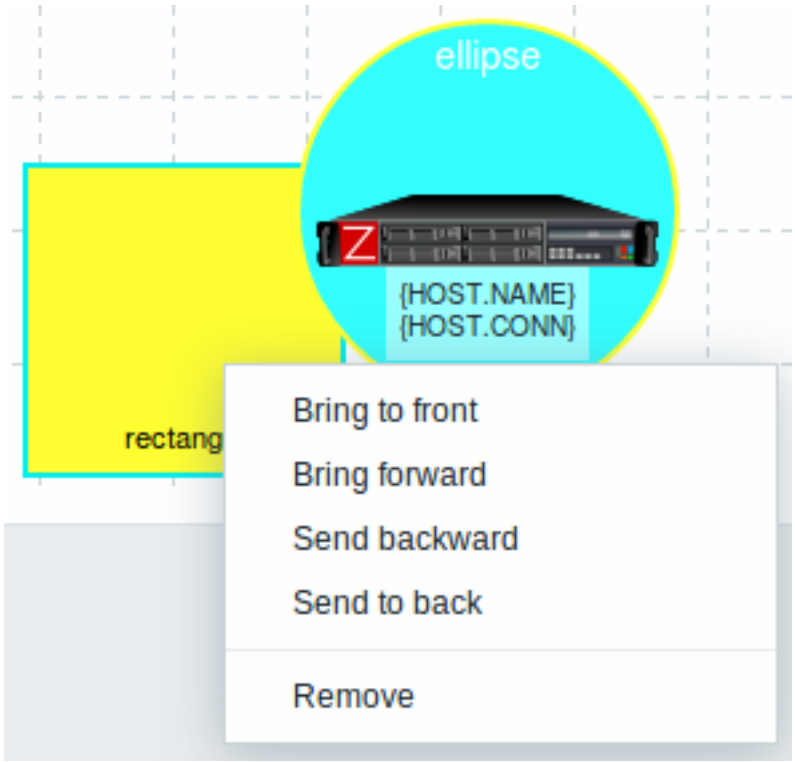
In addition to shapes, it is also possible to add some lines. Lines can be used to link elements or shapes in a map.

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.

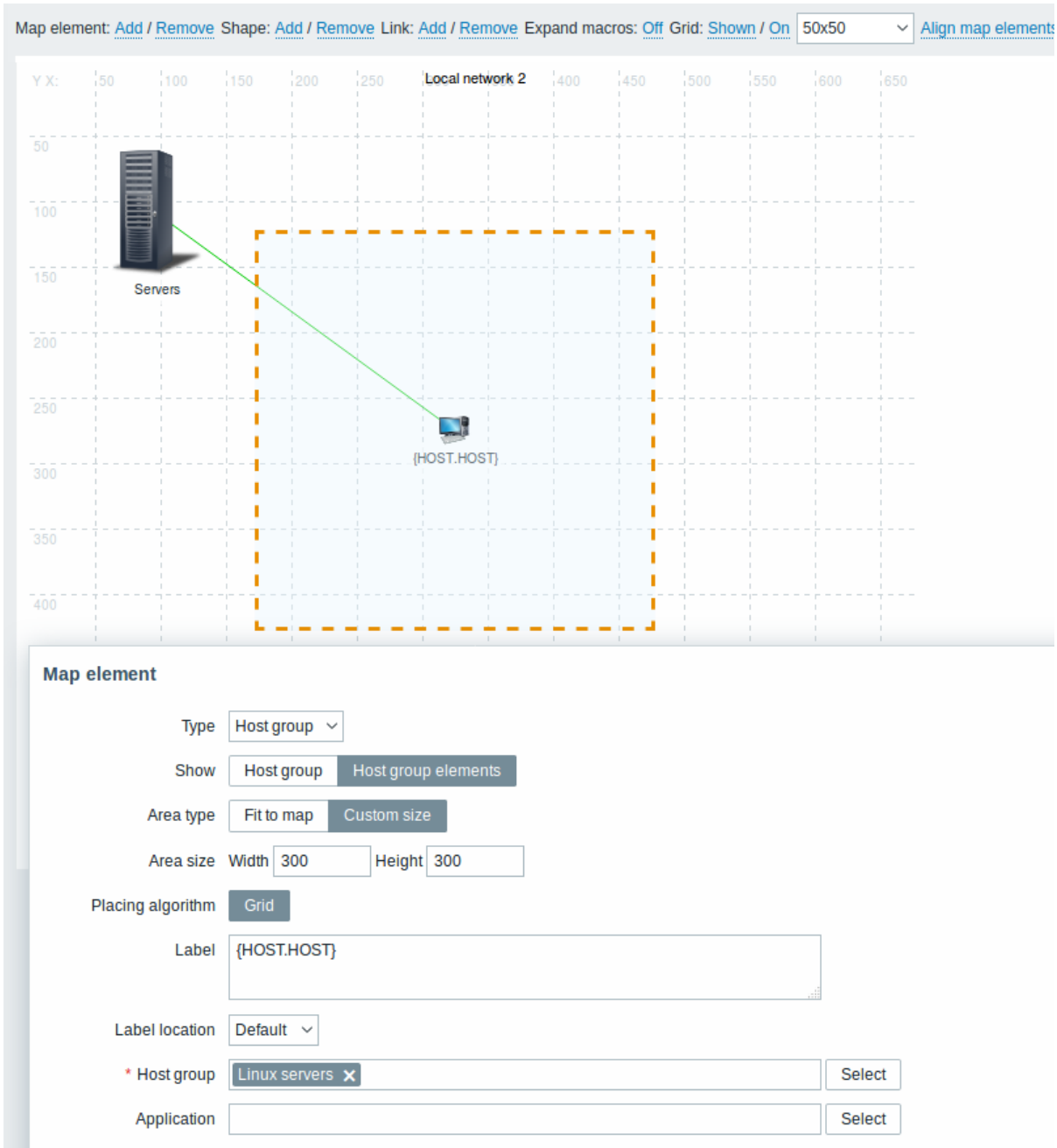


2 Host group elements

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.

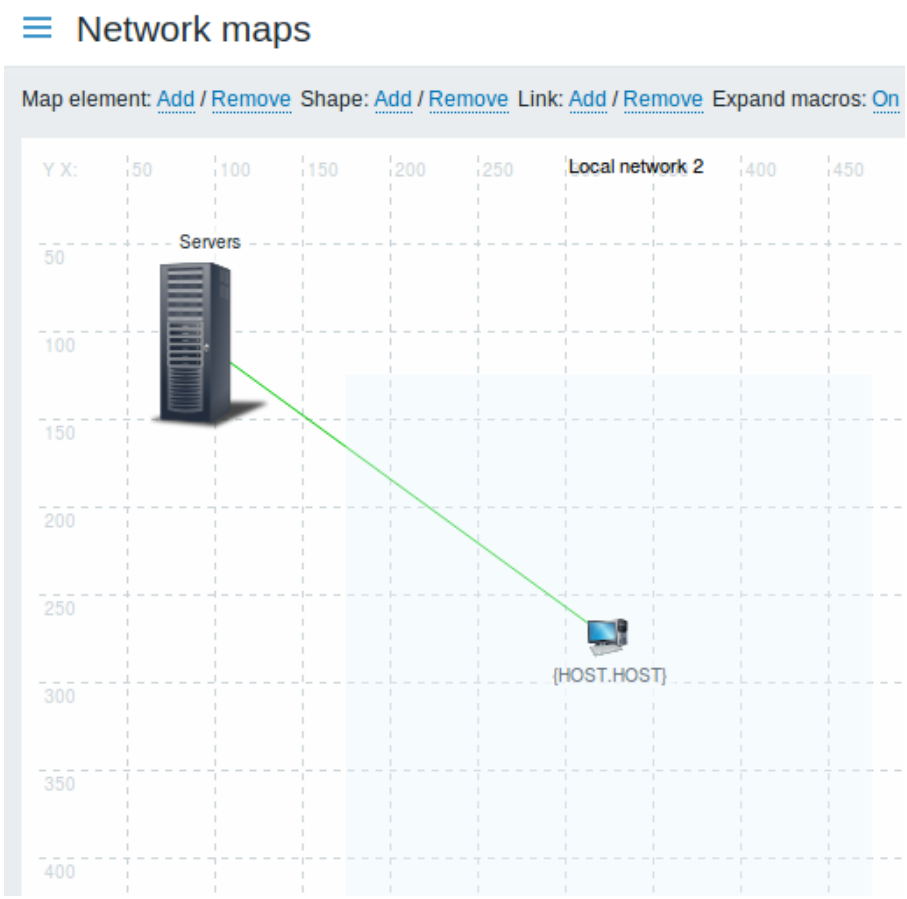
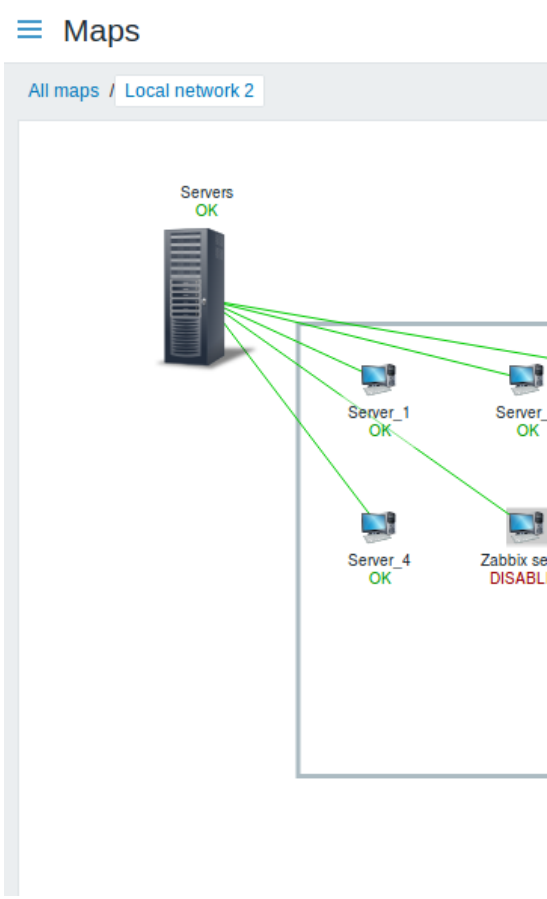
This table consists of parameters typical for *Host group* element type:

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.

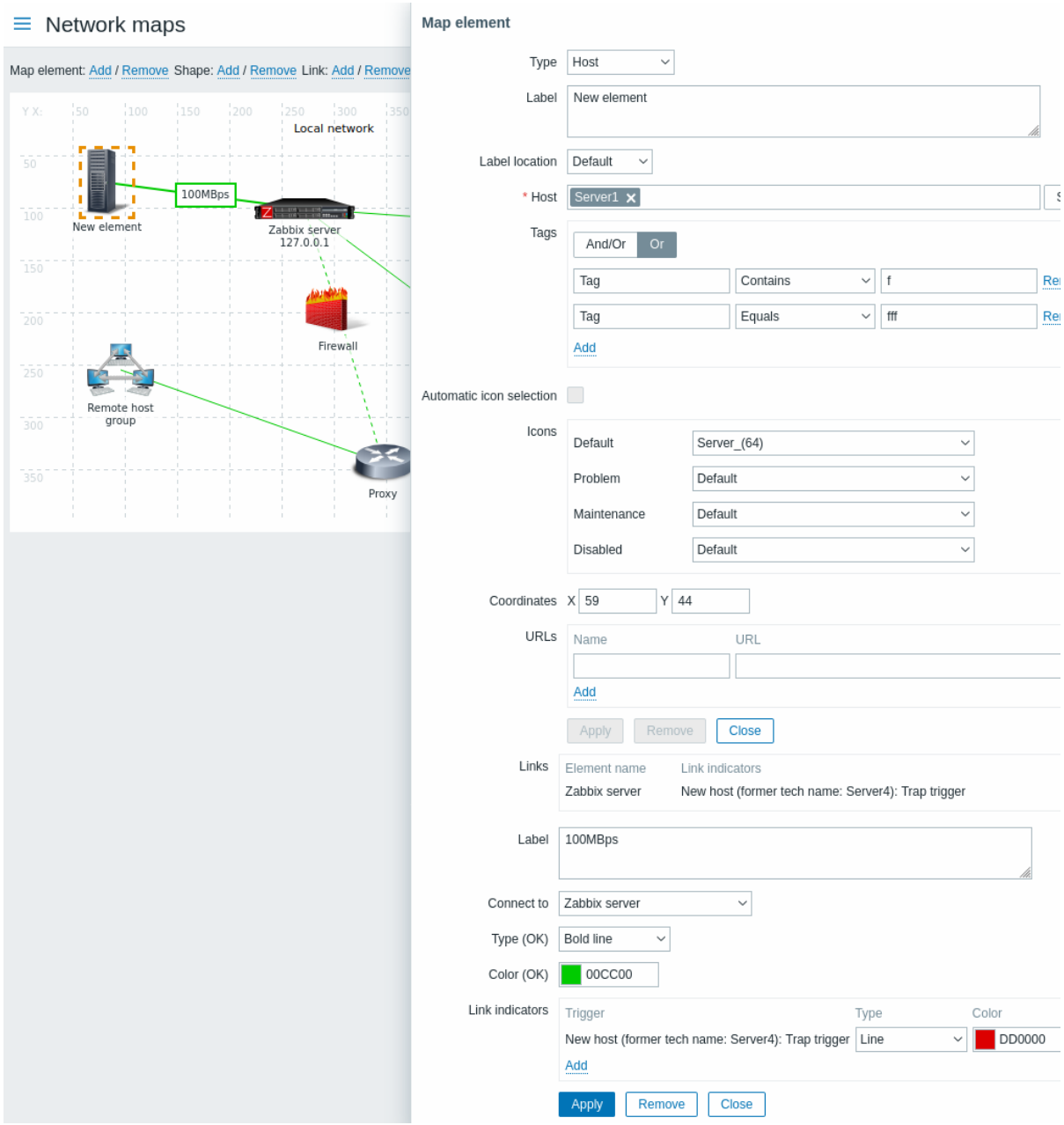
When you configure a link, you set the default link type and color. When you assign triggers to a link, you can assign different link types and colors with these triggers.

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



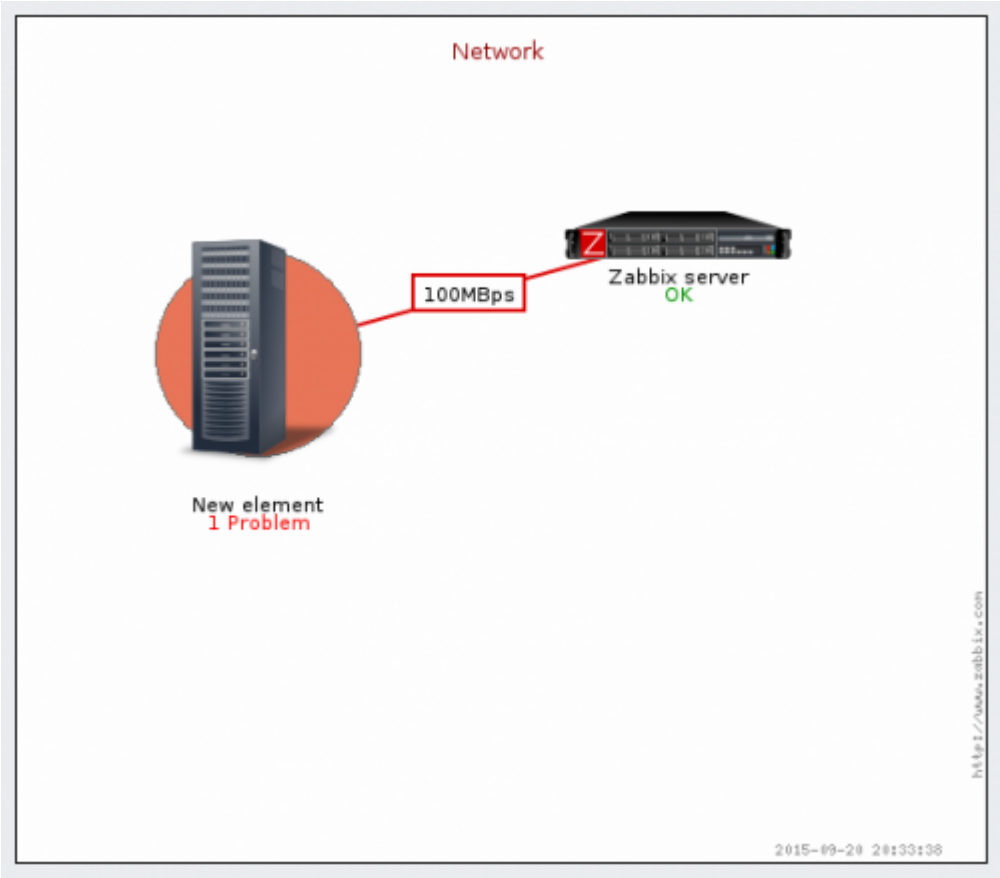
All mandatory input fields are marked with a red asterisk.

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.

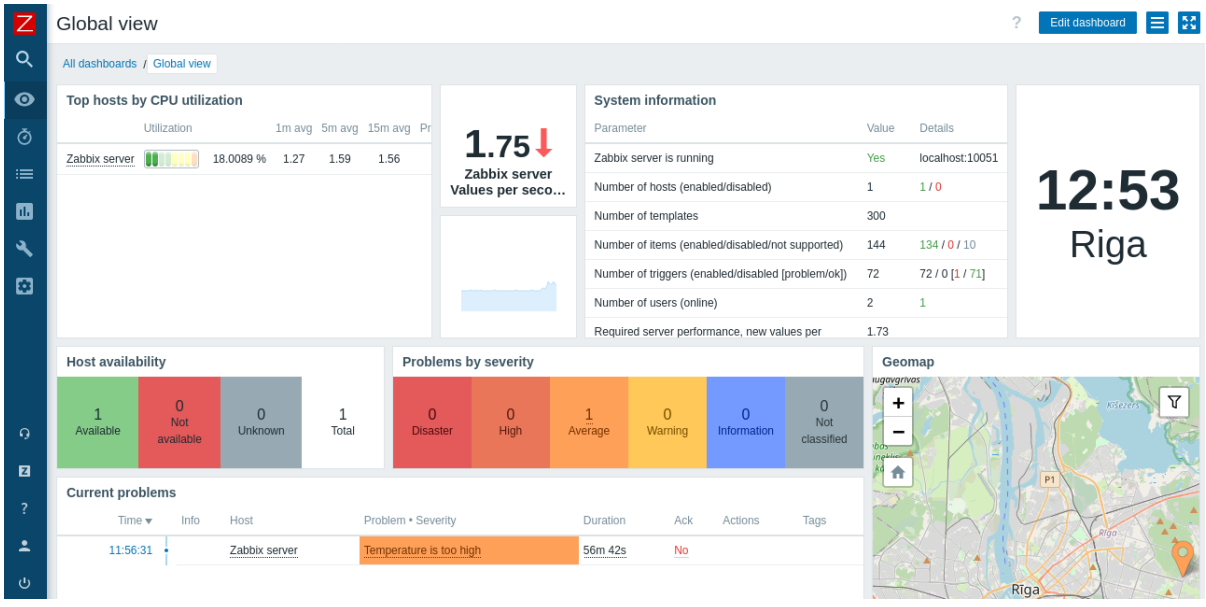


Note:
If multiple triggers go into a problem state, the problem with the highest severity will determine the link style and color. If multiple triggers with the same severity are assigned to the same map link, the one with the lowest ID takes precedence. Note also that:

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.



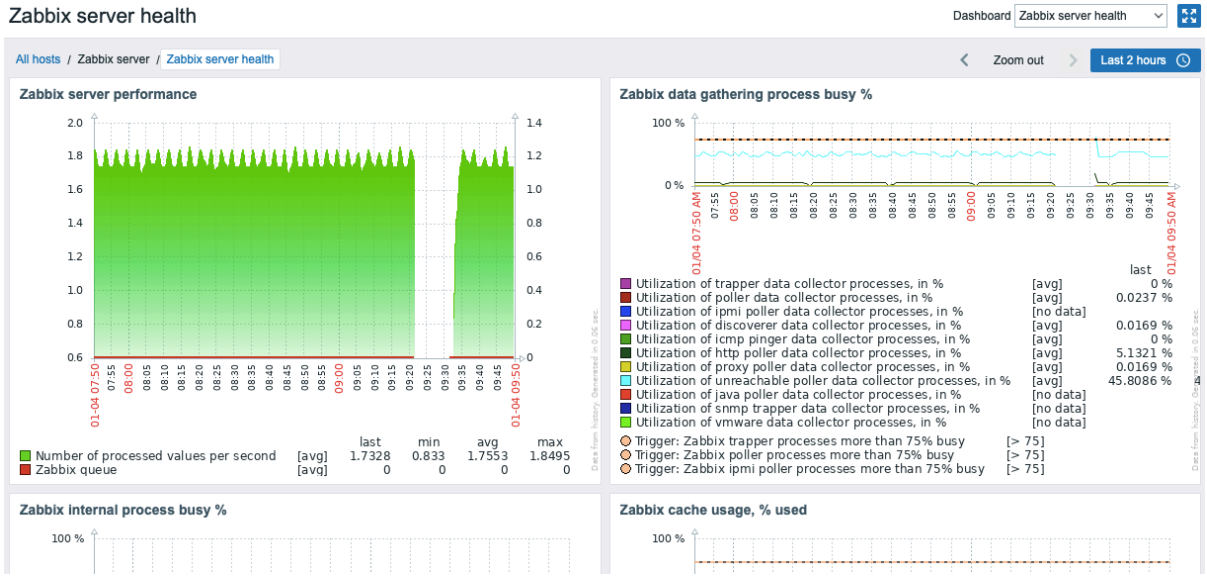
4 Host dashboards

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 *Monitoring* → *Dashboard* 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.

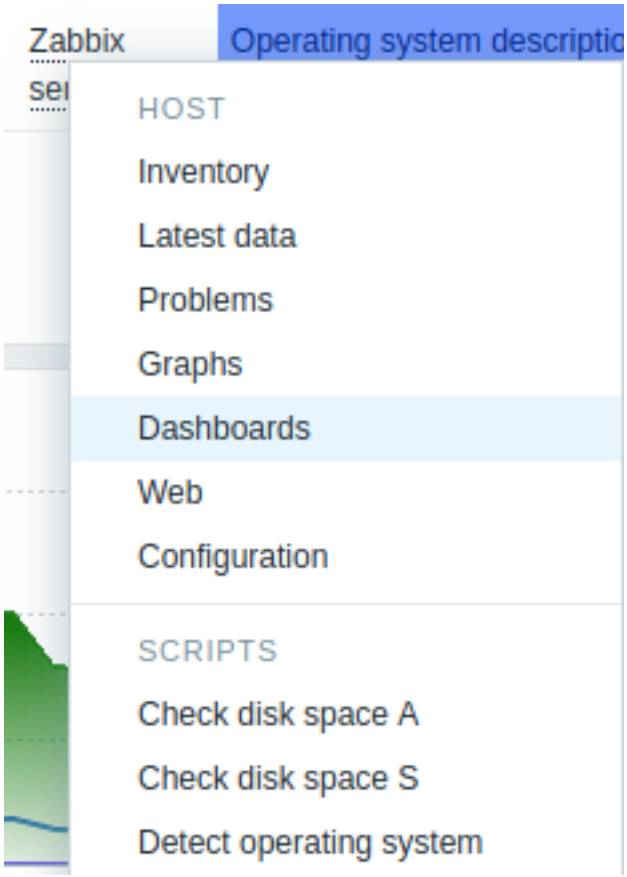
Widgets of the host dashboards cannot be edited.

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 drop-down 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 *Configuration* → *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 *Configuration* → *Templates* you can import them manually from this directory.
- It is also possible to download the template from Zabbix git repository directly (make sure the template is compatible with your Zabbix version).

Please use the sidebar to access information about specific template types and operation requirements.

See also:

- Template import
- Linking a template

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:

- Apache by HTTP
- Asterisk by HTTP
- AWS by HTTP
- AWS EC2 by HTTP
- AWS RDS instance by HTTP
- AWS S3 bucket by HTTP
- Azure 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
- GitLab by HTTP
- Hadoop by HTTP
- HAProxy by HTTP
- HashiCorp Consul Cluster by HTTP
- HashiCorp Consul Node by HTTP
- HashiCorp Vault by HTTP

- Hikvision camera by HTTP
- InfluxDB by HTTP
- HPE MSA 2040 Storage by HTTP
- HPE MSA 2060 Storage by HTTP
- HPE Primera by HTTP
- HPE Synergy by HTTP
- Jenkins by HTTP
- Kubernetes API server by HTTP
- Kubernetes Controller manager by HTTP
- Kubernetes kubelet by HTTP
- Kubernetes nodes by HTTP
- Kubernetes Scheduler by HTTP
- Kubernetes cluster state by HTTP

- Microsoft SharePoint by HTTP
- NetApp AFF A700 by HTTP
- NGINX by HTTP
- NGINX Plus by HTTP
- OpenWeatherMap 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
- ZooKeeper by HTTP

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

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
- WildFly Domain by JMX
- WildFly Server by JMX

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

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 *Configuration* → *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 *Configuration* → *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

Template name	Vendor	Device family	Known models	OS	MIBs used	Tags
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
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
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
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

Template name	Vendor	Device family	Known models	OS	MIBs used	**Tags**
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
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

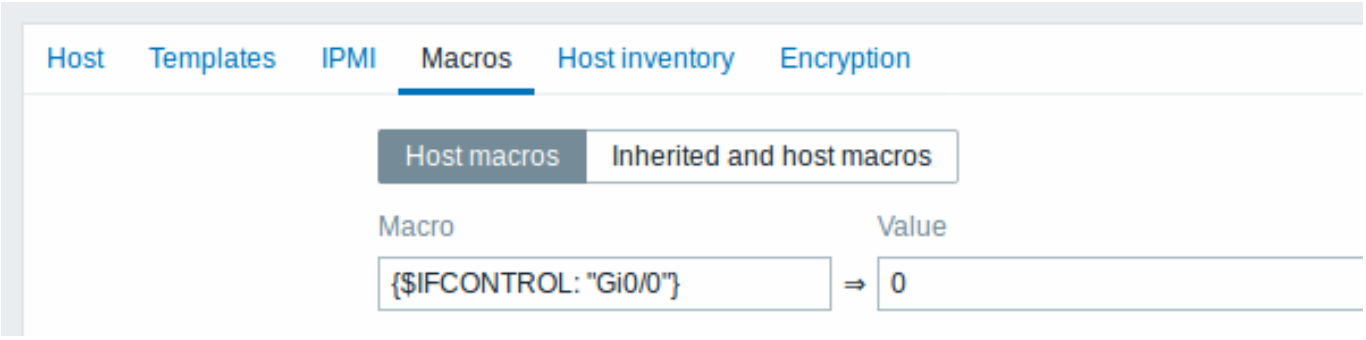
Templates were designed with the following in mind:

- 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:

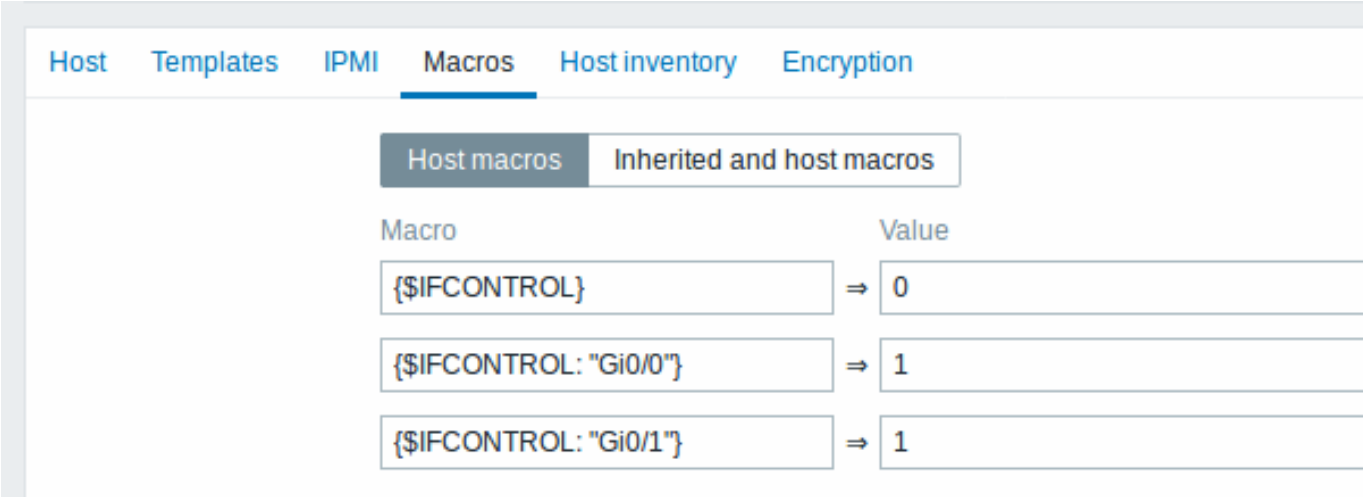
```
{$IFCONTROL:"{#IFNAME}"}=1 and last(/Alcatel Timetra TiMOS SNMP/net.if.status[ifOperStatus.{#SNMPINDEX}])
```

- 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.

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 Agent 2
- SMART by Zabbix agent 2
- SMART by Zabbix agent 2 active
- Systemd by Zabbix agent 2

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
- Nginx by Zabbix agent
- PHP-FPM by Zabbix agent
- RabbitMQ cluster by Zabbix agent
- MySQL by Zabbix agent
- PostgreSQL

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

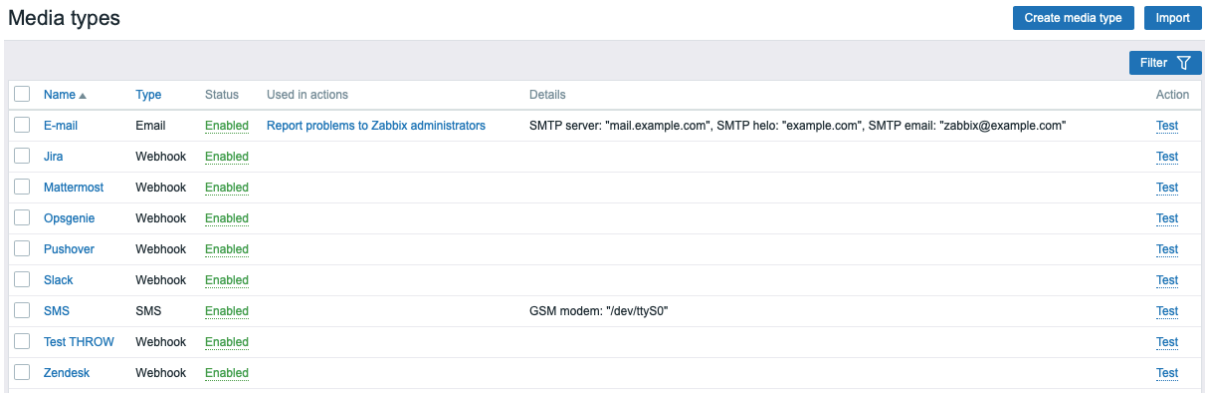
Overview

Media are the delivery channels used for sending notifications and alerts from Zabbix.

You can configure several media types:

- E-mail
- SMS
- Custom alertscripts
- Webhook

Media types are configured in *Administration → Media types*.



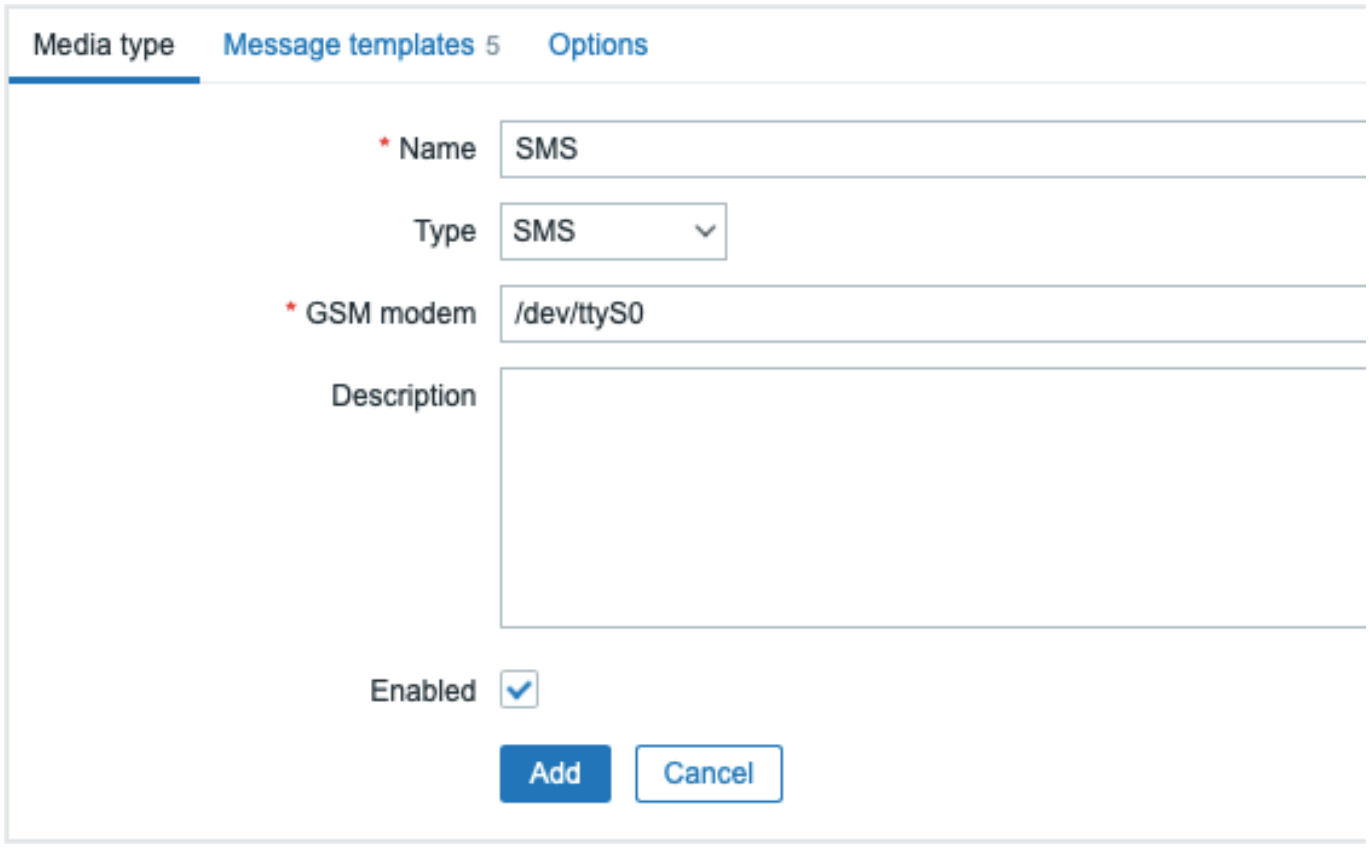
Some media types come pre-defined in the default dataset. You just need to finetune their parameters to get them working.

It is possible to test if a configured media type works, by clicking on *Test* in the last column (see Media type testing 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:

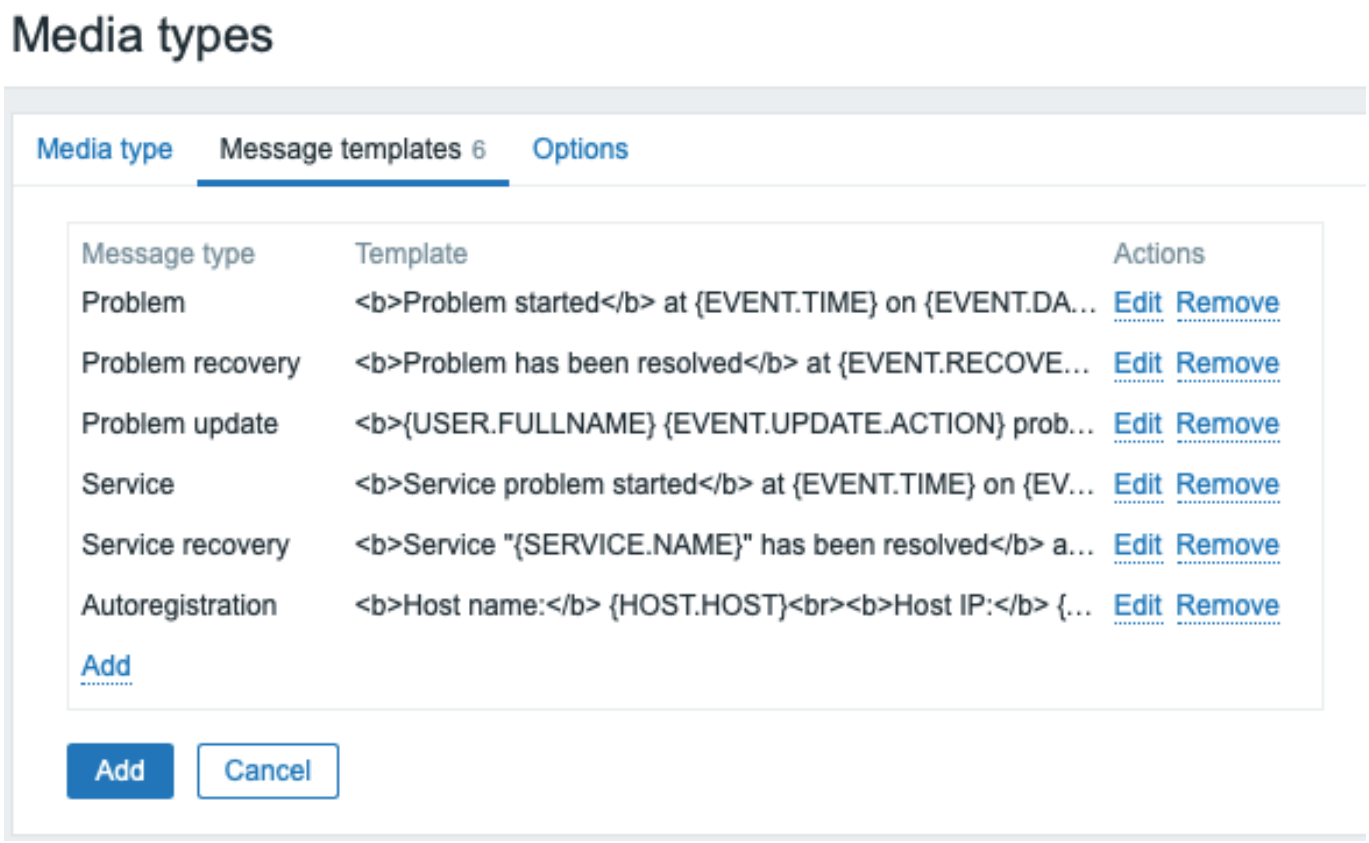
Parameter	Description
Name	Name of the media type.
Type	Select the type of media.
Description	Enter a description.
Enabled	Mark the checkbox to enable the media type.

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:

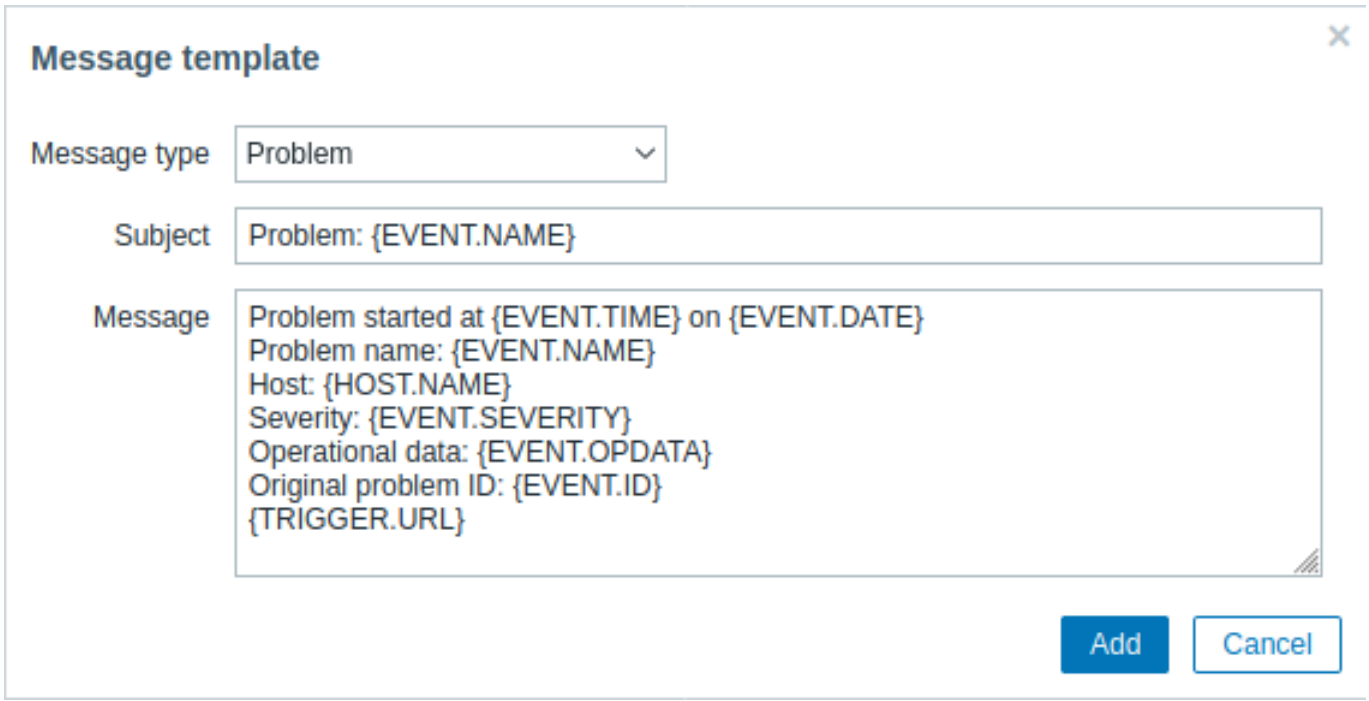
- Problem
- Problem recovery

- 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)}`).

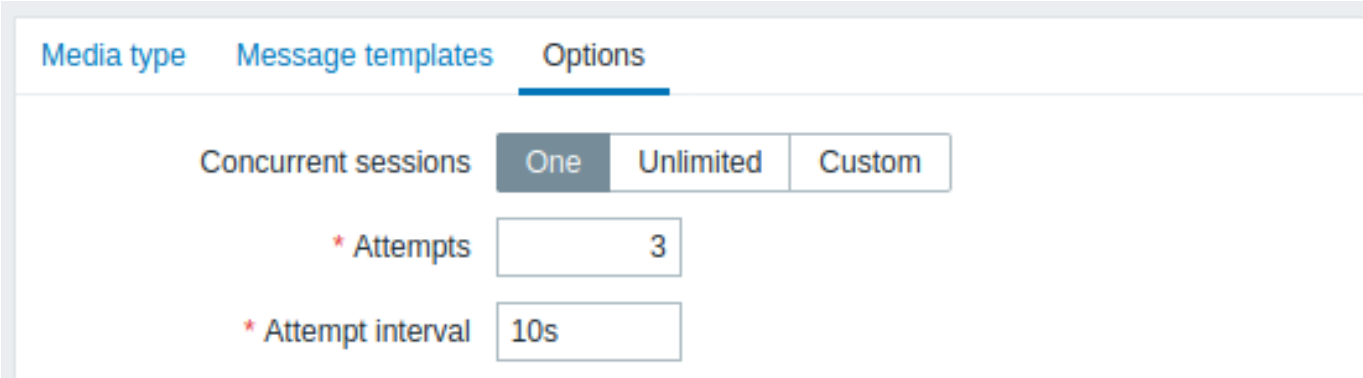
To make changes to an existing message template: In the *Actions* column click on Edit to edit the template or click on Remove to delete the message template.

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.

Media type testing

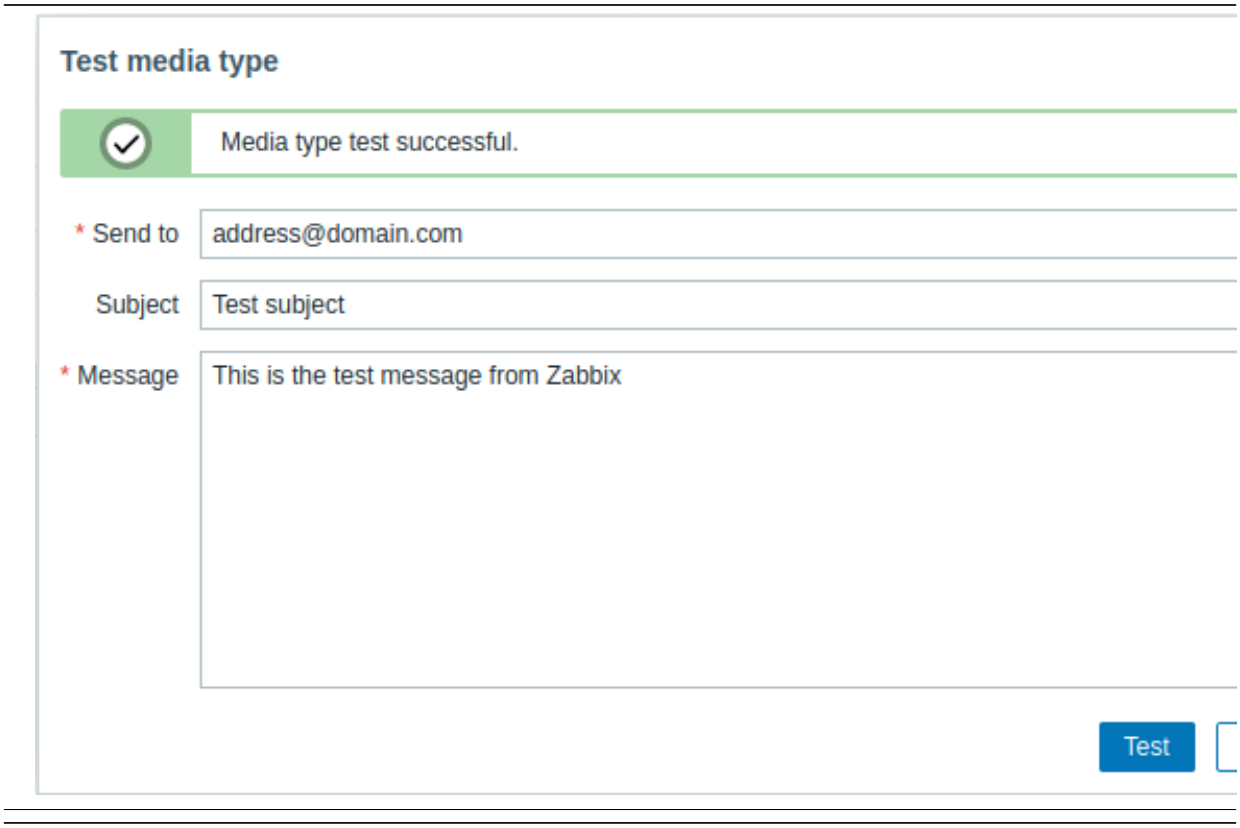
It is possible to test if a configured media type works.

E-mail

For example, to test an e-mail media type:

- Locate the relevant e-mail 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 and with body and optional subject
- Send a test message by clicking on *Test*

Test success or failure message will be displayed in the same window:

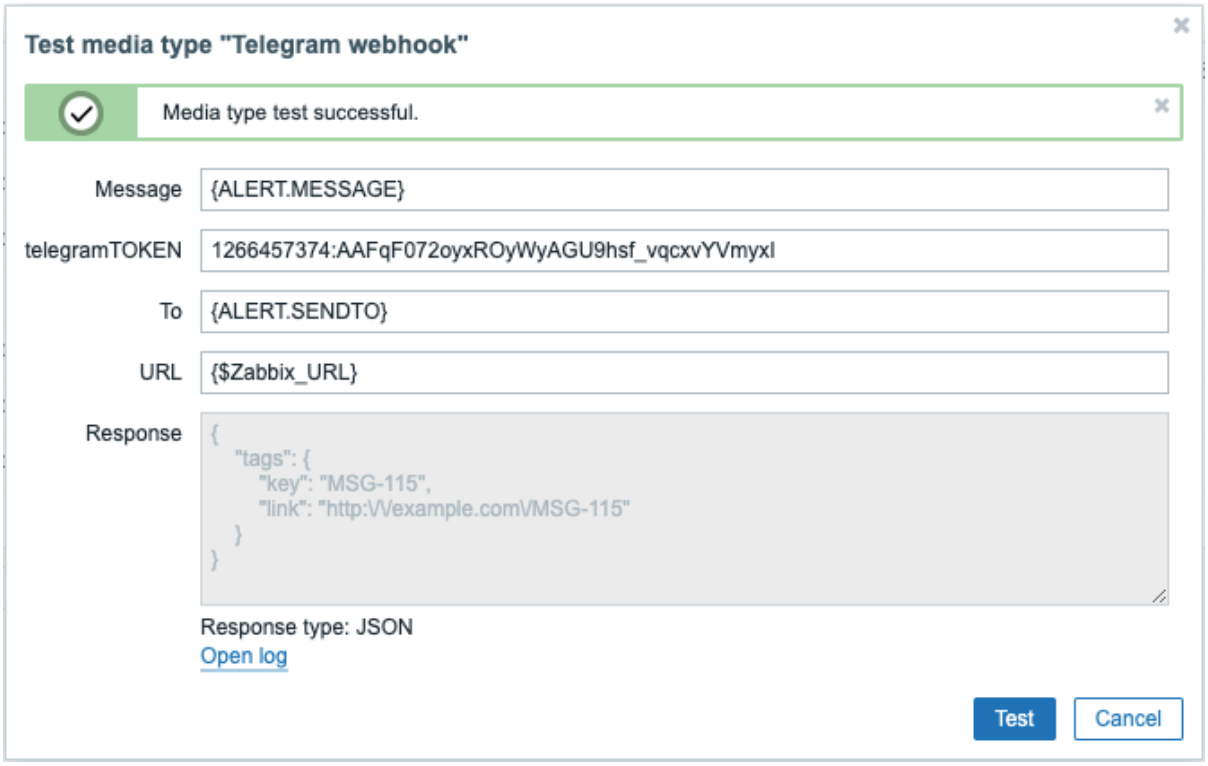


Webhook

To test a webhook media type:

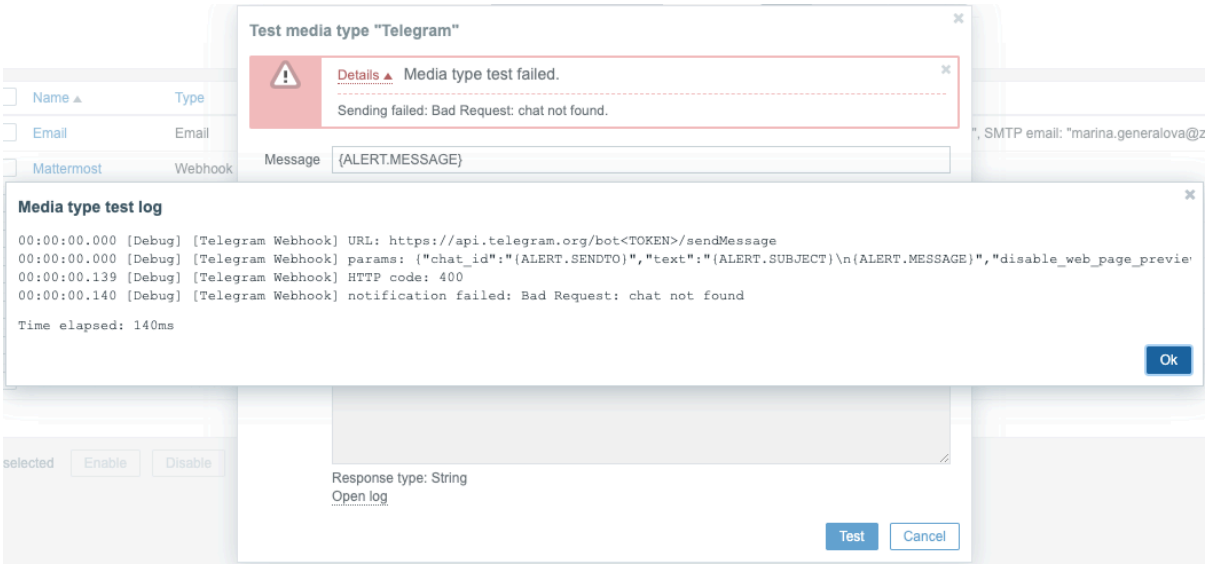
- Locate the relevant webhook in the list of media types
- Click on *Test* in the last column of the list (a testing window will open)
- Edit the webhook parameter values, if needed
- Click on *Test*

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

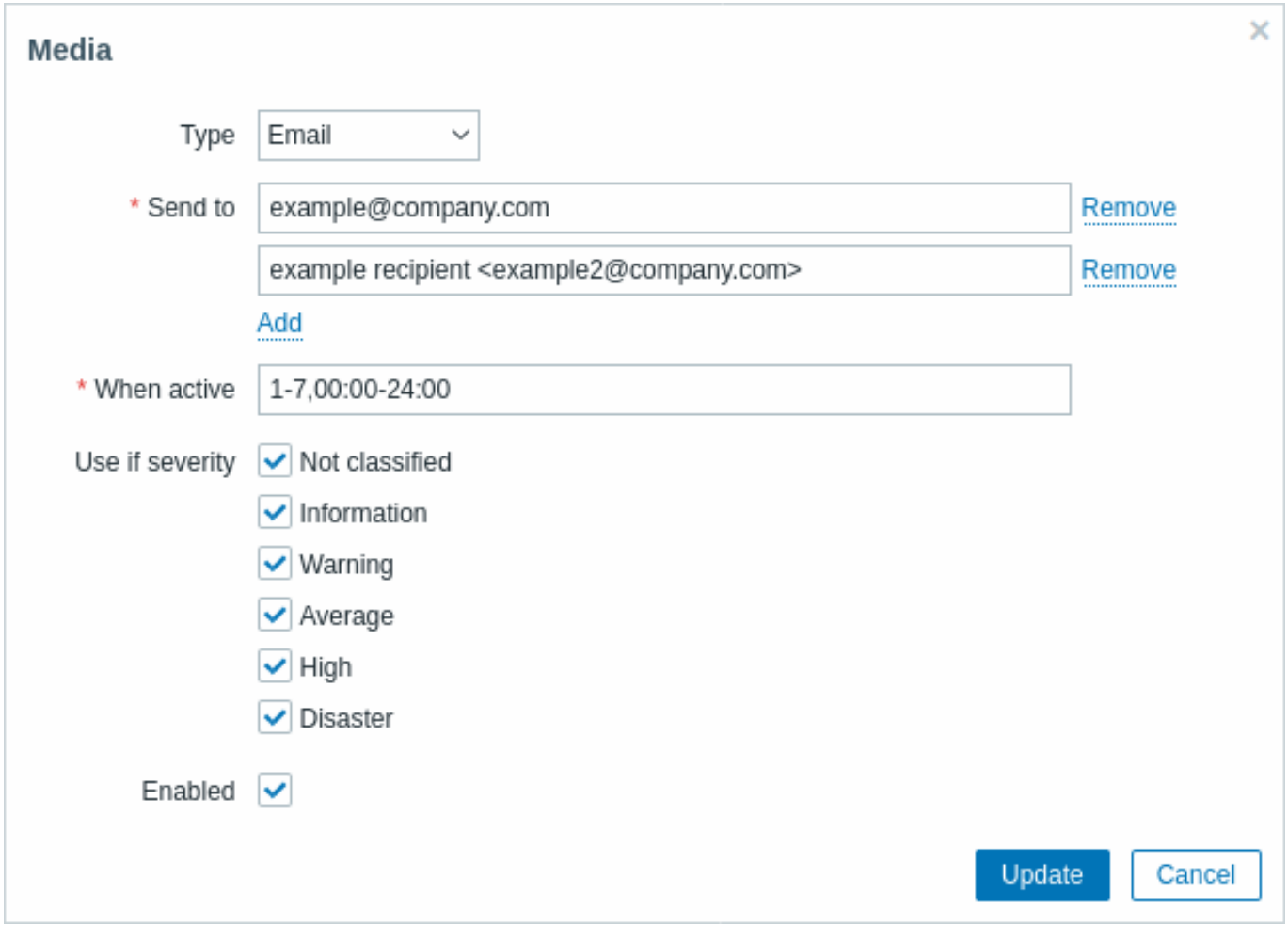
- *"Media type test failed."* message is displayed, followed by additional failure details.

User media

To receive notifications of a media type, a medium (e-mail 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 *Administration* → *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 names of all configured media types.
Send to	Provide required contact information where to send messages. For an e-mail 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 e-mail addresses provided. It is also possible to specify recipient name in the *Send to* field of the e-mail recipient in a format 'Recipient name <address1@company.com>'. Note, that if a recipient name is provided, an e-mail 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.

Parameter	Description
Status	Status of the user media. **Enabled** - is in use. **Disabled** - is not being used.

1 E-mail

Overview

To configure e-mail as the delivery channel for messages, you need to configure e-mail as the media type and assign specific addresses to users.

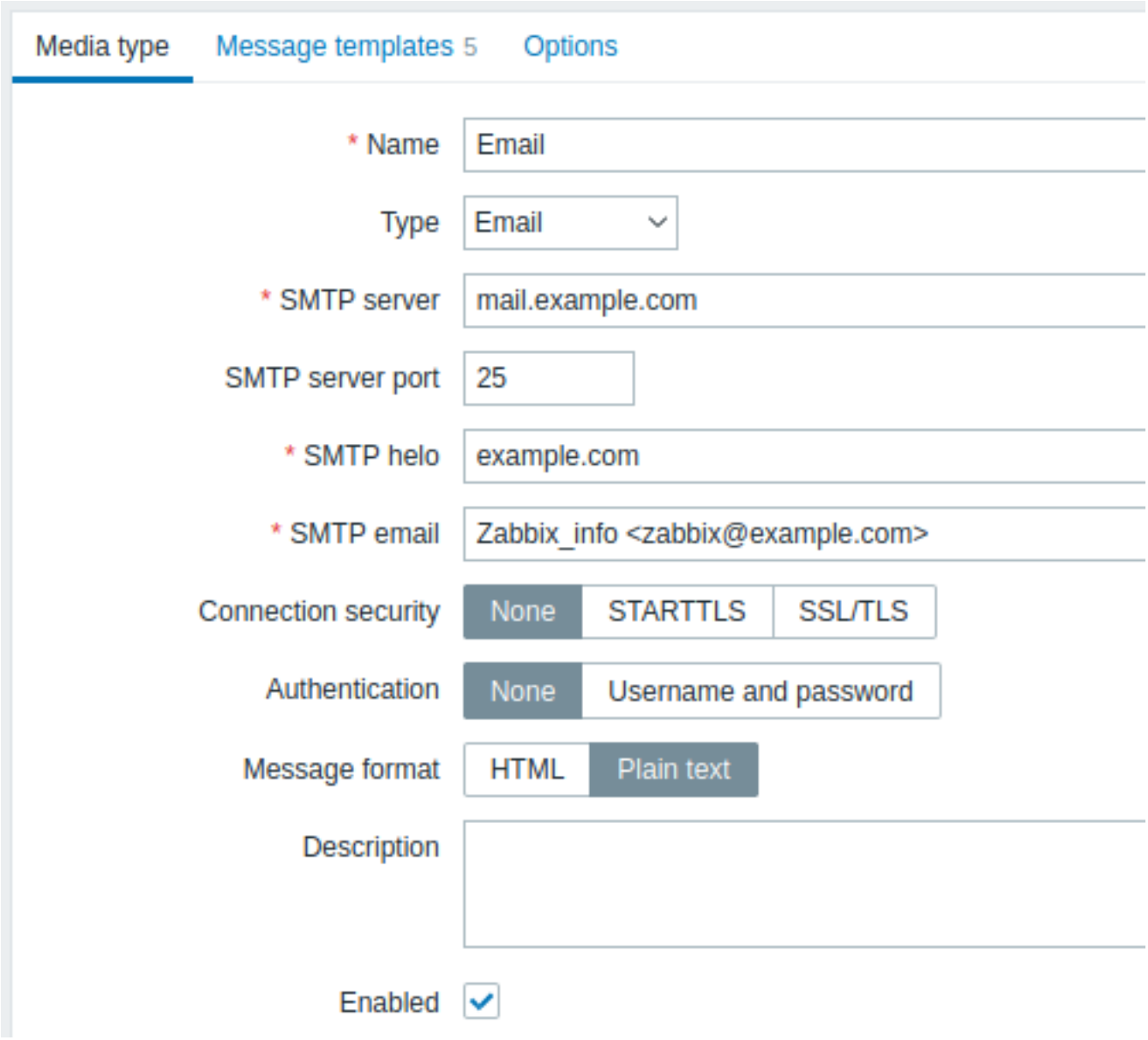
Note:
Multiple notifications for single event will be grouped together on the same email thread.

Configuration

To configure e-mail as the media type:

- Go to *Administration* → *Media types*
- Click on *Create media type* (or click on *E-mail* 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 e-mail media type:

Parameter	Description
SMTP server	Set an SMTP server to handle outgoing messages.
SMTP server port	Set the SMTP server port to handle outgoing messages. This option is supported *starting with Zabbix 3.0*.
SMTP helo	Set a correct SMTP helo value, normally a domain name.
SMTP 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 e-mail 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)
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 This option is supported *starting with Zabbix 3.0*.
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. This option is supported *starting with Zabbix 3.0*.
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. This option is supported *starting with Zabbix 3.0*.
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" This option is supported *starting with Zabbix 3.0*.
Username	User name to use in authentication. This sets the value of CURLOPT_USERNAME. This option is supported *starting with Zabbix 3.0*.
Password	Password to use in authentication. This sets the value of CURLOPT_PASSWORD. This option is supported *starting with Zabbix 3.0*.
Message format	Select message format: **HTML** - send as HTML **Plain text** - send as plain text

Attention:
To make SMTP authentication options available, Zabbix server should be compiled with the --with-libcurl compilation option with cURL 7.20.0 or higher.

See also common media type parameters for details on how to configure default messages and alert processing options.

User media

Once the e-mail media type is configured, go to the *Administration* → *Users* section and edit user profile to assign e-mail 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 *Administration → 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 *Administration → 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 alertscripts

Overview

If you are not satisfied with 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.

Alert scripts are executed on Zabbix server. These scripts are located in the directory defined in the server configuration file **AlertScriptsPath** variable.

Here is an example alert script:

```
#####!/bin/bash

to=$1
subject=$2
body=$3

cat <<EOF | mail -s "$subject" "$to"
$body
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 case Zabbix will try to repeat failed execution.

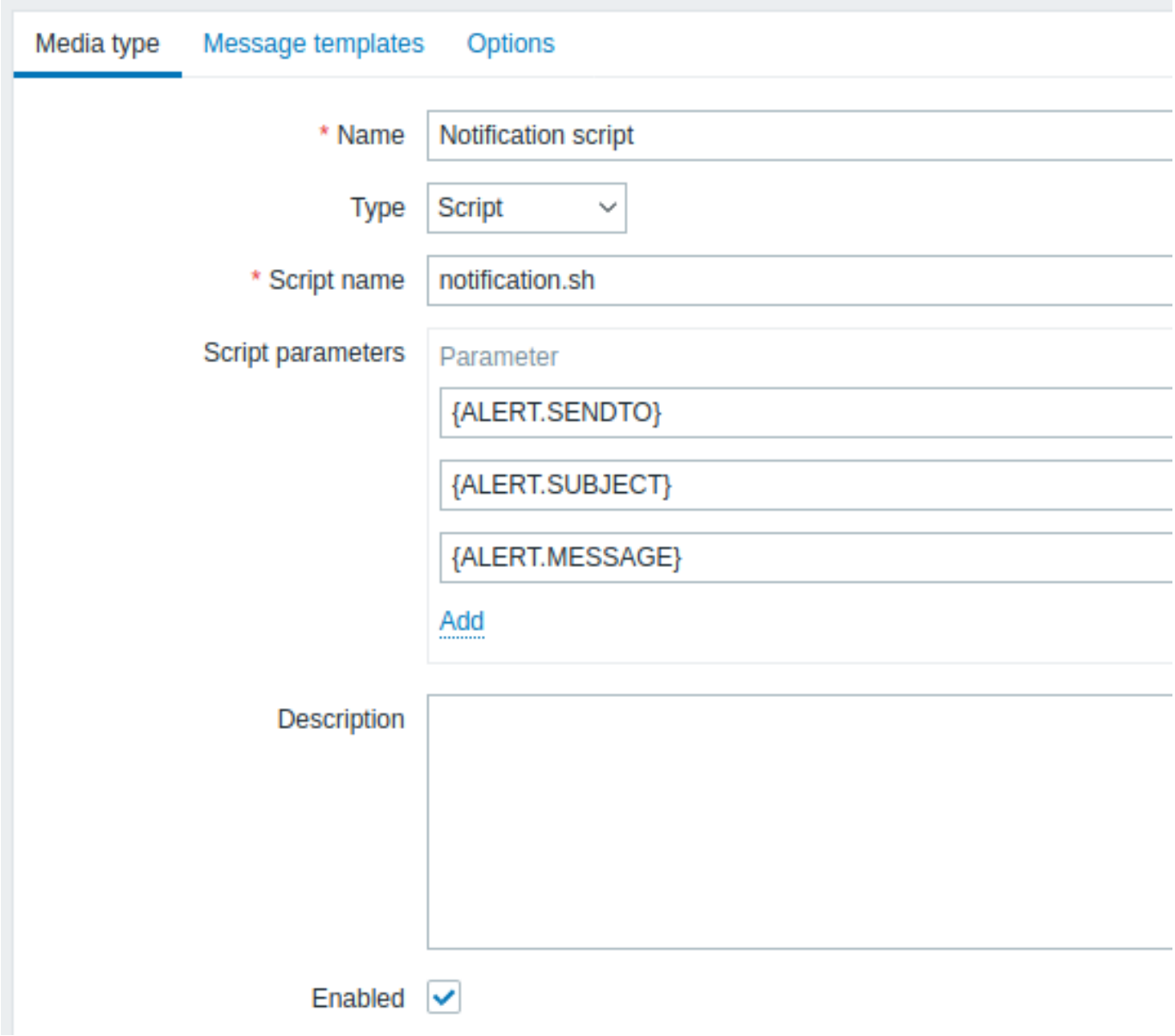
Environment variables are not preserved or created for the script, so they should be handled explicitly.

Configuration

To configure custom alertscripts as the media type:

- Go to *Administration* → *Media types*
- 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.
Script parameters	Add command-line parameters to the script. {ALERT.SENDTO}, {ALERT.SUBJECT} and {ALERT.MESSAGE} macros are supported in script parameters. Customizing script parameters is supported since Zabbix 3.0.

See common media type parameters for details on how to configure default messages and alert processing options.

Warning:
Even if an alertscript doesn't use default messages, message templates for operation types used by this media type must still be defined, otherwise a notification will not be sent.

Attention:
As parallel processing of media types is implemented since Zabbix 3.4.0, it is important to note that with more than one script media type configured, these scripts may be processed in parallel by alerter processes. The total number of alerter processes is limited by the StartAlerters parameter.

User media

Once the media type is configured, go to the *Administration → Users* section and edit user profile to assign media of this 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 a user media, a *Send to* field cannot be empty. If this field will not be used in an alertscript, 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 to use predefined webhook media types for pushing Zabbix notifications to:

- brevis.one
- Discord
- Express.ms messenger
- Github issues
- GLPi
- iLert
- iTop
- Jira
- Jira Service Desk
- ManageEngine ServiceDesk
- 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

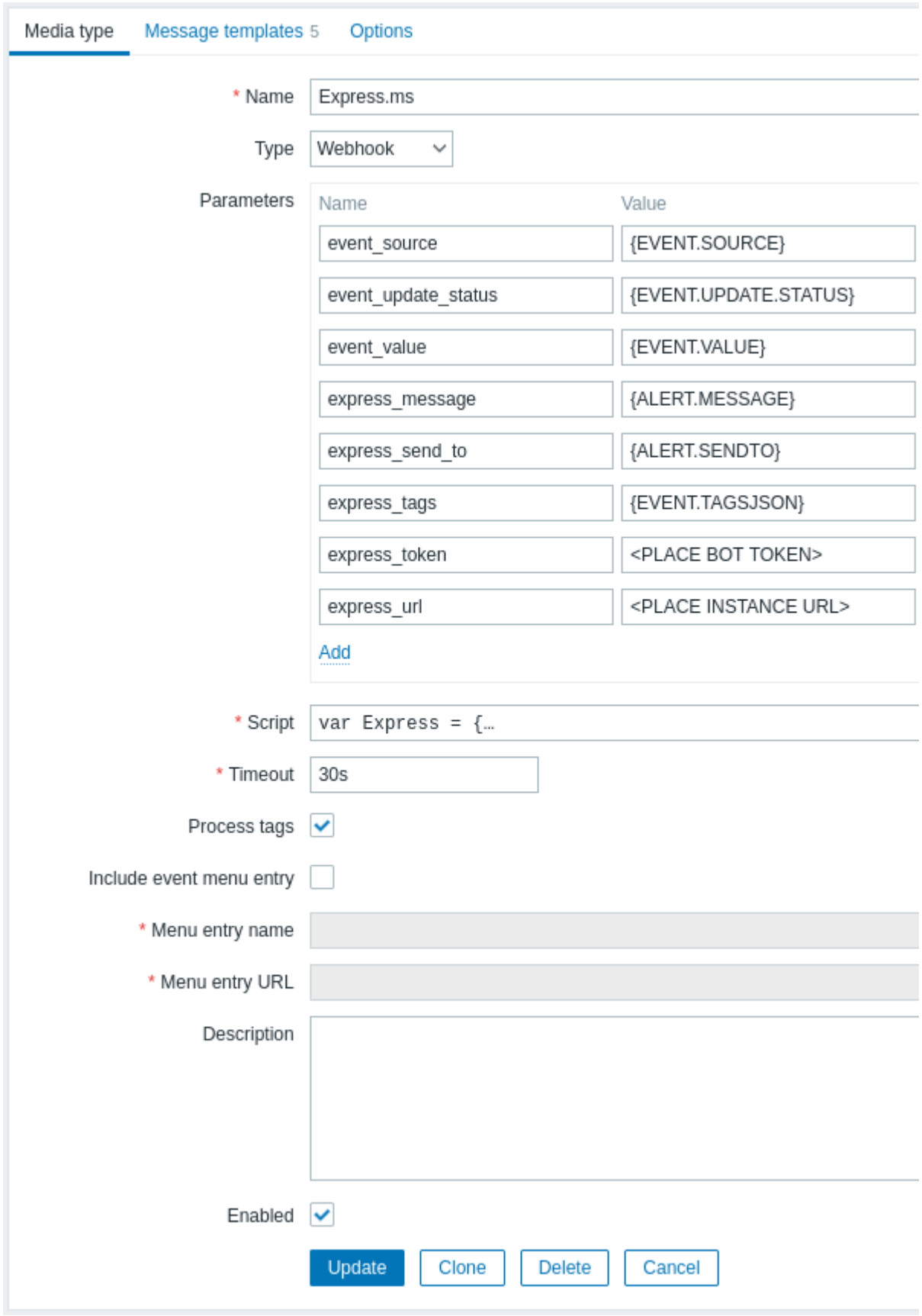
To start using a webhook integration:

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 *Administration* → *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. All macros that are supported in problem notifications are supported in the parameters. 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. 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 the already existing (if any) problem event tags in Zabbix. If a webhook uses tags (the Process tags checkbox is marked), the webhook should always return a JSON object containing at least an empty object for tags:`var result = {tags: {}};`. Examples of tags that can be returned: *Jira ID: PROD-1234*, *Responsible: John Smith*, *Processed:<no value>*, etc.
Include event menu entry	Mark the checkbox to include an entry in the event menu linking to the created external ticket. If marked, the webhook should not be used to send notifications to different users (consider creating a dedicated user instead) or in several alert actions related to 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 selected.
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 selected.

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.

User media

Once the media type is configured, go to the *Administration* → *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 *Administration → 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:

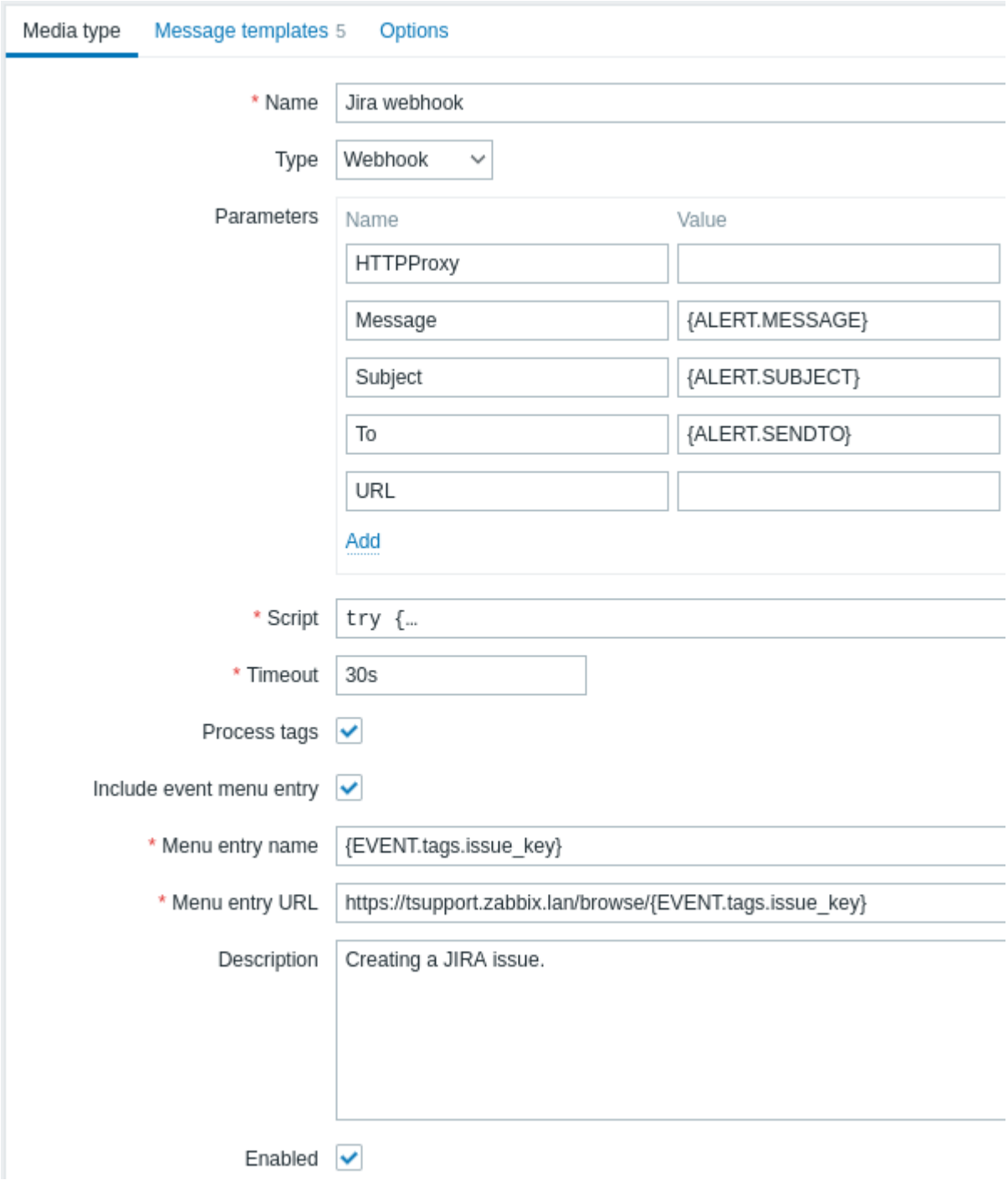
- If a webhook uses tags to store ticket\message ID and to follow up with update\resolve operations, this webhook should not be used in several alert actions for a single problem event. If {EVENT.TAGS.<name>} already exists, and is updated in the webhook, then its resulting value is not defined. For such a case, a new tag name should be used in the webhook to store updated values. This applies to Jira, Jira Service Desk, Mattermost, Opsgenie, OTRS, Redmine, ServiceNow, Slack, Zammad, and Zendesk webhooks provided by Zabbix and to the majority of webhooks that utilize *Include event menu entry* option. Using the webhook in several operations is allowed if those operations or escalation steps belong to the same action. It is also ok to use this webhook in different actions if the actions will not be applied to the same problem event due to different filter conditions.
- When using a webhook in actions for internal events: in the action operation configuration, check the *Custom message* checkbox and define the custom message, otherwise, a notification will not be sent.

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://tsupport.zabbix.lan/rest/api/2/issue/',
        JSON.stringify({"fields": fields})
    );

    if (req.getStatus() != 201) {
        throw 'Response code: ' + req.getStatus();
    }

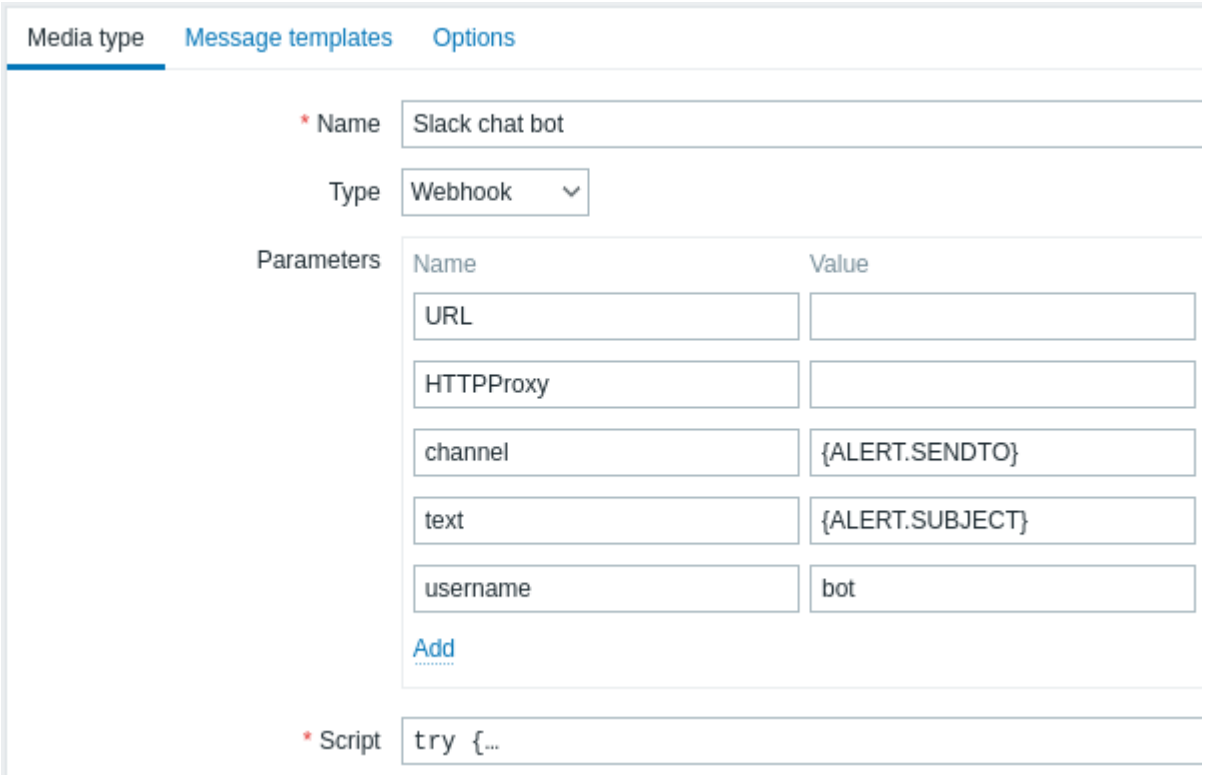
    resp = JSON.parse(resp);
    result.tags.issue_id = resp.id;
    result.tags.issue_key = resp.key;

    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

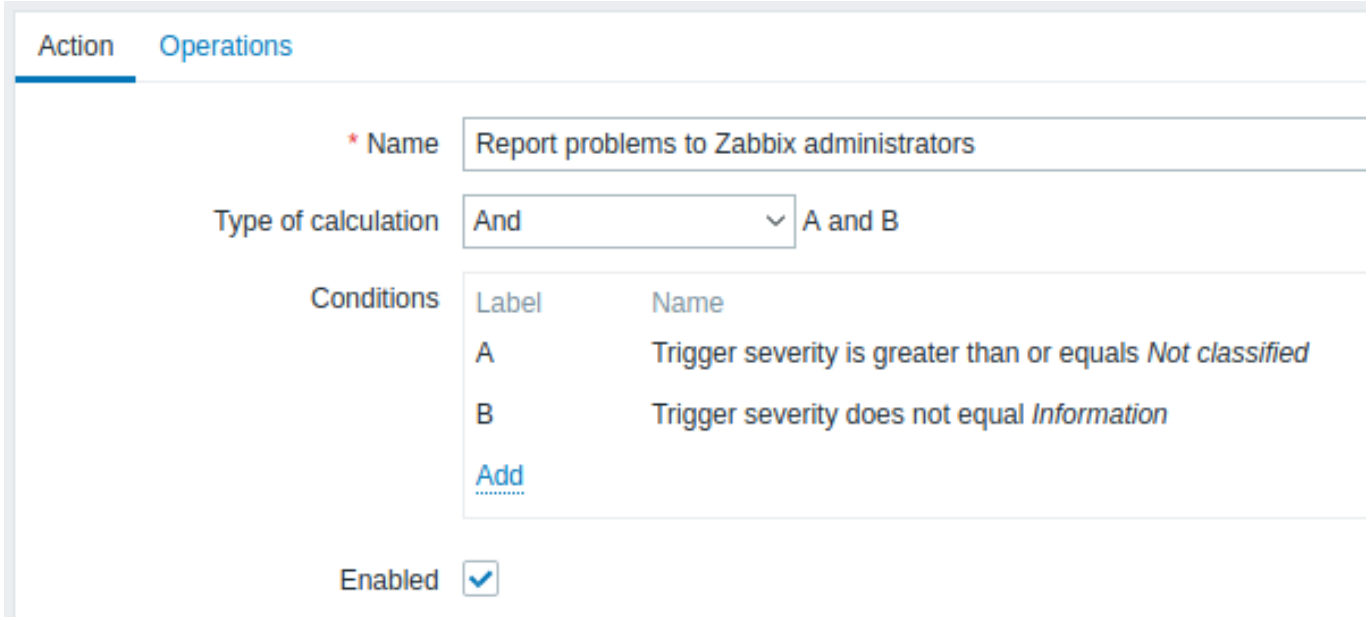
Configuring an action

To configure an action, do the following:

- Go to *Configuration -> 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

Note that service actions can be configured in the service action section.

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.
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.
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.

Condition type	Supported operators	Description
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.
Trigger name	contains does not contain	Specify a string in the trigger name or a string to exclude. **contains** - event is generated by a trigger, containing this string in the name. **does not contain** - this string cannot be found in the trigger name. *Note*: Entered value will be compared to trigger 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.

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.

Condition type	Supported operators	Description
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).
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
Trigger name contains 'Database is down'
Trigger name contains 'Database is unavailable'

is evaluated as

(Host group equals Oracle servers **or** Host group equals MySQL servers**) and (**Trigger name contains 'Database is down' **or** Trigger 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

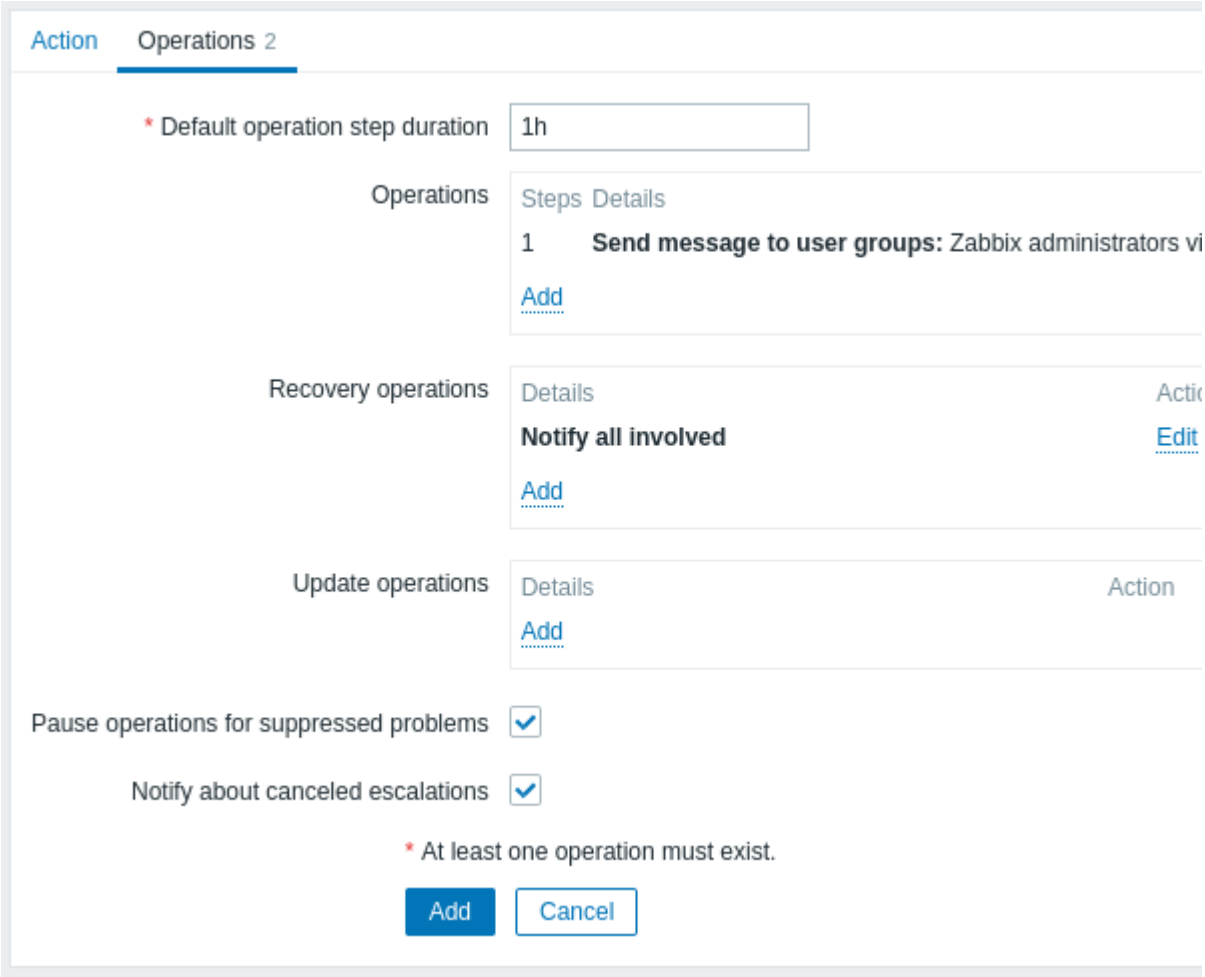
> **Attention:**
> Zabbix server does not create alerts if access to the host is explicitly "denied" for the user defined as action operation recipient or if the user has no rights defined to the host at all.

For discovery and autoregistration events, there are additional operations 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:

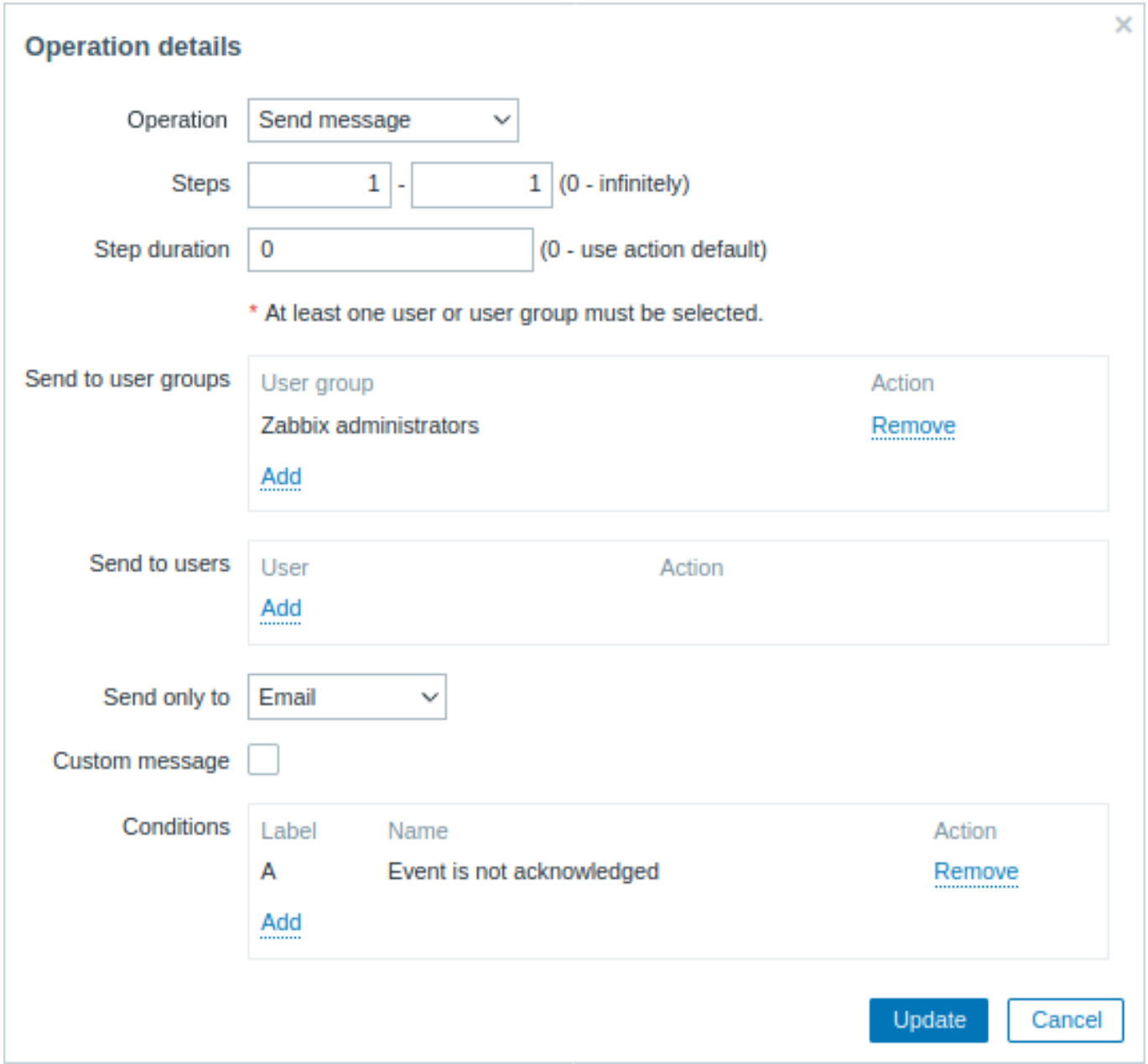
Parameter	Description
Default operation step duration	Duration of one operation step by default (60 seconds to 1 week). For example, an hour-long step duration means that if an operation is carried out, an hour will pass before the next step. Time suffixes are supported, e.g. 60s, 1m, 2h, 1d, since Zabbix 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 (e-mail, 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 (e-mail, 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 (e-mail, 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.

Parameter	Description
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*.
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 popup 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 *Step duration* Operation type: send message	
Send to user groups	Click on *Add* to 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	Click on *Add* to 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: **Not ack** - only when the event is unacknowledged **Ack** - only when the event is acknowledged. *Conditions* option is not available for *Service actions*.

When done, click on *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 evens.

- 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 e-mails from Zabbix, e-mail servers/clients must support standard 'SMTP/MIME e-mail' 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 e-mail' 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

The following tutorial provides step-by-step instructions on how to set up remote commands.

Configuration

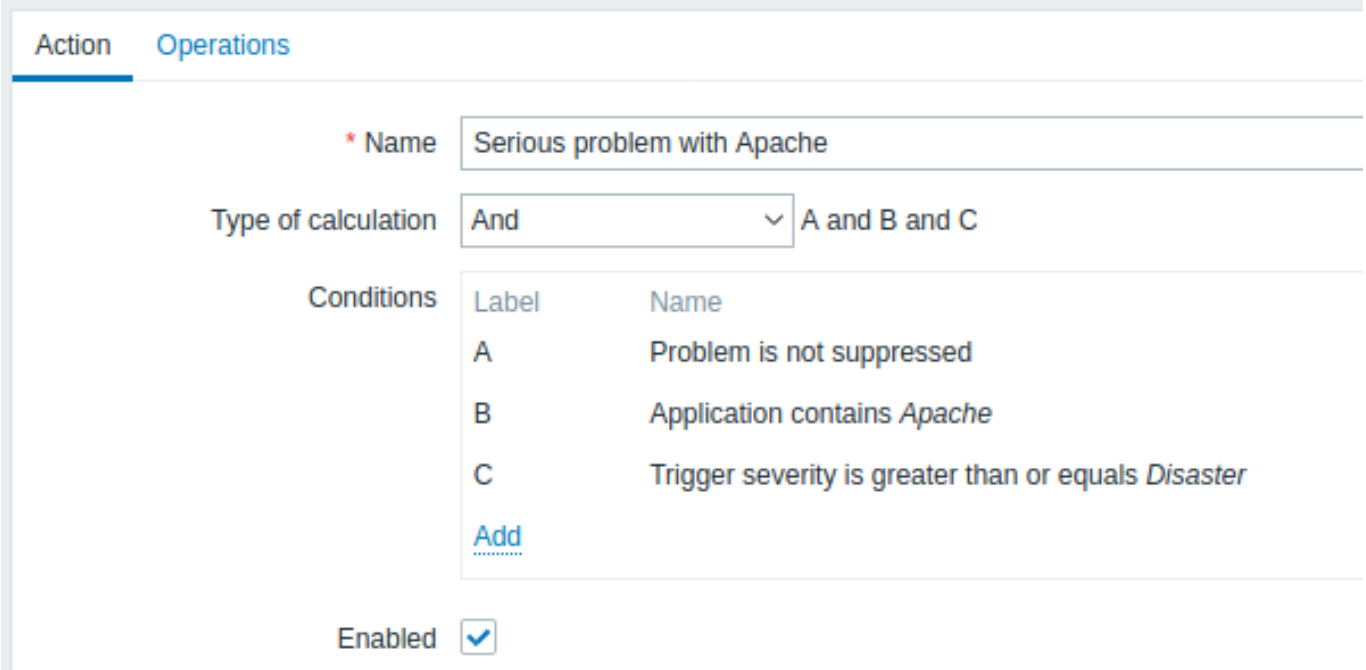
Those remote commands that are executed on Zabbix agent (custom scripts) must be first enabled in the agent configuration.

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.

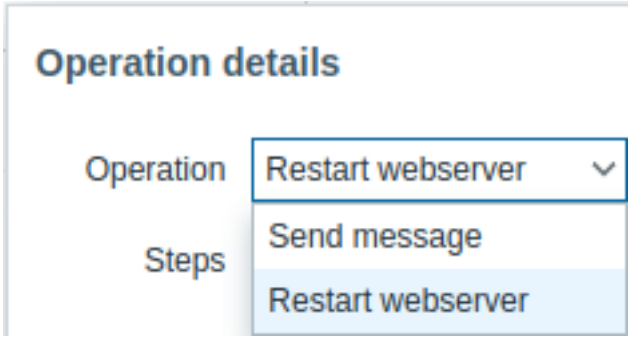
Attention:
Remote commands do not work with active Zabbix agents.

Then, when configuring a new action in *Configuration* → *Actions*:

- Define the appropriate conditions. In this example, set that the action is activated upon any disaster problems with one of Apache applications:



- In the *Operations* tab, click on Add in the Operations/Recovery operations/Update operations block
- From the Operation dropdown field select one of the predefined scripts



- Select the target list for the script

Predefined scripts

All scripts (webhook, script, SSH, Telnet, IPMI) that are available for action operations 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.

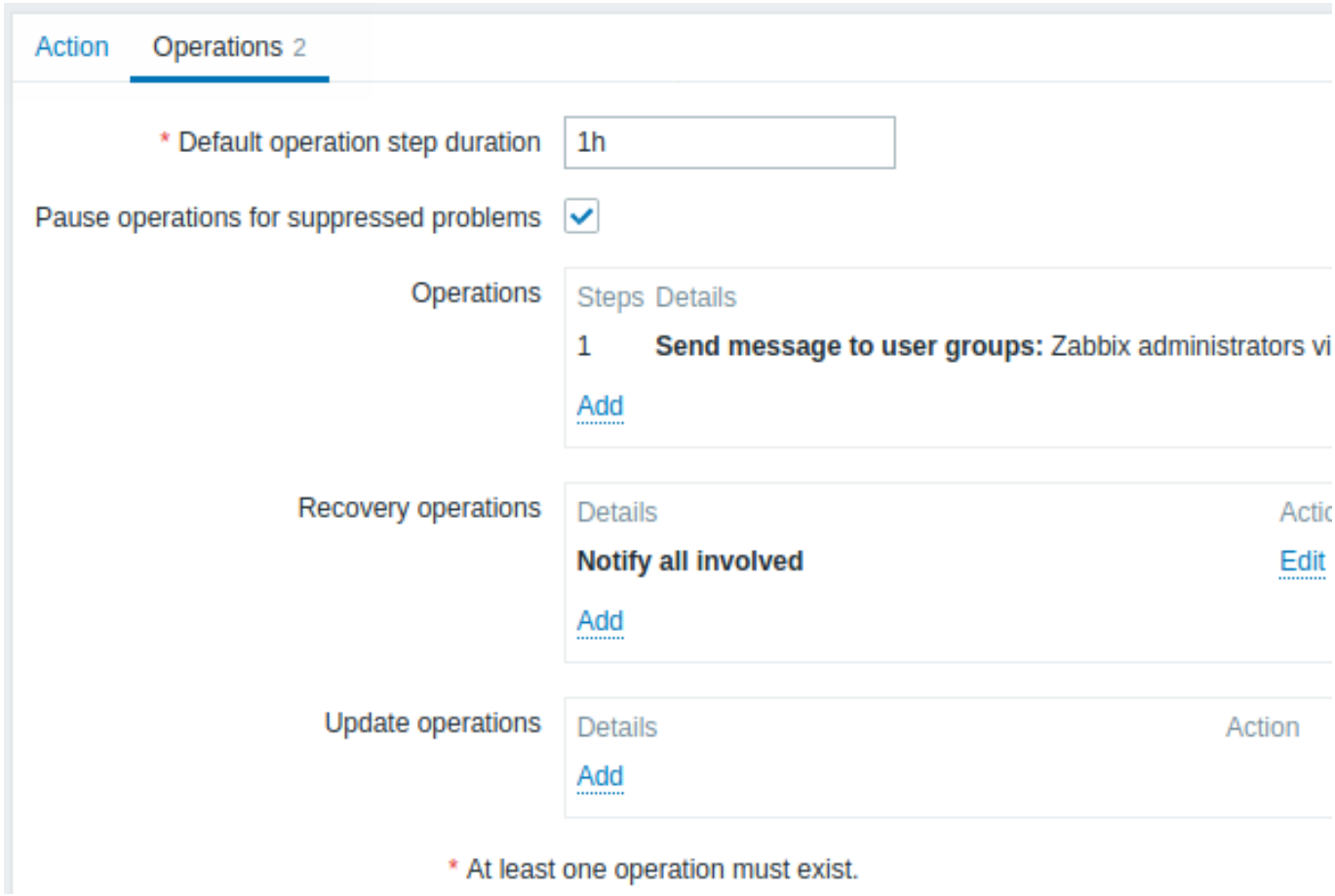
Use cases

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.

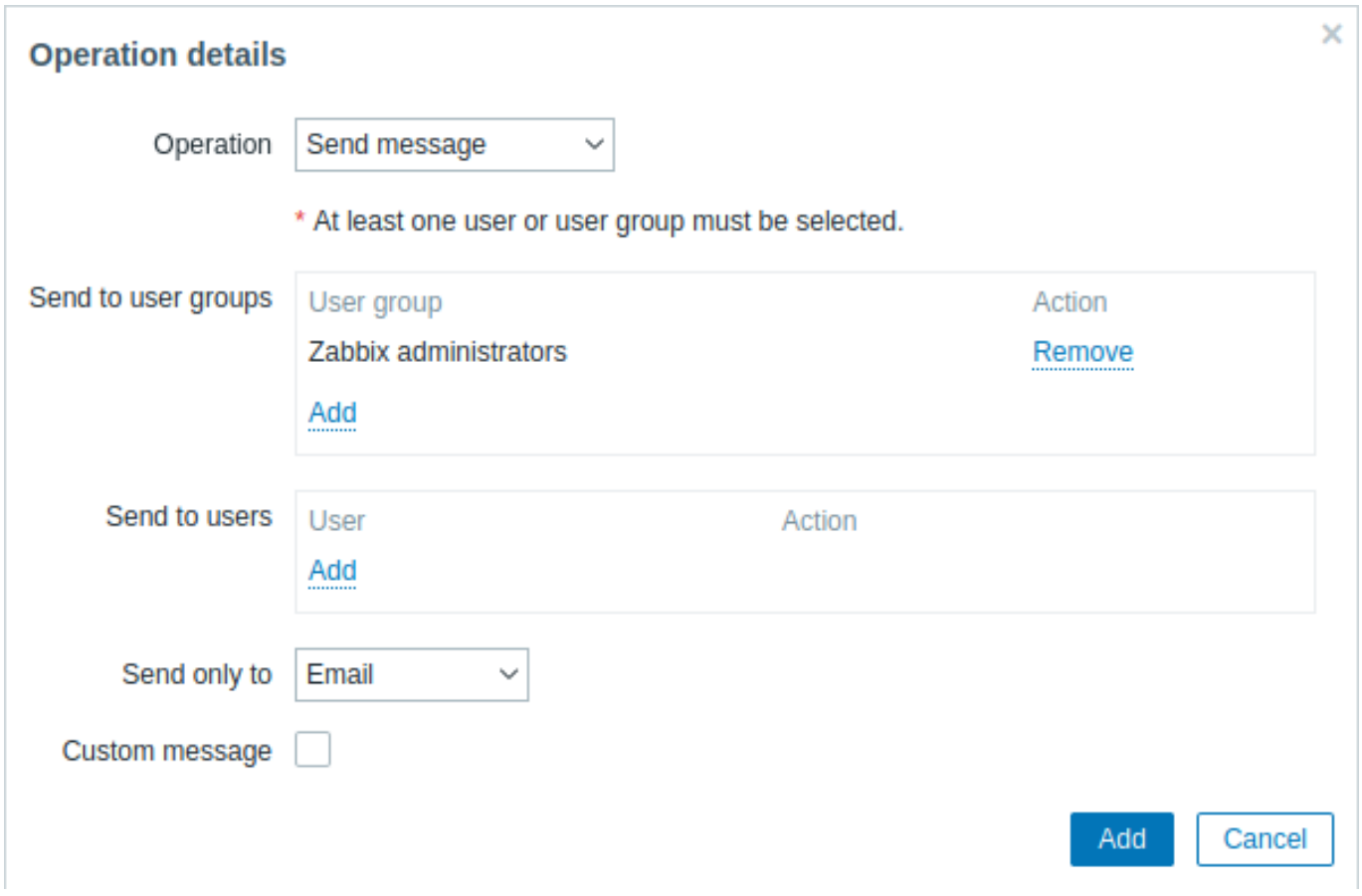
Configuring a recovery operation

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 popup 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	Click on *Add* to 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	Click on *Add* to 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

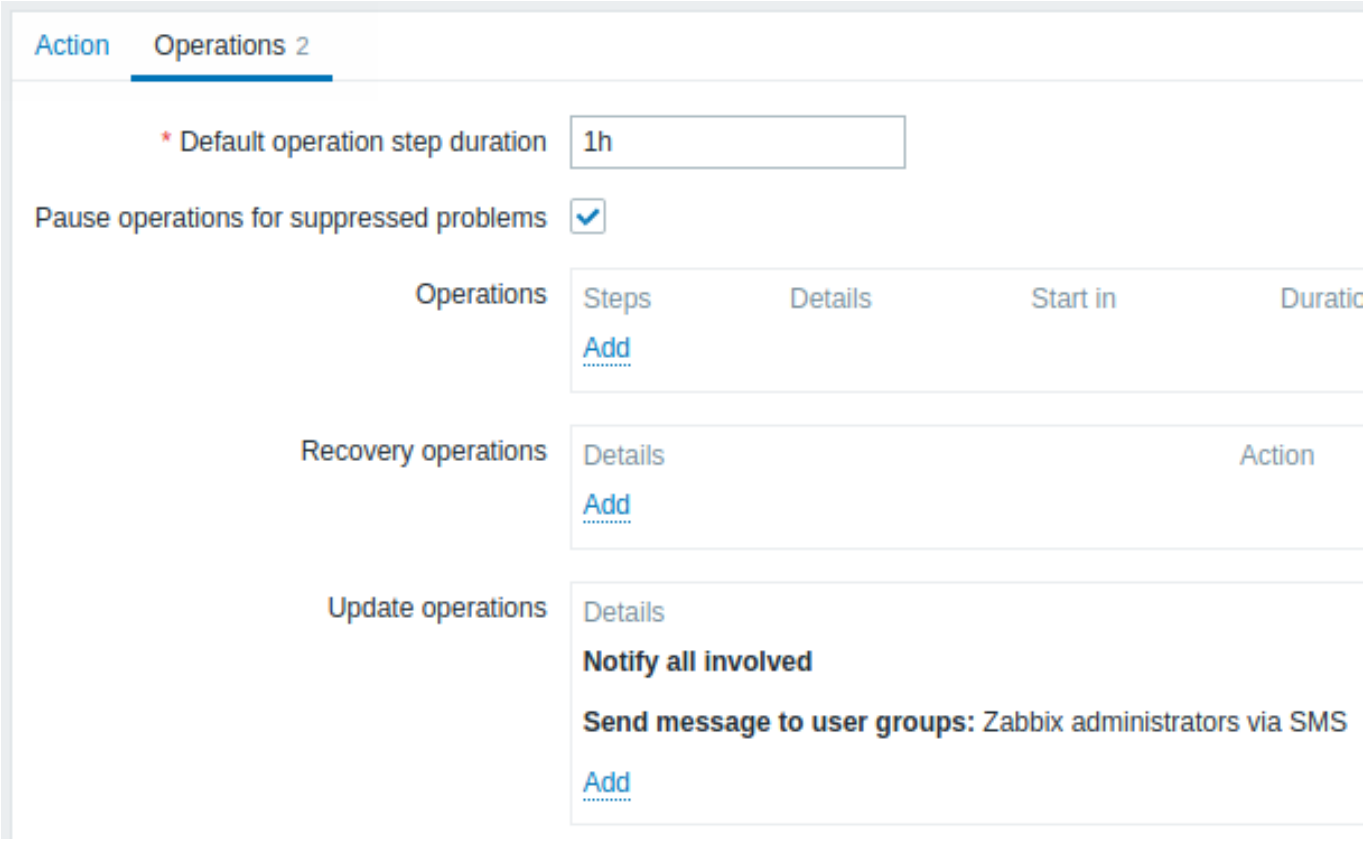
Update operations are available in actions with the following event sources:

- *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.

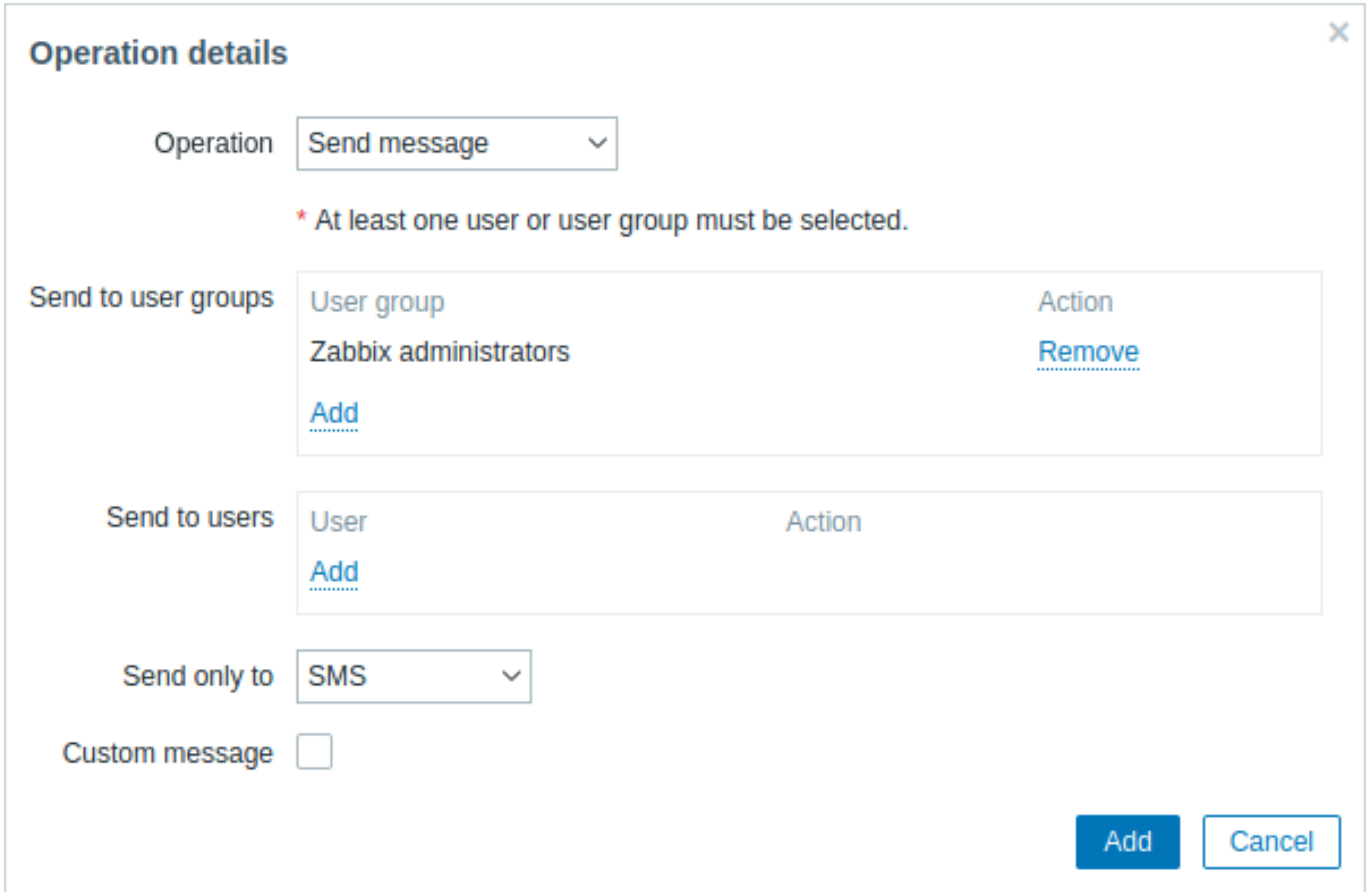
Configuring an update operation

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 popup window will open where you can edit the operation step 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.

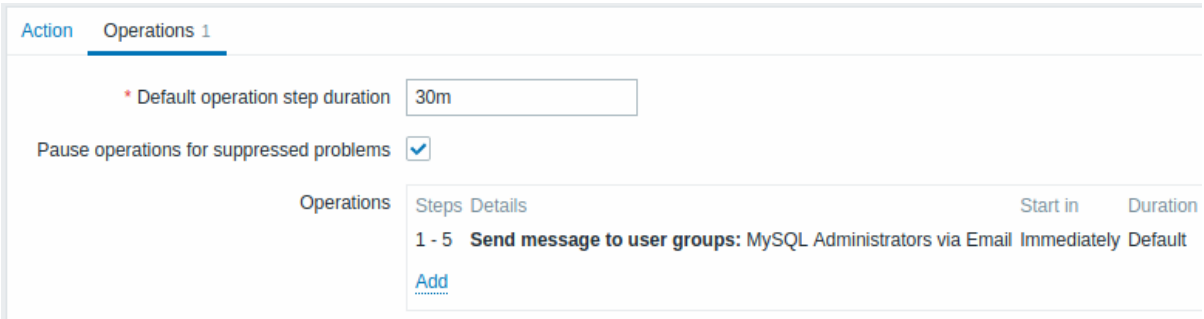
Situation	Behavior
*The time period defined in the **Time period** action condition ends after the initial notification is sent*	All remaining escalation steps are executed. The *Time period* condition cannot stop operations; it has effect with regard to when actions are started/not started, not operations.
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 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).

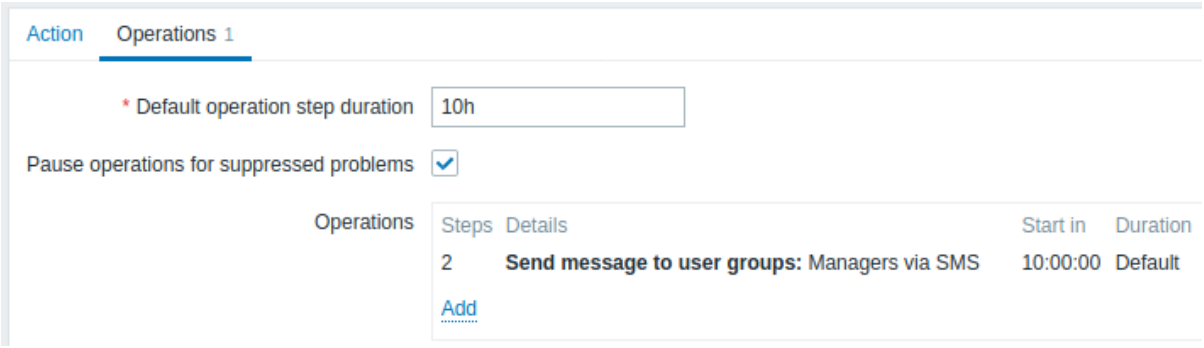
If the problem is resolved and a recovery message is configured, it will be sent to those who received at least one problem message within this escalation scenario.

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'



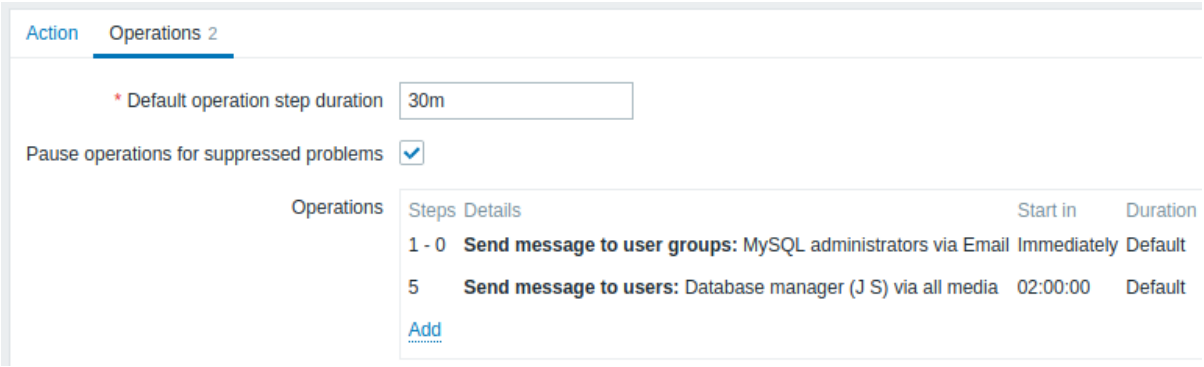
A notification will only be sent at Step 2 of the escalation scenario, or 10 hours after the problem starts.

You can customize the message text to something like 'The problem is more than 10 hours old'.

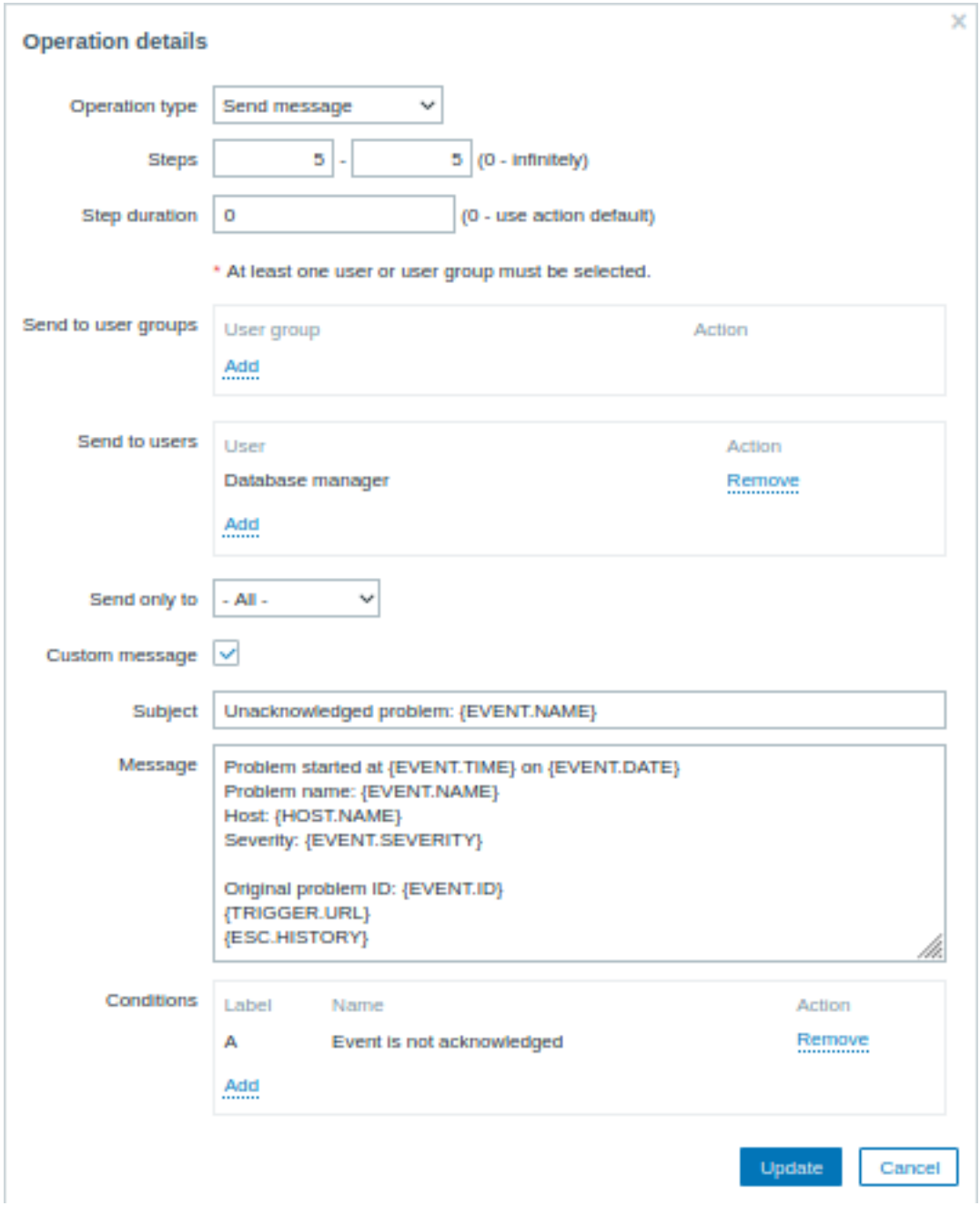
Example 3

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:



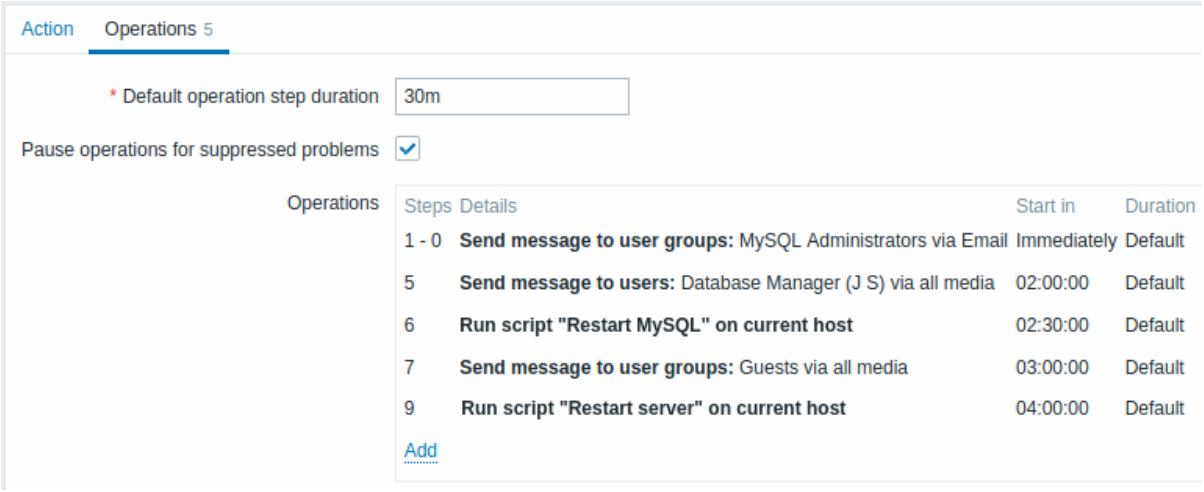
Note the use of {ESC.HISTORY} macro in the customized message. The macro will contain information about all previously executed steps on this escalation, such as notifications sent and commands executed.

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.

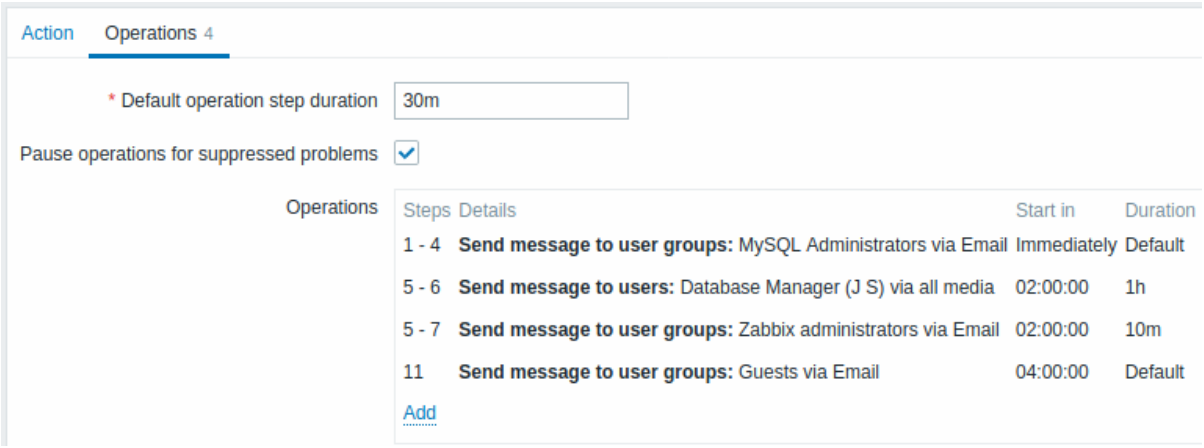
If the problem still exists, after another 30 minutes Zabbix will send a message to all guest users.

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.

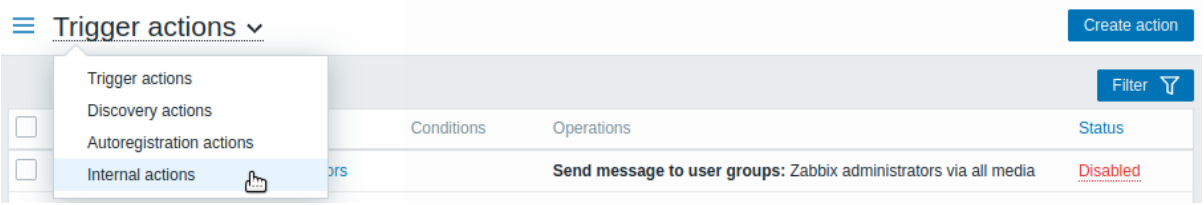
Step 1

Configure some media, such as e-mail, 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 *Configuration* → *Actions* and select *Internal actions* from the third level menu (or page title dropdown).

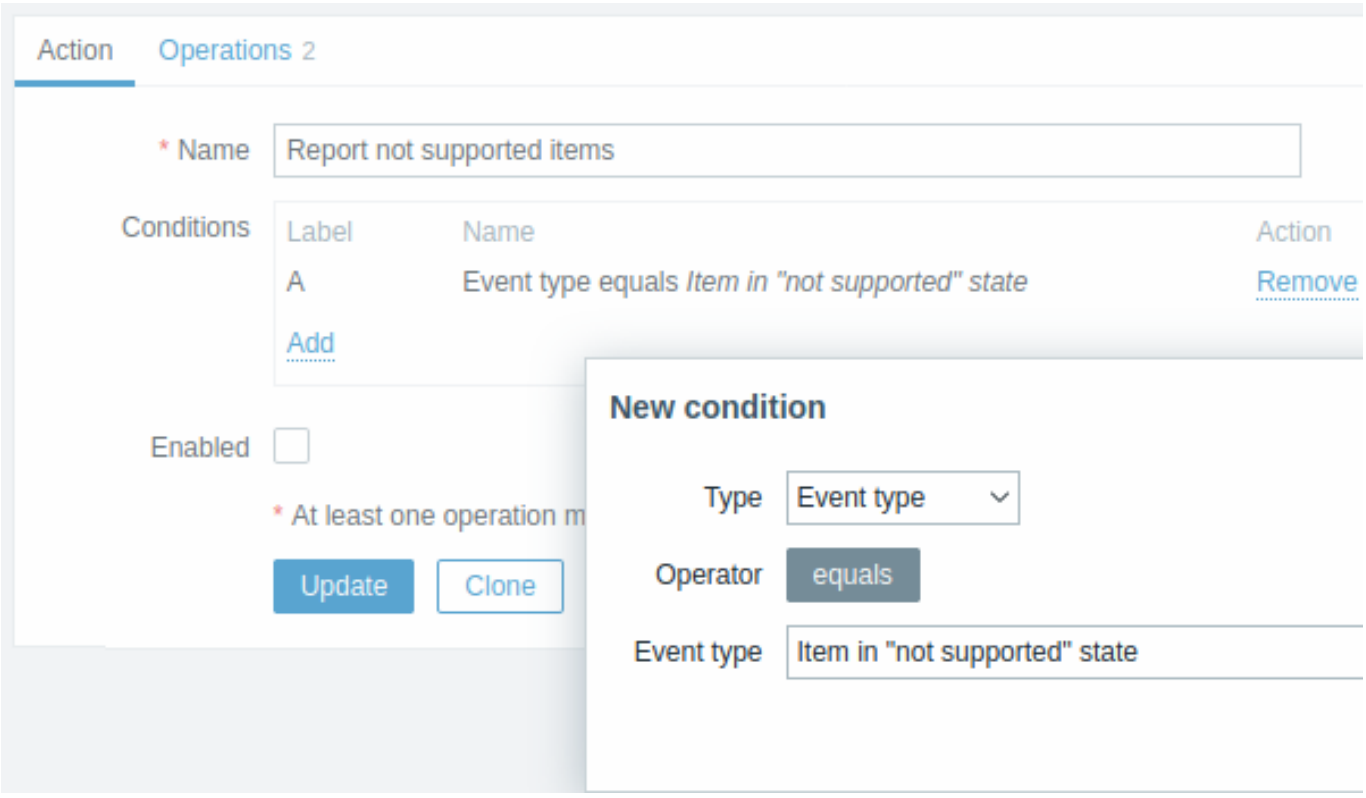


Click on *Create action* to the right to open an action configuration form.

Step 3

In the **Action** tab enter a name for the action. Then click on *Add* in the condition block to add a new condition.

In the new condition popup window select *Event type* as the condition type and then select *Item in "not supported" state* as the event type value.

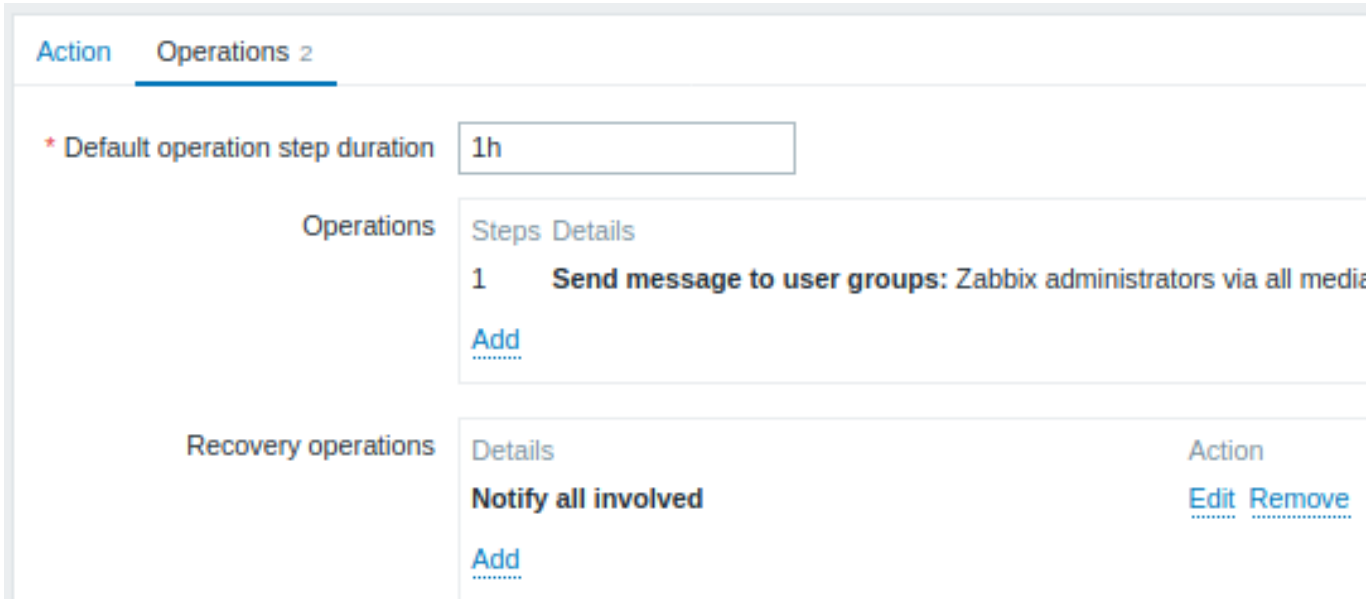


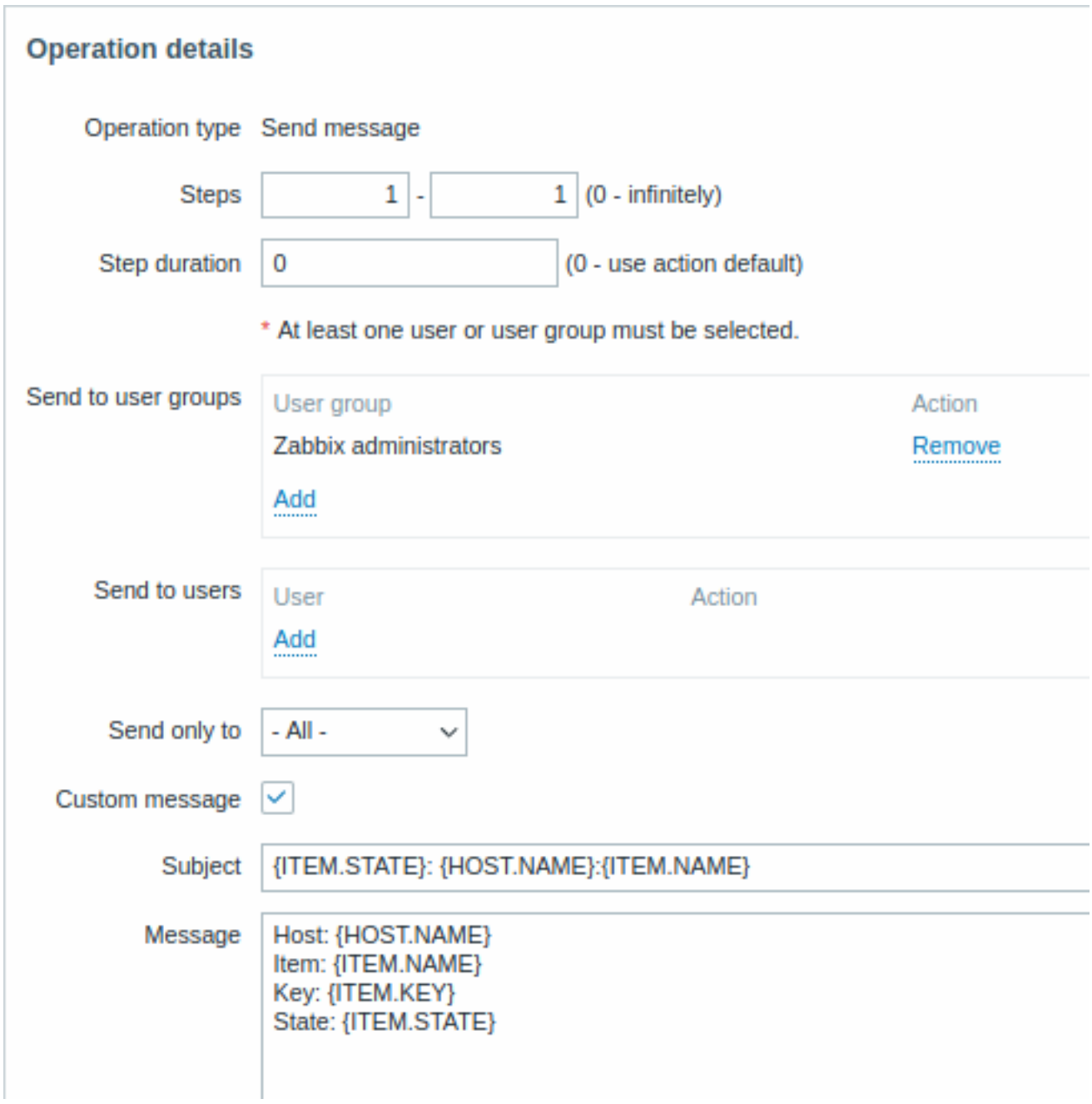
Don't forget to click on *Add* to actually list the condition in the *Conditions* block.

Step 4

In the **Operations** tab, click on *Add* in the *Operations* block and select some recipients of the message (user groups/users) and the media types (or 'All') to use for delivery.

Select *Custom message* checkbox if you wish to enter the custom subject/content of the problem message.





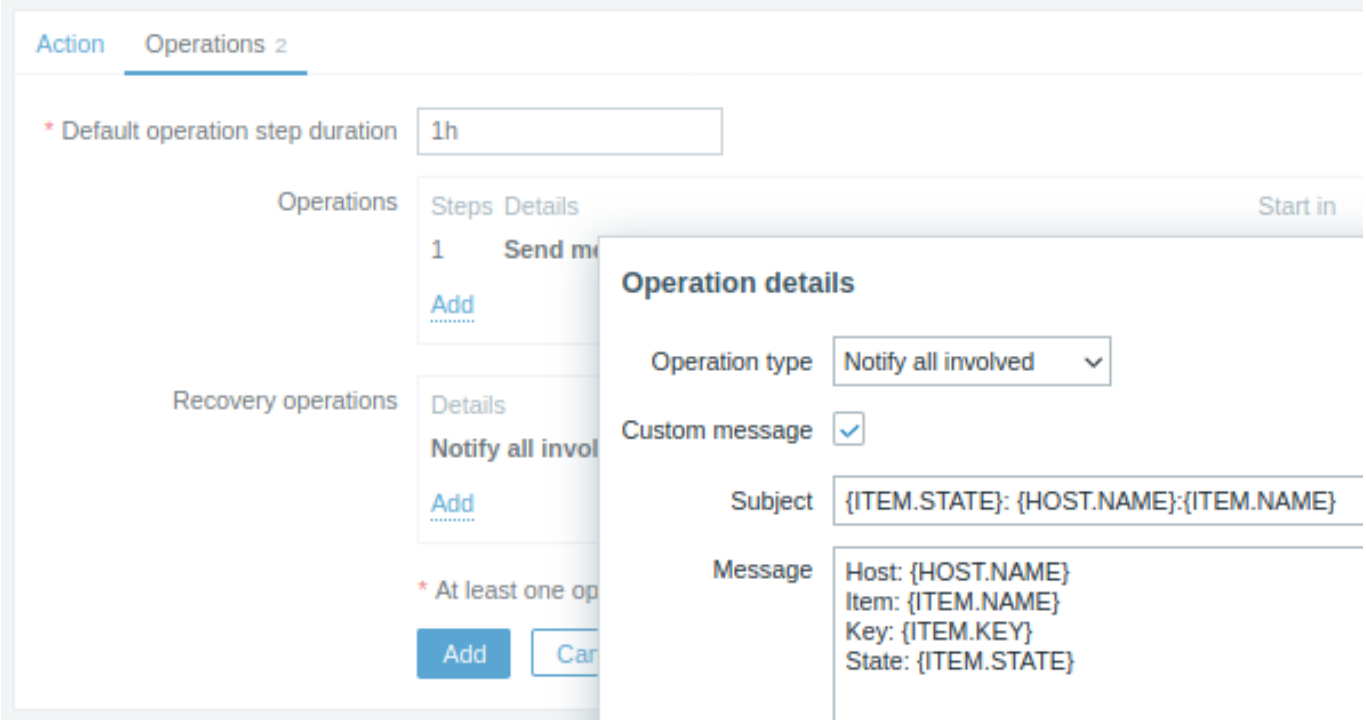
Click on *Add* to actually list the operation in the *Operations* block.

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, select the operation type, the recipients of the message (user groups/users) and the media types (or 'All') to use for delivery.

Select *Custom message* checkbox if you wish to enter the custom subject/content of the problem message.



Click on *Add* in the *Operation details* popup 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

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. 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)

If a macro function is applied to the macro in locations not supporting macro functions then the function is ignored.

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
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")}	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.

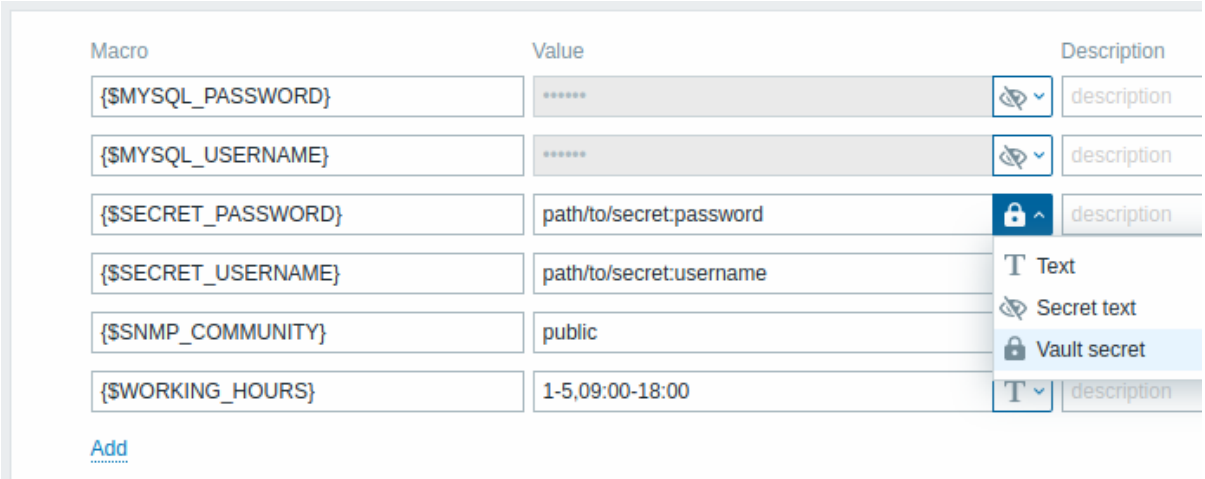
Configuration

To define user macros, go to the corresponding location in the frontend:

- for global macros, visit *Administration* → *General* → *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:

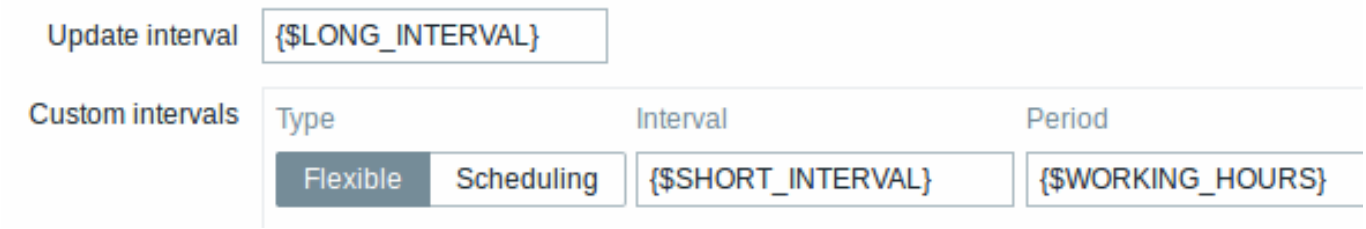
- define {$INTERVAL} macro and use it in the item update interval;
- use {$INTERVAL} as parameter of the agent unavailability trigger:

```
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 *Administration → User → Media*;
- use it to set up more frequent item polling during working hours:

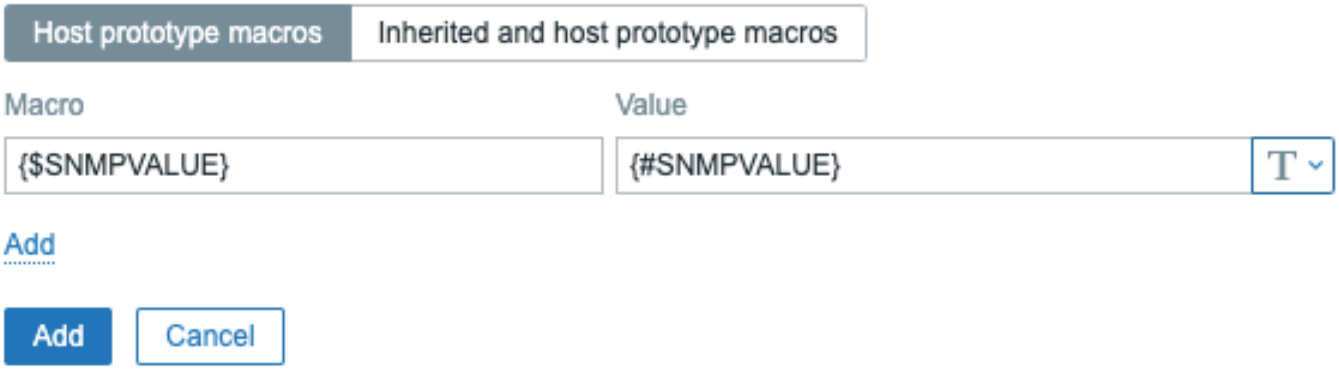


- use it in the *Time period* action condition;
- adjust the working time in *Administration* → *General* → *Macros*, if needed.

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:

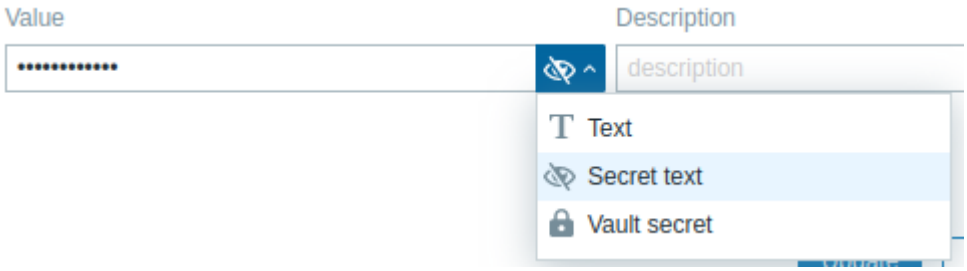
- Secret text
- Vault secret

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).

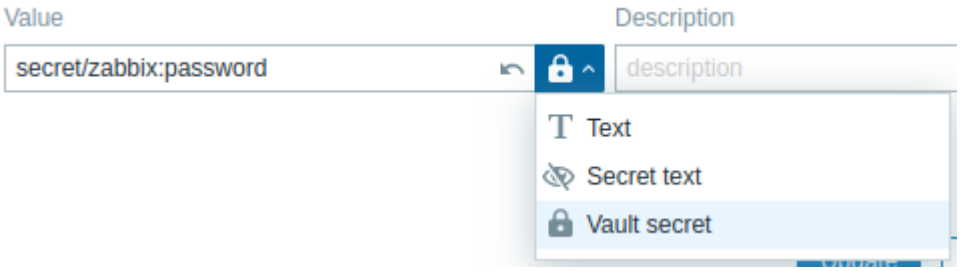


If you change macro value type or press *Set new value*, current value will be erased. To revert the original value, use the backwards arrow at the right end of the *Value* field (only available before saving new configuration). Reverting the value will not expose it.

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 formula
  - 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 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}
```

{HOST.HOST<1-9>} and {ITEM.KEY<1-9>} macros are supported inside expression macros. {ITEM.KEY<1-9>} macros are supported in expression macros since Zabbix 6.2.3.

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}
```

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:

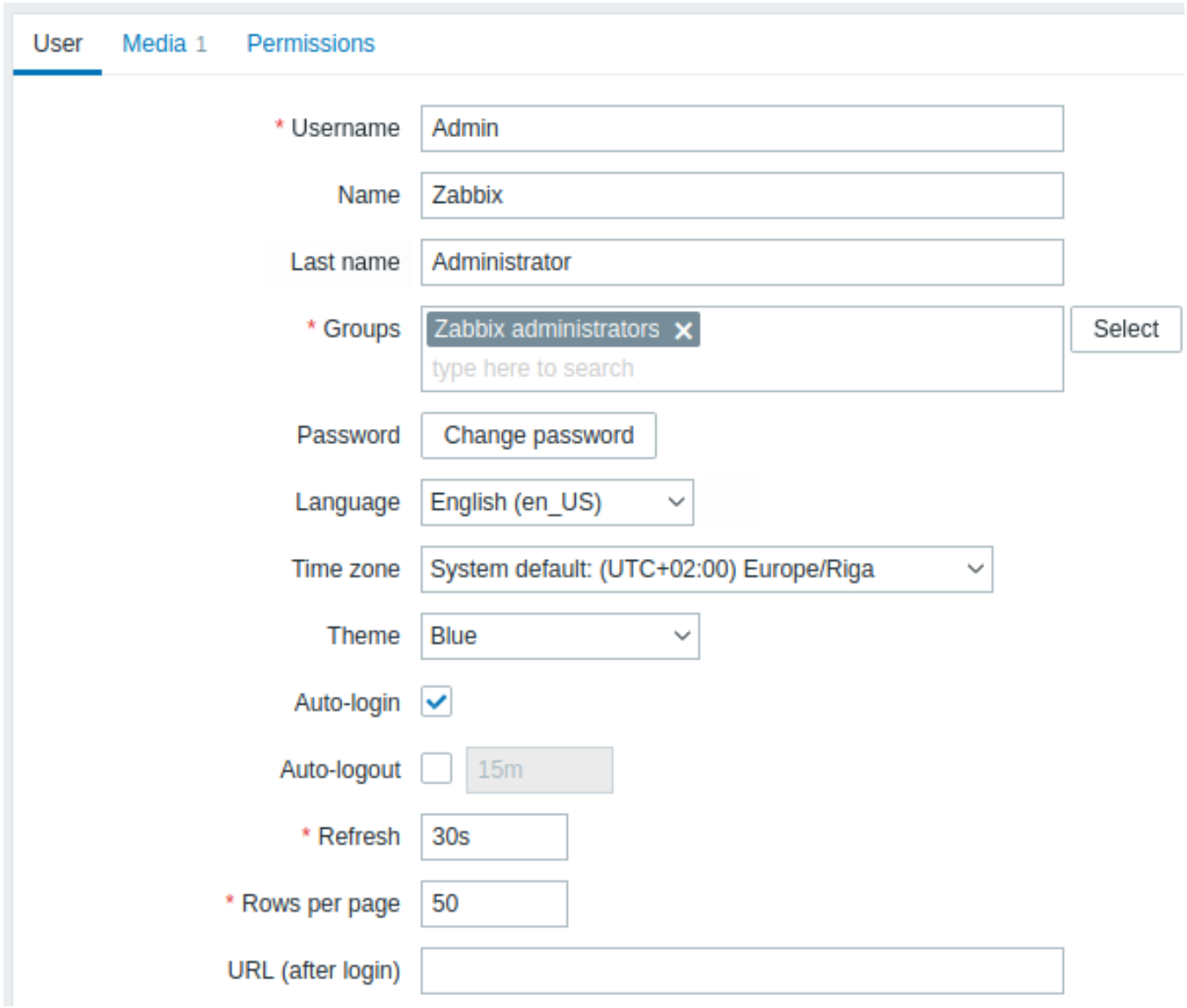
- *Admin* - a Zabbix superuser with full permissions;
- *guest* - a special Zabbix user. The 'guest' user is disabled by default. If you add it to the Guests user group, you may access monitoring pages in Zabbix without being logged in. Note that by default, 'guest' has no permissions on Zabbix objects.

To configure a new user:

- Go to *Administration* → *Users*
- Click on *Create user* (or on the 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. With an existing password, contains a *Password* button, clicking on which opens the password fields. Note that passwords longer than 72 characters will be truncated.
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

Parameter	Description
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.
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.

See the Media types section for details on configuring user media.

Permissions

The *Permissions* tab contains information on:

- The user role. Users cannot change their own role.
- The user type (User, Admin, Super Admin) that is defined in the role configuration.
- Host and template groups the user has access to. Users of type 'User' and 'Admin' do not have access to any groups, templates and hosts by default. To get the access they need to be included in user groups that have access to respective entities.
- Access rights to sections and elements of Zabbix frontend, modules, and API methods. Elements to which access is allowed are displayed in green color. Light gray color means that access to the element is denied.
- Rights to perform certain actions. Actions that are allowed are displayed in green color. Light gray color means that a user does not have the rights to perform this action.

See the User permissions page for details.

2 Permissions

Overview

You can differentiate user permissions in Zabbix by defining the respective user role. Then the unprivileged users need to be included in user groups that have access to host group data.

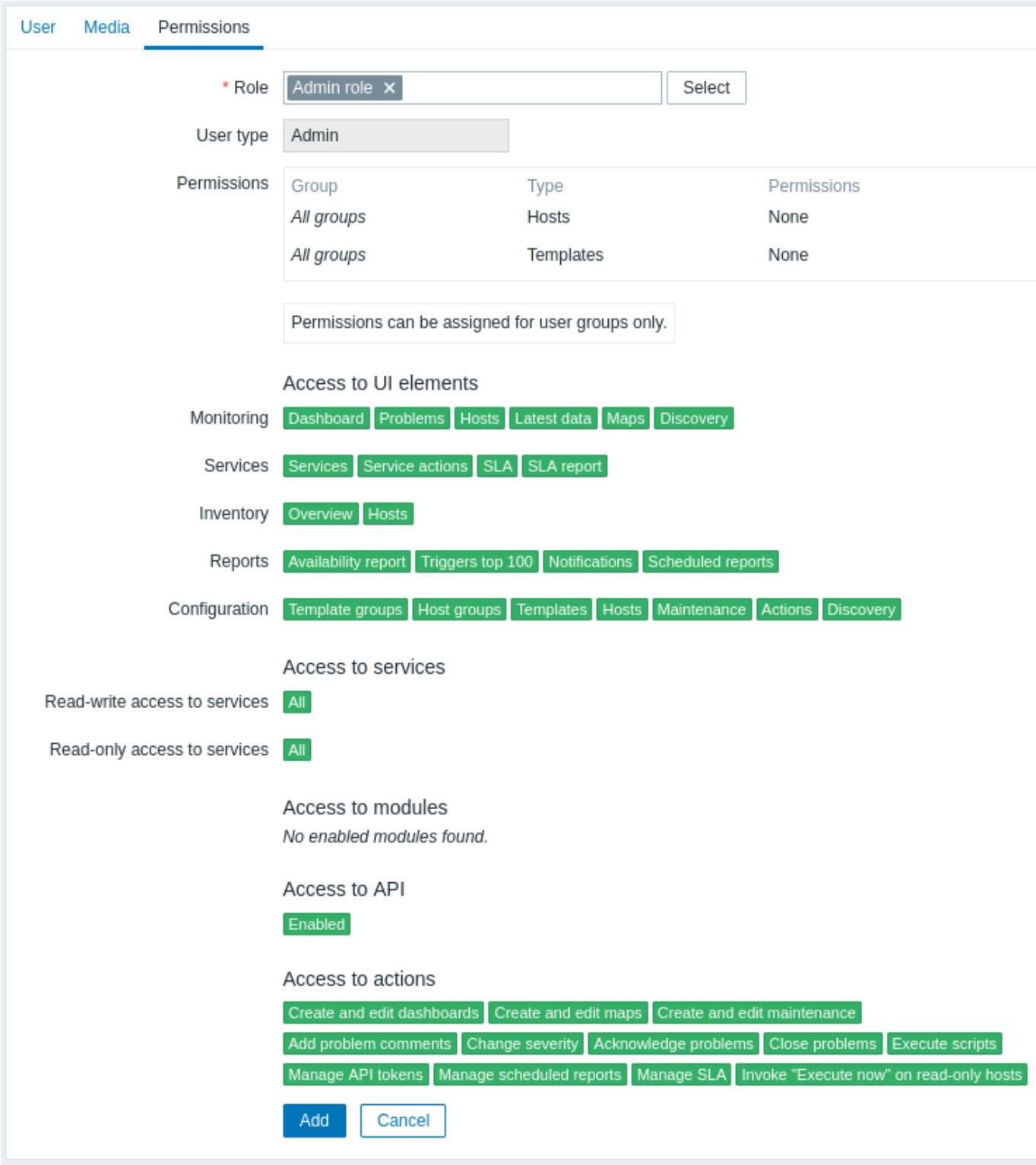
User role

The user role defines which parts of UI, which API methods, and which actions are available to the user. The following roles are pre-defined in Zabbix:

User type	Description
Guest role	The user has access to the Monitoring, Inventory, and Reports menu sections, but without the rights to perform any actions.
User role	The user has access to the Monitoring, Inventory, and Reports menu sections. The user has no access to any resources by default. Any permissions to host or template groups must be explicitly assigned.
Admin role	The user has access to the Monitoring, Inventory, Reports and Configuration menu sections. The user has no access to any host groups by default. Any permissions to host or template groups must be explicitly given.
Super Admin role	The user 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.

User roles are configured in the *Administration→User roles* section. Super Admins can modify or delete pre-defined roles and create more roles with custom sets of permissions.

To assign a role to the user, go to the Permissions tab in the user configuration form, locate the *Role* field and select a role. Once a role is selected a list of associated permissions will be displayed below.



Permissions to groups

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.

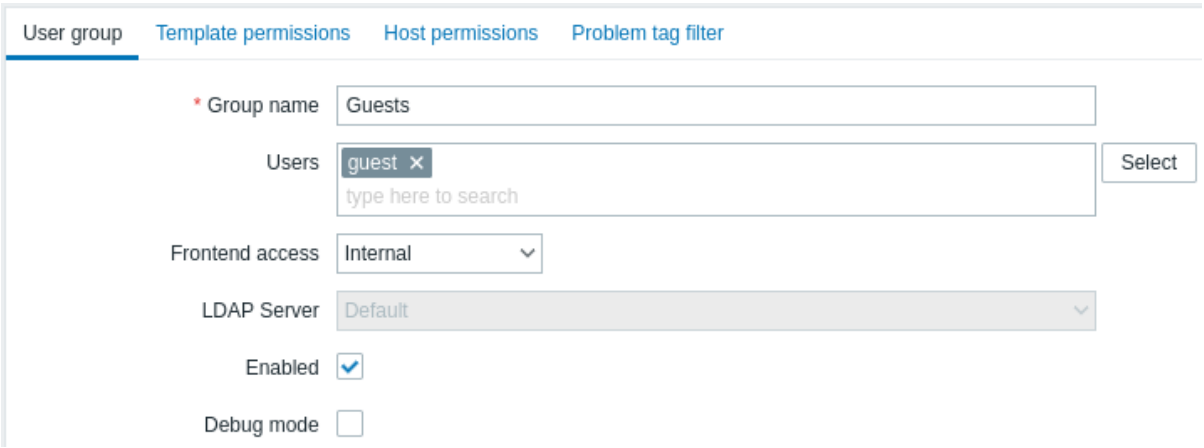
A user can belong to any number of groups.

Configuration

To configure a user group:

- Go to *Administration → User groups*
- Click on *Create user group* (or on the group name to edit an existing group)
- Edit group attributes in the form

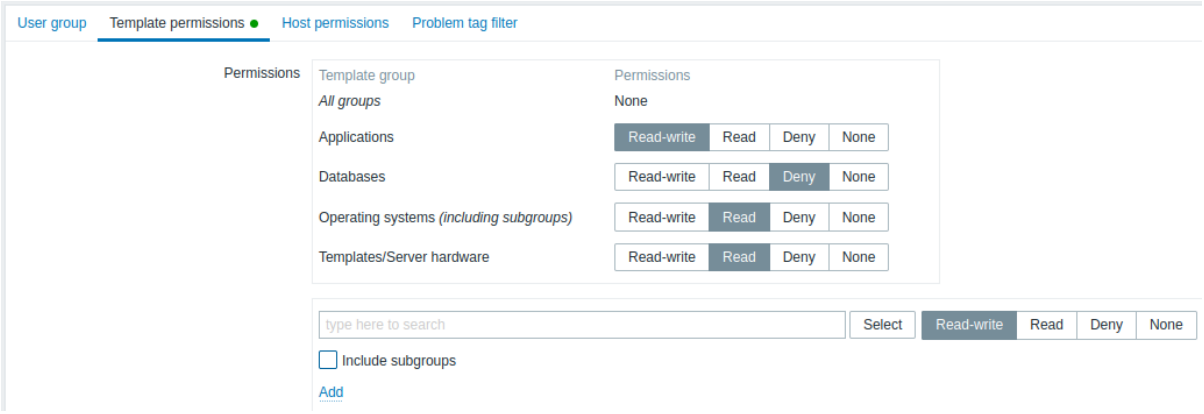
The **User group** tab contains general group attributes:



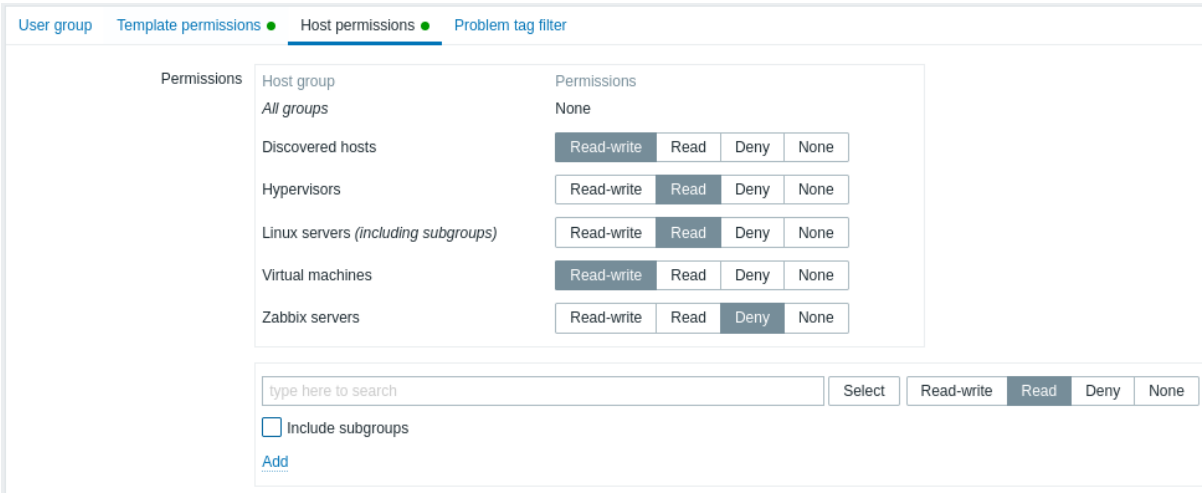
All mandatory input fields are marked with a red asterisk.

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:

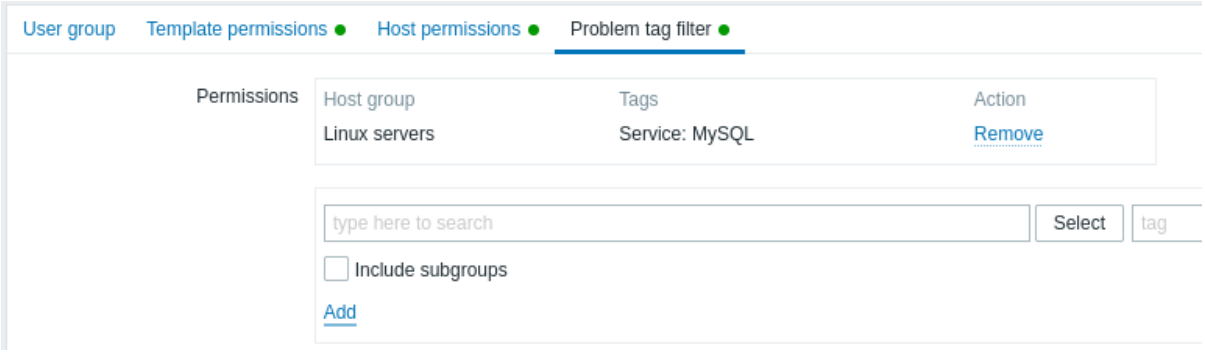
- **Read-write** - read-write access to a group;
- **Read** - read-only access to a group;
- **Deny** - access to a group denied;
- **None** - no permissions are set.

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 .
```

CyberArk configuration

This section explains how to configure Zabbix to retrieve secrets from CyberArk Vault CV12.

The vault should be installed and configured as per the official CyberArk documentation.

To learn about configuring TLS in Zabbix, see Storage of secrets section.

Database credentials

Access to a secret with database credentials is configured for each Zabbix component separately.

Server and proxies

To obtain database credentials for Zabbix server or proxy from the vault, specify the following configuration parameters in the configuration file:

- *Vault* - specifies which vault provider should be used.

- *VaultURL* - vault server HTTP[S] URL.
- *VaultDBPath* - query to the vault secret containing database credentials. The credentials 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 these configuration parameters (except VaultDBPath) 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.

Example

In zabbix_server.conf, specify:

```
Vault=CyberArk
VaultURL=https://127.0.0.1:1858
VaultDBPath=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_server_database
VaultTLSCertFile=cert.pem
VaultTLSKeyFile=key.pem
```

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
```

Vault response, from which the keys "Content" and "UserName" should be retrieved:

```
{
"Content": <password>,
"UserName": <username>,
"Address": <address>,
"Database" :<Database>,
"PasswordChangeInProcess":<PasswordChangeInProcess>
}
```

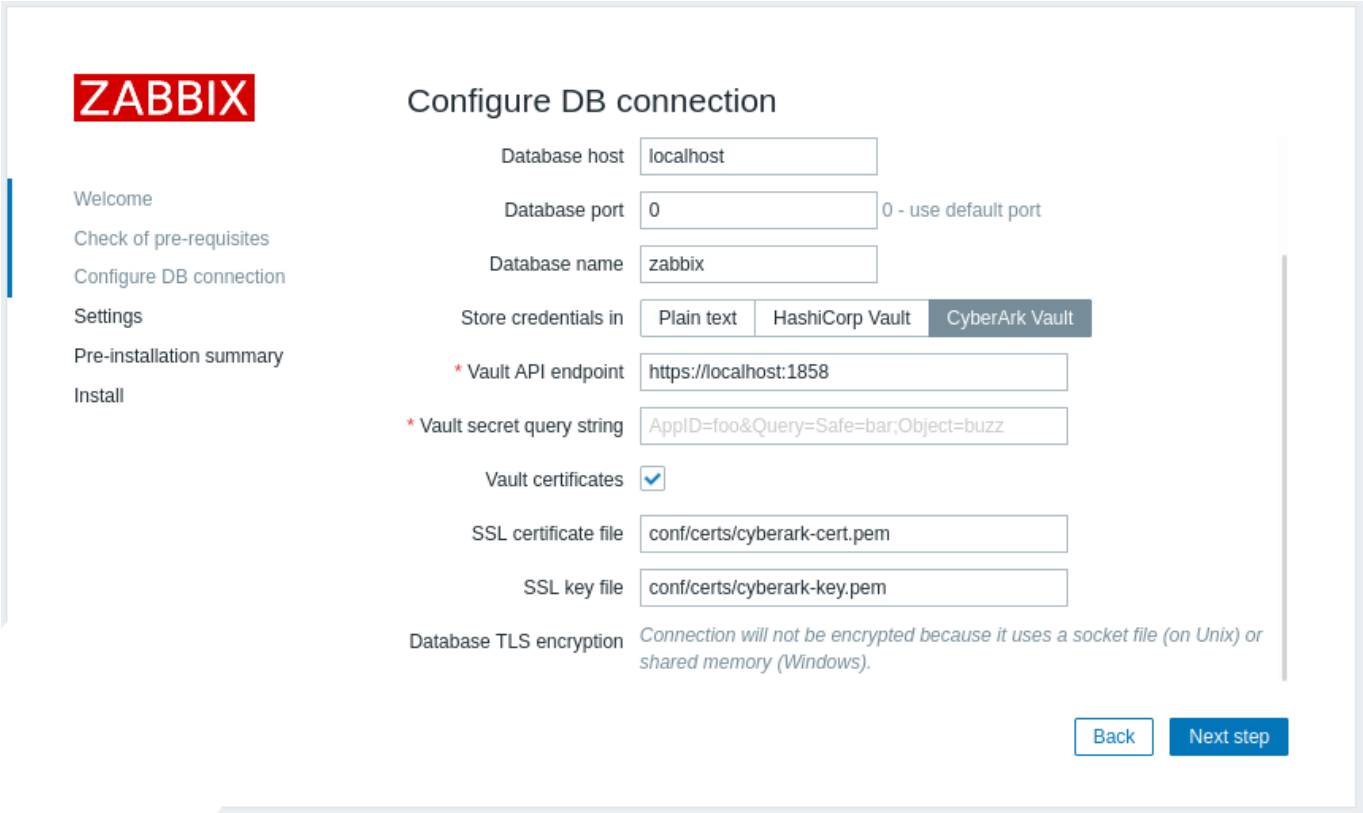
As a result, Zabbix will use the following credentials for database authentication:

- Username: <username>
- Password: <password>

Frontend

To obtain database credentials for Zabbix frontend from the vault, specify required settings during frontend installation.

At the *Configure DB Connection* step, set *Store credentials in* parameter to CyberArk Vault.



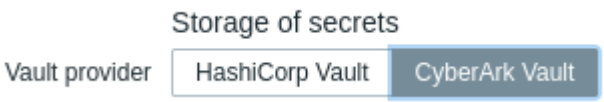
Then, fill in additional parameters:

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:key`
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 SSL certificate file. The file must be in PEM format. If the certificate file contains also 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:

- Set the *Vault provider* parameter in the *Administration -> General -> Other* web interface section to CyberArk Vault.



- Make sure that Zabbix server is configured to work with CyberArk Vault.

The macro value should contain a query (as `query:key`).

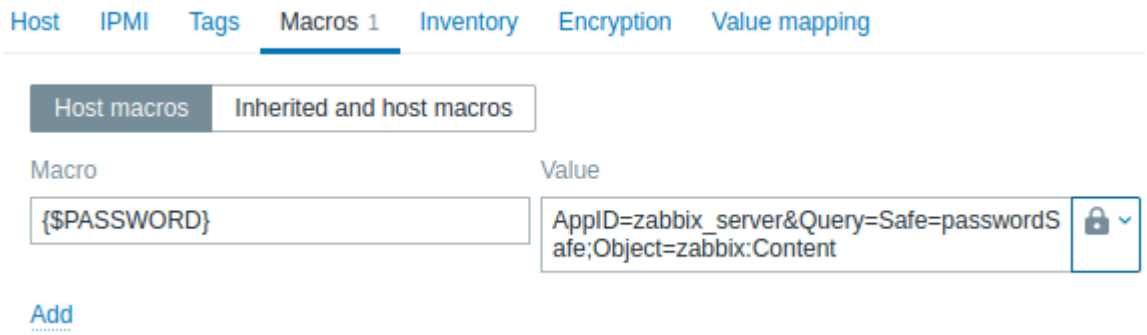
See Vault secret macros for detailed information about macro value processing by Zabbix.

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

In Zabbix: add user macro {$PASSWORD} with type *Vault secret* and value: AppID=zabbix_server&Query=Safe=passwordSafe;Object



Zabbix will send 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
```

Vault response, from which the key ”Content” should be retrieved:

```
{
"Content": <password>,
"UserName": <username>,
"Address": <address>,
"Database" :<Database>,
"PasswordChangeInProcess":<PasswordChangeInProcess>
}
```

Macro resolves to the value: <password>

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 per the official HashiCorp documentation.

To learn about configuring TLS in Zabbix, see Storage of secrets section.

Database credentials

Access to a secret with database credentials is configured for each Zabbix component separately.

Server and proxies

To obtain database credentials for Zabbix server or proxy from the vault, specify the following configuration parameters in the configuration file:

- *Vault* - specifies 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 these configuration parameters (except VaultDBPath) 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.

Zabbix server and Zabbix proxy will additionally read ”VAULT_TOKEN” environment variable once during startup and unset it so that it would not be available through forked scripts; it is an error if both VaultToken and VAULT_TOKEN contain value.

Example

In zabbix_server.conf, specify:

```
Vault=HashiCorp
VaultToken=hvs.CAESIIG_PILmULFYOsEyWHxkZ2mF2a8VPKNLE8eHqd4autYGGh4KHGh2cy5aeTY0NFNSaUp3ZnpWbDF1RUNjUkNTZEg
VaultURL=https://127.0.0.1:8200
VaultDBPath=secret/zabbix/database
```

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 secrets with keys username and password under mount point "secret/" and path "secret/zabbix/data
$ 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 th
$ curl --header "X-Vault-Token: <VaultToken>" https://127.0.0.1:8200/v1/secret/data/zabbix/database
```

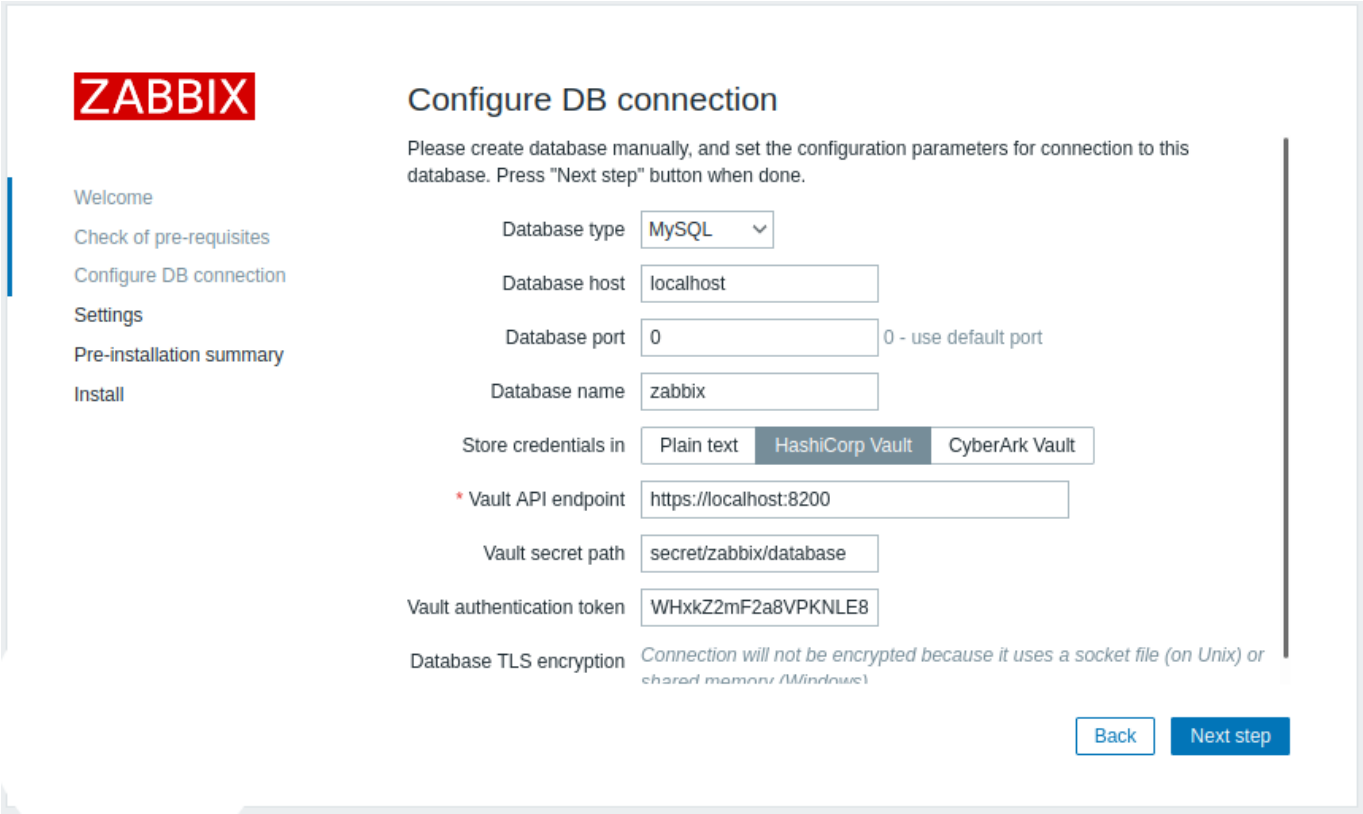
As a result of this configuration, Zabbix server will retrieve the following credentials for database authentication:

- Username: zabbix
- Password: <password>

Frontend

To obtain database credentials for Zabbix frontend from the vault, specify required settings during frontend installation.

At the *Configure DB Connection* step, set *Store credentials in* parameter to HashiCorp Vault.



Then, fill in 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`

Parameter	Mandatory	Default value	Description
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_frontend`
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:

- The *Vault provider* parameter in the *Administration -> General -> Other* web interface section is set to HashiCorp Vault (default).



- Zabbix server is configured to work with HashiCorp Vault.

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 'VaultToken' parameter) must provide read-only access to this path.

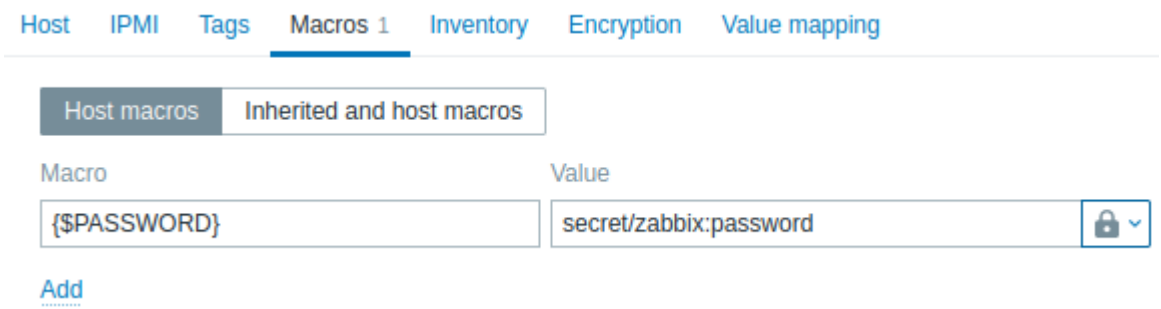
See Vault secret macros for detailed information about macro value processing by Zabbix.

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 "zabbix" is the path) and, in case of Vault macros, a colon can only be used to separate a path/query from a key. It is possible to URL-encode "/" and ":" 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", as "foo%2Fbar/zabbix") and if a mount point name or path need to contain a colon.

Example

In Zabbix: add user macro {$PASSWORD} with type *Vault secret* and value `secret/zabbix:password`



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 th
$ curl --header "X-Vault-Token: <VaultToken>" https://127.0.0.1:8200/v1/secret/data/zabbix
```

Now the macro {$PASSWORD} will resolve to the value: <password>

14 Scheduled reports

Overview

This section provides information about configuring scheduled reports.

> **Attention:**
> Currently the support of scheduled reports is experimental.

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).

> **Note:**
> For multi-page dashboards, only the first page is included in the PDF report.

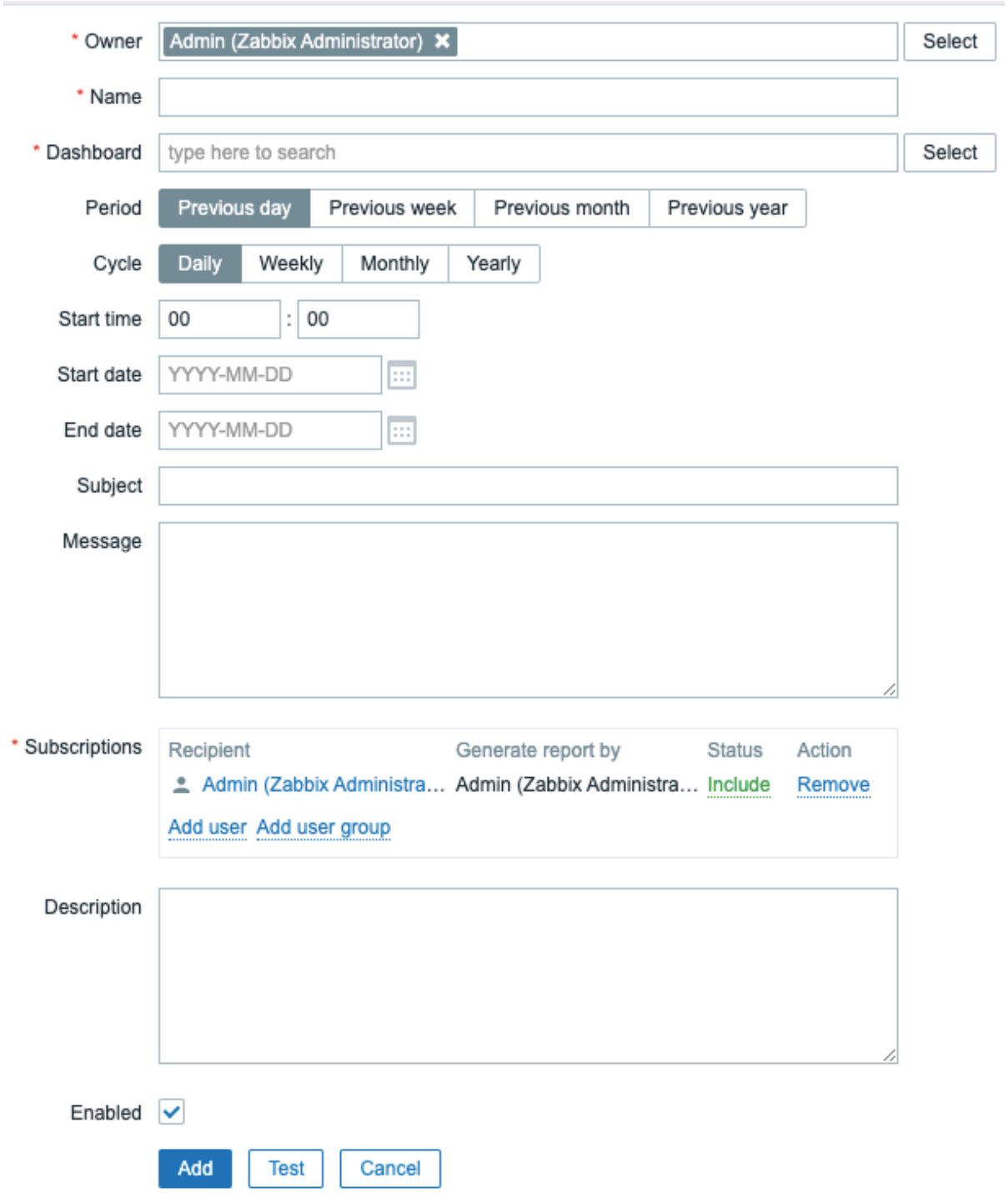
To create a scheduled report in Zabbix frontend, do the following:

- Go to: *Reports* → *Scheduled reports*
- Click on *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, pressing the *Clone* button, and then saving 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 on Select next to the field and select a dashboard from the list in a popup window. 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 one of the available options: *Previous day*, *Previous week*, *Previous month*, *Previous 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 the 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	The date when regular report generation should be started
End date	The 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. Press *Add user* or *Add user group* to add more recipients. Press on the username to edit settings: *Generate report by* - whether the report should be generated on behalf of the report owner or the recipient. *Status* - select *Include* to send the report to user or *Exclude* to prevent sending the report to this user. At least one user must have *Include* status. *Exclude* status can be used to exclude specific users from a user group that is included. Note that users with insufficient permissions***** 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.

*Users with insufficient permissions are users who have a role based on the Admin user type and are not members of the user group the recipient or the report owner is a member of.

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.

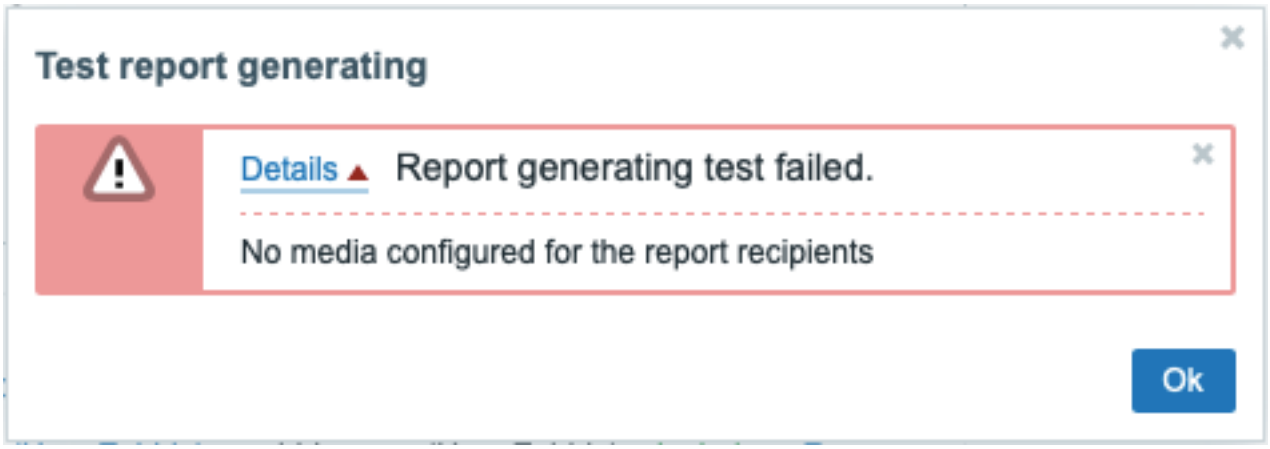
Testing

To test a report, click on the Test button at the bottom of the report configuration form.

Note:
Test button is not available, if a 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 'generated by' user settings are ignored.

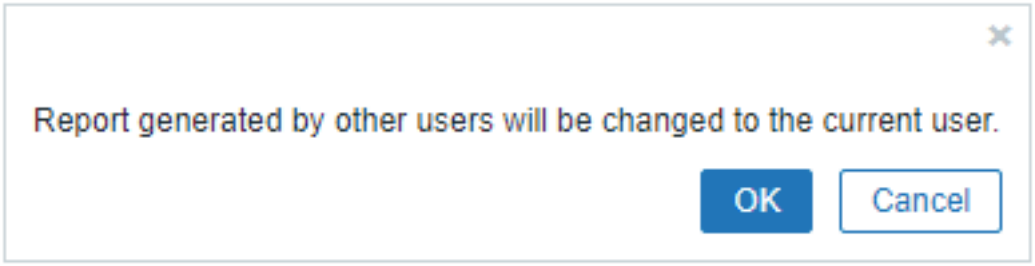
If the configuration is incorrect, an error message is displayed describing the possible cause.



Updating a report

To update an existing report, press on the report name, then make required configuration changes and press Update button.

If an existing report is updated by another user and this user changes the Dashboard, upon pressing the Update button a warning message ”Report generated by other users will be changed to the current user” will be displayed.



Pressing OK at this step will lead to the following changes:

- *Generated by* settings will be updated to display the user who edited the report last (unless *Generated 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.

Pressing Cancel will close the popup window and cancel the report update.

Cloning a report

To quickly clone an existing report, press 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 that clones a report has no permissions to a dashboard, the Dashboard field will be cleared.
- If the user that clones a report has no permissions to some users or user groups in the Subscriptions list, inaccessible recipients will not be cloned.
- *Generated by* settings will be updated to display the current user (unless *Generated by* is set to the Recipient).

Change required settings and the report name, then press Add.

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.

- Service actions

In this section you can 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.

- SLA

In this section you can define service level agreements and set service level objectives for specific services.

- SLA report

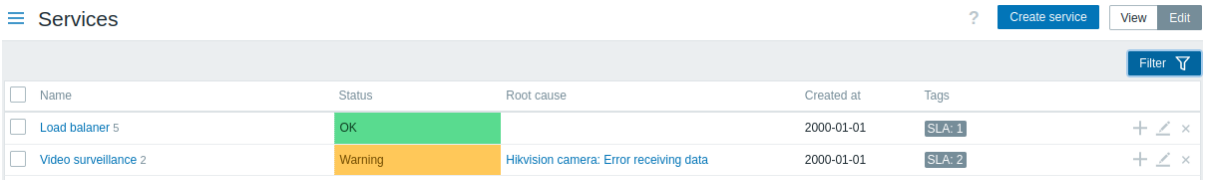
In this section you can view SLA reports.

See also:

- SLA monitoring configuration example
- Notes about upgrading services from Zabbix versions below 6.0

1 Service tree

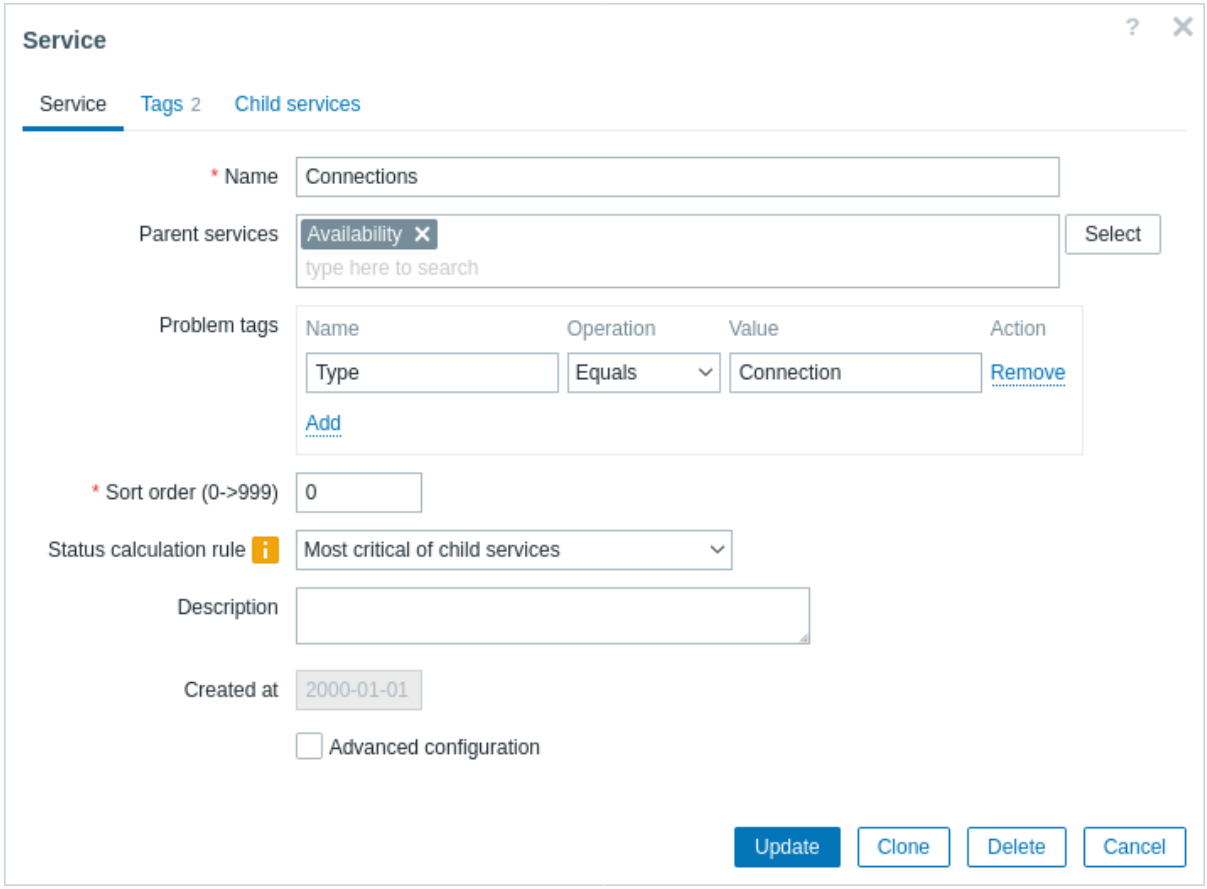
Service tree is configured in the *Services->Services* menu section. In the upper right corner, switch from View to the Edit mode.



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.

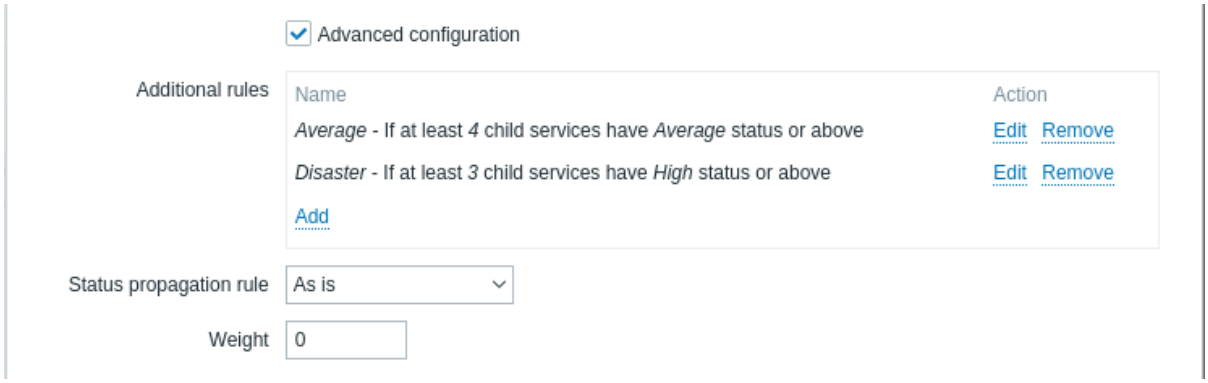
Service configuration In the **Service** tab, specify required service parameters:



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.
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).

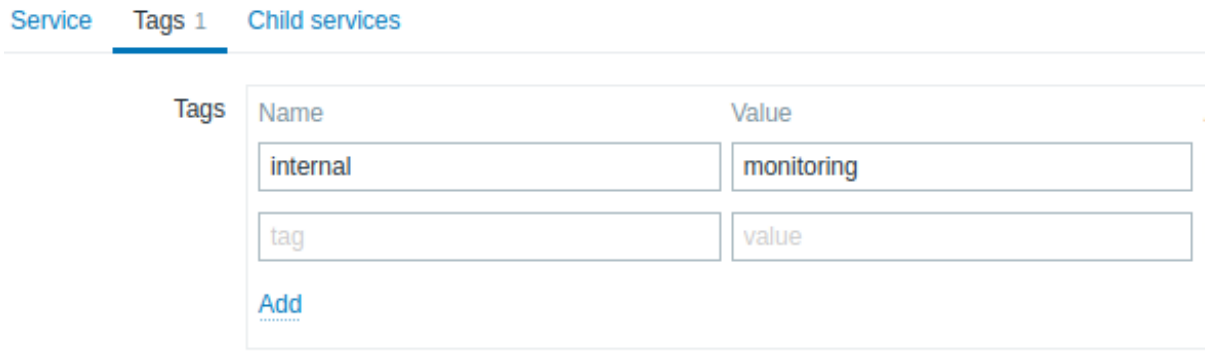
The **Child services** tab allows to specify dependant services. Click on *Add* to add a service from the list of existing services. If you want to add a new child service, save this service first, then click on a plus icon next to the service that you have just created.

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 *Tabs* 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.

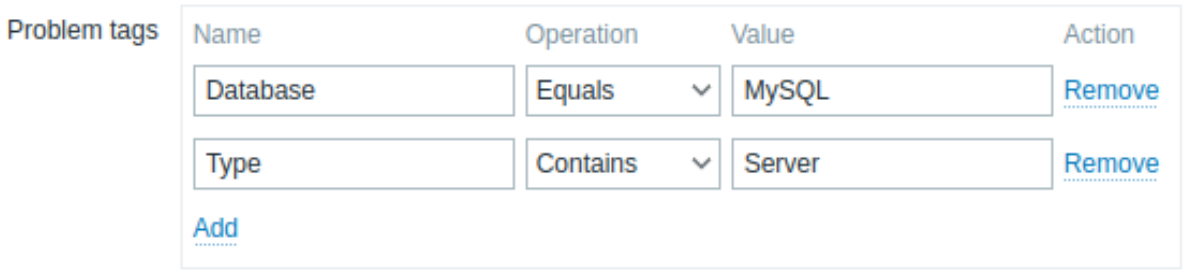


Problem tags

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.

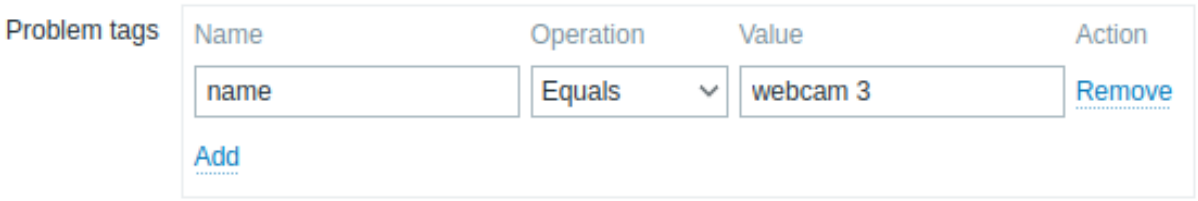


Note:
A problem in Zabbix inherits tags from the whole chain of templates, hosts, items, web scenarios, and triggers. Any of these tags can be used for matching problems to services.

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 Service actions

Overview In this section you can view and configure service actions.

Service actions are useful if you want some operations taking place as a result of service status change (OK ⇔ PROBLEM), for example:

- send message
- restart webserver

Service actions are functionally similar to other action types in Zabbix (for example, trigger actions).

Configuration To create a new service action, go to the *Service actions* subsection of the *Services* menu, then click on *Create action* in the upper right corner.

Service actions are configured in the same way as other types of actions in Zabbix. For more details, see configuring actions.

The key differences are:

- User access to service actions depends on access rights to services granted by user's role.
- Service actions support different set of conditions.

Conditions 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.
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 *Administration->Media types* menu. Otherwise, the notifications will not be sent.

3 SLA

Overview Once the services are created, you can start monitoring whether their performance is on track with Service Level Agreement (SLA).

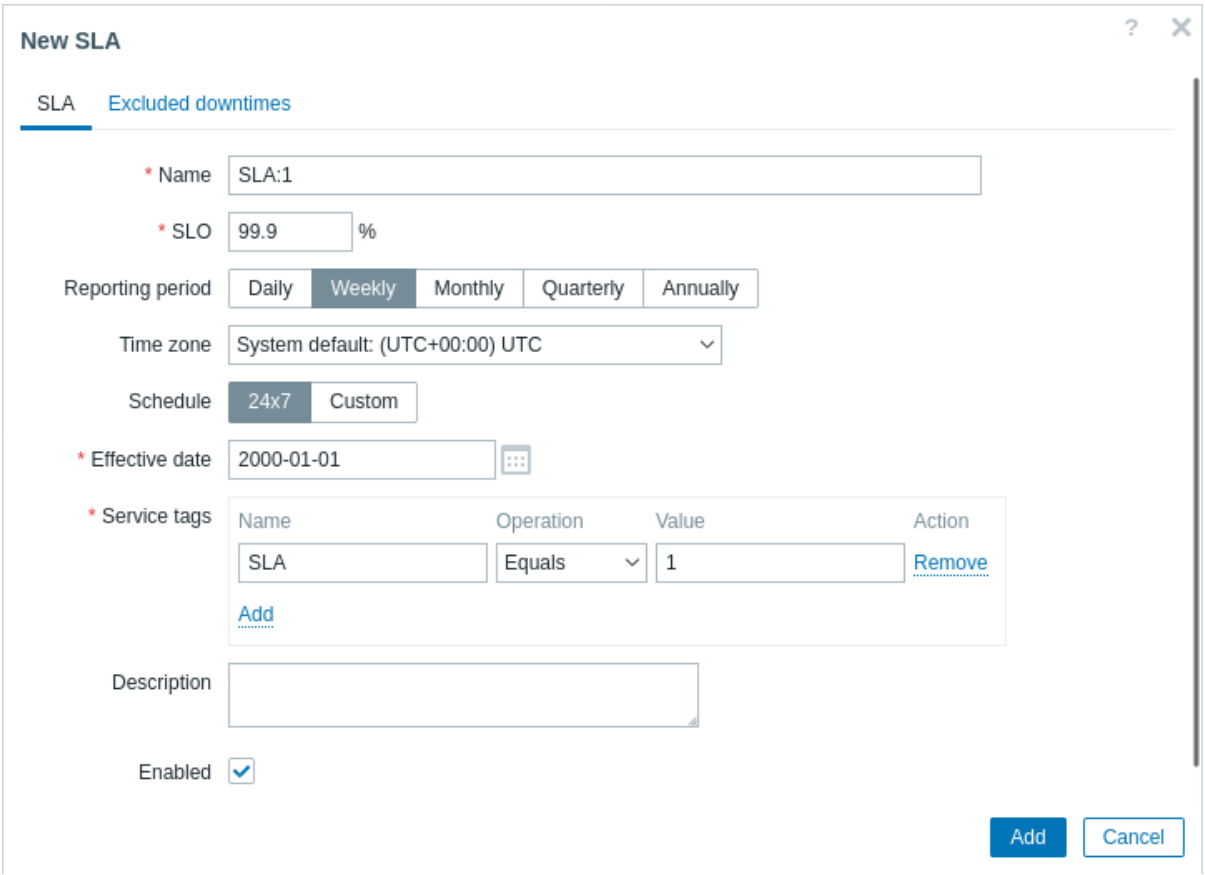
Services->SLA menu section allows to configure SLAs for various services. An SLA in Zabbix defines service level objective (SLO), expected uptime schedule and planned downtimes.

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.

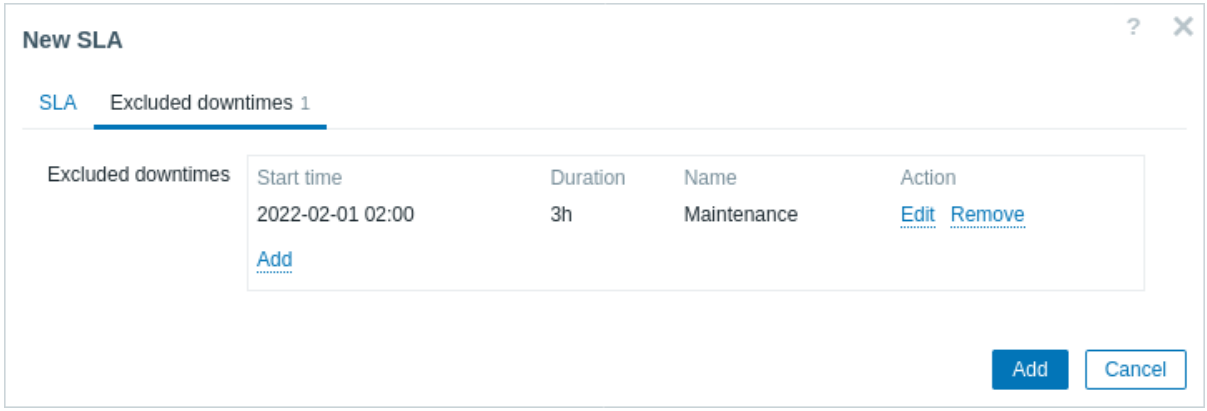
The **SLA** tab allows to specify general SLA parameters.



Parameter	Description
Name	Enter the SLA name.
SLO	Enter the service level objective (SLO) as percentage.
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.

Parameter	Description
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.

4 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`

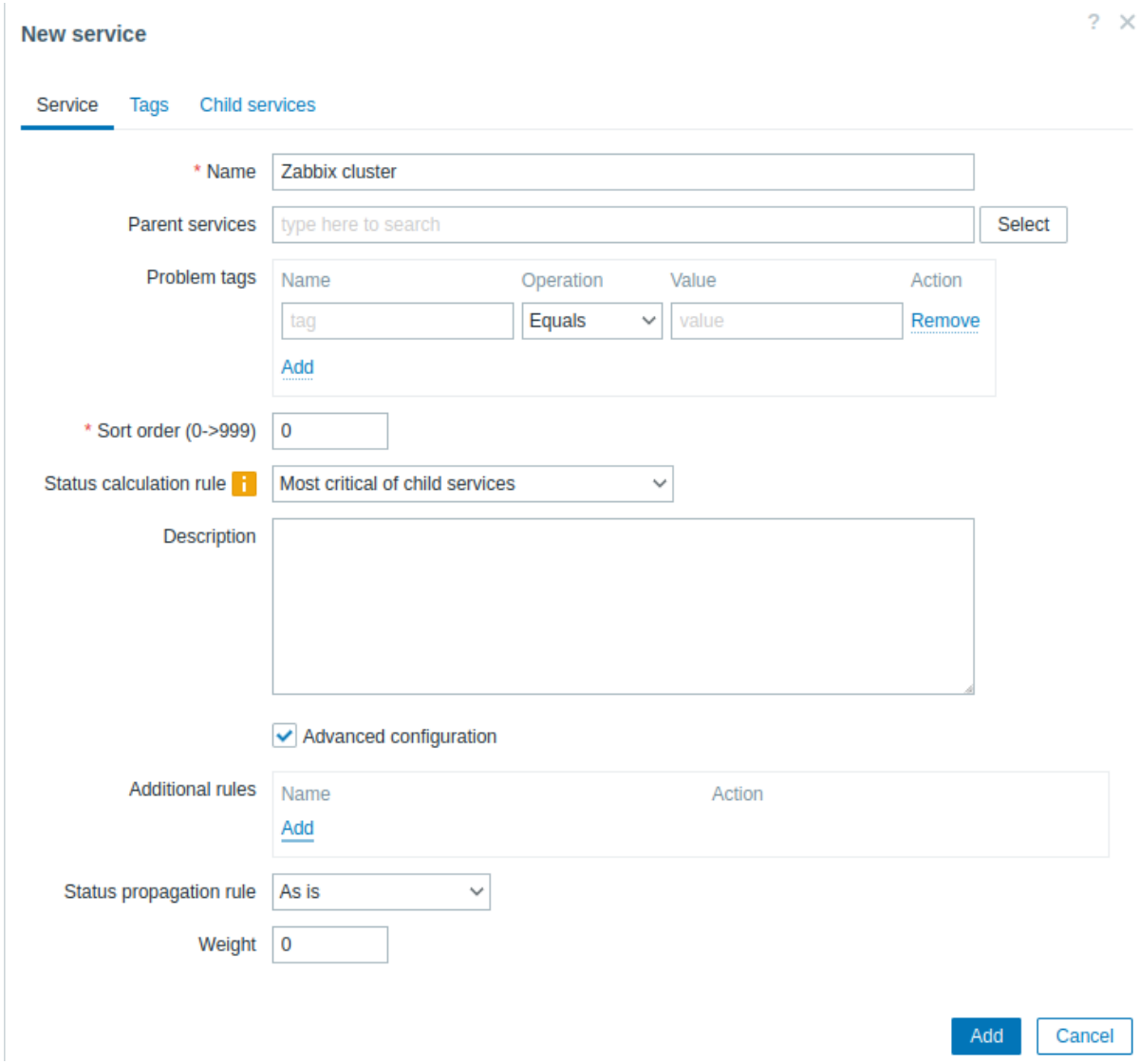
Service tree The next step is to build the service tree. In this example, the infrastructure is very basic and consists of three services: *Zabbix cluster* (parent) and two child services *Zabbix server node 1* and *Zabbix server node 2*.

```
Zabbix cluster
|
|- Zabbix server node 1
|- Zabbix server node 2
```

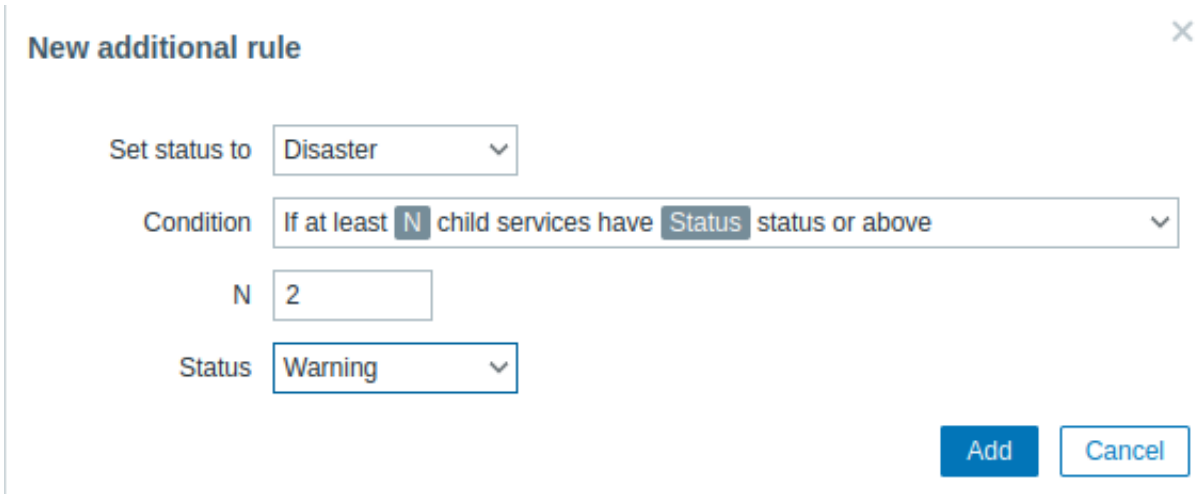
At the Services page, turn on *Edit* mode and press Create service:



In the service configuration window, enter name *Zabbix cluster* and mark the checkbox *Advanced configuration*.



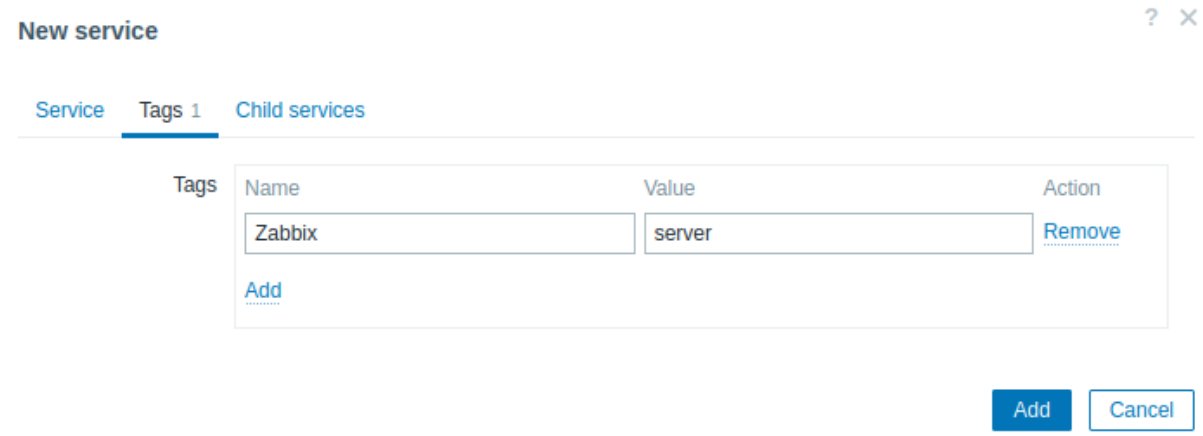
Configure additional rule:



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:

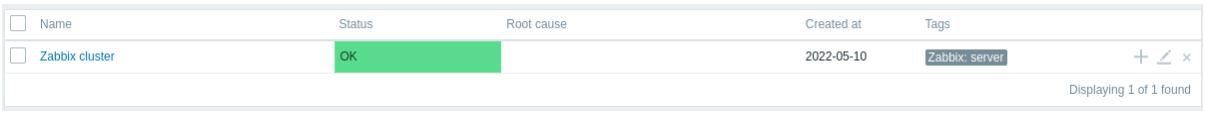
- Set status to: Disaster
- Condition: If at least `N` child services have `Status` status or above
- N: 2
- Status: Warning

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.

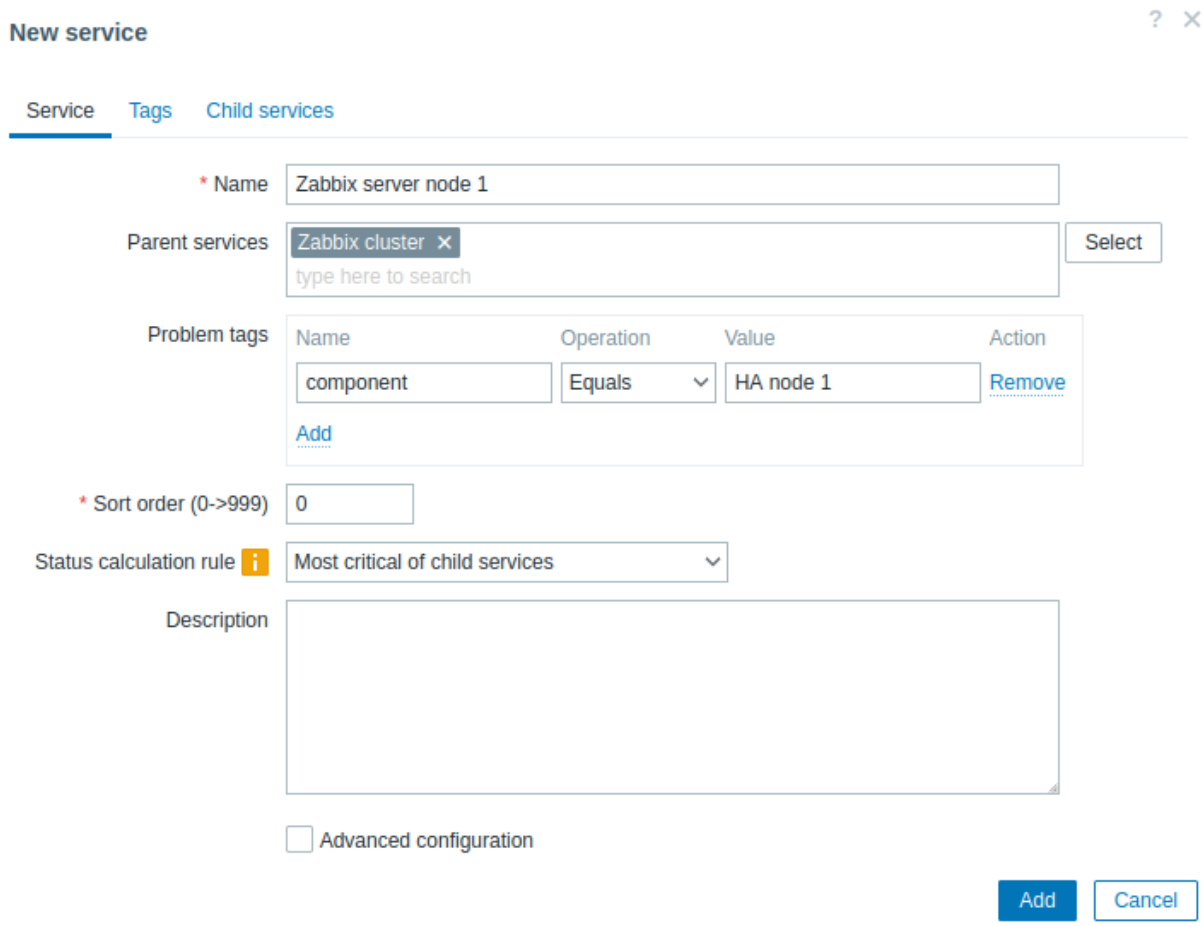
To add a child service, press on the plus icon next to the Zabbix cluster service (the icon is visible only in *Edit* mode).



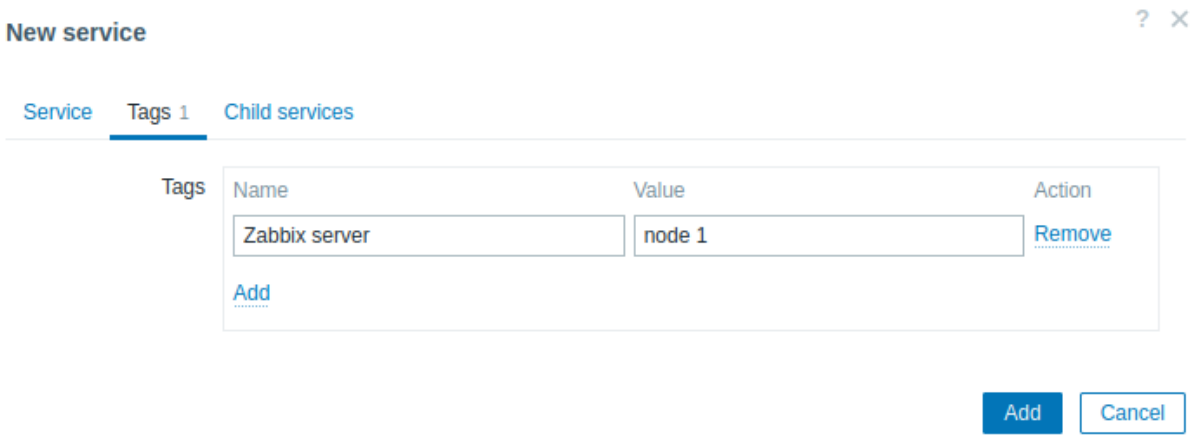
In the service configuration window, enter name *Zabbix server node 1*. Note that the Parent services parameter is already pre-filled with *Zabbix cluster*.

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 Service Level Agreement (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.

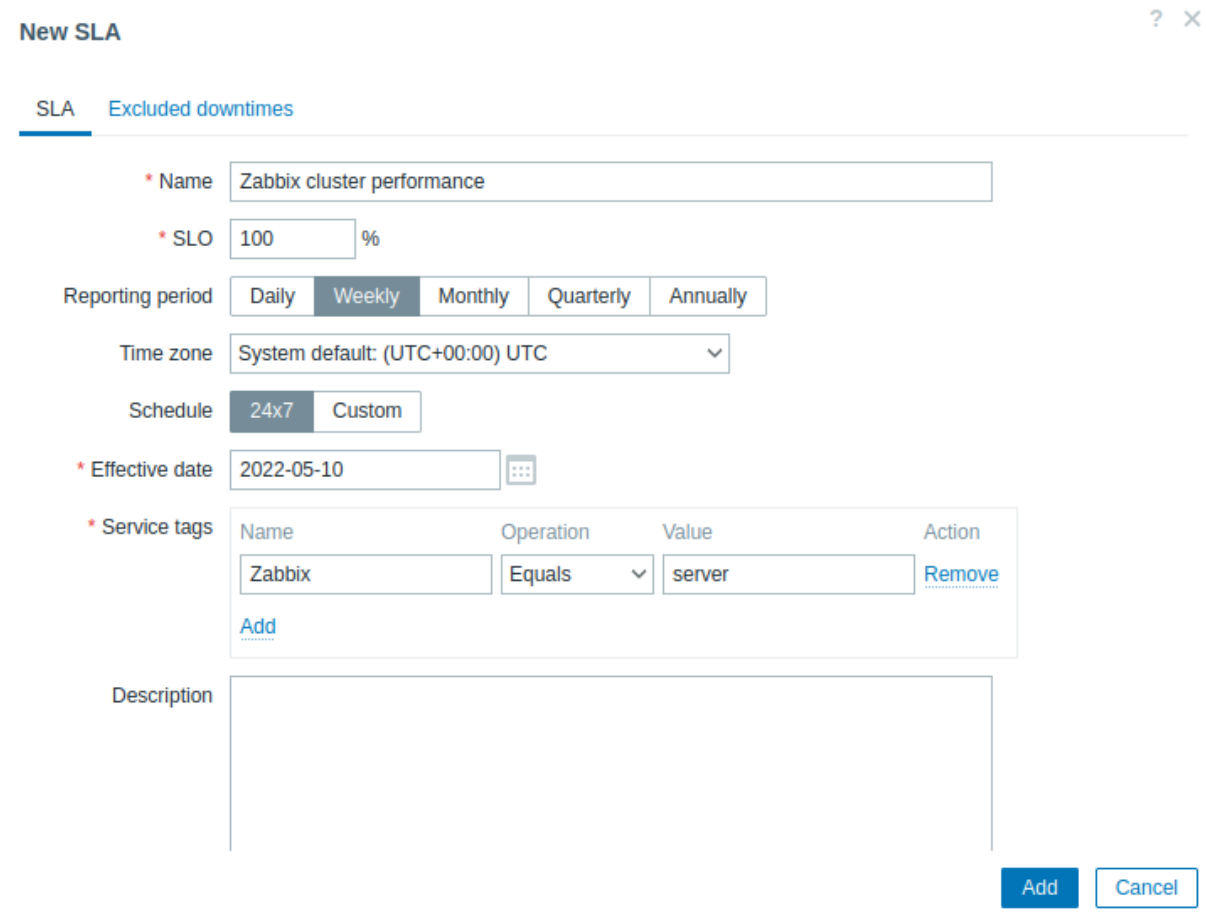
SLA In this example, expected Zabbix cluster performance is 100% excluding semi-annual one hour maintenance period.

First, you need to add a new Service Level Agreement (SLA).

Go to the *Services->SLA* menu section and press Create SLA. Enter name *Zabbix cluster performance* and set the SLO to 100%.

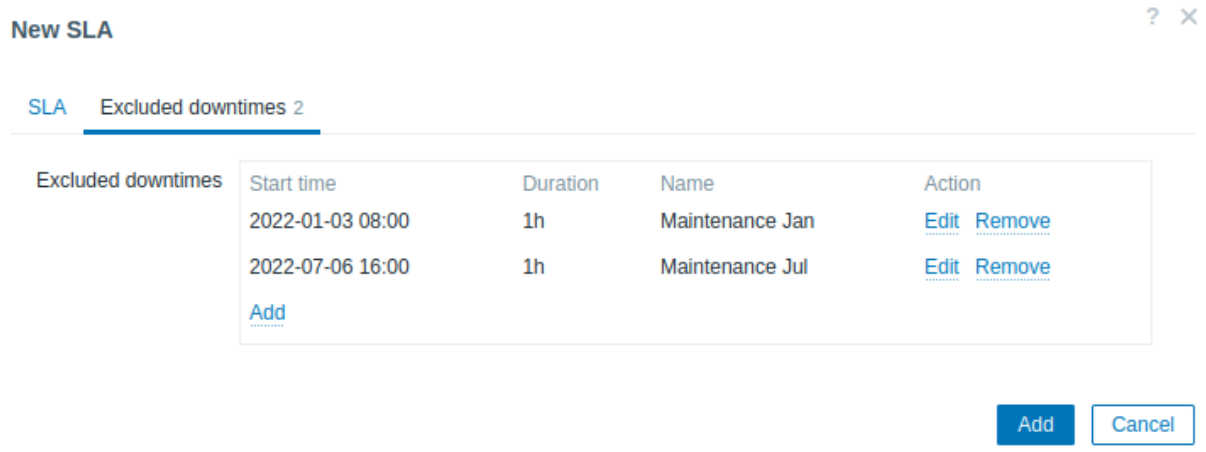
The service Zabbix cluster has a service tag `Zabbix:server`. To use this SLA for measuring performance of Zabbix cluster, in the *Service tags* parameter, specify:

- 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.

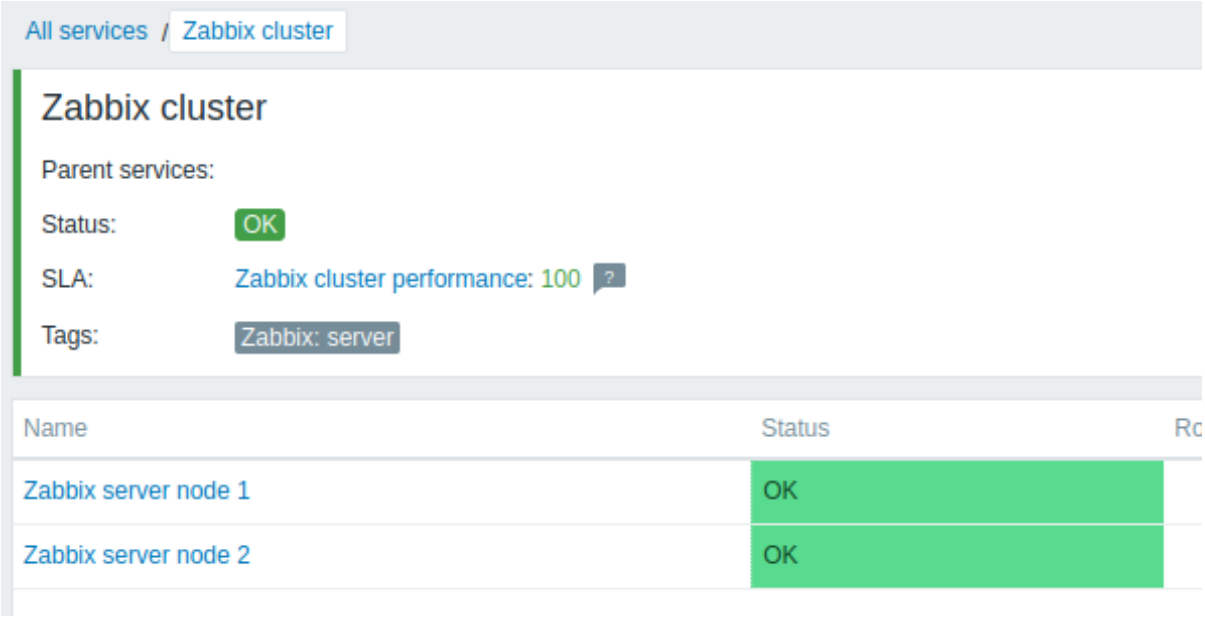


Press Add to save the new SLA.

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.

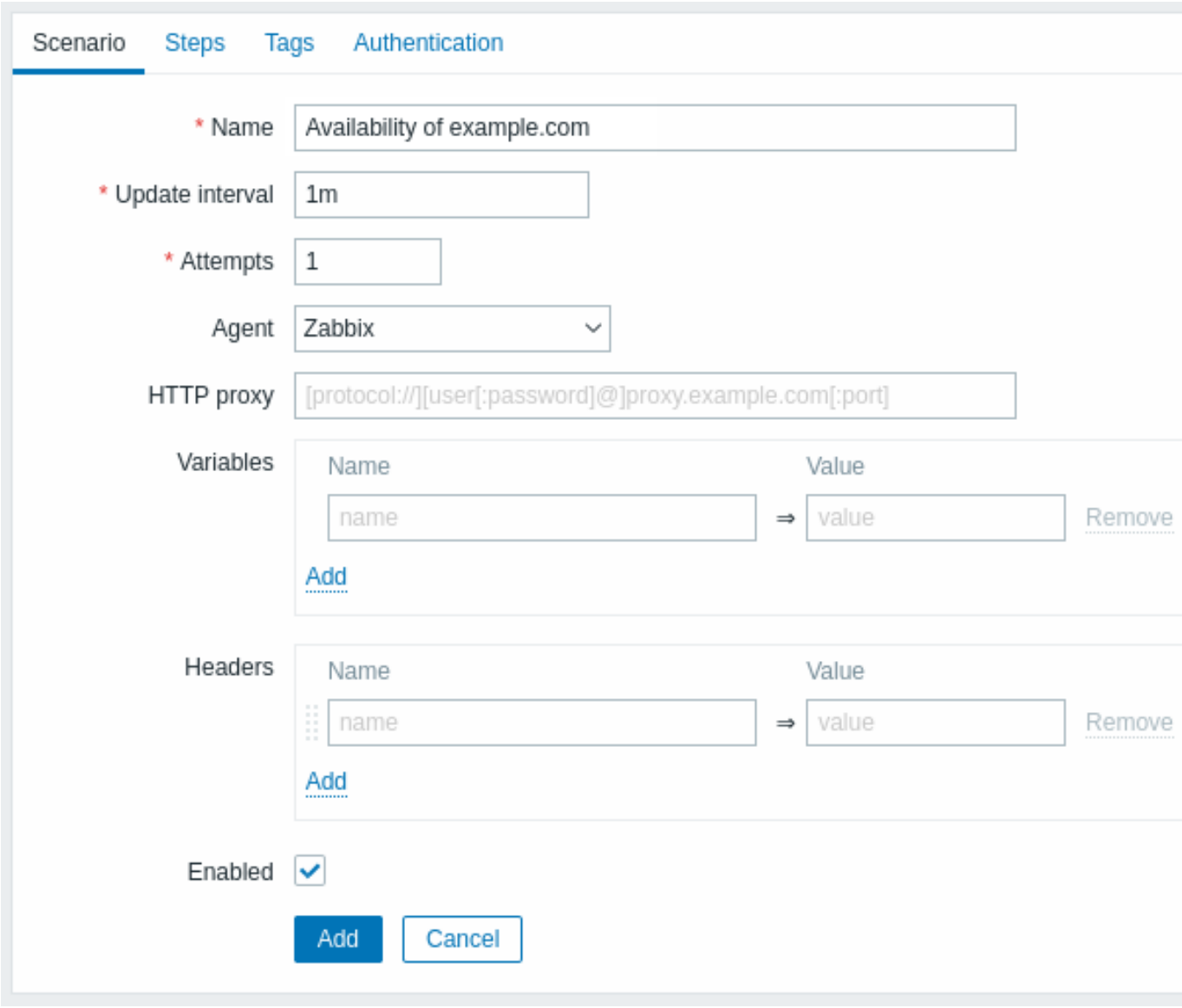
See also known issues for web monitoring using HTTPS protocol.

Configuring a web scenario To configure a web scenario:

- Go to: *Configuration* → *Hosts* (or *Templates*)
- Click on *Web* in the row of the host/template

- Click on *Create 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.

Parameter	Description
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.
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.

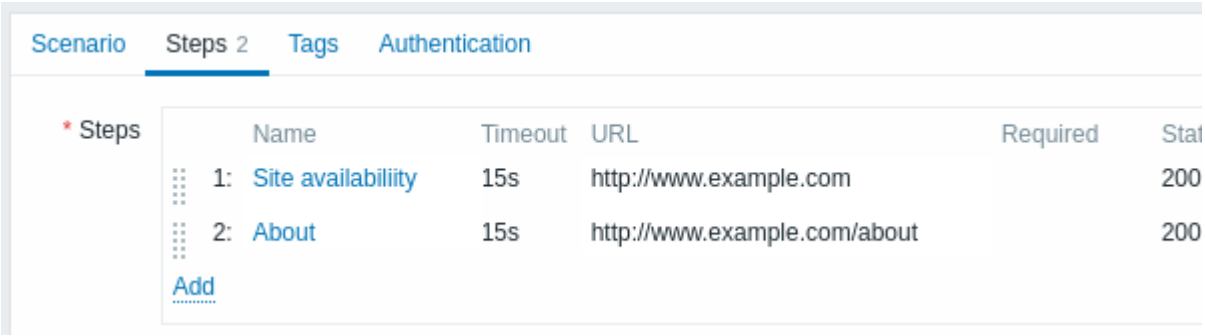
Note that when editing an existing scenario, two extra buttons are available in the form:

Clone	Create another scenario based on the properties of the existing one.

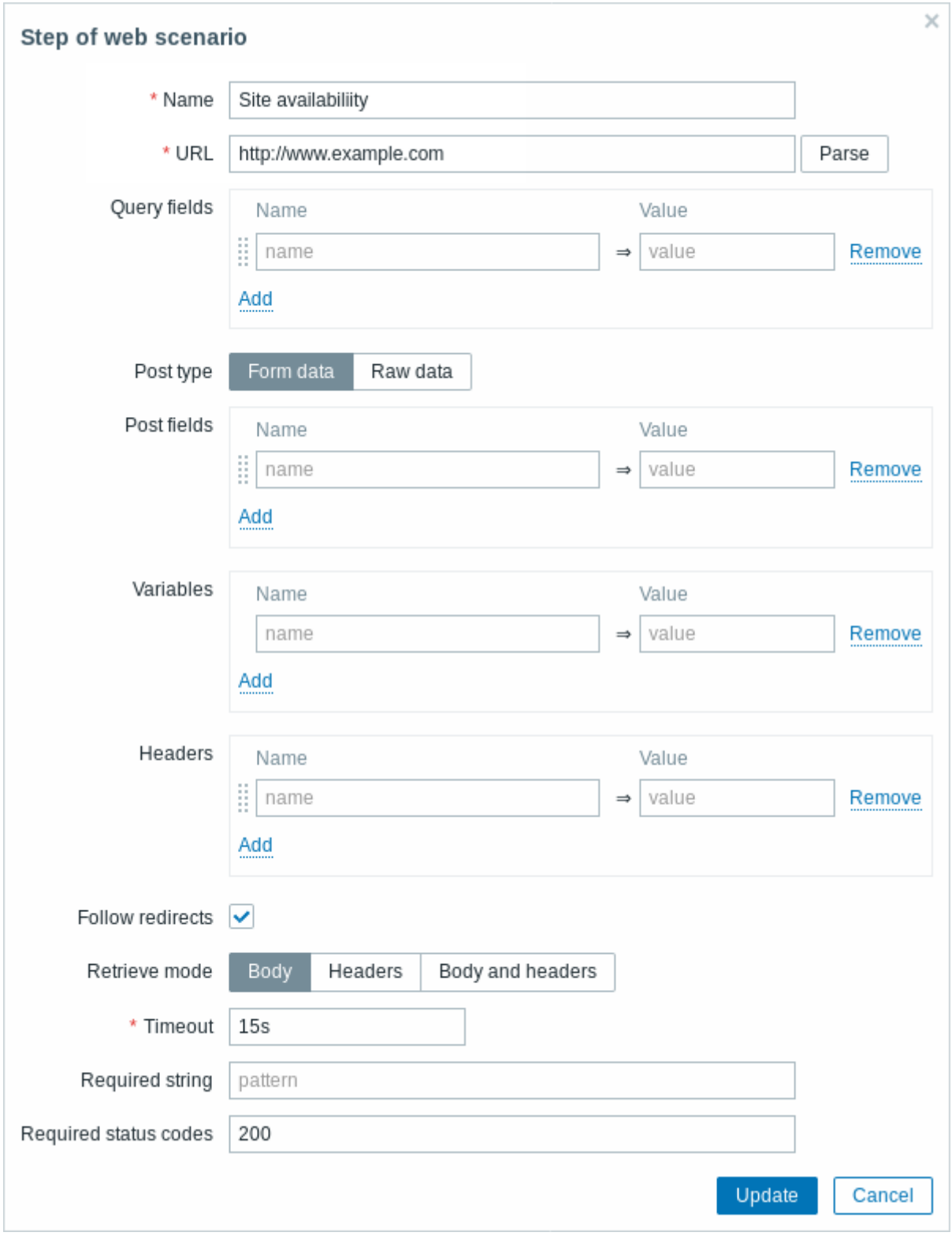
Clear history and trends	Delete history and trend data for the scenario. This will make the server perform the scenario immediately after deleting the data.

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.

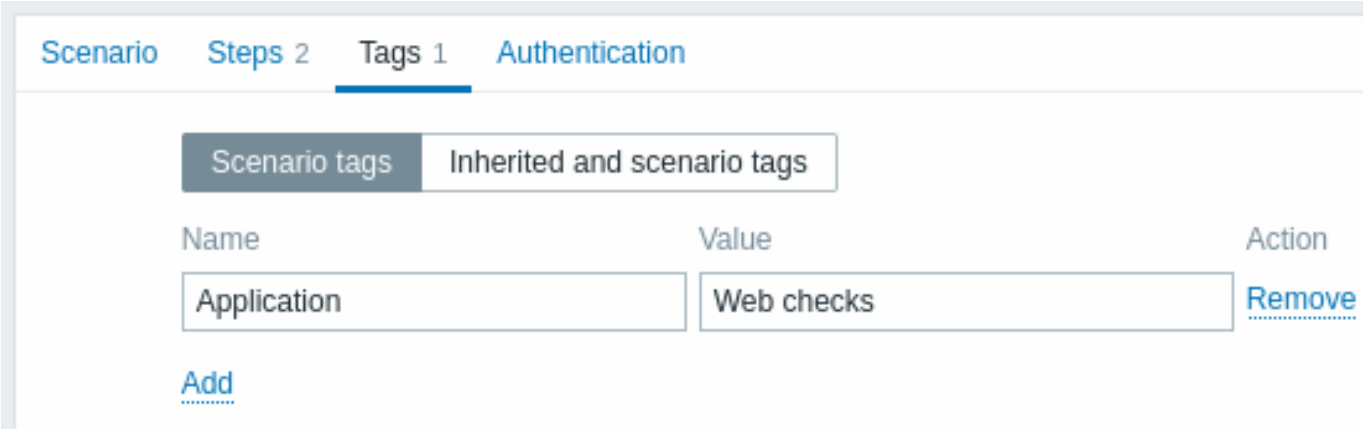
Parameter	Description
URL	URL to connect to and retrieve data. For example: https://www.example.com http://www.example.com/download Domain names can be specified in Unicode characters. They are automatically punycode-converted to ASCII when executing the web scenario step. The *Parse* button can be used 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.

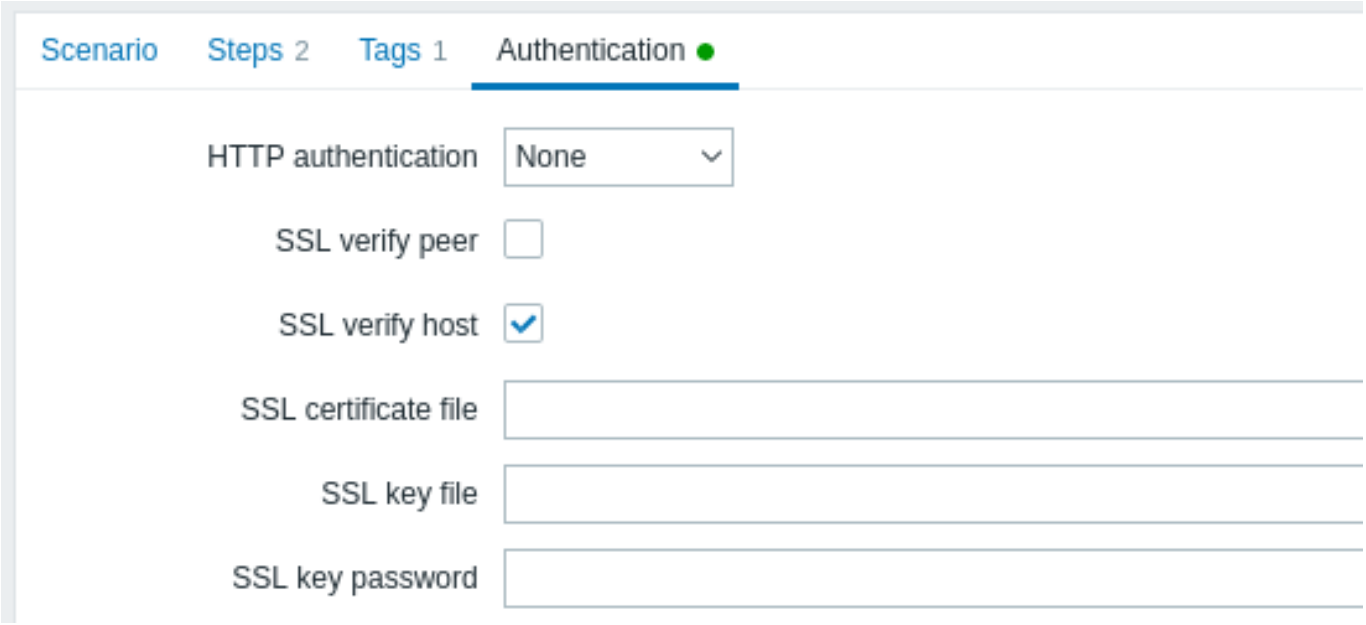
See also a real-life example of how web monitoring steps can be configured.

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 *Monitoring* → *Dashboard* by a 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.

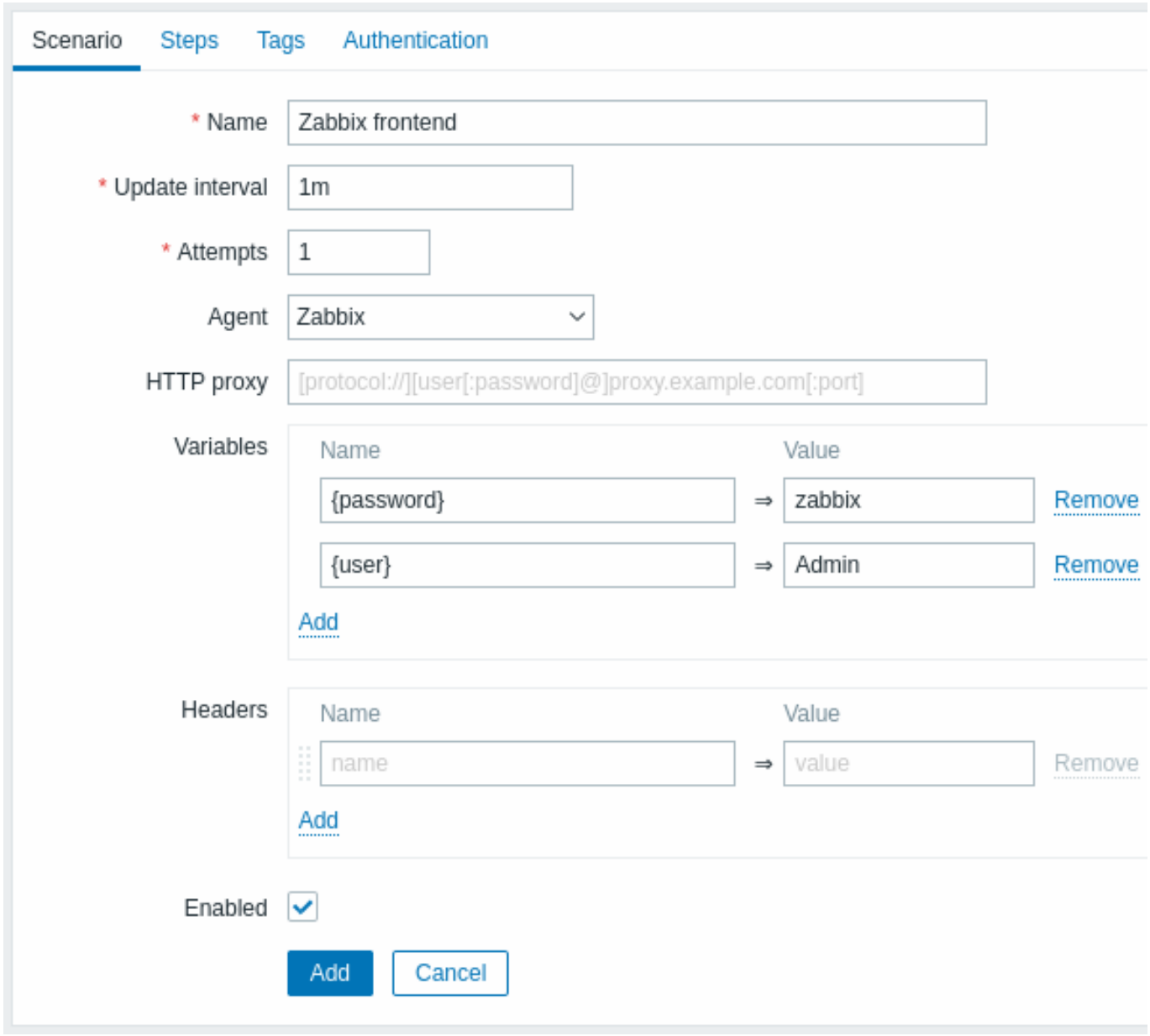
Scenario

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 *Configuration* → *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.

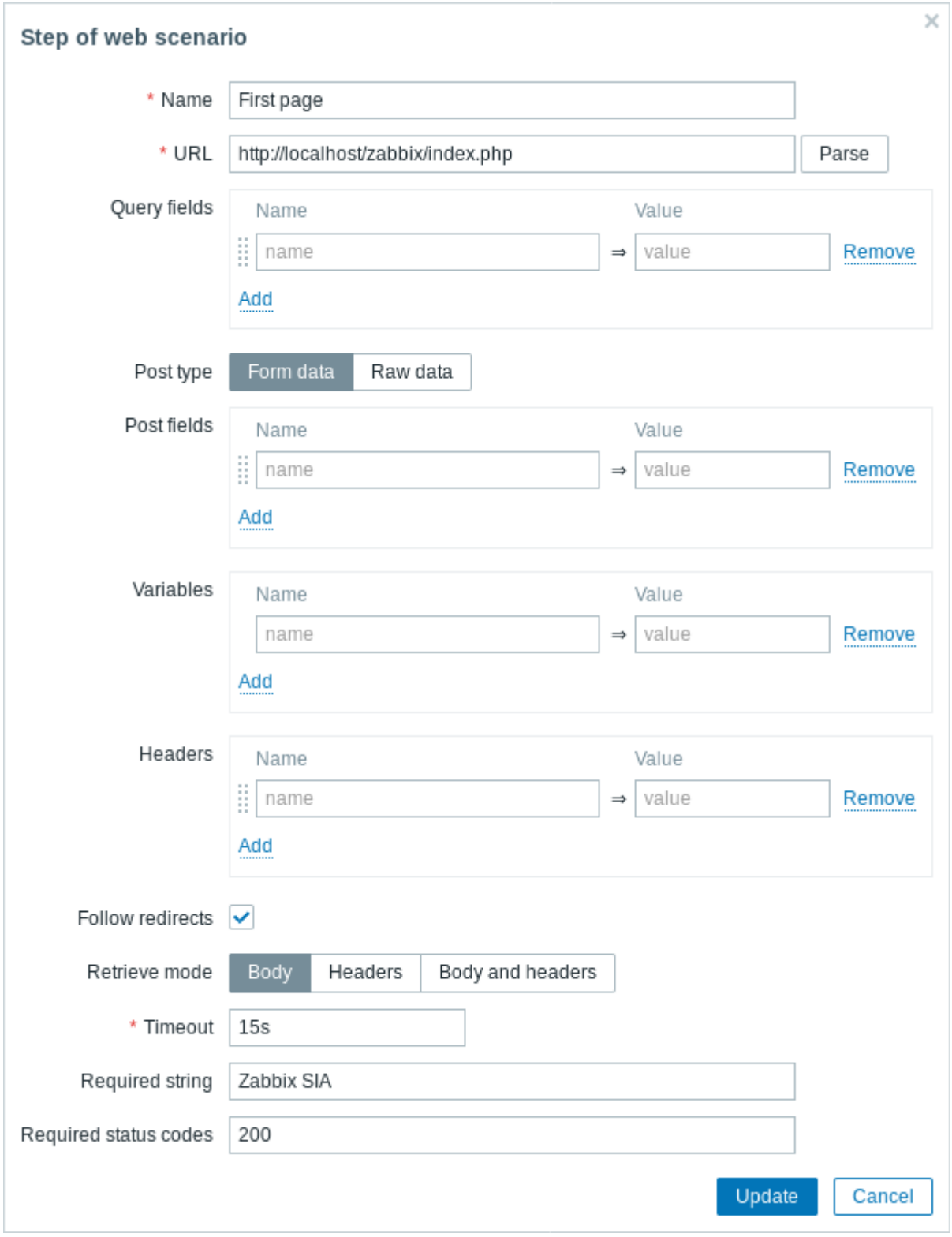
Step 2

Define steps for the scenario.

Click on *Add* button in the *Steps* tab to add individual steps.

Web scenario step 1

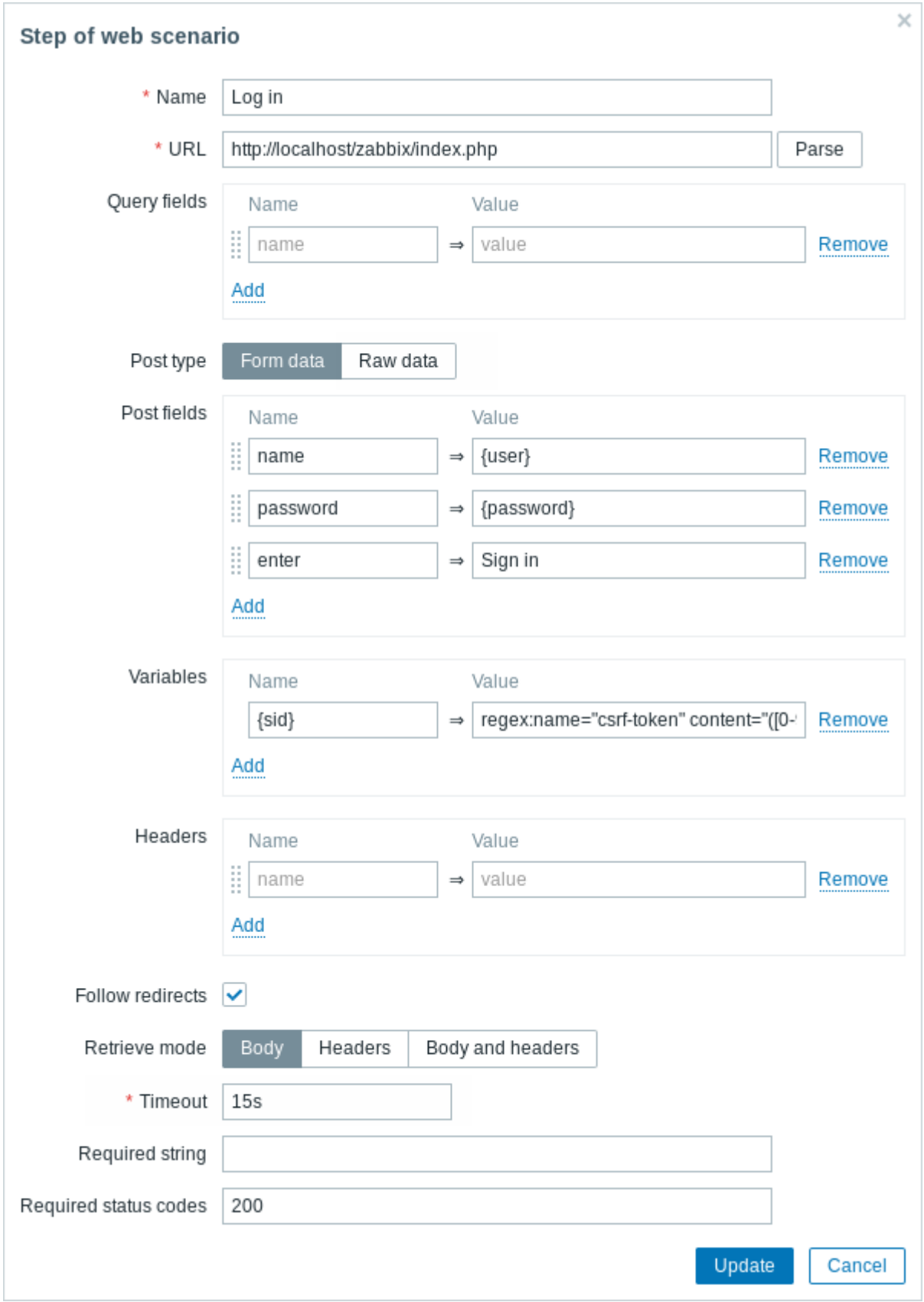
We start by checking that the first page responds correctly, returns with HTTP response code 200 and contains text ”Zabbix SIA”.



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}.

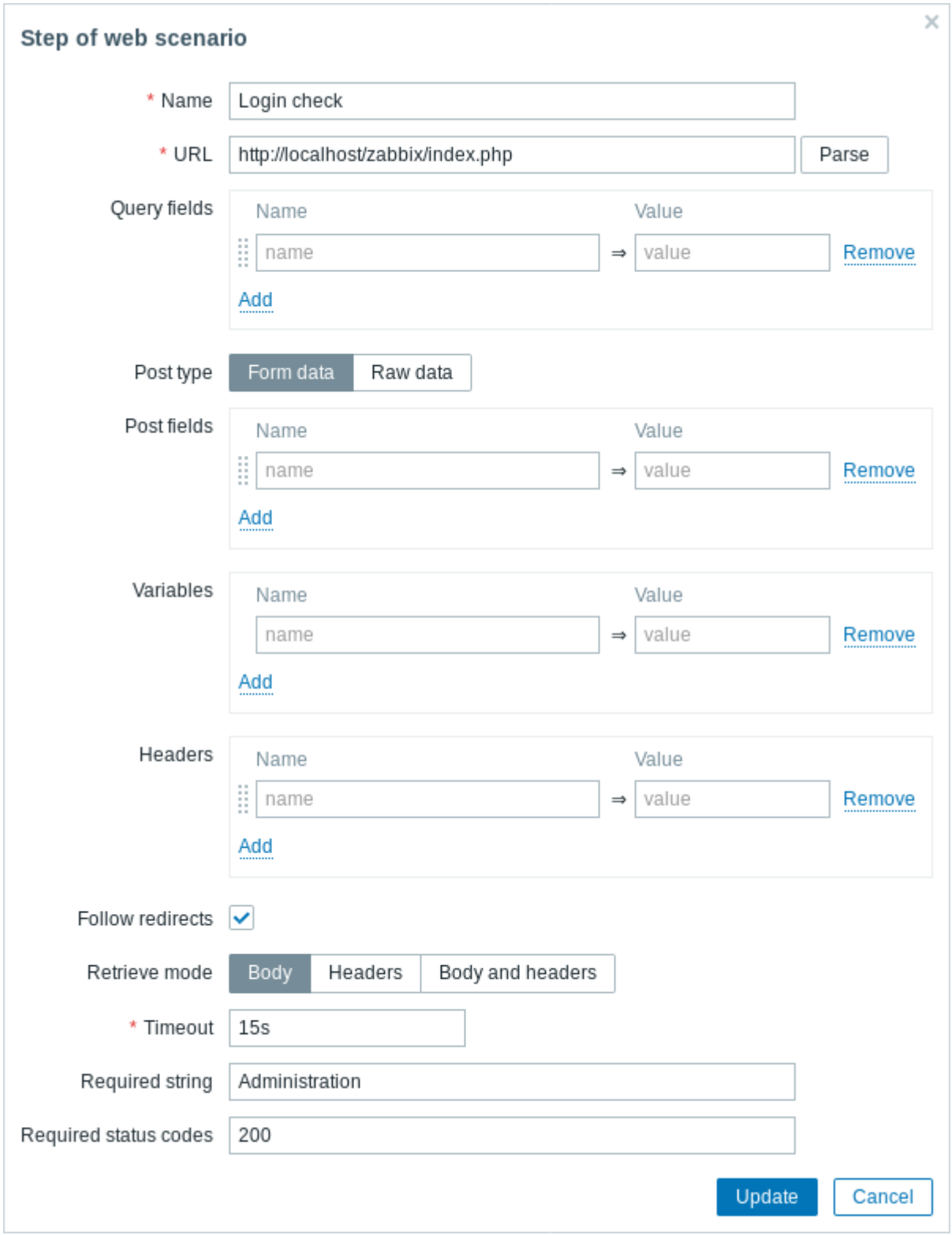


Attention:
Note that Zabbix frontend uses JavaScript redirect when logging in, thus first we must log in, and only in further steps we may check for logged-in features. Additionally, the login step must use full URL to **index.php** file.

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.

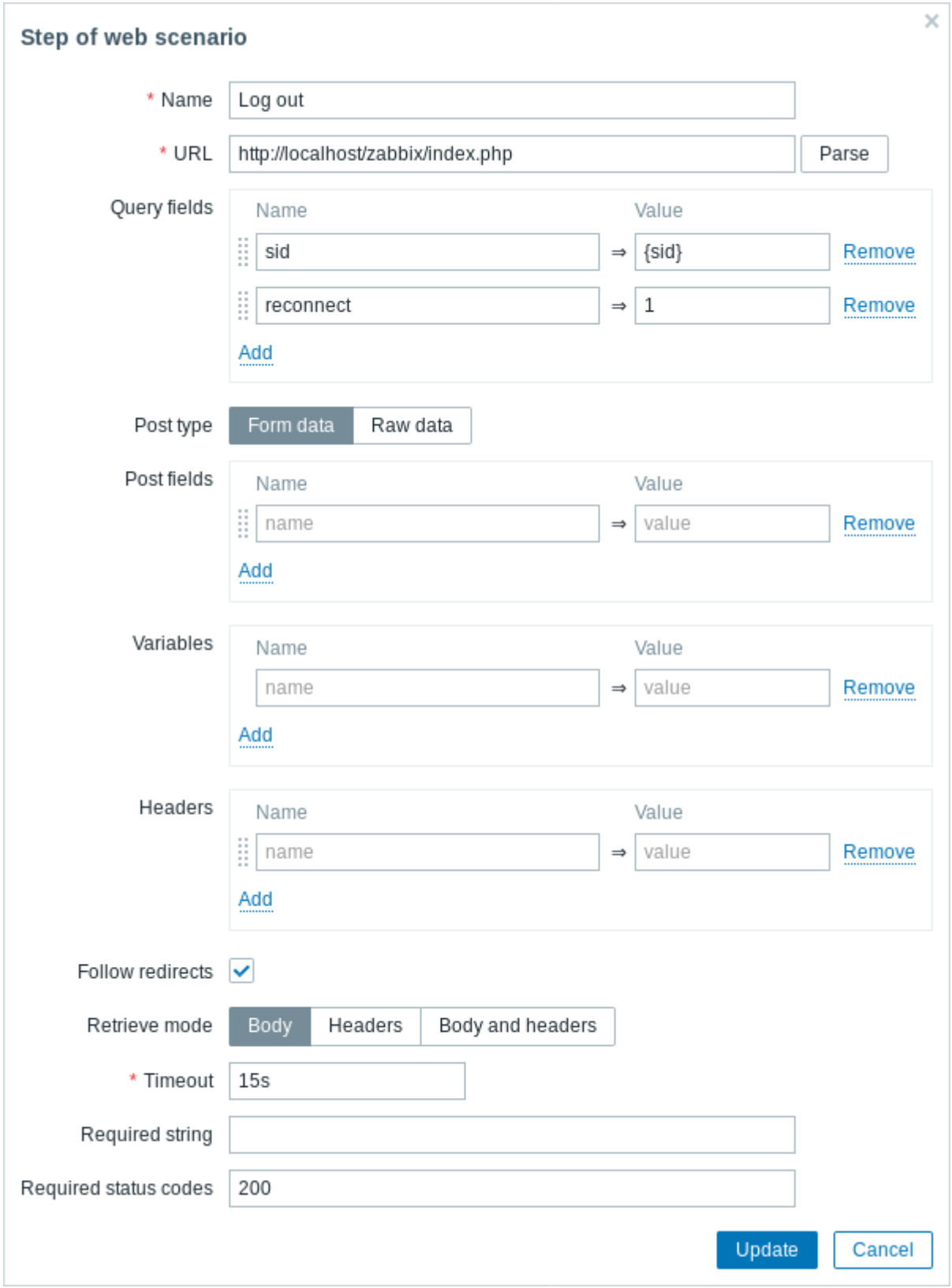
Web scenario step 3

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**.



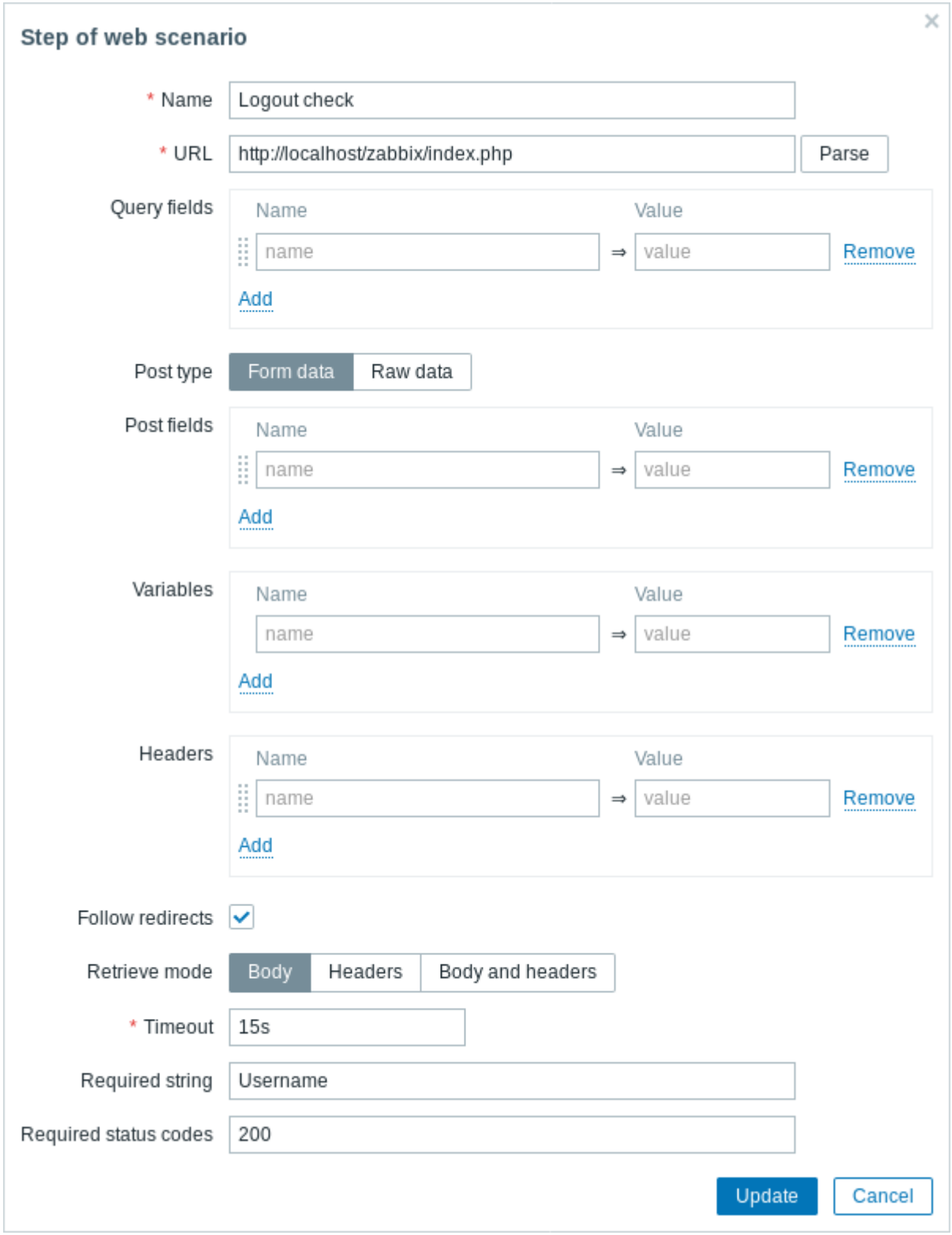
Web scenario step 4

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.



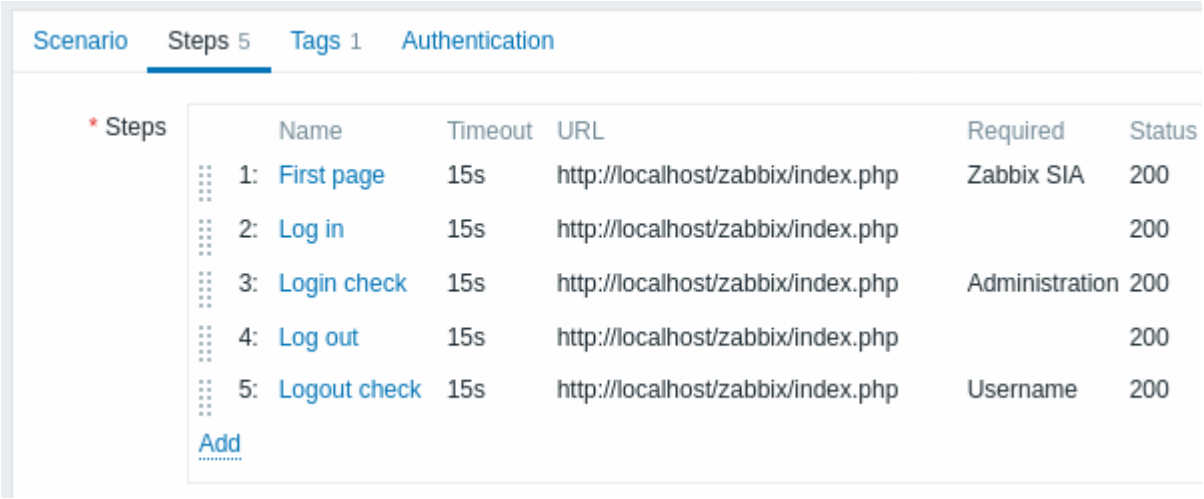
Web scenario step 5

We can also check that we have logged out by looking for the **Username** string.



Complete configuration of steps

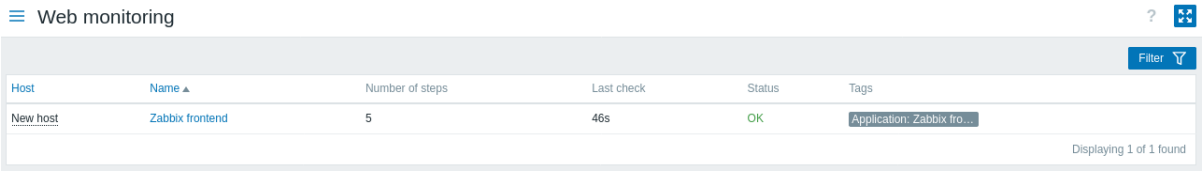
A complete configuration of web scenario steps should look like this:



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 Support of monitoring VMware environments is available in Zabbix starting with version 2.2.0.

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.

The default dataset in Zabbix offers several ready-to-use templates for monitoring VMware vCenter or ESX hypervisor.

The minimum required VMware vCenter or vSphere version is 5.1.

Data collection The virtual machine monitoring is performed in two steps:

1. *vmware collector* Zabbix processes gather virtual machine data. The processes obtain necessary information from VMware web services over the SOAP protocol, pre-process it and store in Zabbix server shared memory.
2. Poller processes retrieve the data using Zabbix simple check VMware keys.

Zabbix divides collected data into two types: VMware configuration data and VMware performance counter data. Both types are collected independently by *vmware collectors*. 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).

The following statistics are available based on the VMware performance counter information:

- datastore;
- disk device;
- CPU;
- power;
- network interface;
- custom performance counter items;
- etc.

See VMware item keys for the full list of items that obtain data from VMware performance counters.

Configuration If Zabbix server is compiled from sources, it should be compiled with the **--with-libxml2** and **--with-libcurl** compilation options to enable virtual machine monitoring. Zabbix packages are compiled with these options enabled.

The following configuration file options can be used to tune the Virtual machine monitoring:

- **StartVMwareCollectors** - the number of pre-forked *vmware collector* instances.
  This value depends on the number of VMware services you are going to monitor. For the most cases this should be:
  *servicenum < StartVMwareCollectors < (servicenum * 2)*
  where *servicenum* is the number of VMware services. E. g., if you have 1 VMware service to monitor set StartVMwareCollectors to 2, if you have 3 VMware services set it to 5. In most cases, this value should not be less than 2 and should not be 2 times greater than the number of monitored VMware services. Required number of collectors also depends on your VMware environment size and *VMwareFrequency* and *VMwarePerfFrequency* configuration parameters (see below).
- **VMwareCacheSize**
- **VMwareFrequency**
- **VMwarePerfFrequency**
- **VMwareTimeout**

For more details, see the configuration file pages for Zabbix server and proxy.

> **Attention:**
> To support datastore capacity metrics, Zabbix requires VMware configuration parameter *vpxd.stats.maxQueryMetrics* to be at least 64. See also the VMware knowledge base article.

Discovery Zabbix can use a low-level discovery rule (for example, *vmware.hv.discovery[{$VMWARE.URL}]*) to automatically discover VMware hypervisors and virtual machines. Host prototypes can be created with the low-level discovery rule. When virtual machines are discovered, these prototypes become real hosts. See low-level discovery for more information.

Ready-to-use templates The default dataset in Zabbix offers several ready-to-use templates for monitoring VMware vCenter or directly ESX hypervisor. These templates contain pre-configured LLD rules as well as a number of built-in checks for monitoring virtual installations.

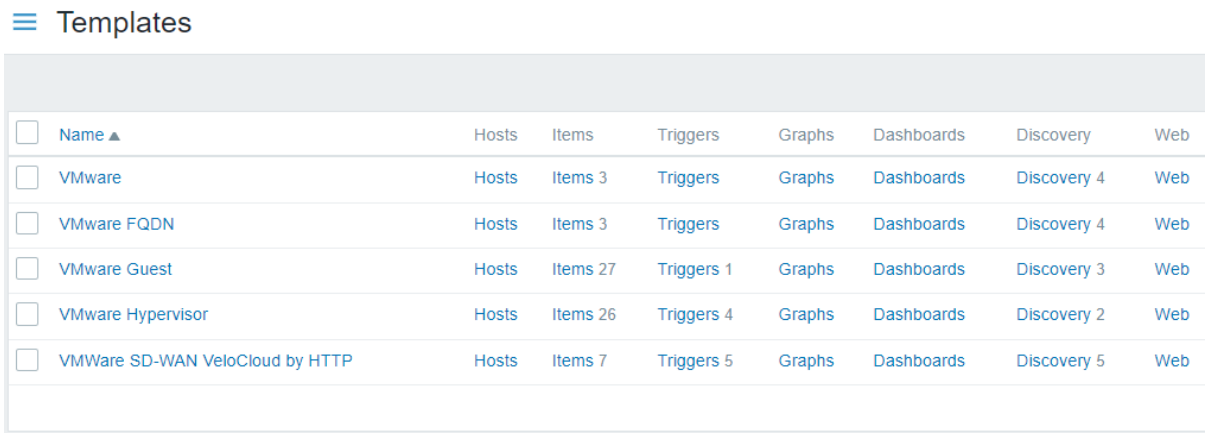
Templates for VMware vCenter and ESX hypervisor monitoring:

- *VMware* - uses UUID data for corresponding macros;
- *VMware FQDN* - uses FQDN data for corresponding macros.

> **Note:**
> In order for the *VMware FQDN* template to work correctly each monitored VM should have a unique OS name compliant with FQDN rules and VMware Tools must be installed on every machine. If these conditions are met, it is recommended to use *VMware FQDN* template. The creation of *VMware FQDN* template became possible after introducing the ability to create hosts with custom interfaces in Zabbix 5.2.
 A classic *VMware* template is still available and can be used if FQDN requirements cannot be met. Please keep in mind, that the *VMware* template has a known issue. Hosts for discovered virtual machines will be created with the names saved in the vCenter (for example, VM1, VM2, etc.). If Zabbix agent active is installed on these hosts later with autoregistration enabled, the autoregistration process will read host names as they have been registered upon launch (for example, vm1.example.com, vm2.example.com, etc.) and create new hosts since no name matches have been found. As a result, there will be two duplicate hosts for each machine with different names.

Templates used by discovery (normally, these templates should not be manually linked to a host):

- *VMware Hypervisor*;
- *VMware Guest*.



Host configuration To use VMware simple checks the host must have the following user macros defined:

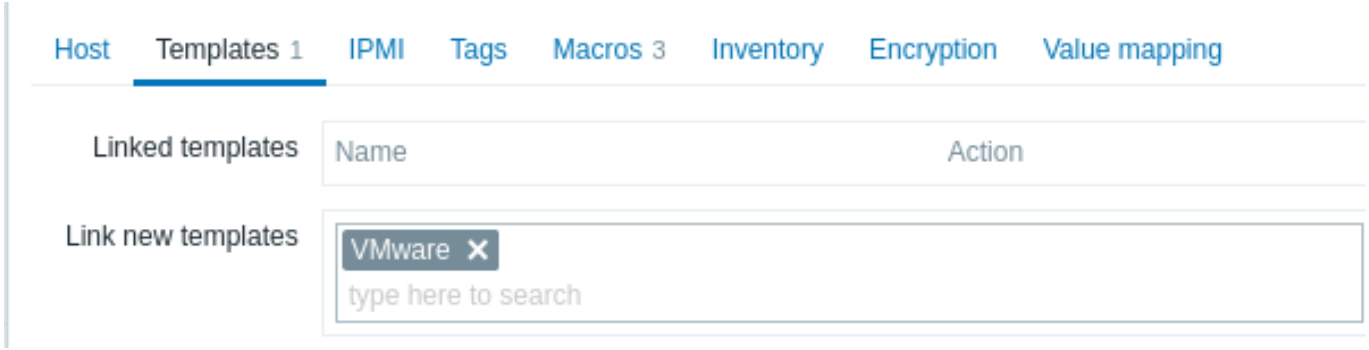
- **{$VMWARE.URL}** - VMware service (vCenter or ESX hypervisor) SDK URL (https://servername/sdk)
- **{$VMWARE.USERNAME}** - VMware service user name
- **{$VMWARE.PASSWORD}** - VMware service {$VMWARE.USERNAME} user password

Example The following example demonstrates how to quickly setup VMware monitoring on Zabbix:

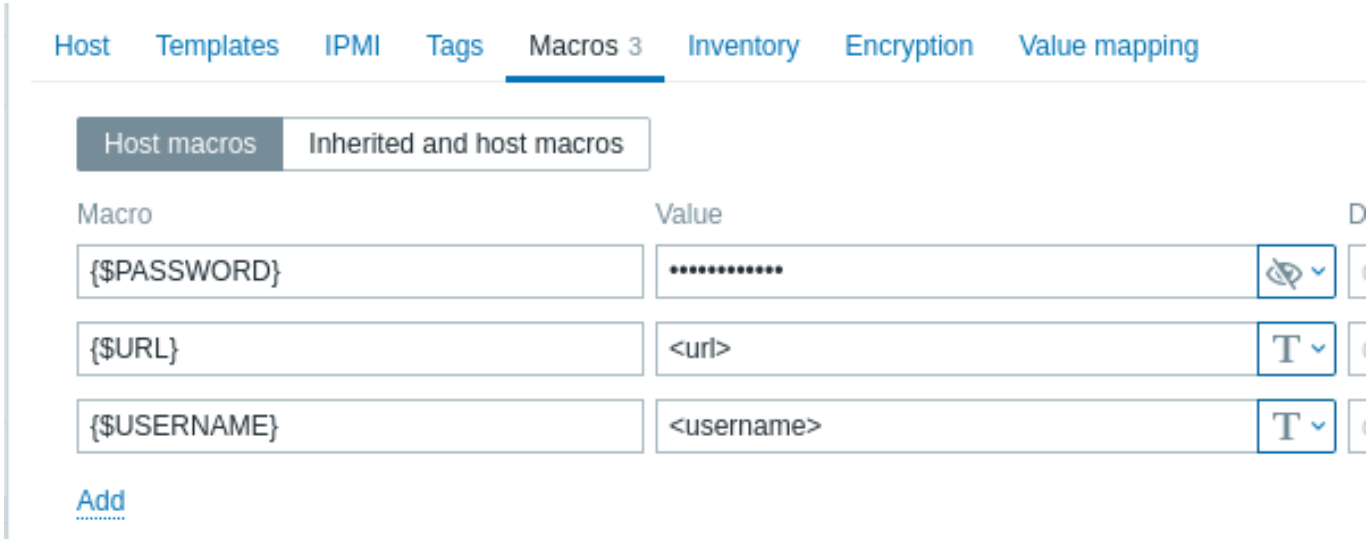
Note:
It is assumed that VMware installation is up and running.

1. Set the StartVMwareCollectors option in Zabbix server configuration file to 1 or more (the default value is 0).
2. Create a new host:

- *Host name* any name of your choice.
- *Templates* - type or select *VMware*



- *Host group* - any host group
- *Interface* - any dummy interface; this field is required, but the interface will not be used.
- *Macros* - set host macros required for VMware authentication: {$USERNAME}, {$PASSWORD}, {$URL}.



- click on the **Add** button to save the host.

Extended logging The data gathered by VMware collector can be logged for detailed debugging using debug level 5. This level can be set in server and proxy configuration files or using a runtime control option (`-R log_level_increase="vmware collector,N"`, where N is a process number).

For example, if debug level 4 has already been set, you can increase it to level 5 by running:

```
Increase log level of all vmware collectors:
shell> zabbix_server -R log_level_increase="vmware collector"

Increase log level of the second vmware collector:
shell> zabbix_server -R log_level_increase="vmware collector,2"
```

If extended logging of VMware collector data is not required it can be stopped using the `-R log_level_decrease` option.

Troubleshooting

- In case of unavailable metrics, please make sure 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. See ZBX-12094 for additional details.
- 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*. See the VMware knowledge base article for details.

VMware monitoring item keys

This page provides details on the simple checks that can be used to monitor VMware environments. The metrics are grouped by the monitoring target.

General service metrics

Key			
Description	Return value	Parameters	Comments
vmware.eventlog[<url>,<mode>]			
VMware event log.	Log	**url** - VMware service URL **mode** - *all* (default), *skip* - skip processing of older data	There must be only one vmware.eventlog[] item key per URL. See also: example of filtering VMware event log records.
vmware.fullname[<url>]			
VMware service full name.	String	**url** - VMware service URL	
vmware.version[<url>]			
VMware service version.	String	**url** - VMware service URL	

Cluster

Key			
Description	Return value	Parameters	Comments
vmware.cl.perfcounter[<url>,<id>,<path>,<instance>]			
VMware cluster performance counter metrics.	Integer	**url** - VMware service URL **id** - VMware cluster ID **path** - performance counter path[1] **instance** - performance counter instance	`id` can be received from vmware.cluster.discovery[] as {#CLUSTER.ID}
vmware.cluster.alarms.get[<url>,<id>]			

Key			
VMware cluster alarms data.	JSON object	**url** - VMware service URL **id** - VMware cluster ID	
vmware.cluster.discovery[<url>]			
Discovery of VMware clusters.	JSON object	**url** - VMware service URL	
vmware.cluster.property[<url>,<id>,<prop>]			
VMware cluster property.	String	**url** - VMware service URL **id** - VMware cluster ID **prop** - property path	
vmware.cluster.status[<url>,<name>]			
VMware cluster status.	Integer: *0* - gray; *1* - green; *2* - yellow; *3* - red	**url** - VMware service URL **name** - VMware cluster name	
vmware.cluster.tags.get[<url>,<id>]			
VMware cluster tags array.	JSON object	**url** - VMware service URL **id** - VMware cluster ID	This item works with vSphere 6.5 and newer (since Zabbix 6.2.4); with vSphere 7.0 Update 2 and newer (before Zabbix 6.2.4).

Datastore

Key			
Description	Return value	Parameters	Comments
vmware.datastore.alarms.get[<url>,<uuid>]			
VMware datastore alarms data.	JSON object	**url** - VMware service URL **uuid** - VMware datastore name	
vmware.datastore.perfcounter[<url>,<uuid>,<path>,<instance>]			
VMware datastore performance counter value.	Integer [2]	**url** - VMware service URL **uuid** - VMware datastore unique ID **path** performance counter path [1] **instance** - performance counter instance. Use empty instance for aggregate values (default)	
vmware.datastore.property[<url>,<uuid>,<prop>]			
VMware datastore property.	String	**url** - VMware service URL **uuid** - VMware datastore name **prop** - property path	
vmware.datastore.tags.get[<url>,<uuid>]			
VMware datastore tags array.	JSON object	**url** - VMware service URL **uuid** - VMware datastore name	This item works with vSphere 6.5 and newer (since Zabbix 6.2.4); with vSphere 7.0 Update 2 and newer (before Zabbix 6.2.4).
vmware.datastore.discovery[<url>]			
Discovery of VMware datastores.	JSON object	**url** - VMware service URL	
vmware.datastore.hv.list[<url>,<datastore>]			

List of datastore hypervisors.	String	**url** - VMware service URL
		datastore - datastore name
			Output example:
			`esx7-01-host.zabbix.sandbox`
			`esx7-02-host.zabbix.sandbox`

vmware.datastore.read[<url>,<datastore>,<mode>]			

Key			
Description	Return value	Parameters	Comments
Amount of time for a read operation from the datastore (millisec-onds).	Integer [2]	**url** - VMware service URL **datastore** - datastore name **mode** - latency (average value, default), maxlatency (maximum value)	
vmware.datastore.size[<url>,<datastore>,<mode>]			
VMware datastore space in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**url** - VMware service URL **datastore** - datastore name **mode** - possible values: total (default), free, pfree (free, percentage), uncommitted	
vmware.datastore.write[<url>,<datastore>,<mode>]			
Amount of time for a write operation to the datastore (millisec-onds).	Integer [2]	**url** - VMware service URL **datastore** - datastore name **mode** - latency (average value, default), maxlatency (maximum value)	

Datacenter

Key			
Description	Return value	Parameters	Comments
vmware.dc.alarms.get[<url>,<id>]			
VMware datacenter alarms data.	JSON object	**url** - VMware service URL **id** - VMware datacenter ID	
vmware.dc.discovery[<url>]			
Discovery of VMware datacenters.	JSON object	**url** - VMware service URL	
vmware.dc.tags.get[<url>,<id>]			
VMware datacenter tags array.	JSON object	**url** - VMware service URL **id** - VMware datacenter ID	This item works with vSphere 6.5 and newer (since Zabbix 6.2.4); with vSphere 7.0 Update 2 and newer (before Zabbix 6.2.4).

vSphere Distributed Switch

Key			
Description	Return value	Parameters	Comments
vmware.dvswitch.discovery[<url>]			
Discovery of VMware vSphere Distributed Switches.	JSON object	**url** - VMware service URL	
vmware.dvswitch.fetchports.get[<url>,<uuid>,<filter>,<mode>]			

Key			
VMware vSphere Distributed Switch ports data.	JSON object	**url** - VMware service URL **uuid** - ID of the switch **filter** - single string with comma-separated criteria for selecting ports **mode** - state (all XML without ”config” XML nodes, default), full	Parameter **filter** supports the criteria available in the VMware data object DistributedVirtualSwitchPortCriteria Example: vmware.dvswitch.fetchports.get[{$VMWARE.URL},{$VM 18,inside:true,nsxPort:true,uplinkPort:false”,state]

Hypervisor

Key			
Description	Return value	Parameters	Comments
vmware.hv.alarms.get[<url>,<uuid>]			
VMware hypervisor alarms data.	JSON object	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.cluster.name[<url>,<uuid>]			
VMware hypervisor cluster name.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.connectionstate[<url>,<uuid>]			
VMware hypervisor connection state.	String: *connected* *disconnected* *notResponding*	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.cpu.usage[<url>,<uuid>]			
VMware hypervisor processor usage (Hz).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.cpu.usage.perf[<url>,<uuid>]			
VMware hypervisor processor usage as a percentage during the interval.	Float	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.cpu.utilization[<url>,<uuid>]			
VMware hypervisor processor usage as a percentage during the interval, depends on power management or HT.	Float	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.datacenter.name[<url>,<uuid>]			
VMware hypervisor datacenter name.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.datastore.discovery[<url>,<uuid>]			
Discovery of VMware hypervisor datastores.	JSON object	**url** - VMware service URL **uuid** - VMware hypervisor host name	

Key			
vmware.hv.datastore.list[<url>,<uuid>]			
List of VMware hypervisor datastores.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	Output example: SSD-RAID1-VAULT1 SSD-RAID1-VAULT2 SSD-RAID10
vmware.hv.datastore.multipath[<url>,<uuid>,<datastore>,<partitionid>]			
Number of available datastore paths.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name **datastore** - datastore name **partitionid** - internal ID of physical device from vmware.hv.datastore.discovery	
vmware.hv.datastore.read[<url>,<uuid>,<datastore>,<mode>]			
Average amount of time for a read operation from the datastore (millisec-onds).	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor host name **datastore** - datastore name **mode** - latency (default)	
vmware.hv.datastore.size[<url>,<uuid>,<datastore>,<mode>]			
VMware datastore space in bytes or in percentage from total.	Integer - for bytes Float - for percentage	**url** - VMware service URL **uuid** - VMware hypervisor host name **datastore** - datastore name **mode** - possible values: total (default), free, pfree (free, percentage), uncommitted	
vmware.hv.datastore.write[<url>,<uuid>,<datastore>,<mode>]			
Average amount of time for a write operation to the datastore (millisec-onds).	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor host name **datastore** - datastore name **mode** - latency (default)	
vmware.hv.discovery[<url>]			
Discovery of VMware hypervisors.	JSON object	**url** - VMware service URL	
vmware.hv.diskinfo.get[<url>,<uuid>]			
VMware hypervisor disk data.	JSON object	**url** - VMware service URL **uuid** - VMware hypervisor host unique ID	
vmware.hv.fullname[<url>,<uuid>]			
VMware hypervisor name.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.cpu.freq[<url>,<uuid>]			
VMware hypervisor processor frequency (Hz).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.cpu.model[<url>,<uuid>]			
VMware hypervisor processor model.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	

Key			
vmware.hv.hw.cpu.num[<url>,<uuid>] Number of processor cores on VMware hypervisor.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.cpu.threads[<url>,<uuid>] Number of processor threads on VMware hypervisor.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.memory[<url>,<uuid>] VMware hypervisor total memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.model[<url>,<uuid>] VMware hypervisor model.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.sensors.get[<url>,<uuid>] VMware hypervisor hardware sensors value.	JSON object	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.serialnumber[<url>,<uuid>] VMware hypervisor serial number.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	This item works with vSphere API 6.7 and newer.
vmware.hv.hw.uuid[<url>,<uuid>] VMware hypervisor BIOS UUID.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.hw.vendor[<url>,<uuid>] VMware hypervisor vendor name.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.maintenance[<url>,<uuid>] VMware hypervisor maintenance status.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	Returns '0' - not in maintenance or '1' - in maintenance
vmware.hv.memory.size.ballooned[<url>,<uuid>] VMware hypervisor ballooned memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.memory.used[<url>,<uuid>] VMware hypervisor used memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.net.if.discovery[<url>,<uuid>] Discovery of VMware hypervisor network interfaces.	JSON object	**url** - VMware service URL **uuid** - VMware hypervisor host name	

Key			
vmware.hv.network.in[<url>,<uuid>,<mode>]			
VMware hypervisor network input statistics (bytes per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor host name **mode** - bps (default), packets, dropped, errors, broadcast	
vmware.hv.network.linkspeed[<url>,<uuid>,<ifname>]			
VMware hypervisor network interface speed.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name **ifname** - interface name	Returns 0, if network interface is down, otherwise speed value of the interface.
vmware.hv.network.out[<url>,<uuid>,<mode>]			
VMware hypervisor network output statistics (bytes per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor host name **mode** - bps (default), packets, dropped, errors, broadcast	
vmware.hv.perfcounter[<url>,<uuid>,<path>,<instance>]			
VMware hypervisor performance counter value.	Integer [2]	**url** - VMware service URL **uuid** - VMware hypervisor host name **path** - performance counter path [1] **instance** - performance counter instance. Use empty instance for aggregate values (default)	
vmware.hv.property[<url>,<uuid>,<prop>]			
VMware hypervisor property.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name **prop** - property path	
vmware.hv.power[<url>,<uuid>,<max>]			
VMware hypervisor power usage (W).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name **max** - maximum allowed power usage	
vmware.hv.sensor.health.state[<url>,<uuid>]			
VMware hypervisor health state rollup sensor.	Integer: *0* - gray; *1* - green; *2* - yellow; *3* - red	**url** - VMware service URL **uuid** - VMware hypervisor host name	The item might not work in the VMware vSphere 6.5 and newer, because VMware has deprecated the *VMware Rollup Health State* sensor.
vmware.hv.sensors.get[<url>,<uuid>]			
VMware hypervisor HW vendor state sensors.	JSON object	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.status[<url>,<uuid>]			
VMware hypervisor status.	Integer: *0* - gray; *1* - green; *2* - yellow; *3* - red	**url** - VMware service URL **uuid** - VMware hypervisor host name	Uses the host system overall status property.
vmware.hv.tags.get[<url>,<uuid>]			
VMware hypervisor tags array.	JSON object	**url** - VMware service URL **uuid** - VMware hypervisor host name	This item works with vSphere 6.5 and newer (since Zabbix 6.2.4); with vSphere 7.0 Update 2 and newer (before Zabbix 6.2.4).
vmware.hv.uptime[<url>,<uuid>]			

Key			
VMware hypervisor uptime (seconds).	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.version[<url>,<uuid>]			
VMware hypervisor version.	String	**url** - VMware service URL **uuid** - VMware hypervisor host name	
vmware.hv.vm.num[<url>,<uuid>]			
Number of virtual machines on VMware hypervisor.	Integer	**url** - VMware service URL **uuid** - VMware hypervisor host name	

Resource pool

Key			
Description	Return value	Parameters	Comments
vmware.rp.cpu.usage[<url>,<rpid>]			
CPU usage in hertz during the interval on VMware Resource Pool.	Integer	**url** - VMware service URL **rpid** - VMware resource pool ID	
vmware.rp.memory[<url>,<rpid>,<mode>]			
Memory metrics of VMware resource pool.	Integer	**url** - VMware service URL **rpid** - VMware resource pool ID **mode** - possible values: *consumed* (default) - amount of host physical memory consumed for backing up guest physical memory pages *ballooned* - amount of guest physical memory reclaimed from the virtual machine by the balloon driver in the guest *overhead* - host physical memory consumed by ESXi data structures for running the virtual machines	

Virtual center

Key			
Description	Return value	Parameters	Comments
vmware.alarms.get[<url>]			
VMware virtual center alarms data.	JSON object	**url** - VMware service URL	

Virtual machine

Key			
Description	Return value	Parameters	Comments
vmware.vm.alarms.get[<url>,<uuid>]			
VMware virtual machine alarms data.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine name	
vmware.vm.attribute[<url>,<uuid>,<name>]			

Key		
VMware virtual machine custom attribute value.	String	**url** - VMware service URL **uuid** - VMware virtual machine host name **name** - custom attribute name
vmware.vm.cluster.name[<url>,<uuid>]		
VMware virtual machine name.	String	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.consolidationneeded[<url>,<uuid>]		
VMware virtual machine disk requires consolidation.	String: *true* - consolidation is needed; *false* - consolidation is not needed.	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.cpu.latency[<url>,<uuid>]		
Percentage of time the virtual machine is unable to run because it is contending for access to the physical CPU(s).	Float	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.cpu.num[<url>,<uuid>]		
Number of processors on VMware virtual machine.	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.cpu.readiness[<url>,<uuid>,<instance>]		
Percentage of time that the virtual machine was ready, but could not get scheduled to run on the physical CPU.	Float	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - CPU instance
vmware.vm.cpu.ready[<url>,<uuid>]		

Key		
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 (%).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.cpu.swapwait[<url>,<uuid>,<instance>]		
Percentage of CPU time spent waiting for swap-in.	Float	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - CPU instance
vmware.vm.cpu.usage[<url>,<uuid>]		
VMware virtual machine processor usage (Hz).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.cpu.usage.perf[<url>,<uuid>]		
VMware virtual machine processor usage as a percentage during the interval.	Float	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.datacenter.name[<url>,<uuid>]		
VMware virtual machine datacenter name.	String	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.discovery[<url>]		
Discovery of VMware virtual machines.	JSON object	**url** - VMware service URL
vmware.vm.guest.memory.size.swapped[<url>,<uuid>]		
Amount of guest physical memory that is swapped out to the swap space (KB).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.guest.osuptime[<url>,<uuid>]		

Key		
Total time elapsed since the last operating system boot-up (in seconds).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.hv.name[<url>,<uuid>]		
VMware virtual machine hypervisor name.	String	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size[<url>,<uuid>]		
VMware virtual machine total memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.ballooned[<url>,<uuid>]		
VMware virtual machine ballooned memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.compressed[<url>,<uuid>]		
VMware virtual machine compressed memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.consumed[<url>,<uuid>]		
Amount of host physical memory consumed for backing up guest physical memory pages (KB).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.private[<url>,<uuid>]		
VMware virtual machine private memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.shared[<url>,<uuid>]		
VMware virtual machine shared memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.swapped[<url>,<uuid>]		

Key		
VMware virtual machine swapped memory size (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.usage.guest[<url>,<uuid>]		
VMware virtual machine guest memory usage (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.size.usage.host[<url>,<uuid>]		
VMware virtual machine host memory usage (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.memory.usage[<url>,<uuid>]		
Percentage of host physical memory that has been consumed.	Float	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.net.if.discovery[<url>,<uuid>]		
Discovery of VMware virtual machine network interfaces.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.net.if.in[<url>,<uuid>,<instance>,<mode>]		
VMware virtual machine network interface input statistics (bytes/packets per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - network interface instance **mode** - bps (default)/pps - bytes/packets per second
vmware.vm.net.if.out[<url>,<uuid>,<instance>,<mode>]		
VMware virtual machine network interface output statistics (bytes/packets per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - network interface instance **mode** - bps (default)/pps - bytes/packets per second
vmware.vm.net.if.usage[<url>,<uuid>,<instance>]		

Key		
VMware virtual machine network utilization (combined transmit-rates and receive-rates) during the interval (KBps).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - network interface instance
vmware.vm.perfcounter[<url>,<uuid>,<path>,<instance>]		
VMware virtual machine performance counter value.	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine host name **path** - performance counter path [1] **instance** - performance counter instance. Use empty instance for aggregate values (default)
vmware.vm.powerstate[<url>,<uuid>]		
VMware virtual machine power state.	Integer: *0* - poweredOff; *1* - poweredOn; *2* - suspended	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.property[<url>,<uuid>,<prop>]		
VMware virtual machine property.	String	**url** - VMware service URL **uuid** - VMware virtual machine host name **prop** - property path
vmware.vm.snapshot.get[<url>,<uuid>]		
VMware virtual machine snapshot state.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.state[<url>,<uuid>]		
VMware virtual machine state.	String: *notRunning* *resetting* *running* *shuttingDown* *standby* *unknown*	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.storage.committed[<url>,<uuid>]		
VMware virtual machine committed storage space (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name
vmware.vm.storage.readoio[<url>,<uuid>,<instance>]		
Average number of outstanding read requests to the virtual disk during the collection interval.	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - disk device instance (mandatory)

Key			
vmware.vm.storage.totalreadlatency[<url>,<uuid>,<instance>]			
The average time a read from the virtual disk takes (milliseconds).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - disk device instance (mandatory)	
vmware.vm.storage.totalwritelatency[<url>,<uuid>,<instance>]			
The average time a write to the virtual disk takes (milliseconds).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - disk device instance (mandatory)	
vmware.vm.storage.uncommitted[<url>,<uuid>]			
VMware virtual machine uncommitted storage space (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name	
vmware.vm.storage.unshared[<url>,<uuid>]			
VMware virtual machine unshared storage space (bytes).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name	
vmware.vm.storage.writeoio[<url>,<uuid>,<instance>]			
Average number of outstanding write requests to the virtual disk during the collection interval.	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - disk device instance (mandatory)	
vmware.vm.tags.get[<url>,<uuid>]			
VMware virtual machine tags array.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine host name	This item works with vSphere 6.5 and newer (since Zabbix 6.2.4); with vSphere 7.0 Update 2 and newer (before Zabbix 6.2.4).
vmware.vm.tools[<url>,<uuid>,<mode>]			
VMware virtual machine guest tools state.	String: *guestToolsExecutingScripts* - VMware Tools is starting; *guestToolsNotRunning* - VMware Tools is not running; *guestToolsRunning* - VMware Tools is running.	**url** - VMware service URL **uuid** - VMware virtual machine host name **mode** - version, status	
vmware.vm.uptime[<url>,<uuid>]			
VMware virtual machine uptime (seconds).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name	

Key			
vmware.vm.vfs.dev.discovery[<url>,<uuid>]			
Discovery of VMware virtual machine disk devices.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine host name	
vmware.vm.vfs.dev.read[<url>,<uuid>,<instance>,<mode>]			
VMware virtual machine disk device read statistics (bytes/operations per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - disk device instance **mode** - bps (default)/ops - bytes/operations per second	
vmware.vm.vfs.dev.write[<url>,<uuid>,<instance>,<mode>]			
VMware virtual machine disk device write statistics (bytes/operations per second).	Integer [2]	**url** - VMware service URL **uuid** - VMware virtual machine host name **instance** - disk device instance **mode** - bps (default)/ops - bytes/operations per second	
vmware.vm.vfs.fs.discovery[<url>,<uuid>]			
Discovery of VMware virtual machine file systems.	JSON object	**url** - VMware service URL **uuid** - VMware virtual machine host name	VMware Tools must be installed on the guest virtual machine.
vmware.vm.vfs.fs.size[<url>,<uuid>,<fsname>,<mode>]			
VMware virtual machine file system statistics (bytes/percentages).	Integer	**url** - VMware service URL **uuid** - VMware virtual machine host name **fsname** - file system name **mode** - total/free/used/pfree/pused	VMware Tools must be installed on the guest virtual machine.

Footnotes

[1] 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]`

See also: Creating custom performance counter names for VMware.

The performance counter group descriptions, counter names and rollup types can be found in VMware documentation.

[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.

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}	JSON object with an array of {instanceName:partitionId}.
	{#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		
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.

Item key		
	{#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: [{”instance”:”name”, ”partitionid”:1}]
	”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.
	{#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.

Item key		
	{#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.
	{#IFMAC}	Interface mac address.
	{#IFCONNECTED}	Interface connection status (0 - disconnected, 1 - connected).
	{#IFTYPE}	Interface type.
	{#IFBACKINGDEVICENAME}	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.

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
}
```

11. Maintenance

Overview You can define maintenance periods for host groups, hosts and specific triggers/services in Zabbix.

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.

A proxy will always collect 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.

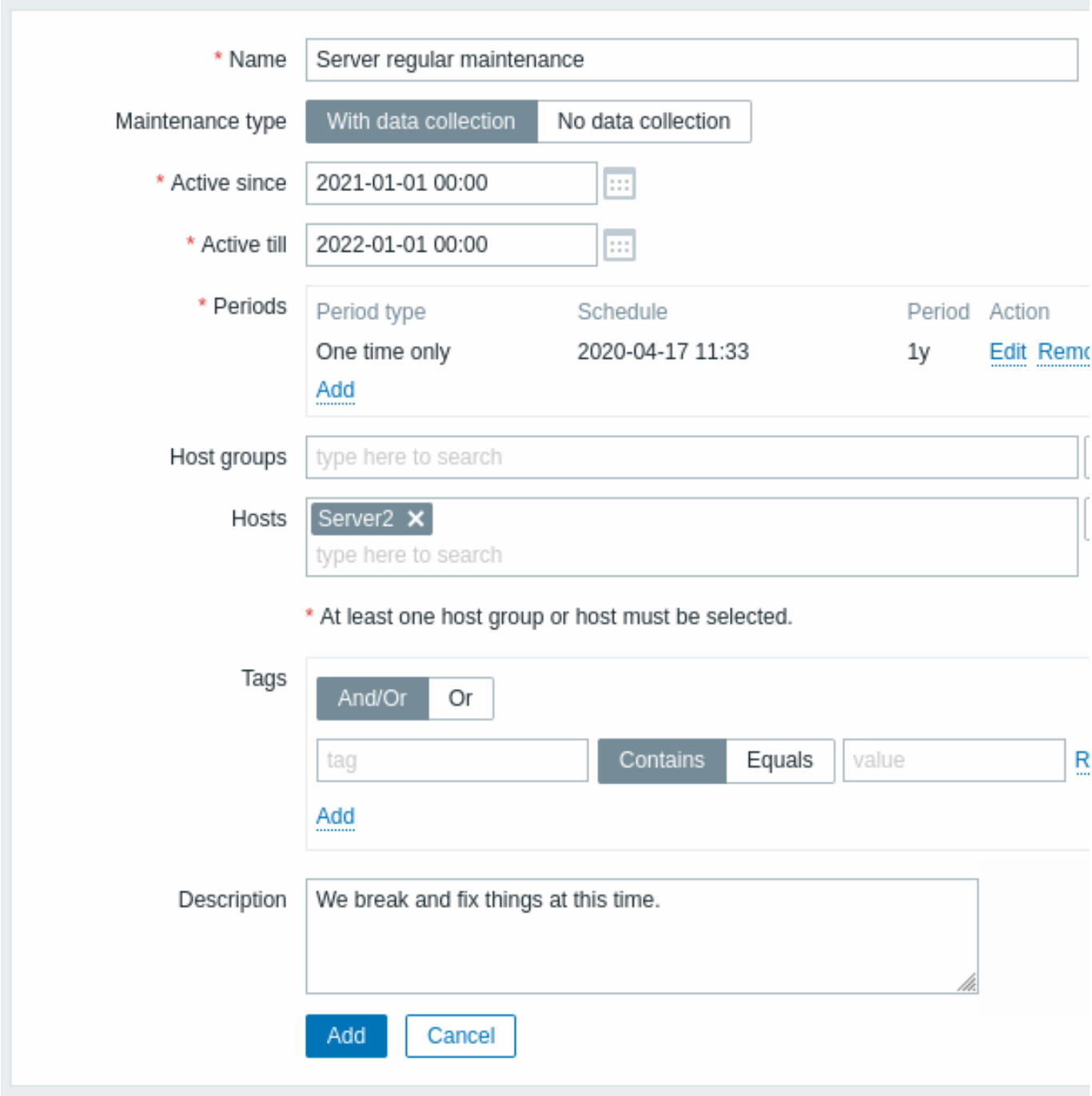
If a timestamped value is sent for a host that is in a "no data" maintenance type (e.g. using Zabbix sender) then this value will be dropped however it is possible to send a timestamped value in for an expired maintenance period and it will be accepted.

If maintenance period, hosts, groups or tags are changed by the user, the changes will only take effect after configuration cache synchronization.

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.

Configuration To configure a maintenance period:

- Go to: *Configuration* → *Maintenance*
- Click on *Create maintenance period* (or on the name of an existing maintenance period)
- Enter maintenance parameters in the form



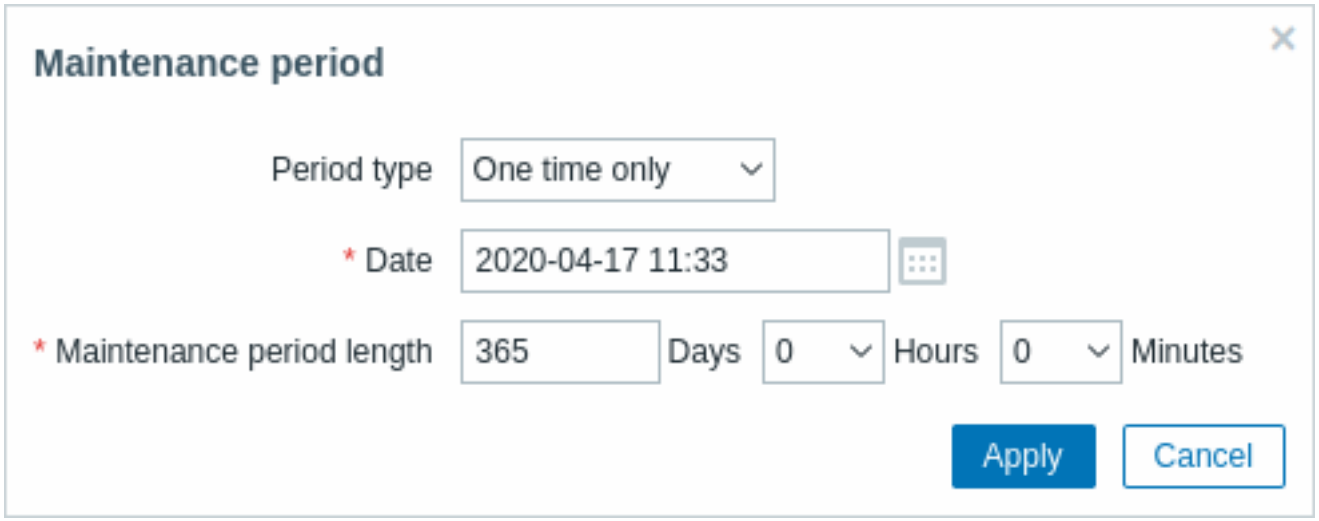
All mandatory input fields are marked with a red asterisk.

Parameter	Description
Name	Name of the maintenance period.

Parameter	Description
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; for that go to the *Periods* tab.
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.
Tags	If maintenance tags are specified, maintenance for the selected hosts will be activated, but only problems with matching tags will be suppressed (i.e. 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	Define the date and time, and the length of the maintenance period.
Daily	*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.

Period type	Description
Weekly	*Every week(s)* - maintenance frequency: 1 (default) - every day, 2 - every two days, 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.
Monthly	*Month* - select all months during which the regular maintenance is carried out. *Date*: **Day of month** - Select this option if the maintenance takes place on the same date each month (for example, every 1st day of the month). Then, select the required day in the new field that appears. *Date*: **Day of week** - Select this option if the maintenance takes place only on certain days (for example, every first Monday of the month). Then, in the drop-down select the required week of the month (first, second, third, fourth, or last) and 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.

When done, press *Add* to add the maintenance period to the *Periods* block.

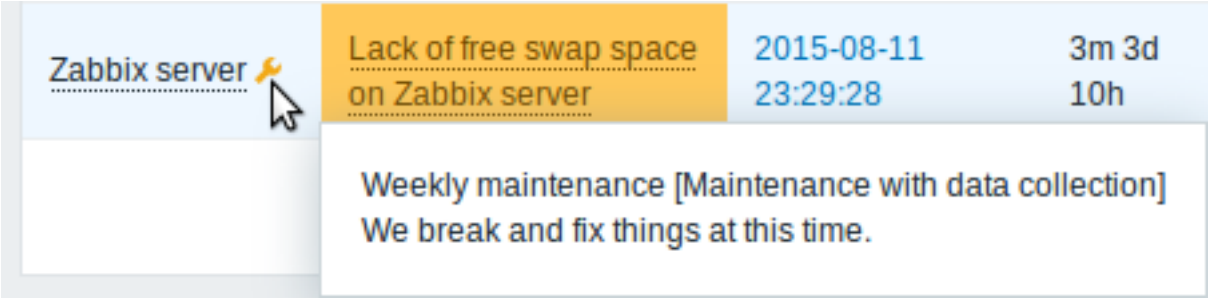
Notes:

- When *Every day/Every week* parameter is greater than 1, the starting day or week is the day/week that the *Active since* time falls on. For example:
  - with *Active since* set to January 1 at 12:00 and a one-hour maintenance set for every two days at 23:00 will result in the first maintenance period starting on January 1 at 23:00, while the second maintenance period will start on January 3 at 23:00;
  - with the same *Active since* time and a one-hour maintenance set for every two days at 01:00, 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.
- Daylight Saving Time (**DST**) changes do not affect how long the maintenance will be. -Let's say we have a two-hour maintenance 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 is set to 1 day it usually starts at 00:00 and finishes at 00:00 the next day:
  - Since Zabbix calculates days in hours, the actual period of the maintenance is 24 hours. -If current time changes forward one hour, the maintenance will stop at 01:00 the next day. -If current time changes back one hour, the maintenance will stop at 23:00 that day. -If a maintenance period starts during the hour, skipped by DST change: -The maintenance will not start.

Display Displaying hosts in maintenance

An orange wrench icon next to the host name indicates that this host is in maintenance in:

- *Monitoring* → *Dashboard*
- *Monitoring* → *Problems*
- *Inventory* → *Hosts* → *Host inventory details*
- *Configuration* → *Hosts* (See 'Status' column)



Maintenance details are displayed when the mouse pointer is positioned over the icon.

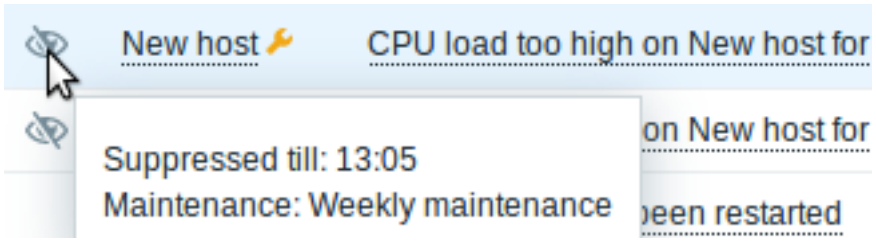
Additionally, hosts in maintenance get an orange background in *Monitoring* → *Maps*.

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:

- *Monitoring* → *Dashboard* (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.

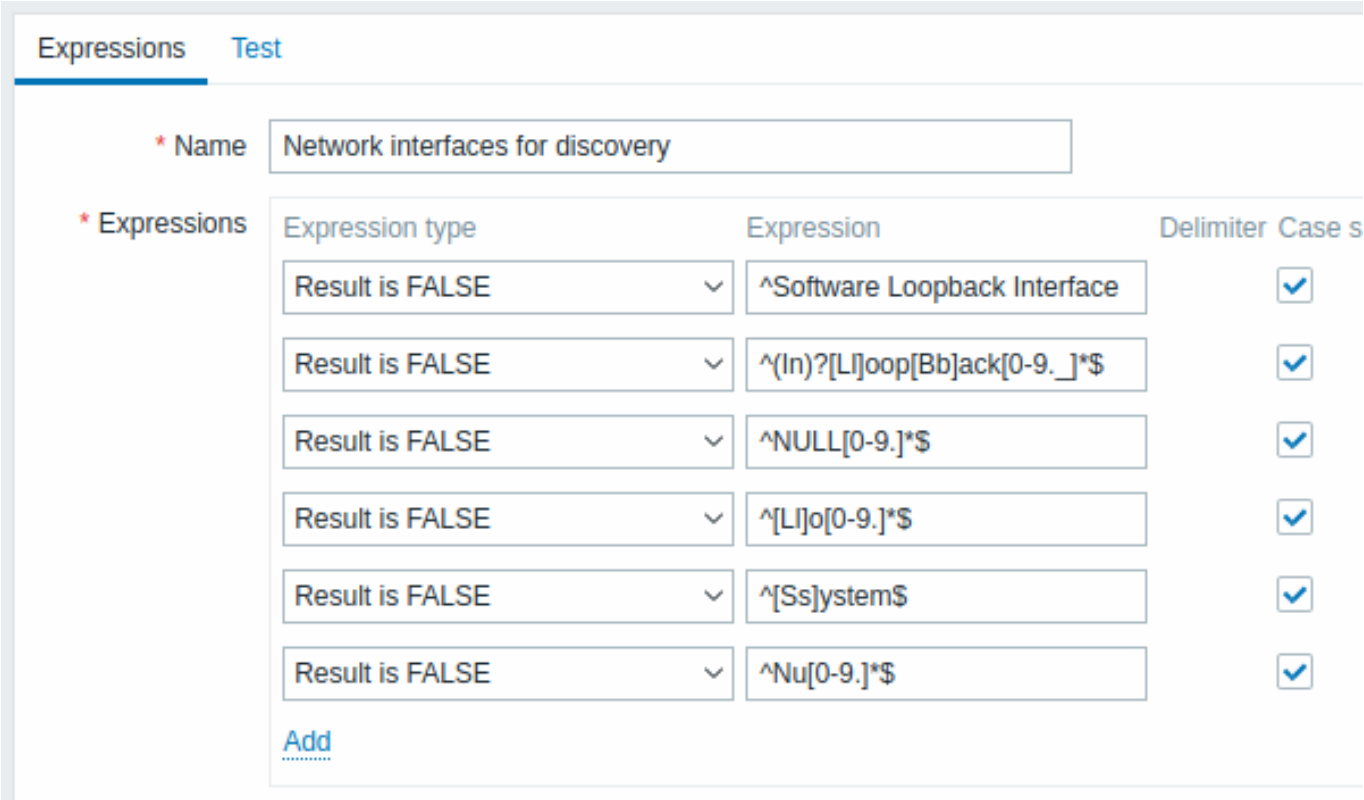
Global regular expressions There is an advanced editor for creating and testing complex regular expressions in Zabbix frontend.

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.



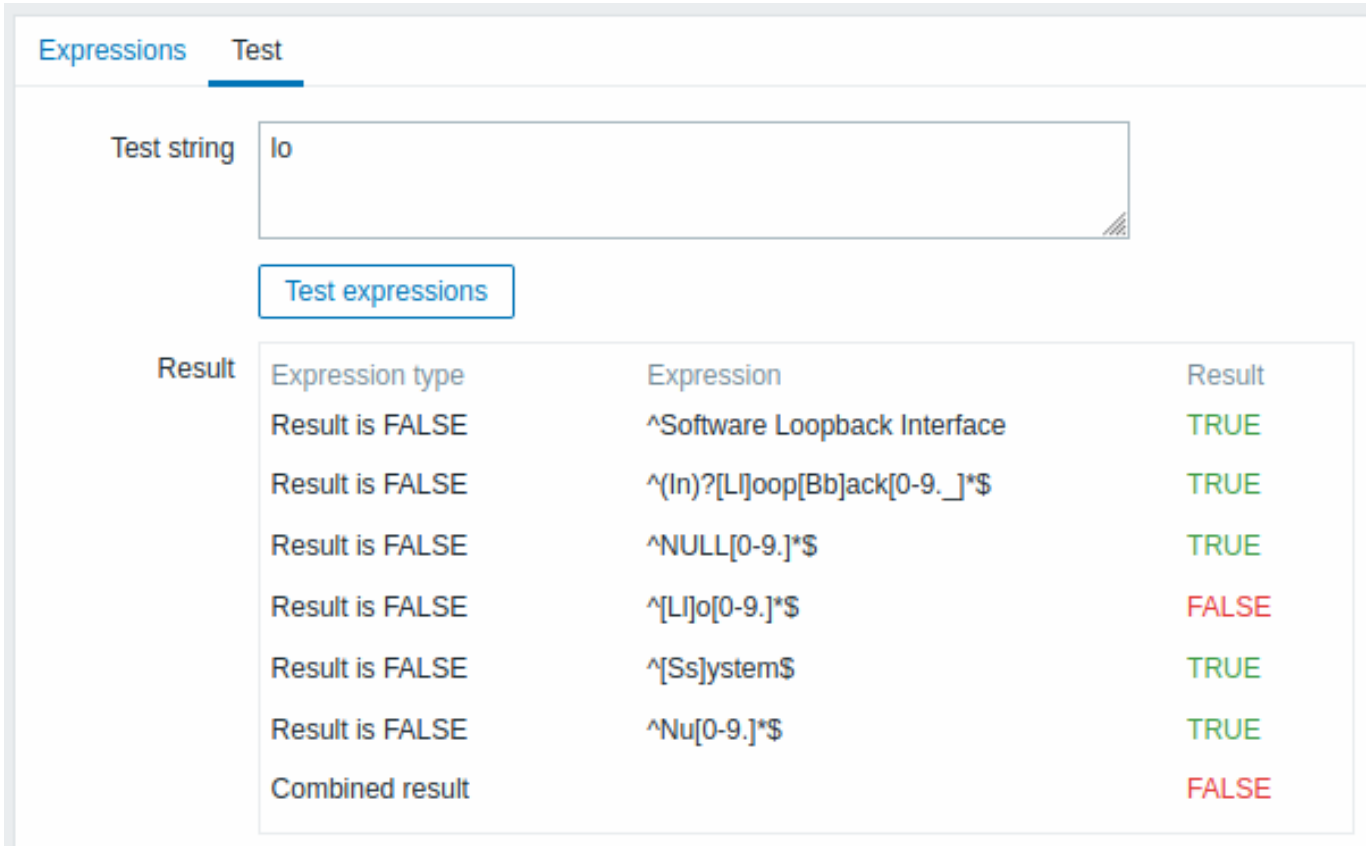
All mandatory input fields are marked with a red asterisk.

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.

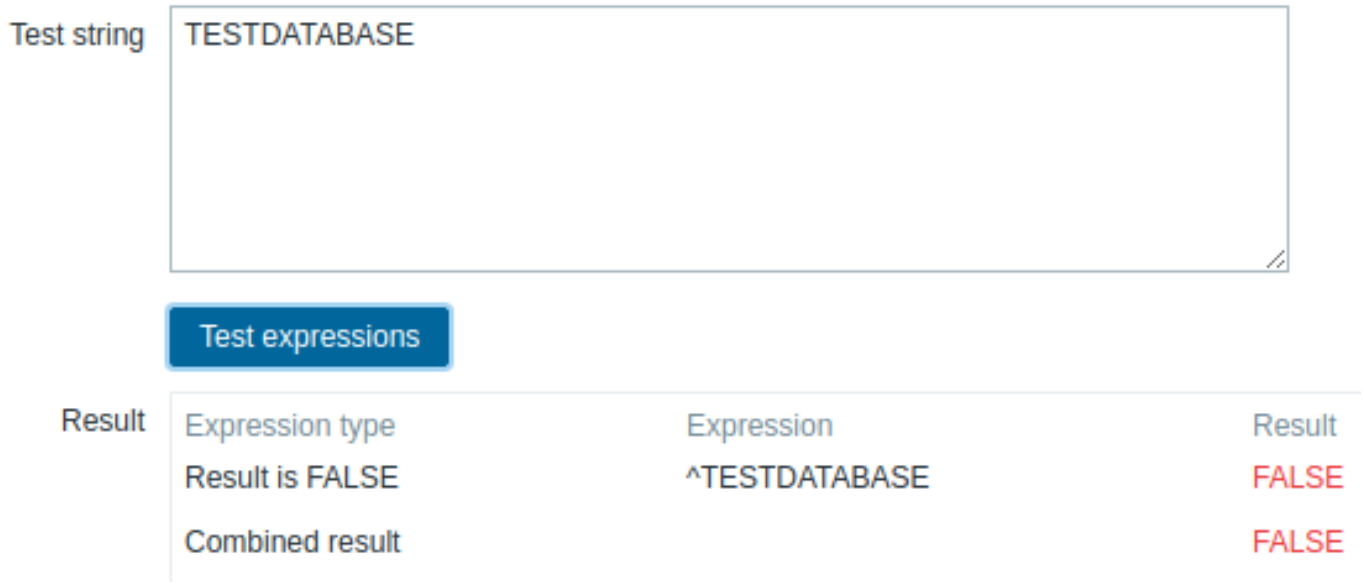
Name	Expression	Matches
File systems for discovery	`^(btrfs|ext2|ext3|ext4|jfs|reiser|xfs|ffs|ufs|jfs2|vxfs|hfs|refs|apfs|ntfs|fat32|zfs)$`	”btrfs” or ”ext2” or ”ext3” or ”ext4” or ”jfs” or ”reiser” or ”xfs” or ”ffs” or ”ufs” or ”jfs” or ”jfs2” or ”vxfs” or ”hfs” or ”refs” or ”apfs” or ”ntfs” or ”fat32” or ”zfs”
Network interfaces for discovery	`^Software Loopback Interface`	Strings starting with ”Software Loopback Interface”.
	`^lo$`	”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$`

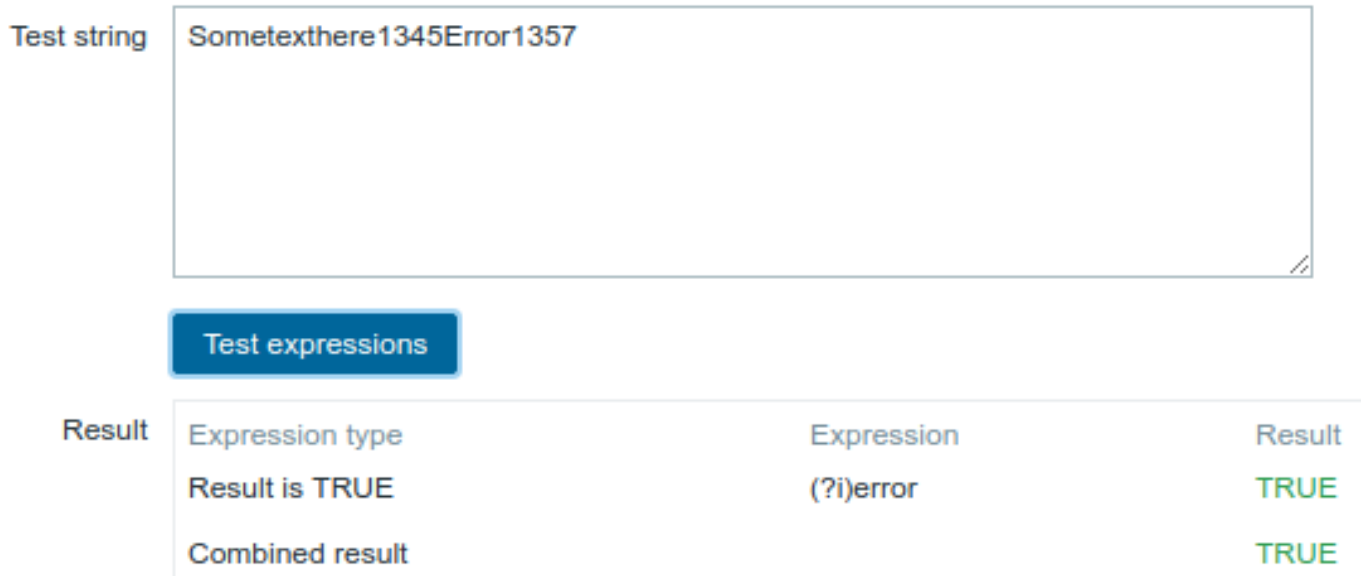


Chosen *Expression type*: ”Result is FALSE”. Doesn’t match name, containing string ”*TESTDATABASE*”.

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.mem[]				
proc.num[]				
sensor[]				`device` and `sensor` parameters on Linux 2.4
system.hw.macaddr[]				`interface` parameter
system.sw.packages[]				`package` parameter
vfs.dir.count[]				`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 Functions for trigers/calculated items	Yes	No	No	`pattern` parameter

Location	Regular expression	Global regular expression	Multiline matching	Comments
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 discovery				
Filters	Yes	Yes	No	*Regular expression* field
Overrides	Yes	No		In *matches*, *does not match* options for *Operation* conditions
Action conditions	Yes	No	No	In *matches*, *does not match* options for *Host name* and *Host metadata* autoregistration conditions
Web monitoring	Yes	No	Yes	*Variables* with a **regex:** prefix *Required string* field
User macro context **Macro functions**	Yes	No	No	In macro context with a **regex:** prefix
regsub() iregsub()	Yes	No	No	`pattern` parameter
Icon mapping	Yes	Yes	No	*Expression* field
Value mapping	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 in the *Ack* column showing the acknowledgment status of problems in:
  - *Monitoring* → *Dashboard* (*Problems* and *Problems by severity* widgets)
  - *Monitoring* → *Problems*
  - *Monitoring* → *Problems* → *Event details*

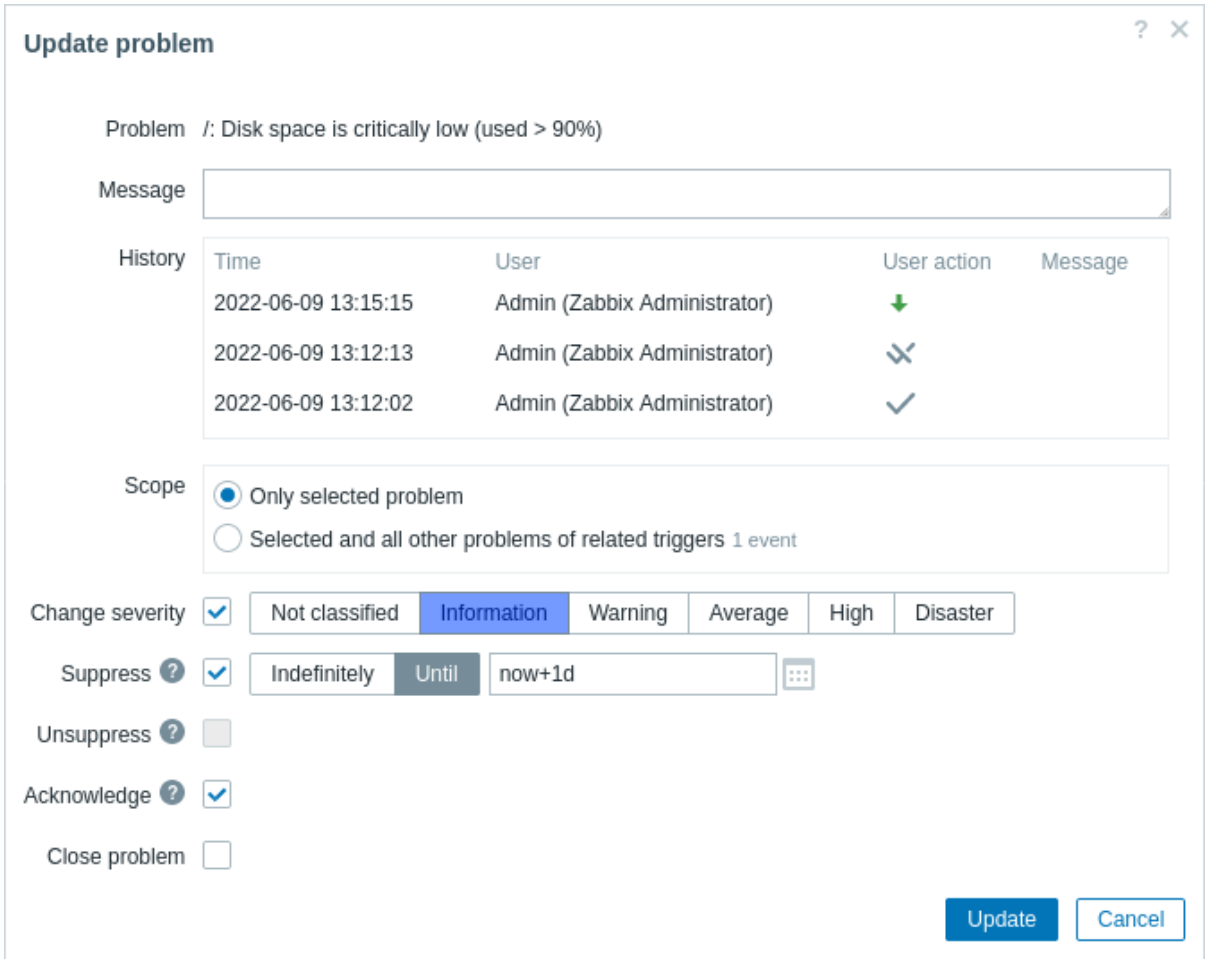
The *Ack* column contains either a 'Yes' or a 'No' link, indicating an acknowledged or an unacknowledged problem respectively. Clicking on the links will take you to the problem update popup window.

- You can click on an unresolved problem cell in:
  - *Monitoring* → *Dashboard* (*Trigger overview* widget)

The popup menu contains an *Acknowledge* 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
- 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.

Parameter	Description
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.
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.

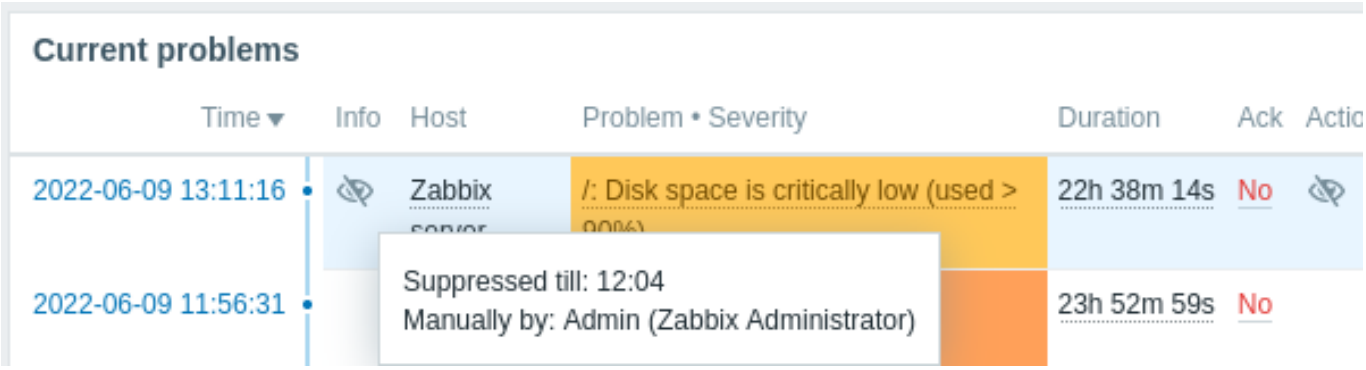
Display

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
- share web scenarios on *share.zabbix.com* - export a template with the web scenarios and upload to *share.zabbix.com*. 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, host/template applications, discoveryRules, itemPrototypes, triggerPrototypes, graphPrototypes are meaningless i.e. it's the same as if it was missing. Other tags, for example, item applications, are meaningful i.e. empty tag means no applications for item, missing tag means don't update applications.
- 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.

```
zabbix_export:
  version: '6.2'
  date: '2020-04-22T06:20:11Z'
```

YAML base format

`zabbix_export:`

Root node for Zabbix YAML export.

`version: '6.2'`

Export version.

`date: '2020-04-22T06:20:11Z'`

Date when export was created in ISO 8601 long format.

Other nodes are dependent on exported objects.

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
    <version>6.0</version>
    <date>2020-04-22T06:20:11Z</date>
</zabbix_export>
```

XML format

`<?xml version="1.0" encoding="UTF-8"?>`

Default header for XML documents.

`<zabbix_export>`

Root element for Zabbix XML export.

`<version>6.0</version>`

Export version.

`<date>2020-04-22T06:20:11Z</date>`

Date when export was created in ISO 8601 long format.

Other tags are dependent on exported objects.

```
{
    "zabbix_export": {
        "version": "6.0",
        "date": "2020-04-22T06:20:11Z"
    }
}
```

JSON format

```
  "zabbix_export":
```

Root node for Zabbix JSON export.

```
      "version": "6.0"
```

Export version.

```
      "date": "2020-04-22T06:20:11Z"
```

Date when export was created in ISO 8601 long format.

Other nodes 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.

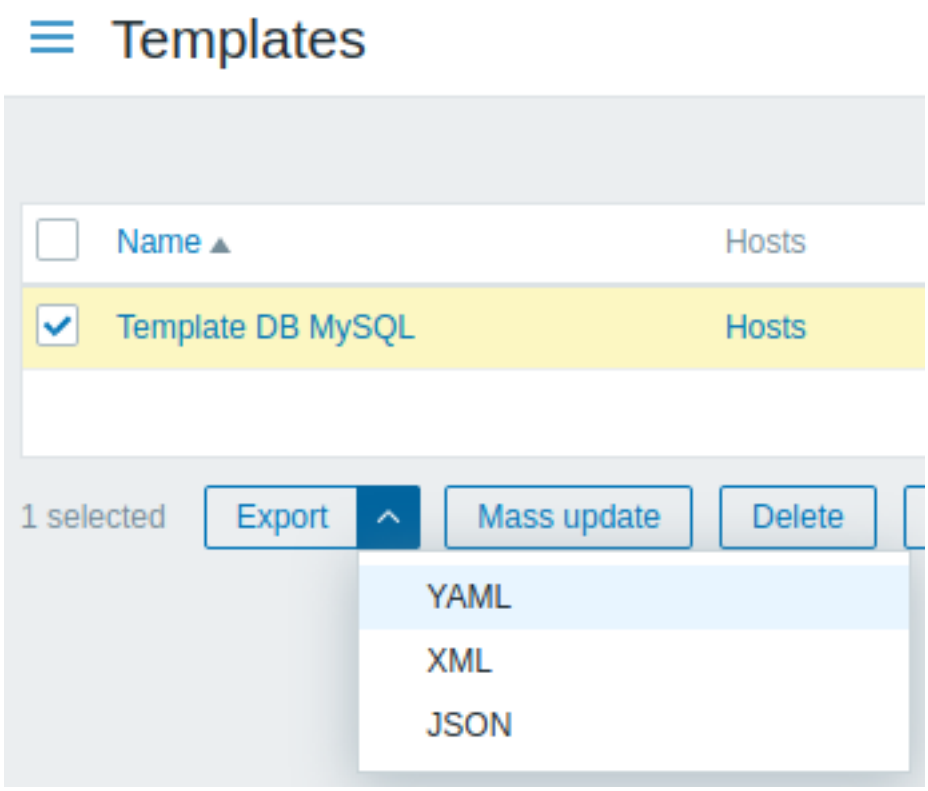
Template export contains:

- 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:

- Go to: *Configuration* → *Templates*
- Mark the checkboxes of the templates to export
- Click on *Export* below the list



Depending on the selected format, templates are exported to a local file with a default name:

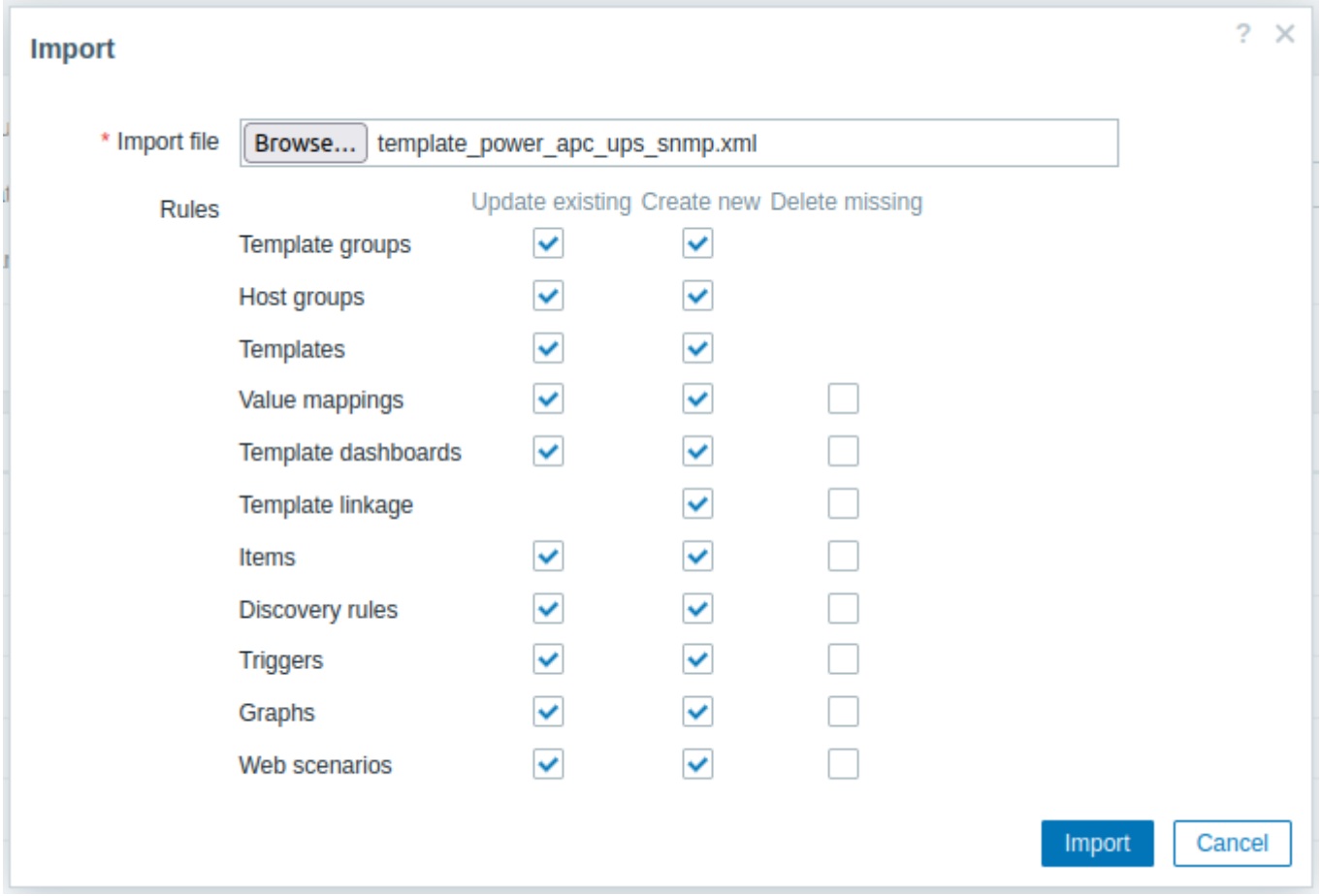
- *zabbix_export_templates.yaml* - in YAML export (default option for export)
- *zabbix_export_templates.xml* - in XML export
- *zabbix_export_templates.json* - in JSON export

Importing

To import templates, do the following:

- Go to: *Configuration* → *Templates*
- Click on *Import* to the right

- Select the import file
- Mark the required options in import rules
- Click on *Import*



All mandatory input fields are marked with a red asterisk.

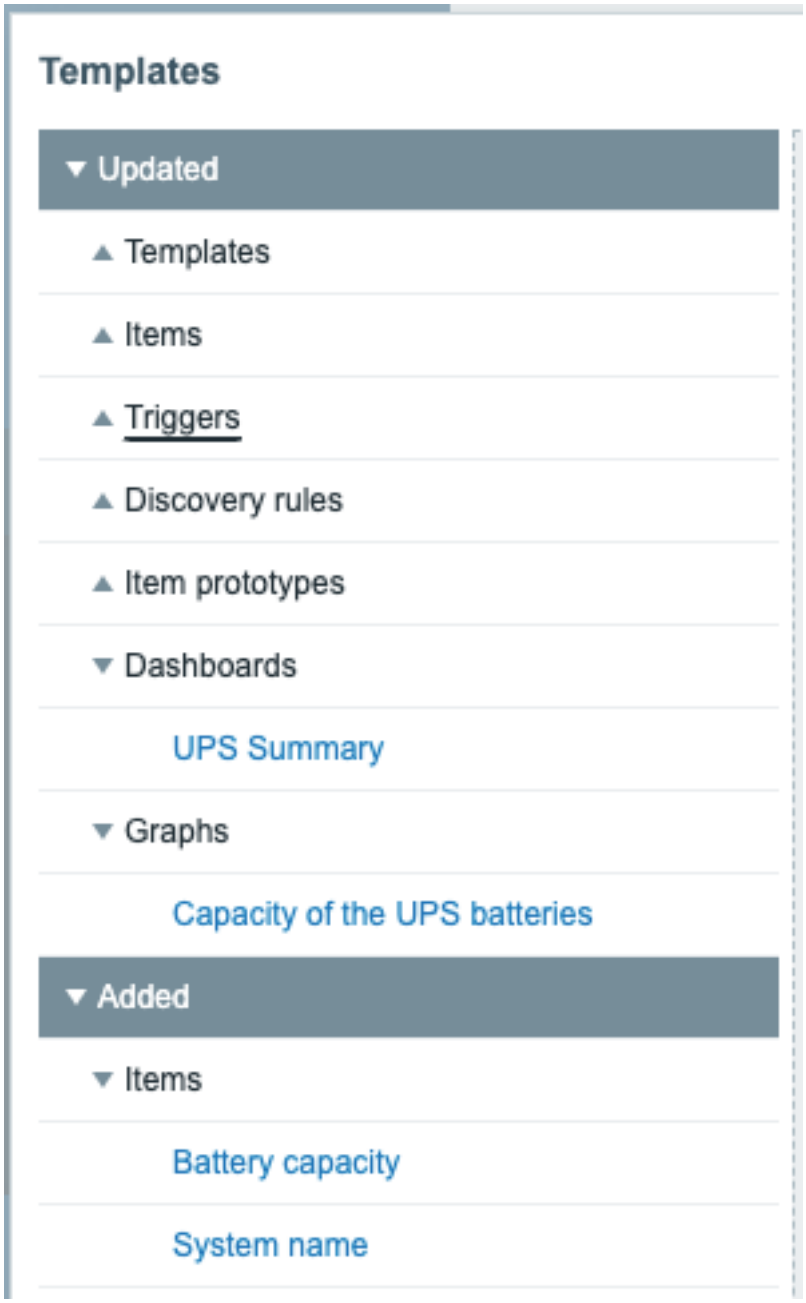
Import rules:

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. Since Zabbix 6.2.7, 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: '6.2'
  date: '2021-08-31T12:40:55Z'
  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
```

```
  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.
groups		-			Root element for template groups.
	name	x	string		Template group name.
templates		-			Root element for linked templates.
	name	x	string		Template 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.
	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.

Element	Element property	Required	Type	Range[1]	Description
	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	A time unit of how long the history data should be stored. Time unit with suffix, user macro or LLD macro.
	trends	-	`string`	Default: 365d	A time unit of how long the trends data should be stored. Time unit with suffix, 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.
	port	-	string		Custom port monitored by the item. Can contain user macros. Used only by SNMP 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.
	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 pooling 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.

Element	Element property	Required	Type	Range[1]	Description
	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.
	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	Validate if host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. Used only by HTTP agent items.
	verify_peer	-	string	0 - NO (default) 1 - YES	Validate if host 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	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.
	name	x	`string`		Trigger name.
	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.
	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.
	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.

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	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	Verify the SSL certificate of the web server.

Element	Element property	Required	Type	Range[1]	Description
	verify_host	-	string	0 - NO (default) 1 - YES	Verify that the Common Name field or the Subject Alternate Name field of the web server certificate matches.
	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.

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:

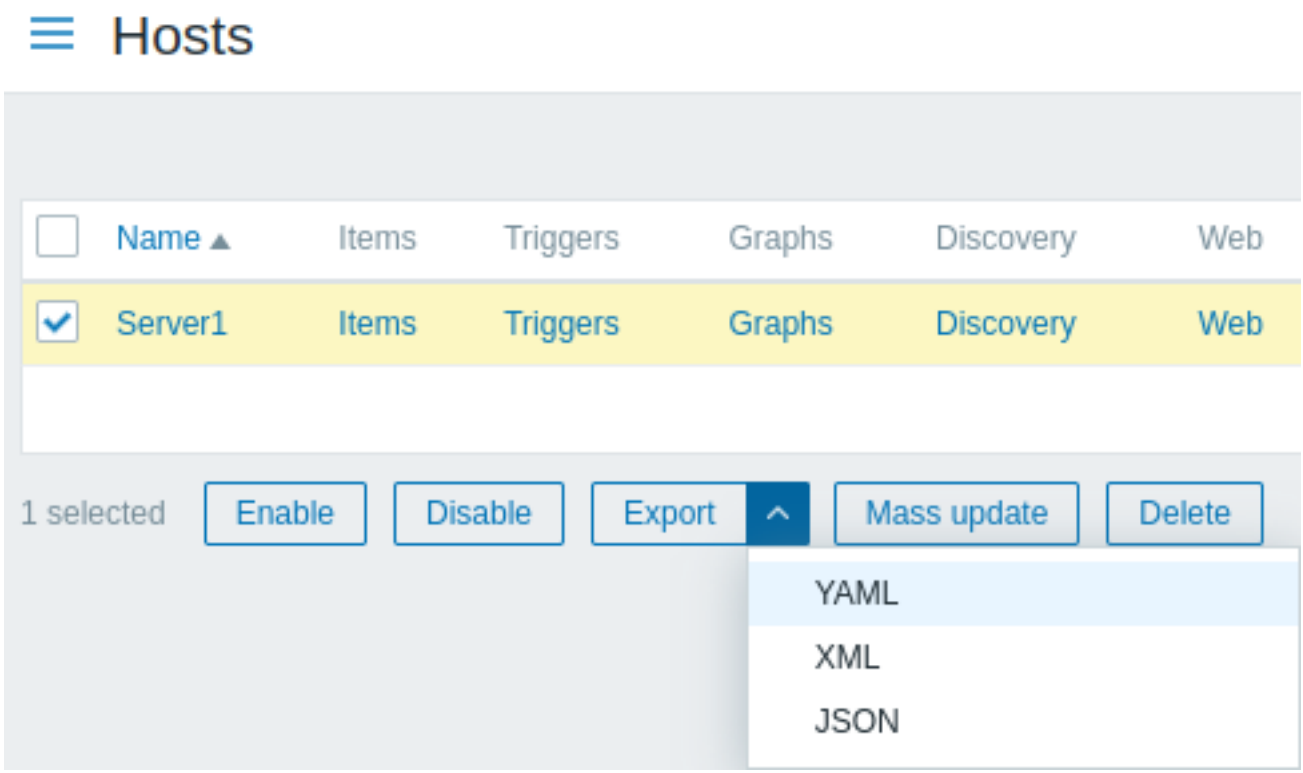
- linked host groups
- host data
- template linkage
- host group linkage
- host interfaces
- directly linked items
- directly linked triggers
- directly linked graphs
- directly linked discovery rules with all prototypes
- directly linked web scenarios
- host macros

- host inventory data
- value maps

Exporting

To export hosts, do the following:

- Go to: *Configuration* → *Hosts*
- Mark the checkboxes of the hosts to export
- Click on *Export* below the list



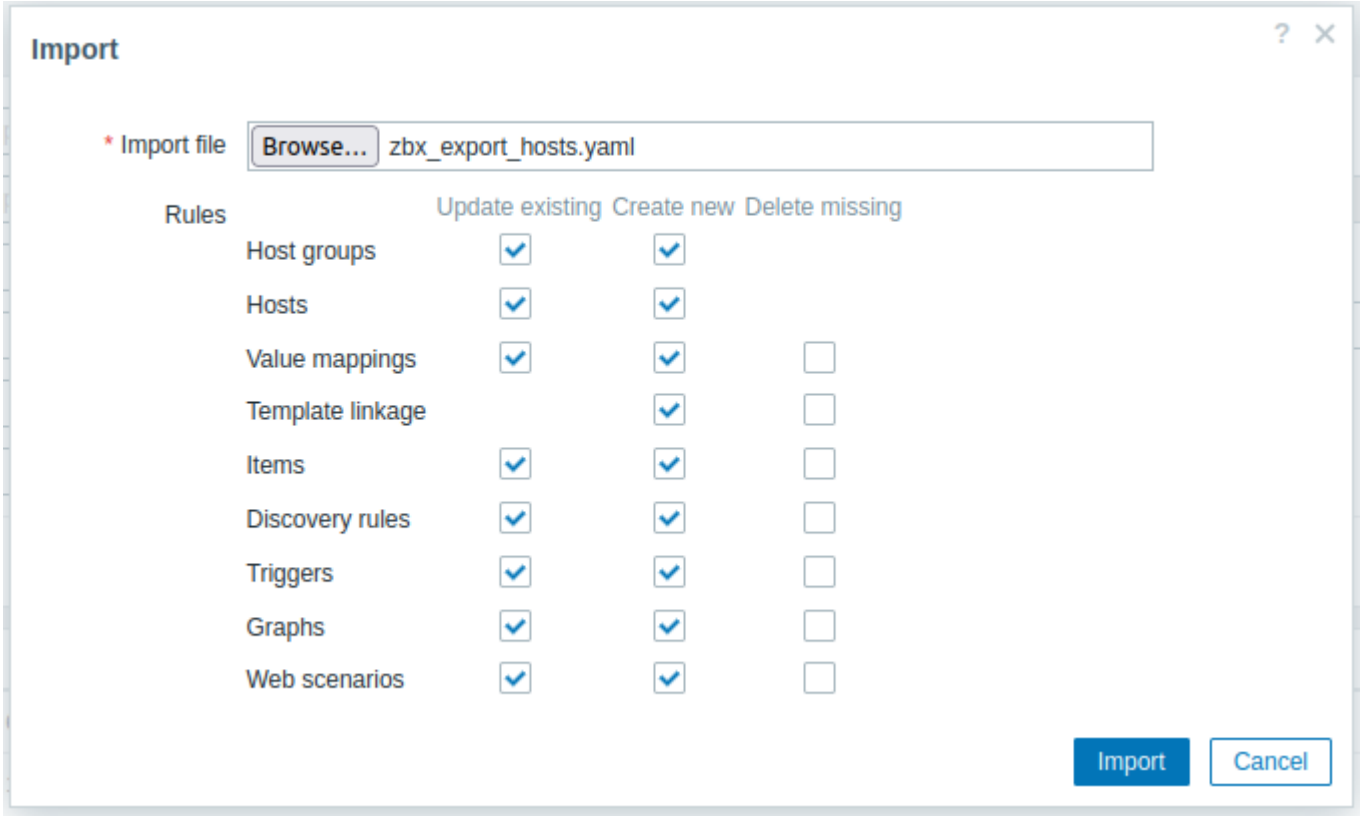
Depending on the selected format, hosts are exported to a local file with a default name:

- *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:

- Go to: *Configuration* → *Hosts*
- Click on *Import* to the right
- Select the import file
- Mark the required options in import rules
- Click on *Import*



A success or failure message of the import will be displayed in the frontend.

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. Since Zabbix 6.2.7, 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).

Export format

Export format in YAML:

```
zabbix_export:
  version: '6.2'
  date: '2021-09-28T12:20:07Z'
  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.

Element	Element property	Required	Type	Range[1]	Description
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.
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.

Element	Element property	Required	Type	Range[1]	Description
	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.
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.

Element	Element property	Required	Type	Range[1]	Description
	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. 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	A time unit of how long the history data should be stored. Time unit with suffix, user macro or LLD macro.
	trends	-	string	Default: 365d	A time unit of how long the trends data should be stored. Time unit with suffix, 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).

Element	Element property	Required	Type	Range[1]	Description
	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.
	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.

Element	Element property	Required	Type	Range[1]	Description
	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 pooling 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.
	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	Validate if host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. Used only by HTTP agent items.

Element	Element property	Required	Type	Range[1]	Description
	verify_peer	-	string	0 - NO (default) 1 - YES	Validate if host 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.
	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.

Element	Element property	Required	Type	Range[1]	Description
	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.
	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.				

Element	Element property	Required	Type	Range[1]	Description
	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.
	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.

Element	Element property	Required	Type	Range[1]	Description
	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.				
	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		-			Root element for trigger prototypes.
	For trigger prototype element tag values, see regular host trigger tags.				
graph_prototypes		-			Root element for graph prototypes.
	For graph prototype element tag values, see regular host graph tags.				
host_prototypes		-			Root element for host prototypes.
	For host prototype element tag values, see regular host tags.				

Element	Element property	Required	Type	Range[1]	Description
item_prototypes		-			Root element for item prototypes.
	For item prototype element tag values, see regular host item tags.				
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.
	name	x	`string`		Trigger name.
	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.
	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.
	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.

Element	Element property	Required	Type	Range[1]	Description
	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.
	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.

Element	Element property	Required	Type	Range[1]	Description
	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	Verify the SSL certificate of the web server.
	verify_host	-	string	0 - NO (default) 1 - YES	Verify that the Common Name field or the Subject Alternate Name field of the web server certificate matches.
	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

Element	Element property	Required	Type	Range[1]	Description
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

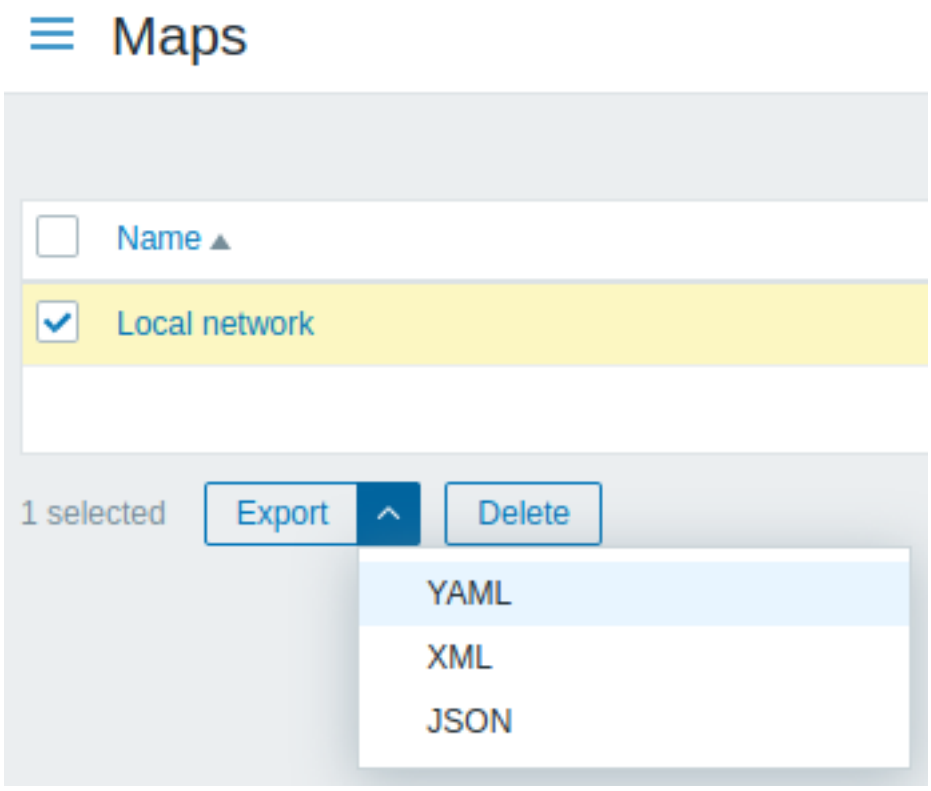
Warning:
Any host groups, hosts, triggers, other maps or other elements that may be related to the exported map are not exported. Thus, if at least one of the elements the map refers to is missing, importing it will fail.

Network map export/import is supported since Zabbix 1.8.2.

Exporting

To export network maps, do the following:

- Go to: *Monitoring* → *Maps*
- Mark the checkboxes of the network maps to export
- Click on *Export* below the list



Depending on the selected format, maps are exported to a local file with a default name:

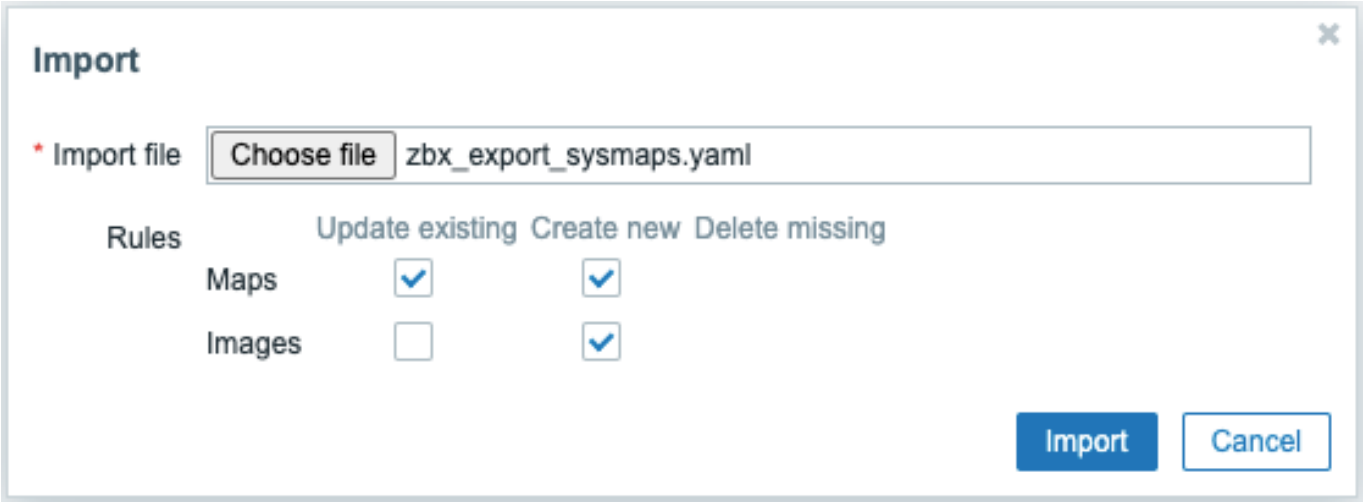
- *zabbix_export_maps.yaml* - in YAML export (default option for export)

- *zabbix_export_maps.xml* - in XML export
- *zabbix_export_maps.json* - in JSON export

Importing

To import network maps, do the following:

- Go to: *Monitoring* → *Maps*
- Click on *Import* to the right
- Select the import file
- Mark the required options in import rules
- Click on *Import*



All mandatory input fields are marked with a red asterisk.

A success or failure message of the import will be displayed in the frontend.

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.

Warning:
If replacing an existing image, it will affect all maps that are using this image.

Export format

Export to YAML:

```
zabbix_export:
  version: '6.2'
  date: '2021-08-31T12:55:10Z'
  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.
	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.

Element	Element property	Type	Range	Description
	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”
	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.

Element	Element property	Type	Range	Description
	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.
	tag			Tag name.
	value			Tag value.
	operator			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.

Element	Element property	Type	Range	Description
	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.
	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.

Element	Element property	Type	Range	Description
	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

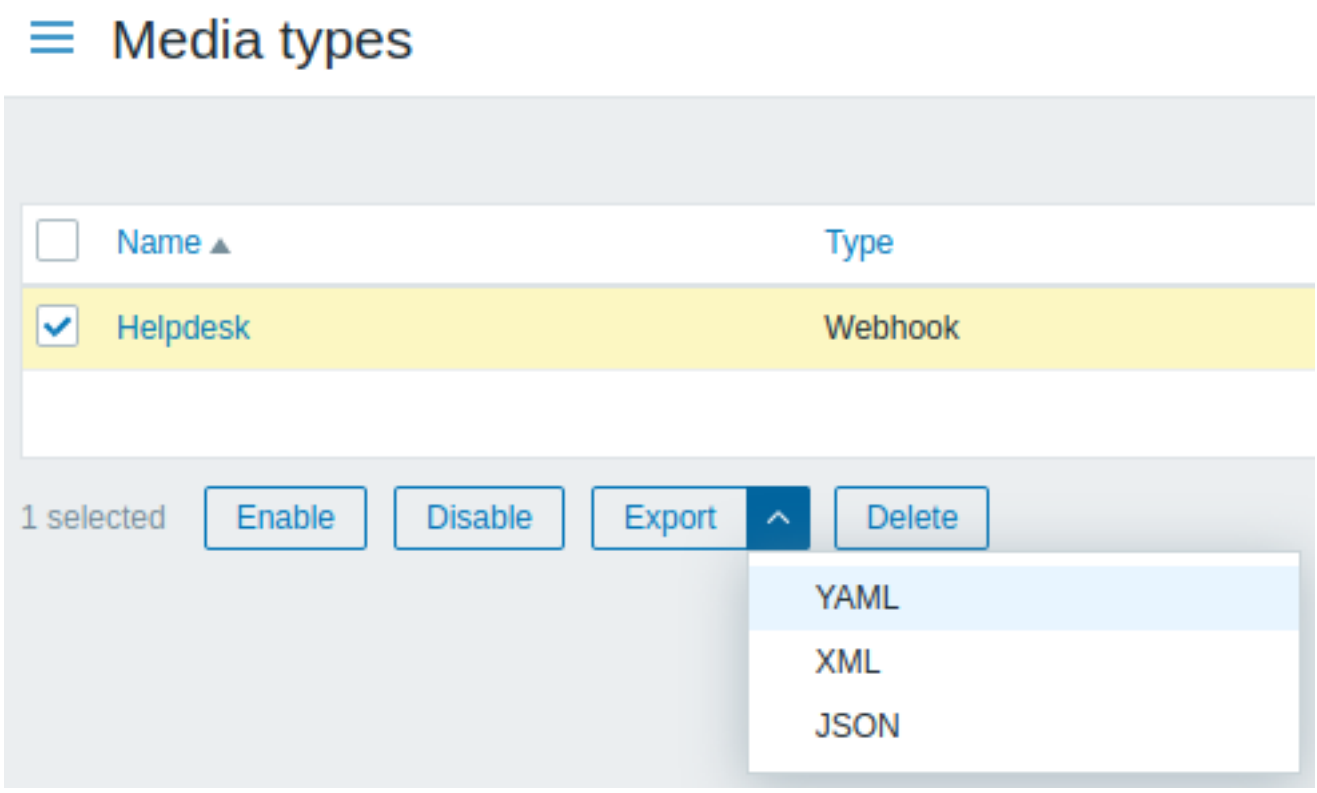
Overview

Media types are exported with all related objects and object relations.

Exporting

To export media types, do the following:

- Go to: *Administration* → *Media types*
- Mark the checkboxes of the media types to export
- Click on *Export* below the list



Depending on the selected format, media types are exported to a local file with a default name:

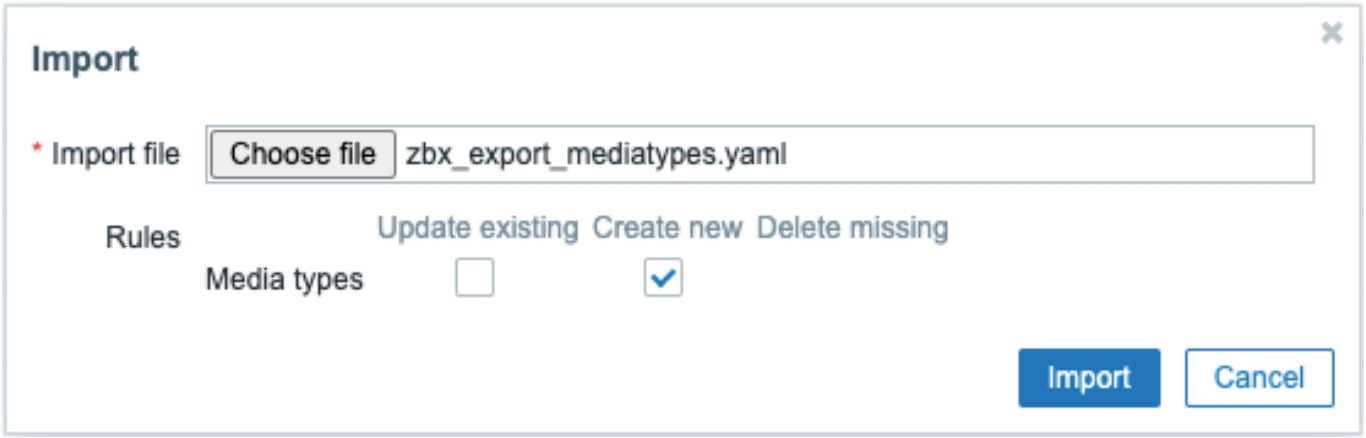
- *zabbix_export_mediatypes.yaml* - in YAML export (default option for export)
- *zabbix_export_mediatypes.xml* - in XML export
- *zabbix_export_mediatypes.json* - in JSON export

Importing

To import media types, do the following:

- Go to: *Administration* → *Media types*

- Click on *Import* to the right
- Select the import file
- Mark the required options in import rules
- Click on *Import*



A success or failure message of the import will be displayed in the frontend.

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.

Export format

Export to YAML:

```
zabbix_export:
  version: '6.2'
  date: '2021-08-31T13:34:17Z'
  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.
	attempt_interval		string	0-60s (default: 10s)	The interval between retry attempts. Accepts seconds and time unit with suffix.

Element	Element property	Required	Type	Range[1]	Description
	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 e-mail 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.
	event_menu_url	-	string		URL of the event menu entry. Supports {EVENT.TAGS.*} macro.

Element	Element property	Required	Type	Range[1]	Description
	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'.

Event	Check of service result
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.

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

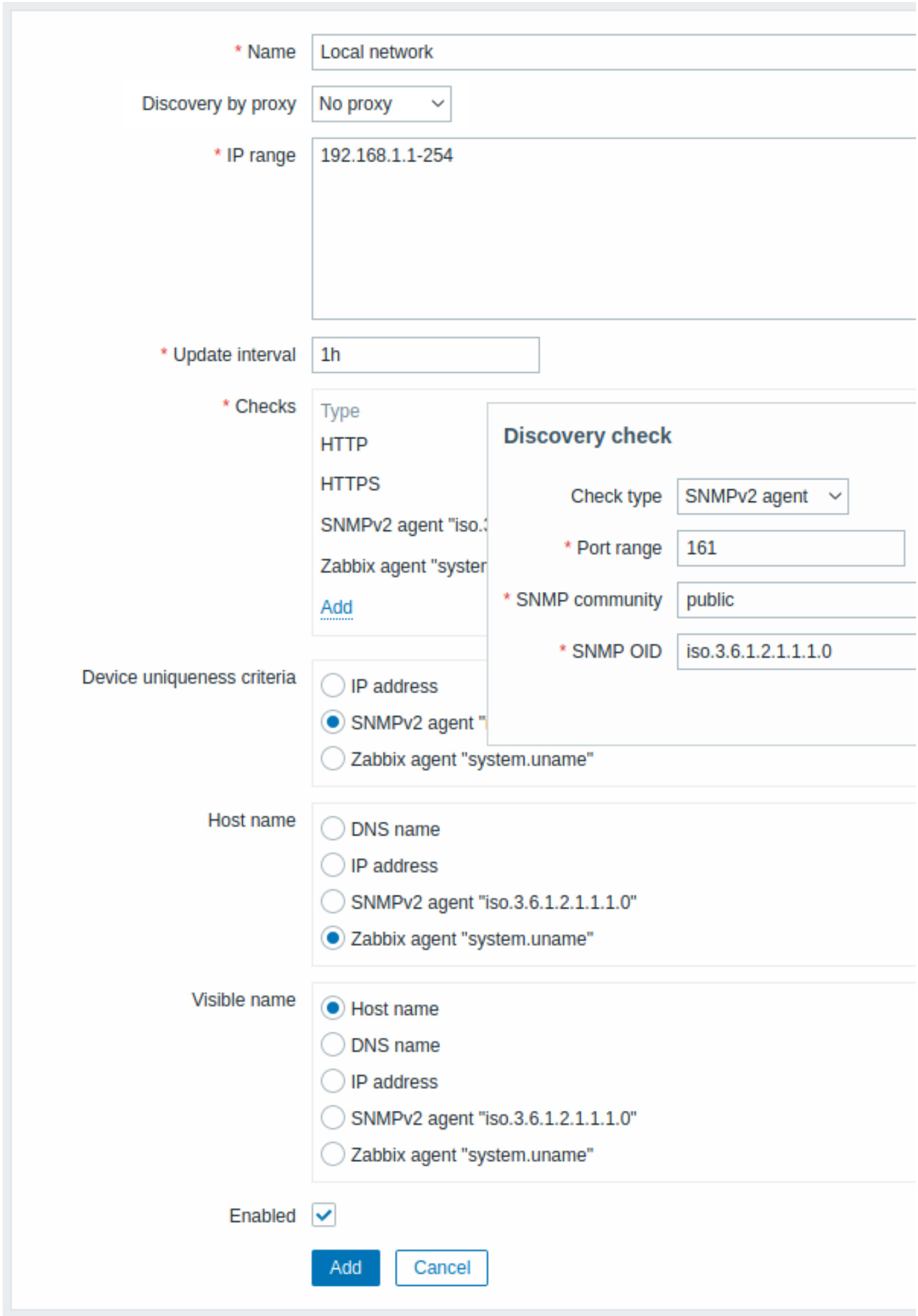
1 Configuring a network discovery rule

Overview

To configure a network discovery rule used by Zabbix to discover hosts and services:

- Go to *Configuration → 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** - no processing of 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.
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.

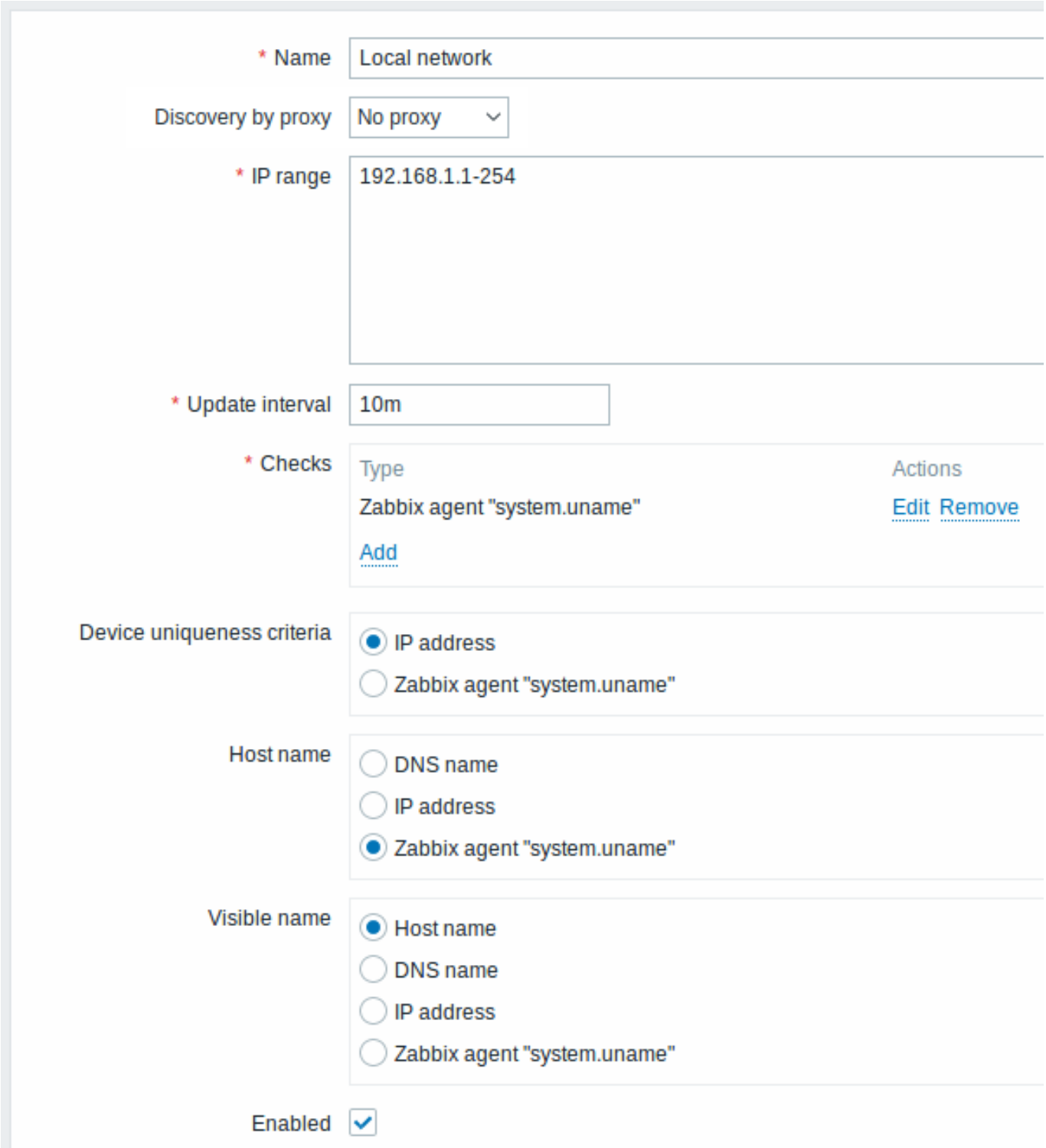
In our scenario we want to:

- 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*.

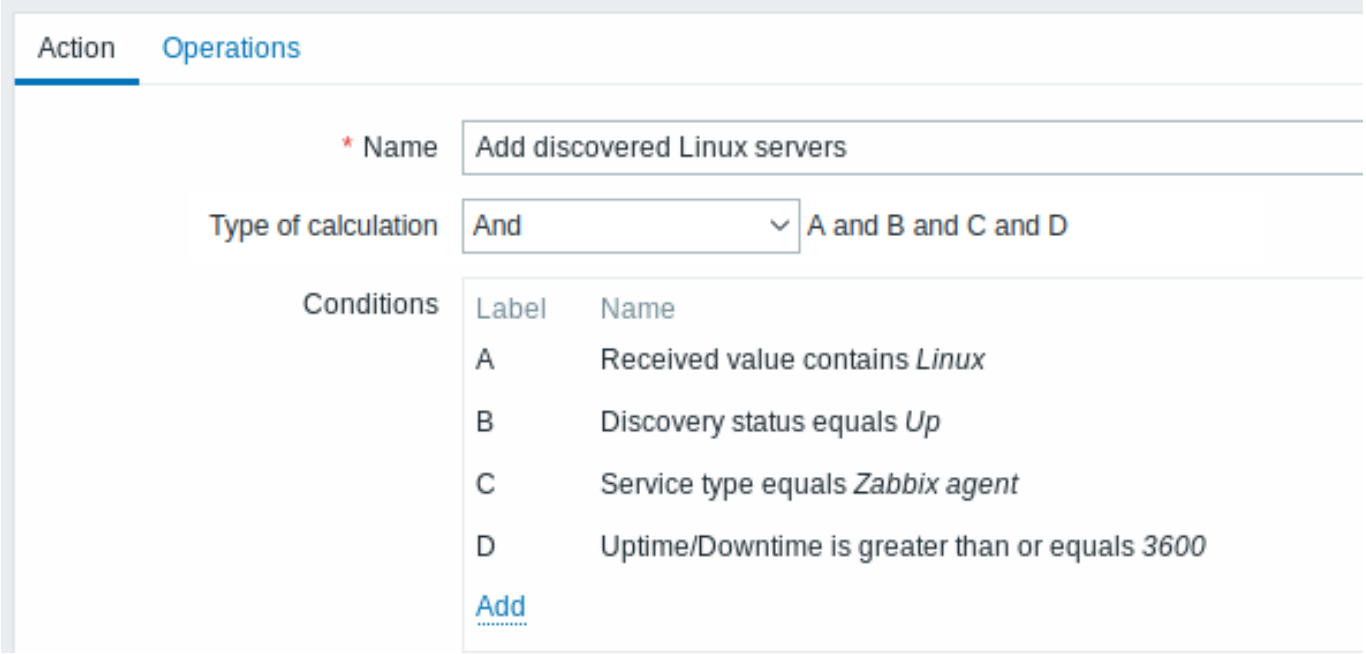
The rule will be executed every 10 minutes.

When this rule is added, Zabbix will automatically start the discovery and generation of the discovery-based events for further

processing.

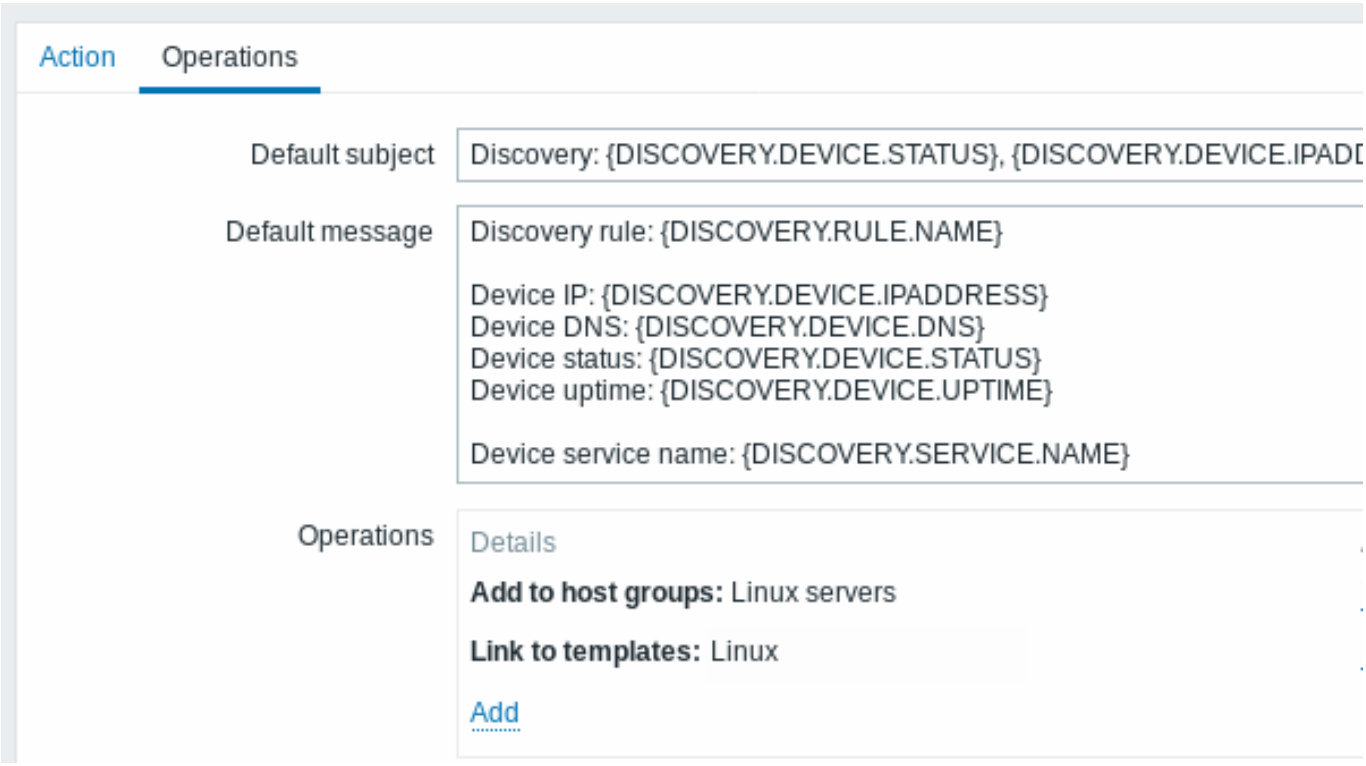
Step 2

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

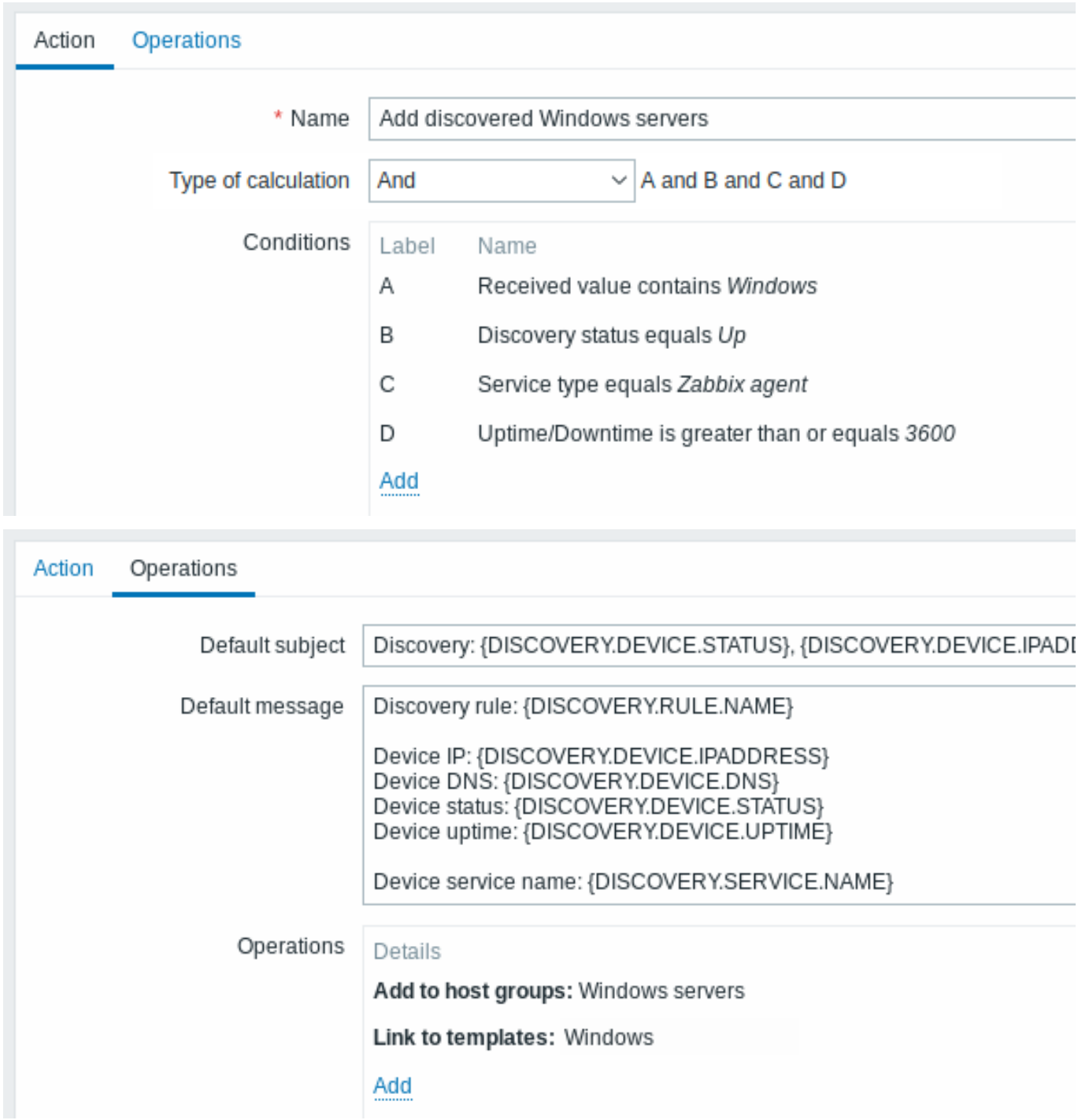


The action will execute the following operations:

- 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.

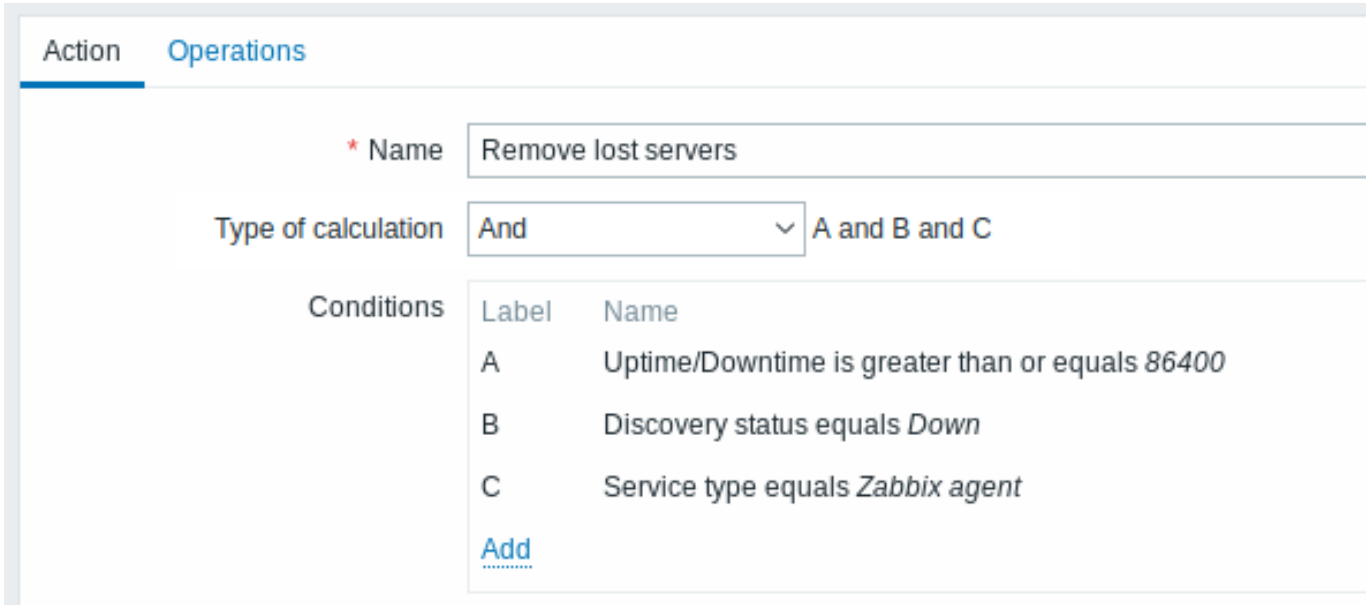
Step 3

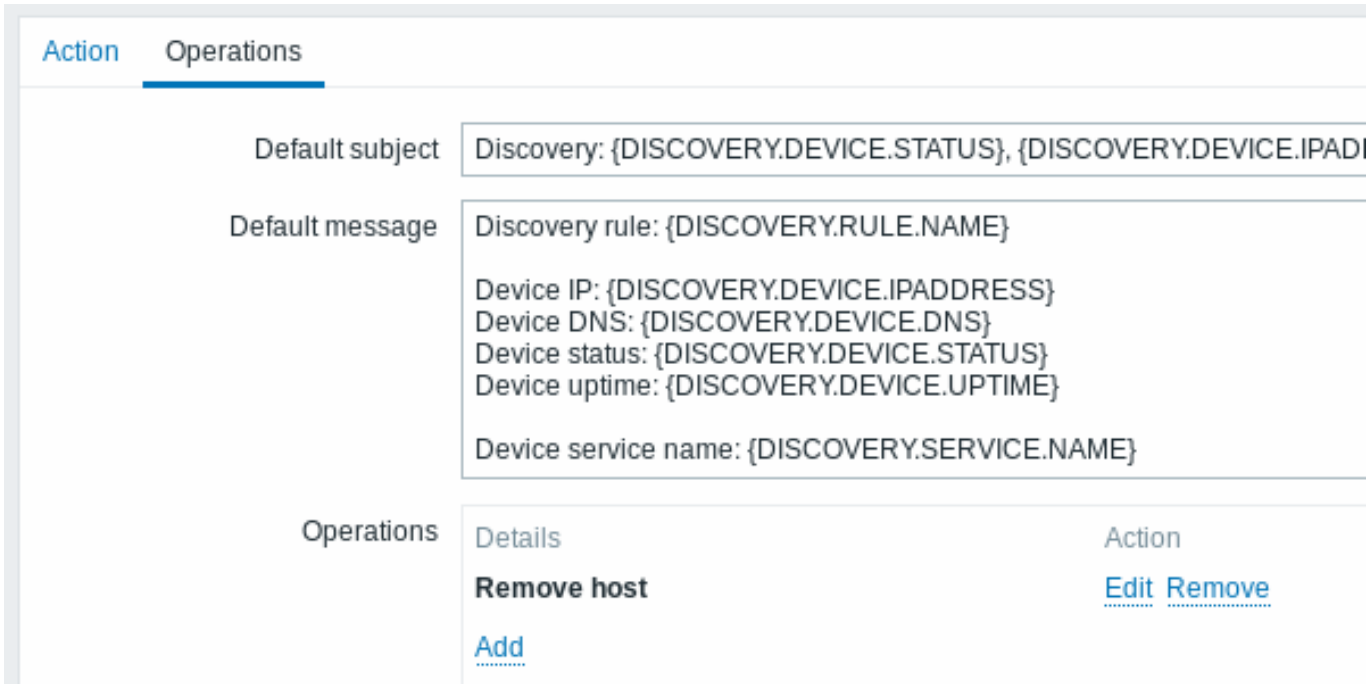
Defining a discovery action for adding the discovered Windows servers to the respective group/template.



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

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 *Configuration* → *Actions*, select *Autoregistration* as the event source 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*, in the Autoregistration section accessible through the dropdown to the right. 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.

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 to templates: Linux

Note:
You can skip an "Add host" operation in this case. Linking to a template requires adding a host first so the server will do that automatically.

The second action:

- Name: Windows host autoregistration
- Conditions: Host metadata contains *Windows*
- Operations: Link to 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 //21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e
* Operations:
    * Send message to users: Admin via all media
    * Add to host groups: Linux servers
    * Link to 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 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08
* Operations:
    * Send message to users: Admin via all media
    * Add to host groups: Linux servers
    * Link to templates: Linux
    * Link to 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 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 ”Configuration” → ”Templates” → ”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.

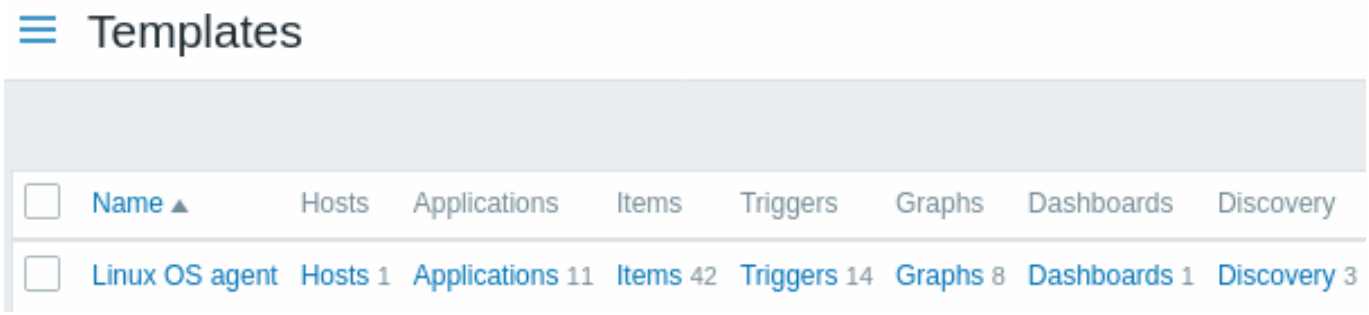
Warning:
As a result of the changes above, newer agents no longer will be able to work with an older Zabbix server.

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: *Configuration* → *Templates* or *Hosts*
- Click on *Discovery* in the row of an appropriate template/host



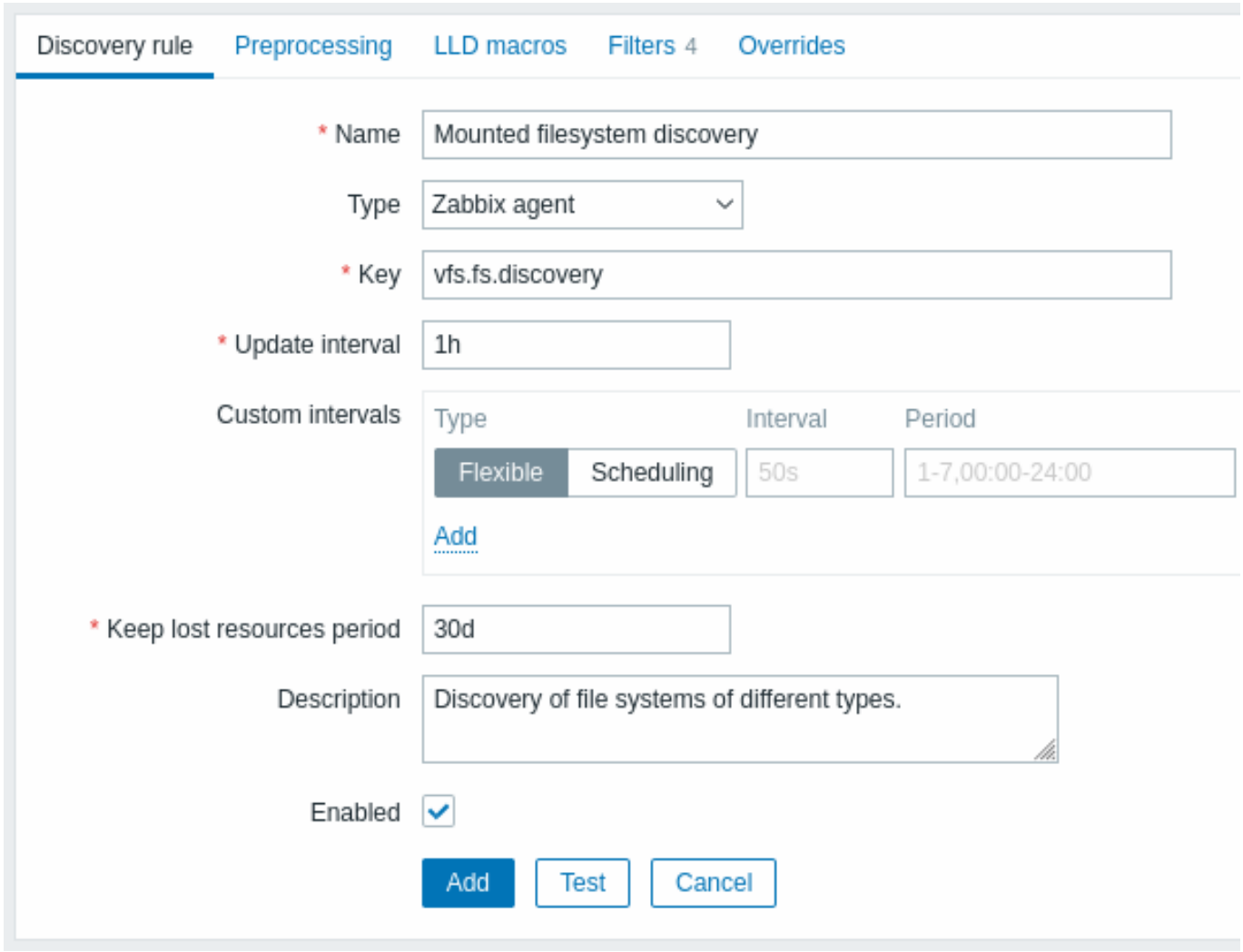
- Click on *Create discovery rule* in the upper right corner of the screen
- Fill in the discovery rule form with the required details

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 key. 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 and their types. 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).
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.

Parameter	Description
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.

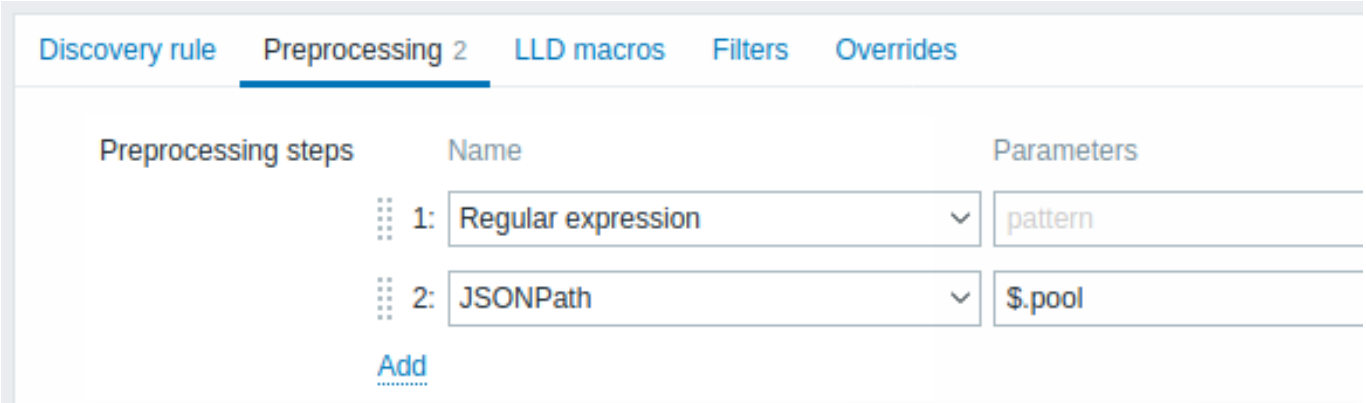
Note:
Discovery rule history is not preserved.

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		
	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.

Type	
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, the item will not become unsupported in case of failed preprocessing step and 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.
Custom scripts	
JavaScript	Enter JavaScript code in the block that appears when clicking in the parameter field or on *Open*. 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.

Type	
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.

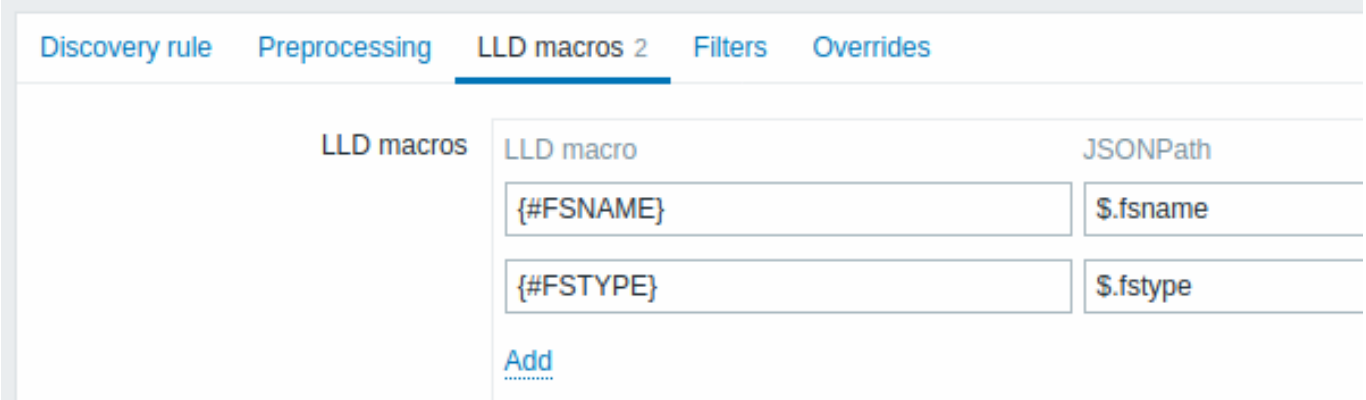
Note that if the discovery rule has been applied to the host via template then the content of this tab is read-only.

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:

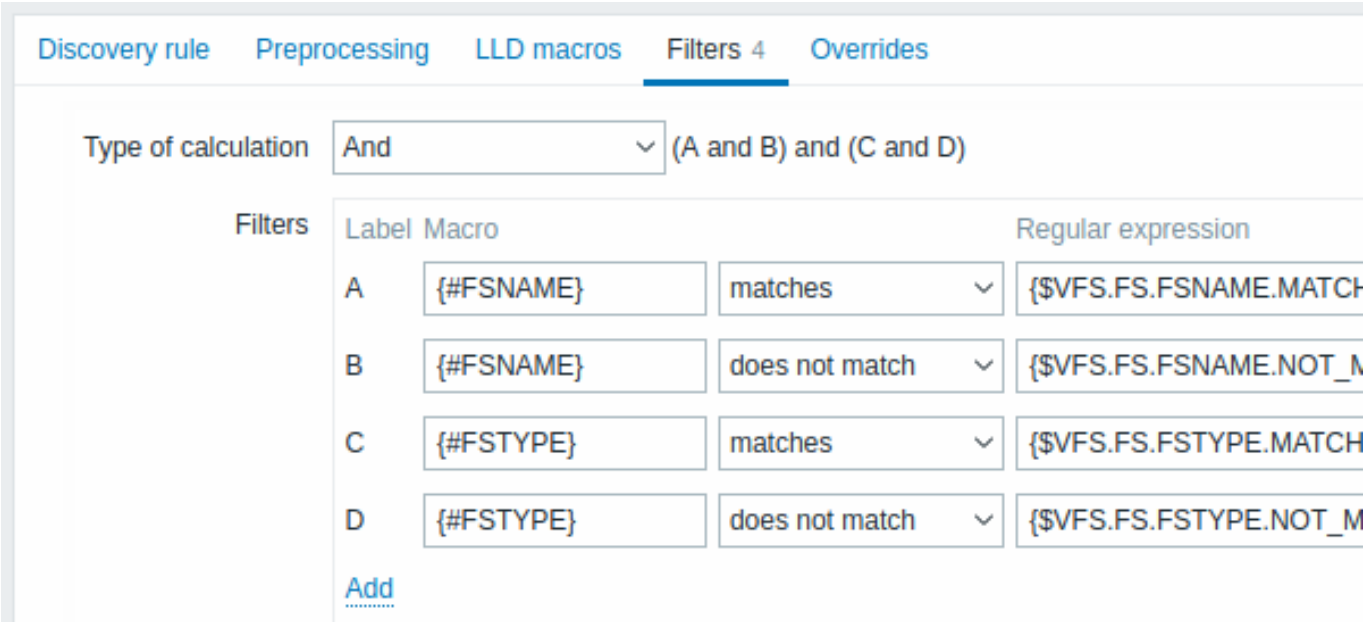


The extracted values can be used in discovered items, triggers, etc. Note that values will be extracted from the result of discovery and any preprocessing steps so far.

Parameter	Description
LLD macro	Name of the low-level discovery macro, using the following syntax: `{#MACRO}`.
JSONPath	Path that is used to extract LLD macro value from a LLD row, using JSONPath syntax. For example, `$.foo` will extract "bar" and "baz" from this JSON: `[{"foo":"bar"}, {"foo":"baz"}]` The values extracted from the returned JSON are used to replace the LLD macros in item, trigger, etc. prototype fields. JSONPath can be specified using the dot notation or the bracket notation. Bracket notation should be used in case of any special characters and Unicode, like `$['unicode + special chars #1']['unicode + special chars #2']`.

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:



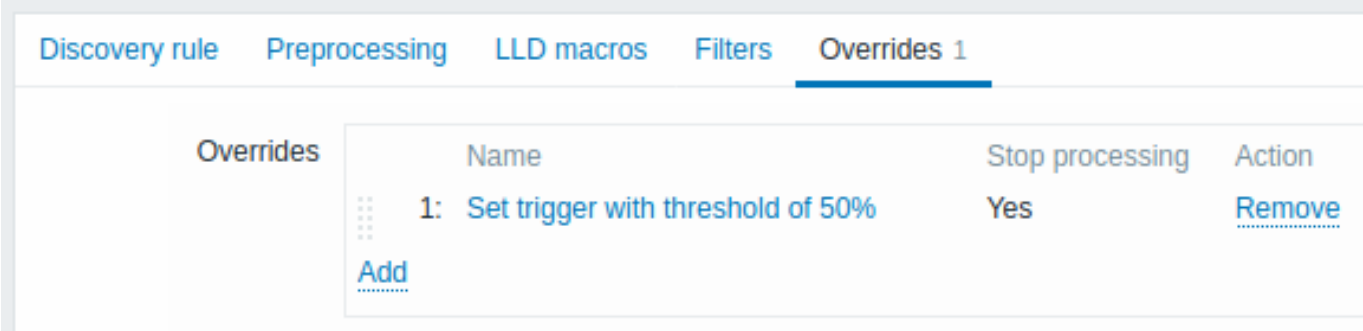
Parameter	Description
Type of calculation	The following options for calculating filters are available: **And** - all filters must be passed; **Or** - enough if one filter is passed; **And/Or** - uses *And* with different macro names and *Or* with the same macro name; **Custom expression** - offers the 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` {#FSDRIVETYPE} macro on Windows is supported since Zabbix **3.0.0**. *Exists* and *does not exist* operators allow to filter entities based on the presence or absence of the specified LLD macro in the response (supported since version 5.4.0). Defining several filters is supported since Zabbix **2.4.0**. 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. This flexible warning logic is supported since Zabbix 6.2.5.

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.

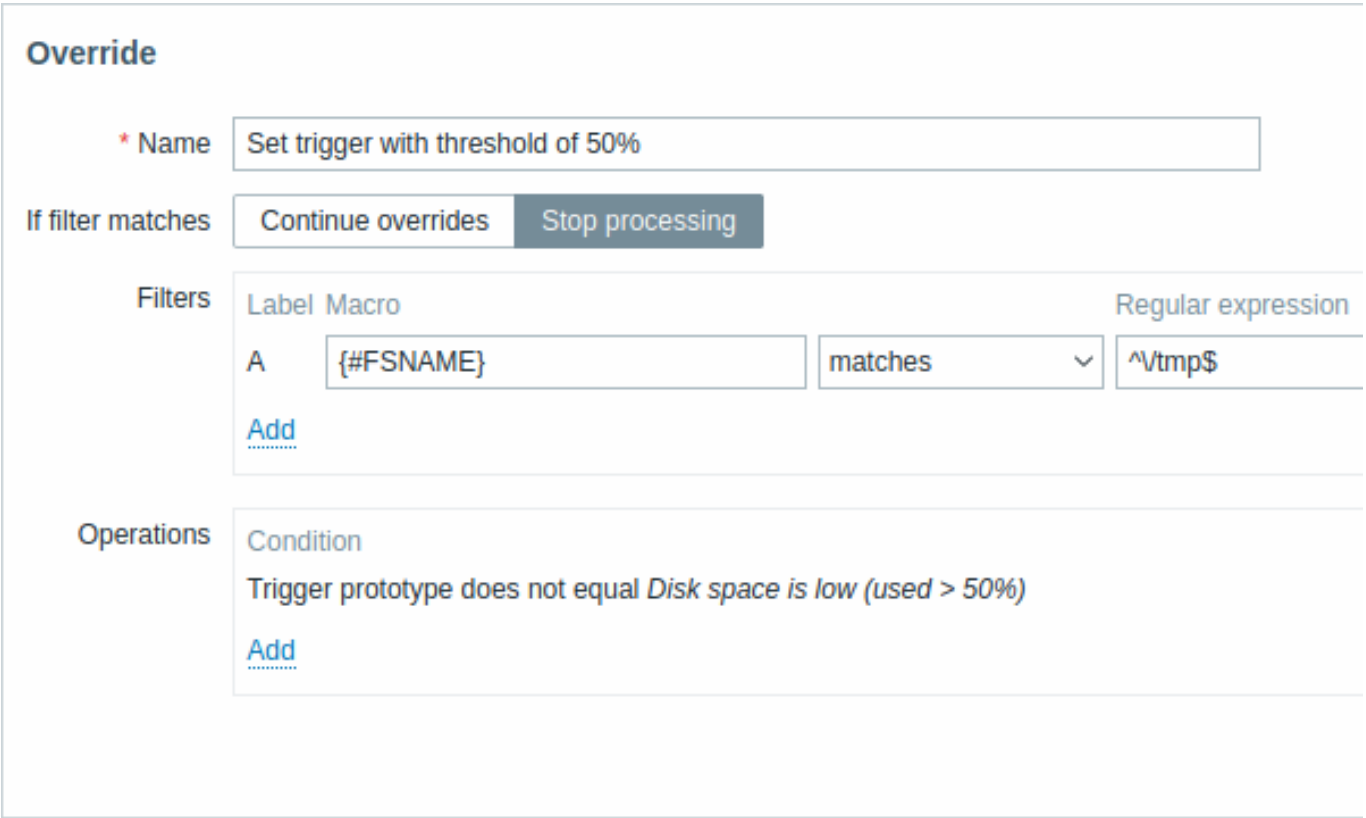
Attention:
Zabbix database in MySQL must be created as case-sensitive if file system names that differ only by case are to be discovered correctly.

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.



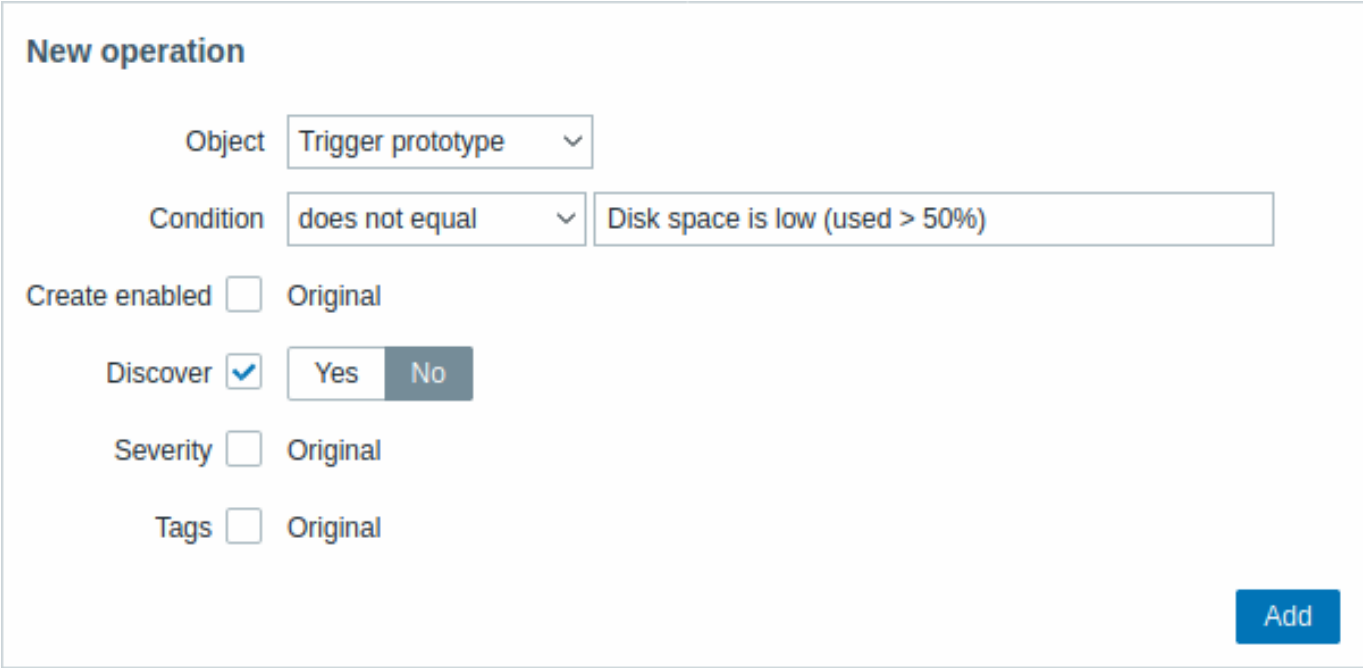
All mandatory parameters are marked with red asterisks.

Parameter	Description
Name	A unique (per LLD rule) override name.

Parameter	Description
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*	

Parameter		Description
	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.
	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.

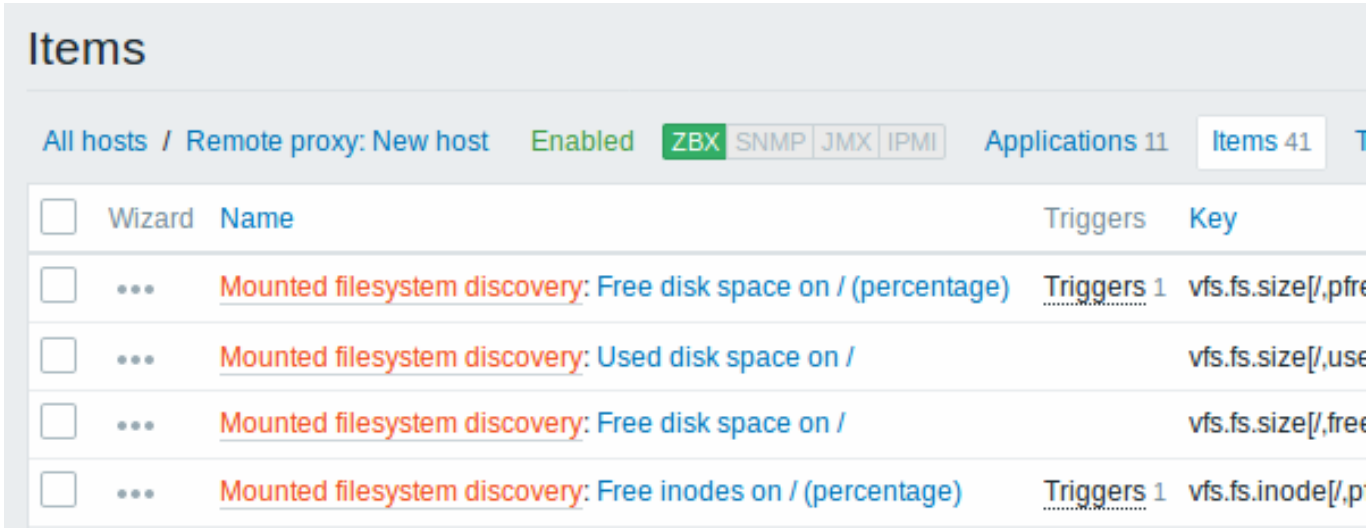
Parameter	Description
Object: *Host prototype*	
Create enabled	When the checkbox is marked, the buttons will appear, allowing to override original host prototype settings: *Yes* - the host will be created in an enabled state. *No* - the host will be created in a disabled state.
Discover	When the checkbox 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.
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.

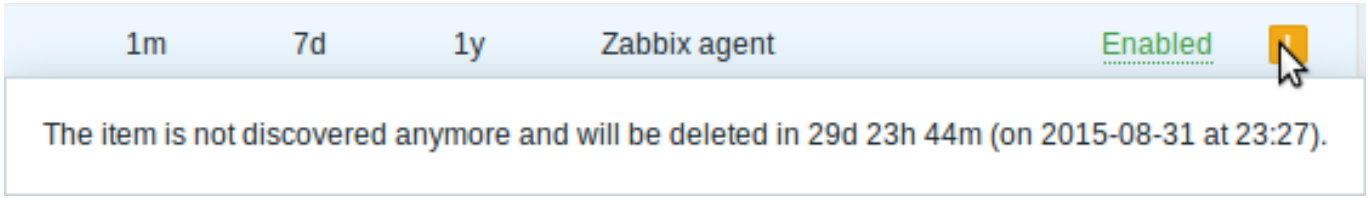
Discovered entities The screenshots below illustrate how discovered items, triggers, and graphs look like in the host's configuration. Discovered entities are prefixed with an orange link to a discovery rule they come from.



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.

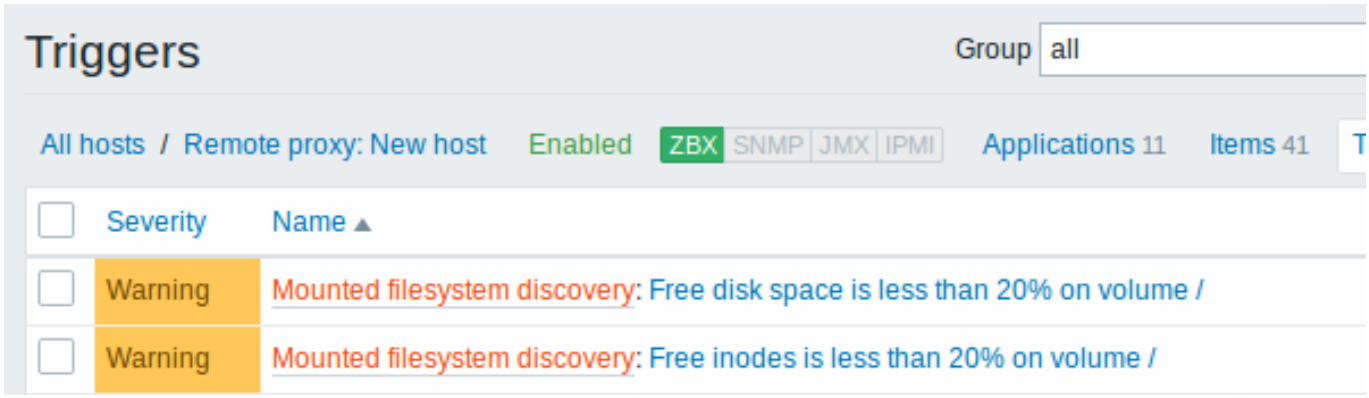
Items (similarly, triggers and graphs) created by a low-level discovery rule will be deleted automatically if a discovered entity (file system, interface, etc) stops being discovered (or does not pass the filter anymore). In this case the items, triggers and graphs will be deleted after the days defined in the *Keep lost resources period* field pass.

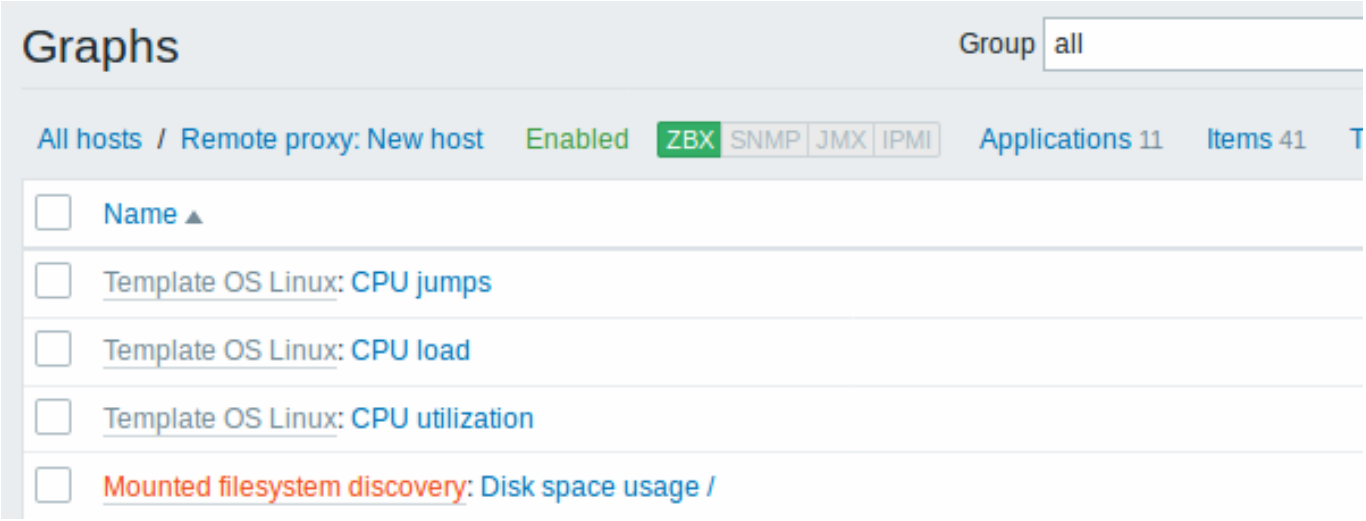
When discovered entities become 'Not discovered anymore', a lifetime indicator is displayed in the item list. Move your mouse pointer over it and a message will be displayed indicating how many days are left until the item is deleted.



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.

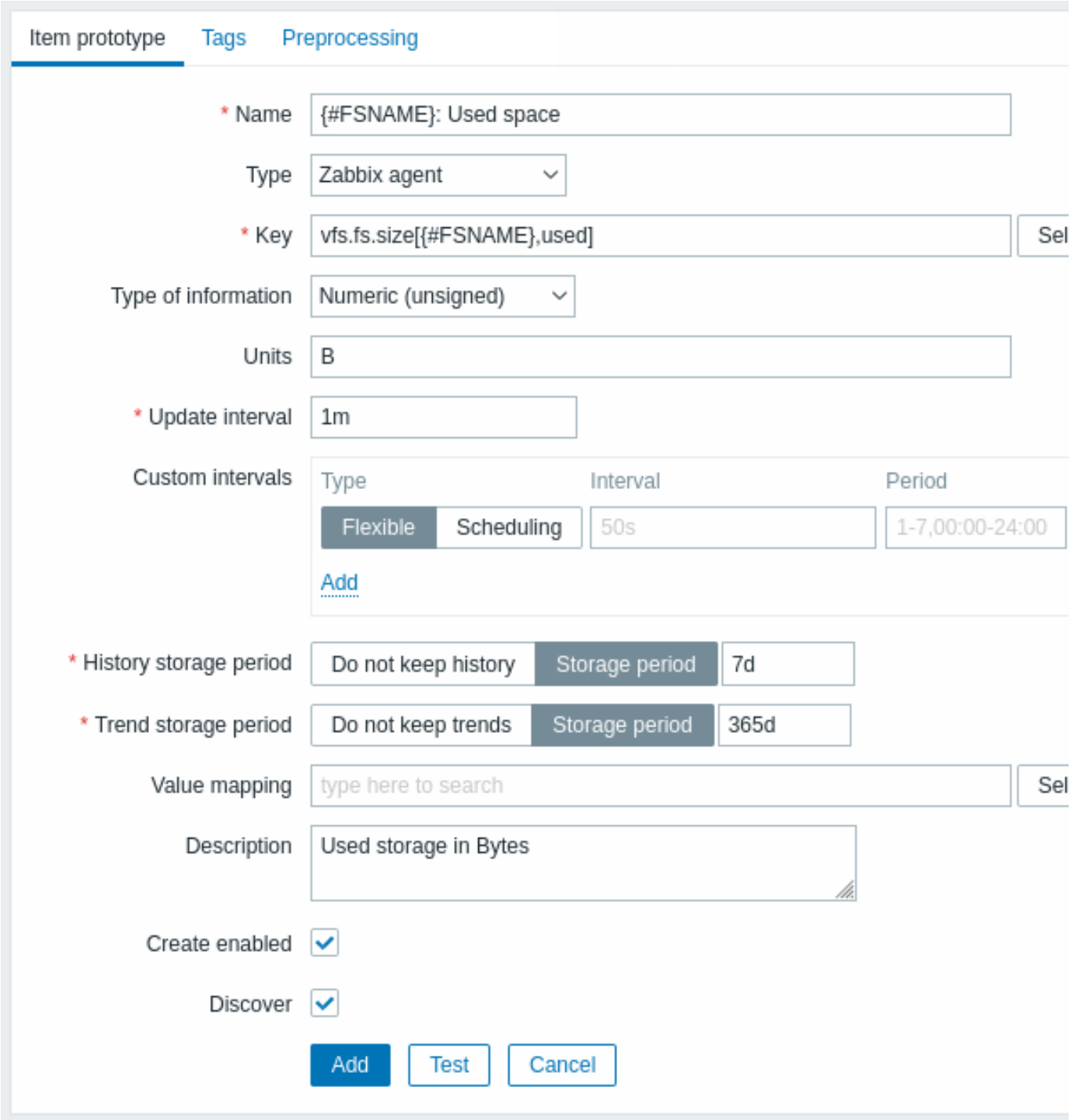
Note:
You don't have to use macro names FSNAME/FSTYPE with custom LLD rules, you are free to use whatever names you like. In case JSONPath is used then LLD row will be an array element that can be an object, but it can be also another array or a value.

Note that, if using a user parameter, the return value is limited to 512 KB. 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

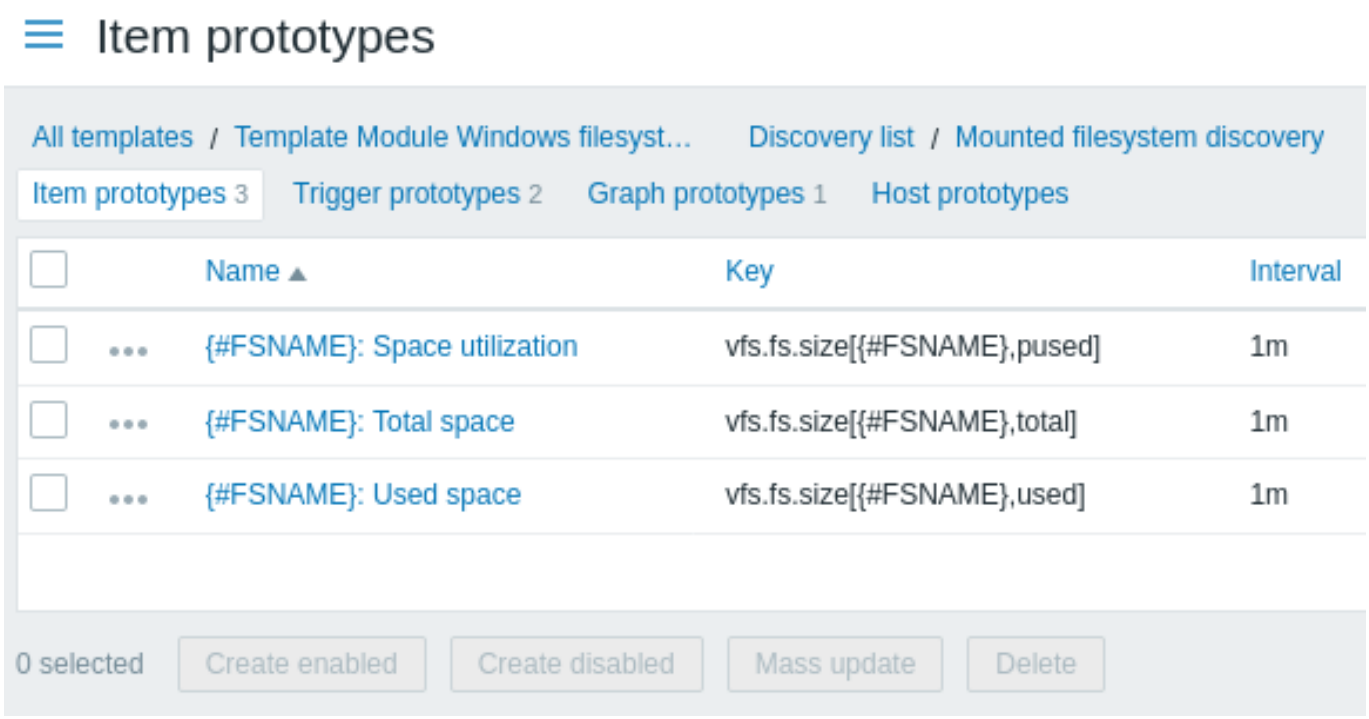
mixed with text are not supported.

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:

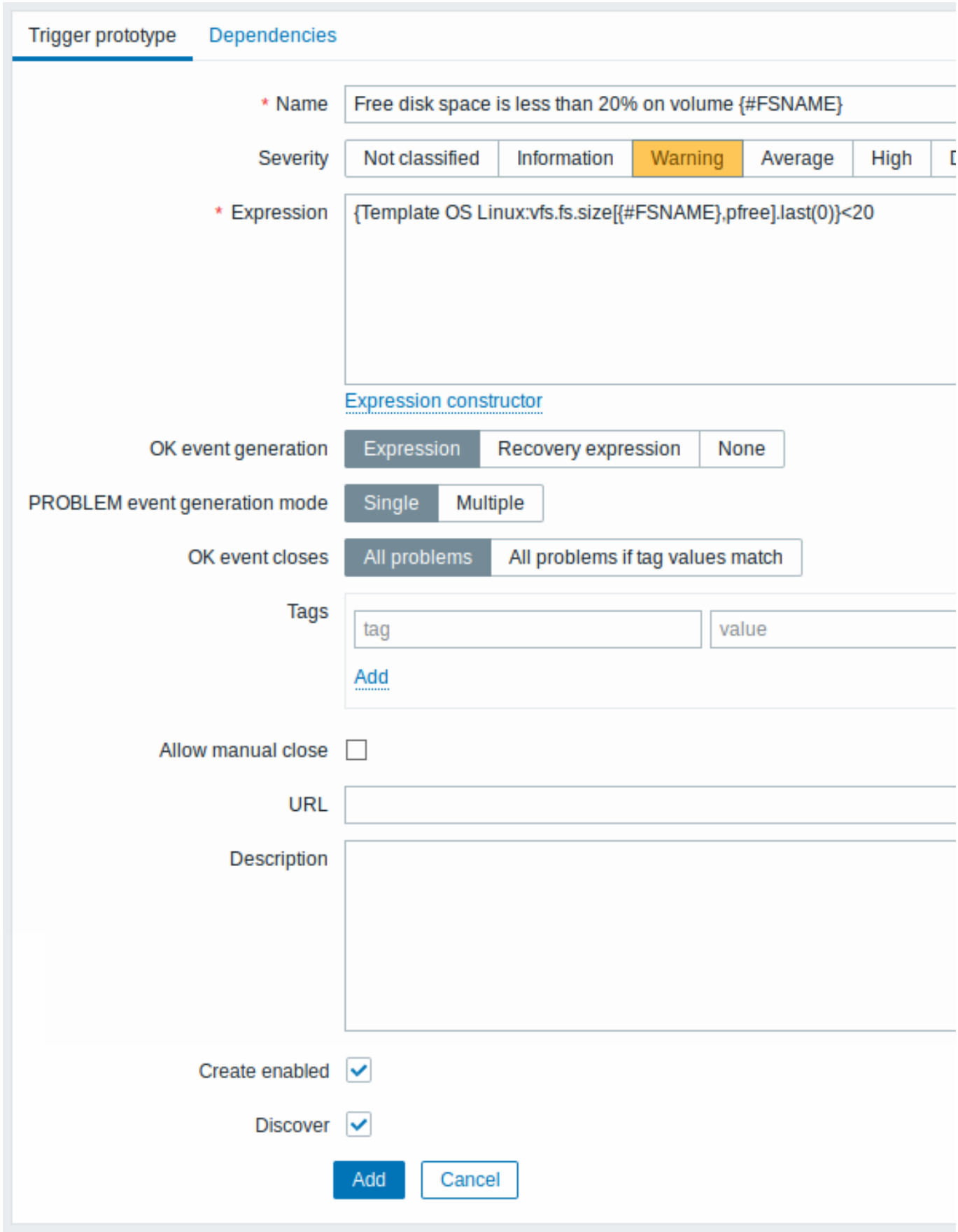


Click on the three-dot icon to open the menu for the specific item prototype with these options:
 - *Create trigger prototype* - create a trigger prototype based on this item prototype - *Trigger prototypes* - click to see a list with links to already-configured trigger prototypes of this item prototype - *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:



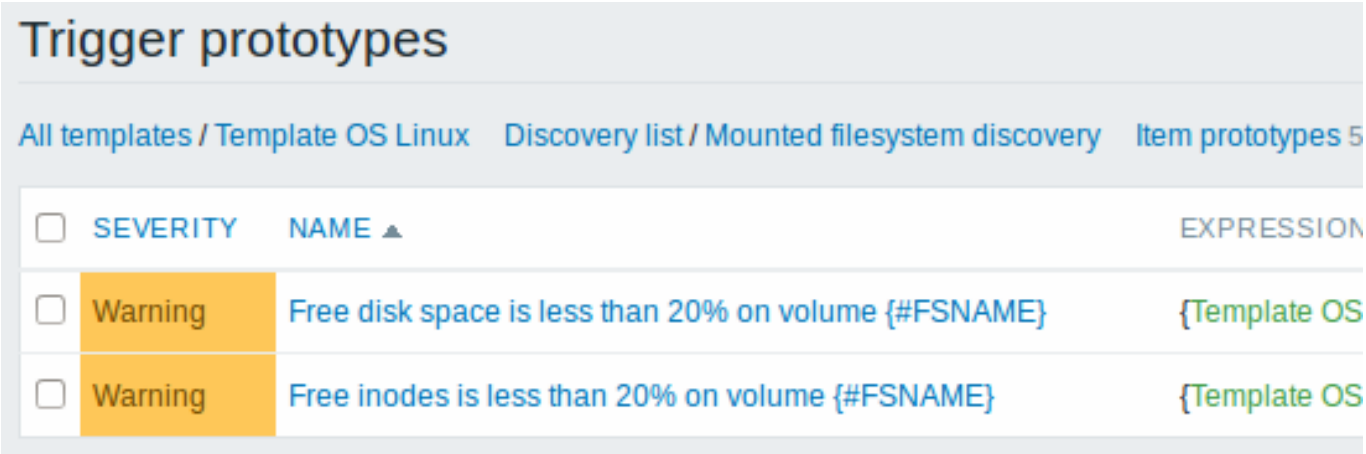
Attributes that are specific for trigger prototypes:

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.

Parameter	Description
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.

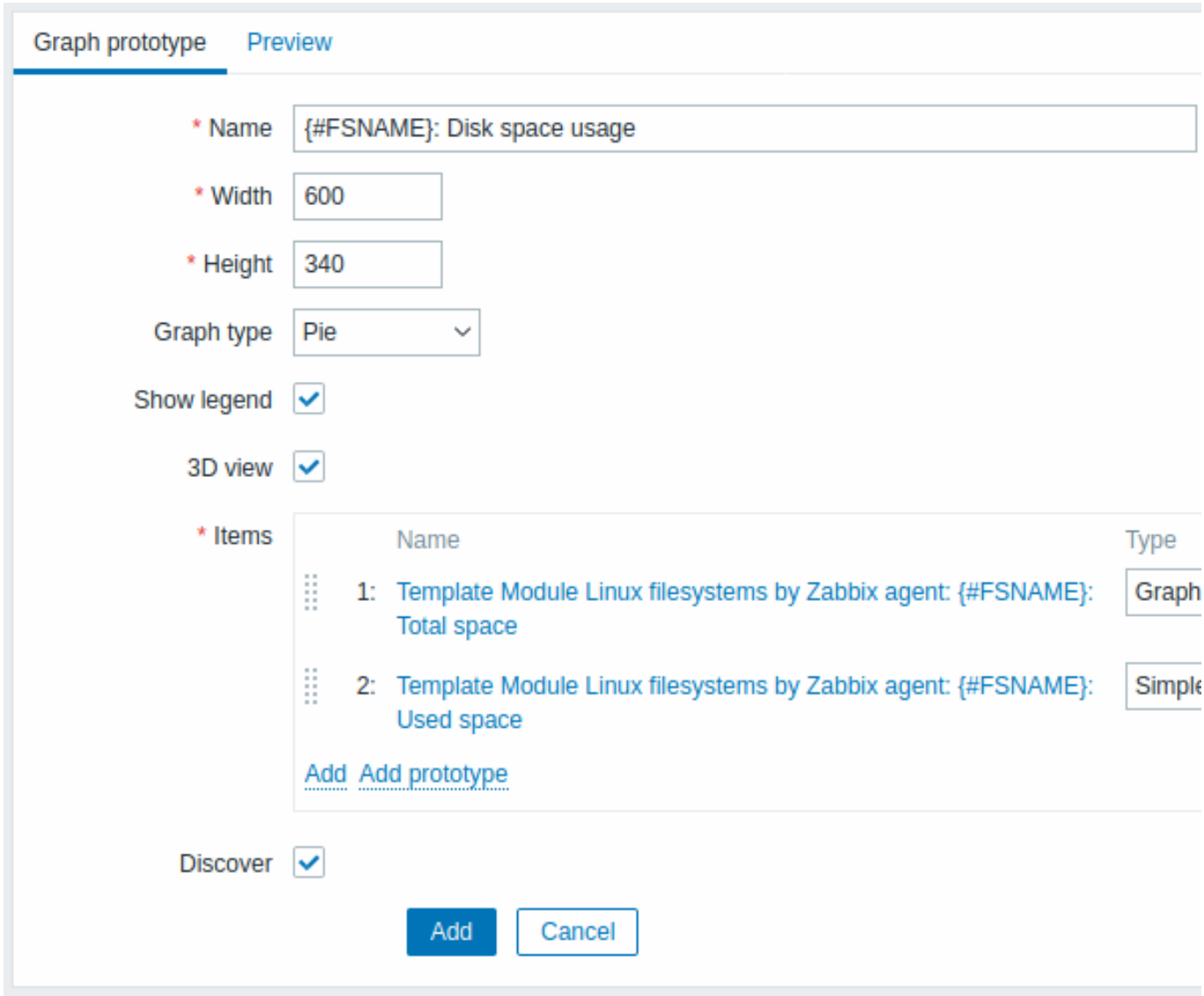
When real triggers are created from the prototypes, there may be a need to be flexible as to what constant ('20' in our example) is used for comparison in the expression. See how user macros with context can be useful to accomplish such flexibility.

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.



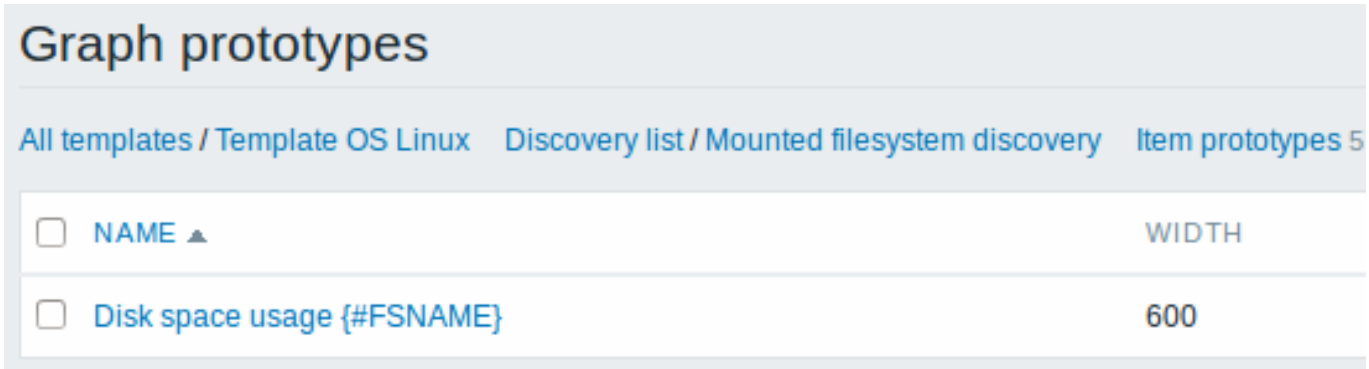
3 Graph prototypes

We can create graph prototypes, too:

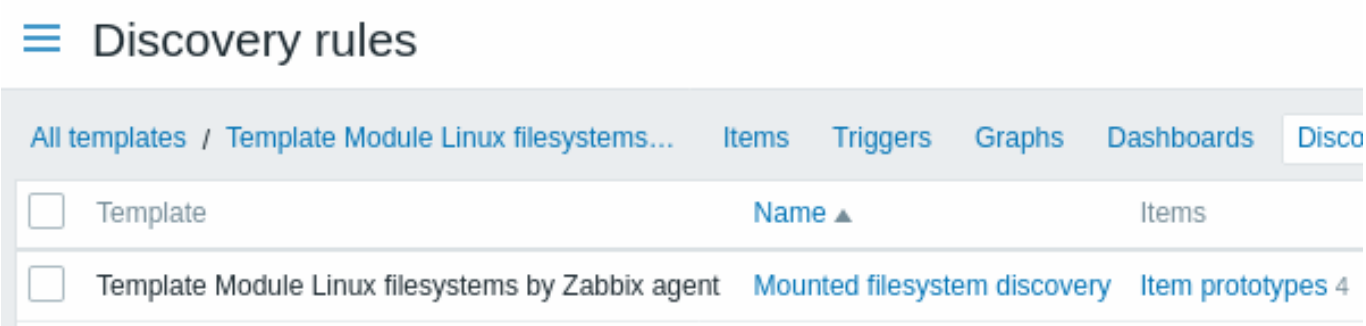


Attributes that are specific for graph prototypes:

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.



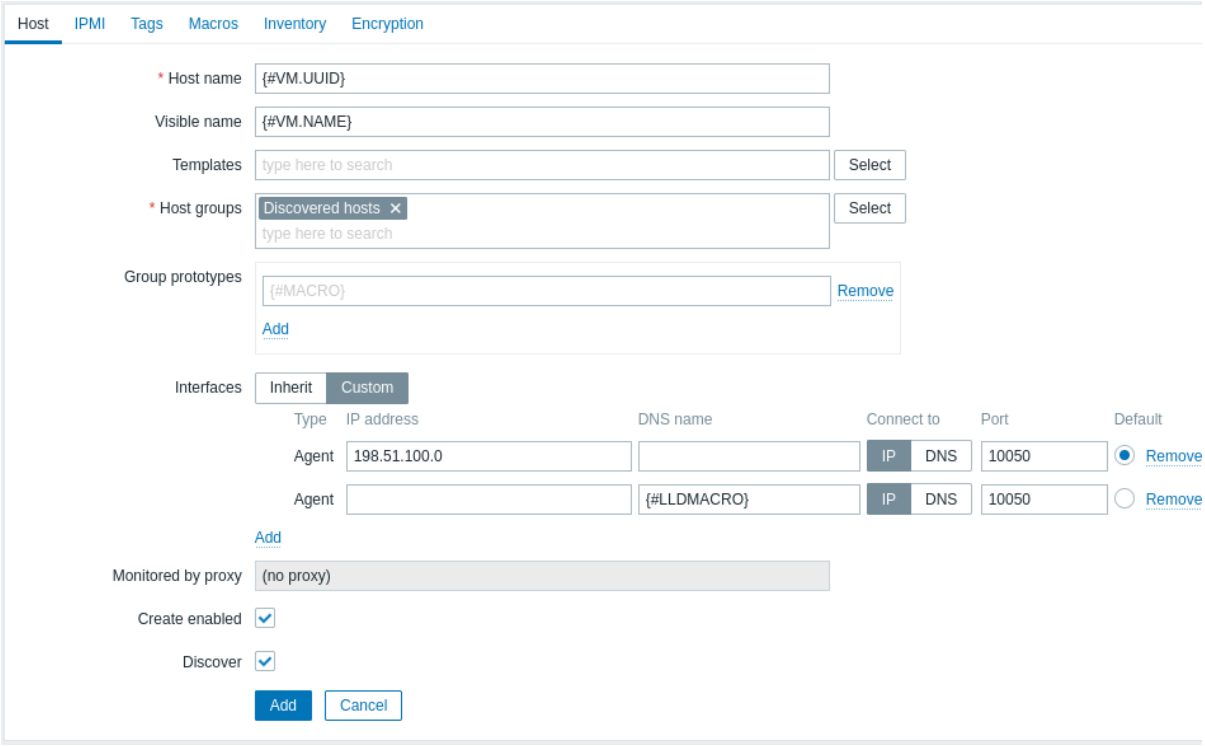
4 Host prototypes

Host prototypes are like blueprints for hosts to be created with a low-level discovery rule. Prototypes, before becoming discovered, cannot have their own items and triggers, other than those from the linked templates.

Host prototype configuration Host prototypes are configured under discovery rules.

To create a host prototype:

1. Navigate to *Configuration → Hosts*.
2. Click on *Discovery* for the required host to navigate to the list of low-level discovery rules configured for the host.
3. Click on *Host prototypes* for the required discovery rule.
4. Click on the *Create host prototype* button in the upper right corner.



In the new window, specify host prototype parameters. Host prototypes have the same parameters as regular hosts, with the following exceptions:

- *Host name* must contain at least one low-level discovery macro to ensure that hosts created from the prototype have unique host names.
- *Group prototypes* allow specifying host group prototypes by using low-level discovery macros.
- *Interfaces* defines whether discovered hosts should inherit the IP of a host the discovery rule belongs to (default) or get custom interfaces.
- *Create enabled* sets the status of discovered hosts, if the checkbox is unmarked the hosts will be created, but disabled.
- *Discover* - if the checkbox is unmarked, the hosts will not be created from the host prototype, unless this setting is overridden in the discovery rule.

Note:
Value maps are not supported for host prototypes.

Low-level discovery macros can be used for host name, visible name, host group prototype, interfaces, tag values, or values of host prototype user macros.

Host interfaces

To add custom interfaces, switch the *Interface* selector from *Inherit* to *Custom* mode. Then press Add and select the required interface type from the menu.

A host prototype may have any of the supported interface types: Zabbix agent, SNMP, JMX, IPMI.

Low-level discovery macros and user macros are supported.

If several custom interfaces are specified, use the *Default* column to specify the primary interface.

Notes:

- If *Custom* is selected, but no interfaces have been specified, the hosts will be created without interfaces.
- If *Inherit* is selected for a host prototype that belongs to a template, discovered hosts will inherit the interface of a host to which the template is linked to.

Warning:
A host will not be created if a host interface contains incorrect data.

Discovered hosts

In the host list, discovered hosts are prefixed with the name of the discovery rule that created them.

The following discovered host parameters are customizable:

- *Templates* - it is possible to link additional templates to these hosts or unlink manually added templates. Templates inherited from a host prototype cannot be unlinked.
- *Description*.
- *Status* - a host can be manually enabled/disabled.
- *Tags* - host tags can be added manually, alongside the tags inherited from the host prototype. Neither manual nor inherited tags can be duplicate, that is, have 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* - host macros can be added manually, alongside the macros inherited from the host prototype. For inherited macros, it is possible to change macro value and type on the host level.
- Host inventory fields.

Other parameters are inherited from the host prototype as read-only.

Discovered hosts can be deleted manually. Hosts that are no longer discovered will be deleted automatically based on the *Keep lost resources period (in days)* value of the discovery rule.

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, because return values are processed without being stored in a database. There's also no limit for custom low-level discovery rules, however, if it is intended to acquire custom LLD data using a user parameter, then the user parameter return value limit applies (512 KB).

If data has to go through Zabbix proxy it has to store this data in database so 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, mountpoint type, filesystem size and inode statistics).

To do that, you may use a combination of:

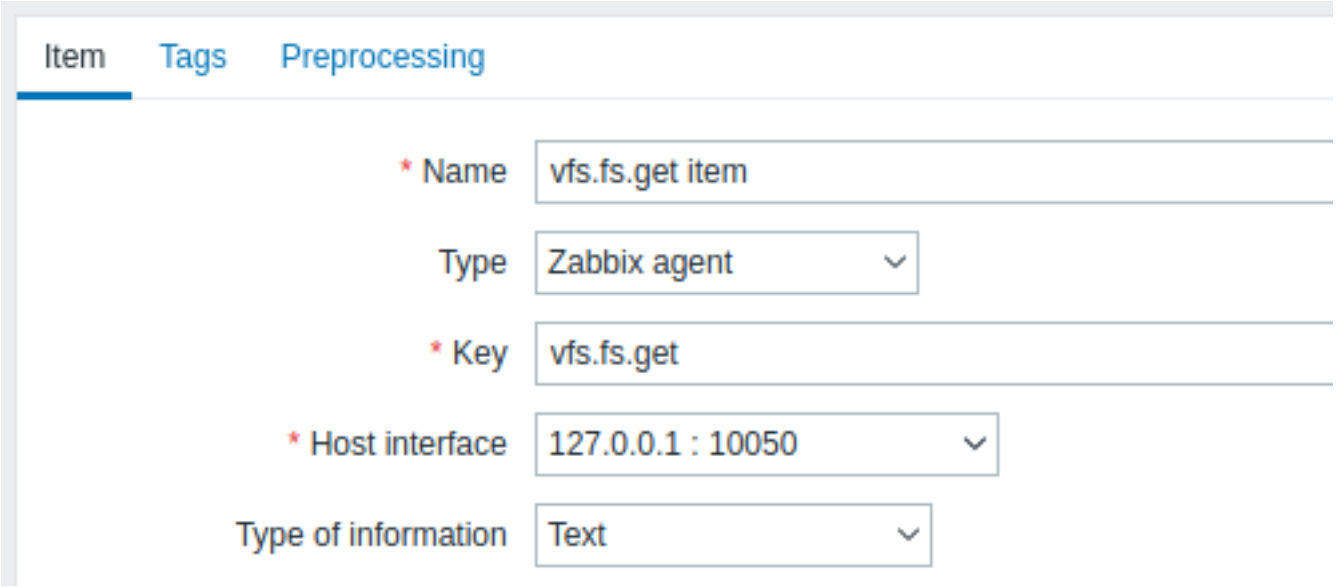
- the `vfs.fs.get` agent item as the master item
- dependent low-level discovery rule and item prototypes

Configuration

Master item

Create a Zabbix agent item using the following key:

`vfs.fs.get`



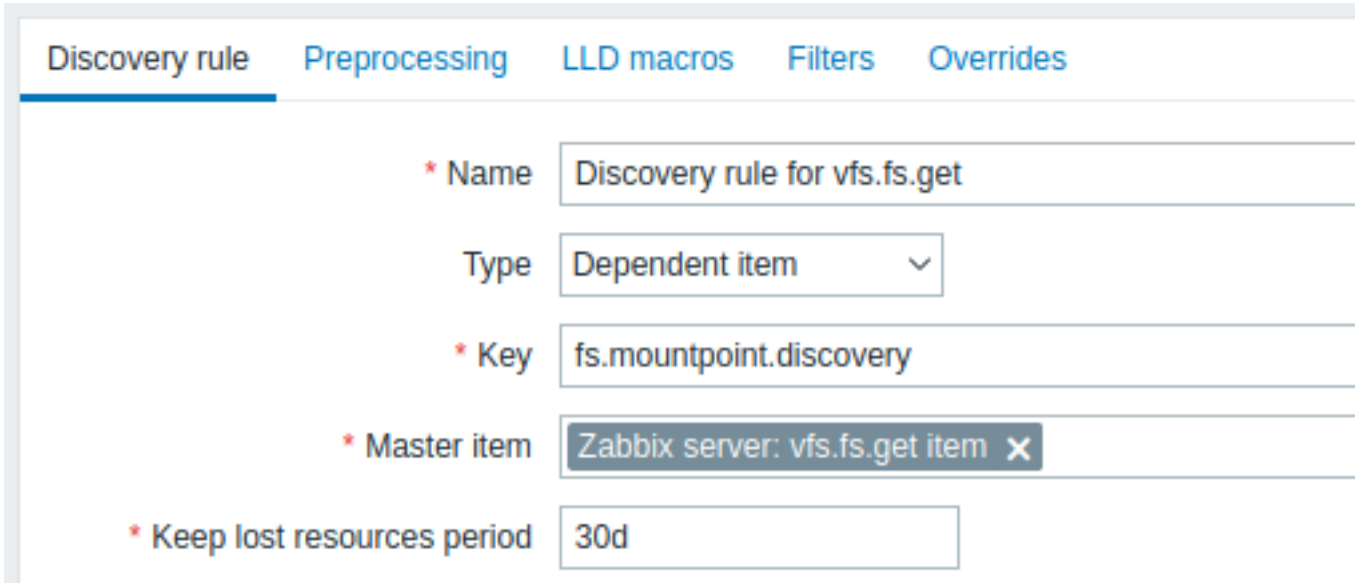
Set the type of information to "Text" for possibly big JSON data.

The data returned by this item will contain something like the following for a mounted filesystem:

```
{
  "fsname": "/",
  "fstype": "rootfs",
  "bytes": {
    "total": 1000,
    "free": 500,
    "used": 500,
    "pfree": 50.00,
    "pused": 50.00
  },
  "inodes": {
    "total": 1000,
    "free": 500,
    "used": 500,
    "pfree": 50.00,
    "pused": 50.00
  }
}
```

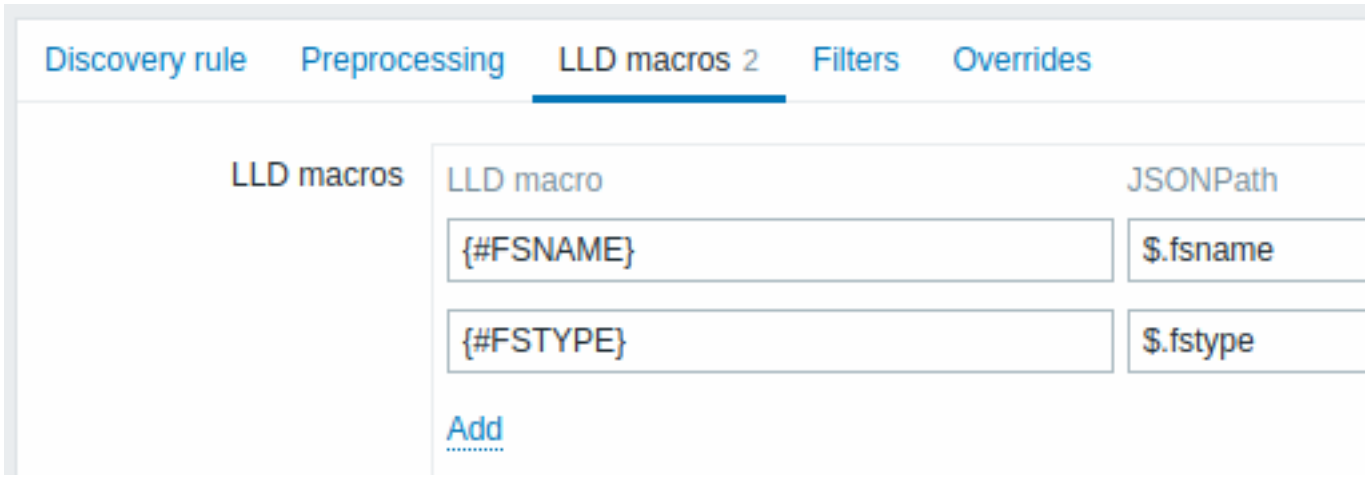
Dependent LLD rule

Create a low-level discovery rule as "Dependent item" type:



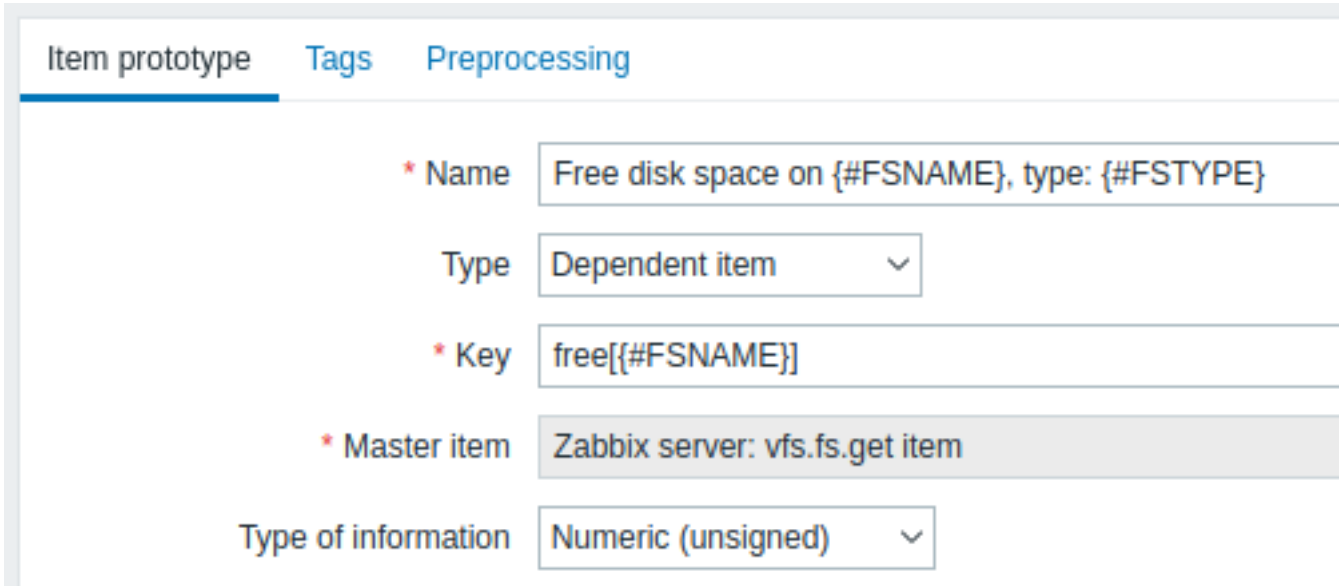
As master item select the `vfs.fs.get` item we created.

In the "LLD macros" tab define custom macros with the corresponding JSONPath:



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}`

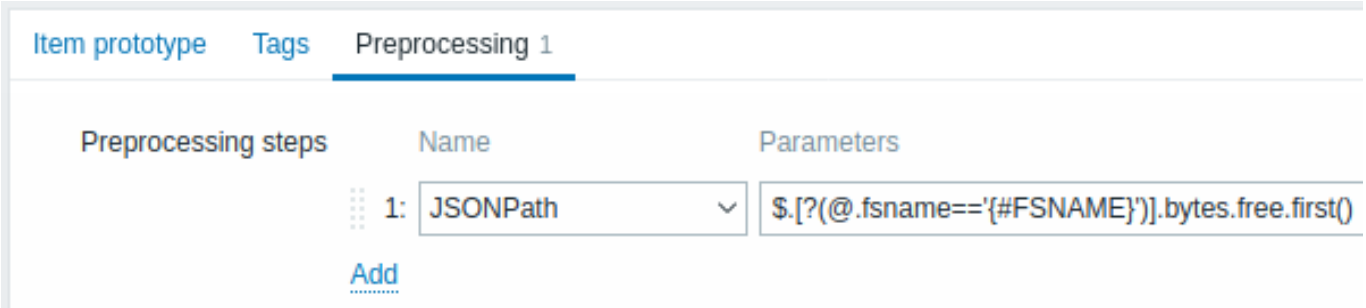
- *Key*: Free[{#FSNAME}]

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.

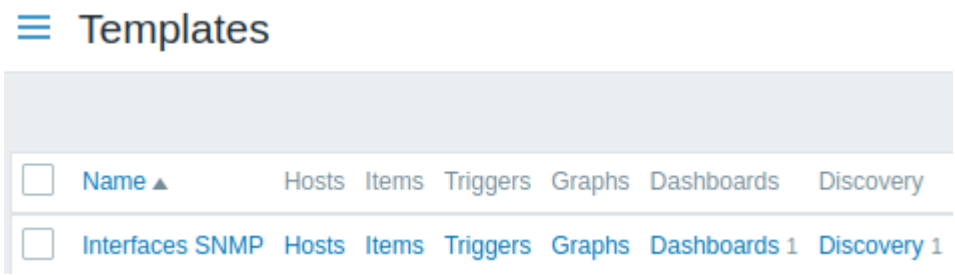
Item key

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.

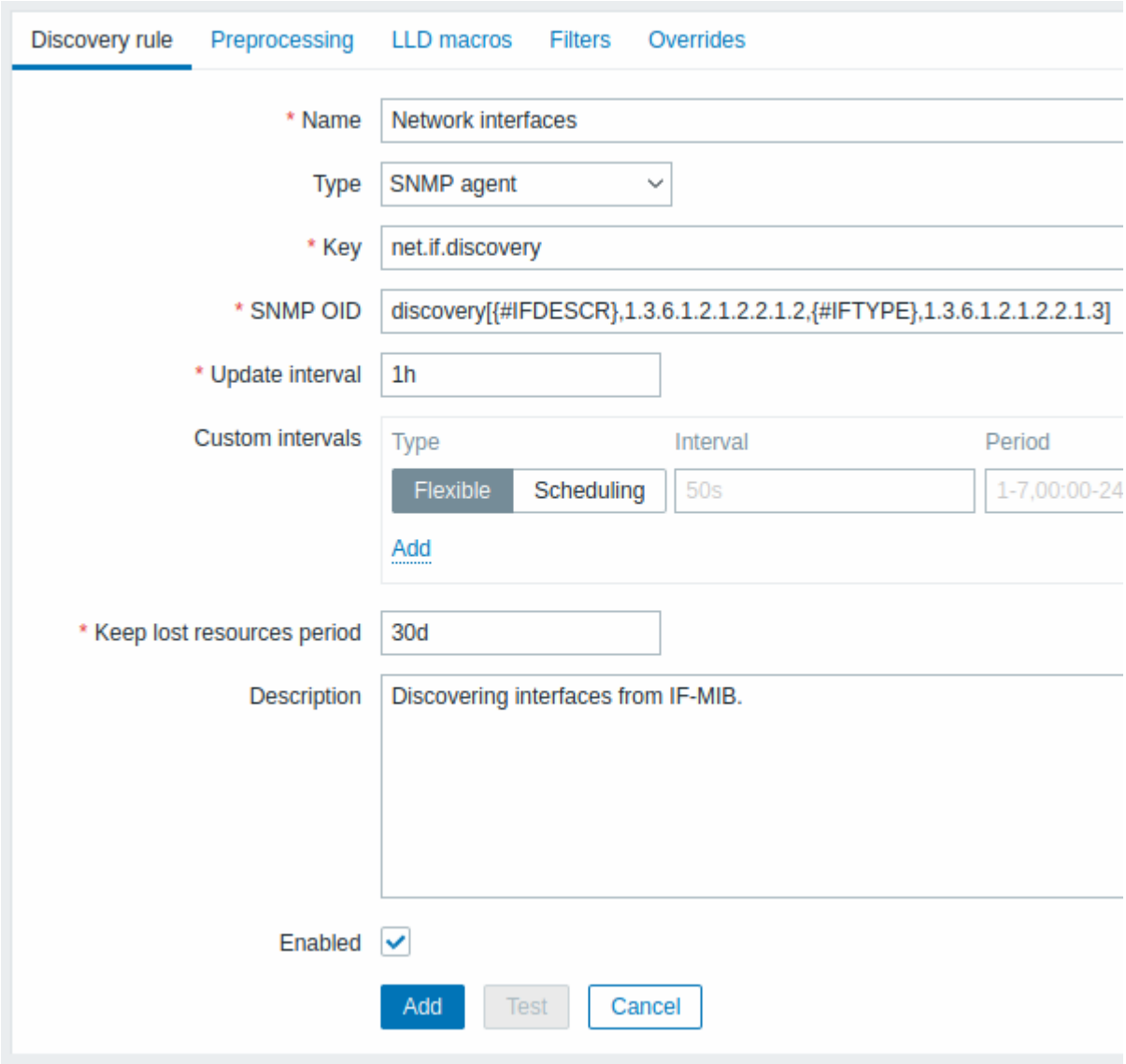
Discovery of SNMP OIDs is supported since Zabbix server/proxy 2.0.

To configure the discovery rule, do the following:

- Go to: *Configuration* → *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"

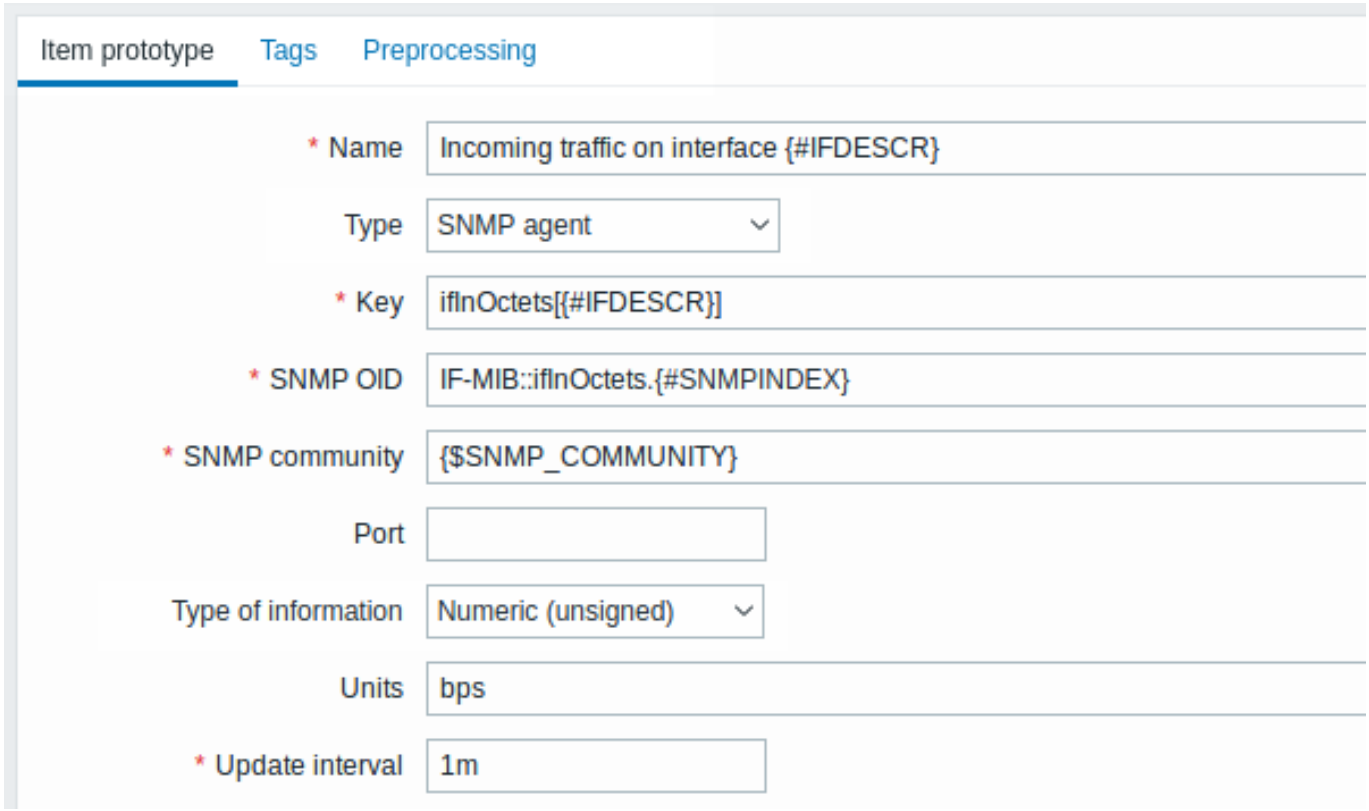
ifAlias.1 "eth0"
ifAlias.2 "eth1"
ifAlias.3 "eth2"
ifAlias.5 "eth4"
```

Then in this case SNMP discovery `discovery[{#IFDESCR}, ifDescr, {#IFALIAS}, ifAlias]` will return the following structure:

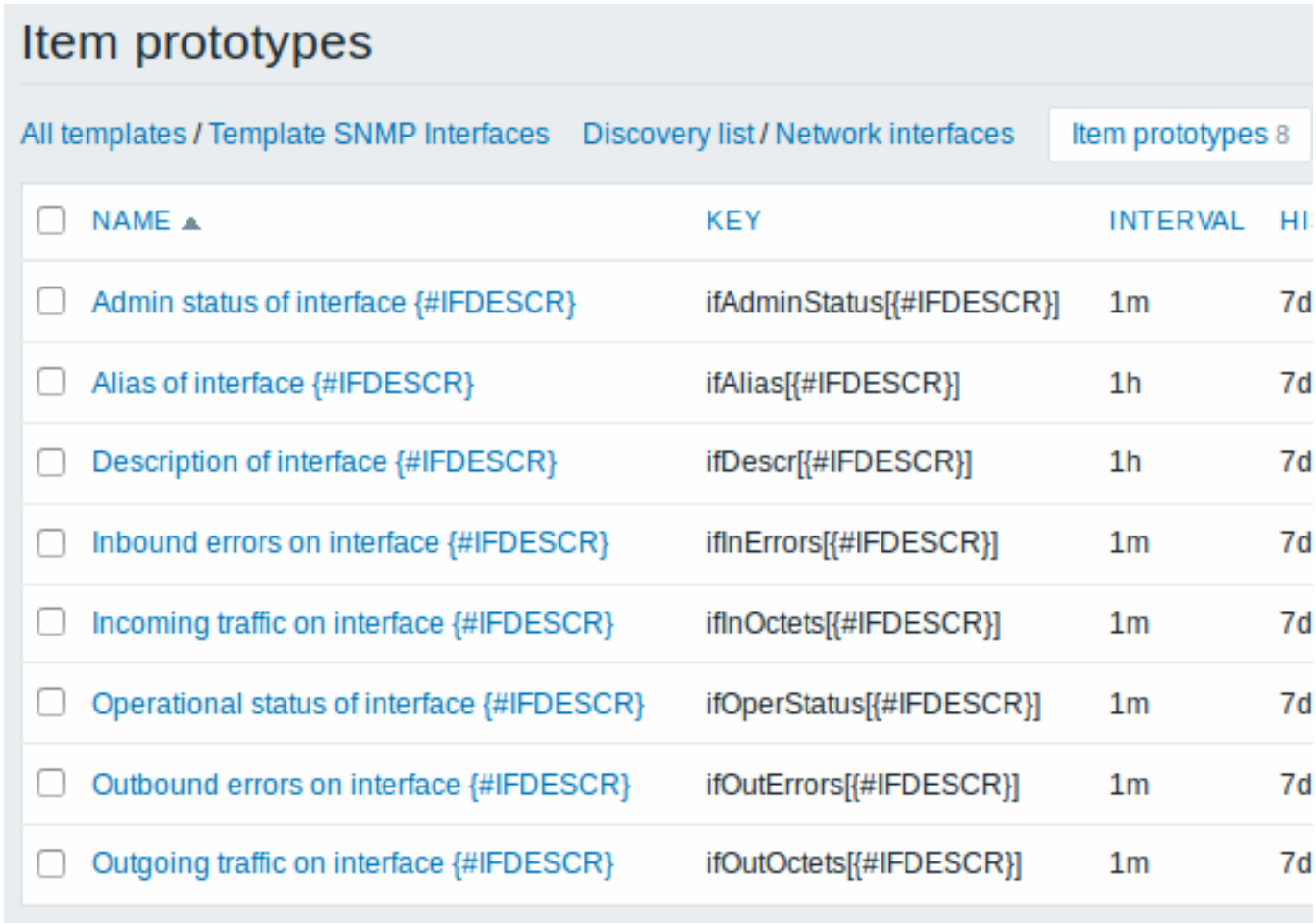
```
[
    {
        "{#SNMPINDEX}": 1,
        "{#IFDESCR}": "Interface #1",
        "{#IFALIAS}": "eth0"
    },
    {
        "{#SNMPINDEX}": 2,
        "{#IFDESCR}": "Interface #2",
        "{#IFALIAS}": "eth1"
    },
    {
        "{#SNMPINDEX}": 3,
        "{#IFALIAS}": "eth2"
    },
    {
        "{#SNMPINDEX}": 4,
        "{#IFDESCR}": "Interface #4"
    },
    {
        "{#SNMPINDEX}": 5,
        "{#IFALIAS}": "eth4"
    }
]
```

Item prototypes

The following screenshot illustrates how we can use these macros in item prototypes:



Again, creating as many item prototypes as needed:



Trigger prototypes

The following screenshot illustrates how we can use these macros in trigger prototypes:

Trigger prototype | Tags | Dependencies

* Name: Interface {#IFDESCR}: Link down

Event name: Interface {#IFDESCR}: Link down

Operational data: Current state: {ITEM.LASTVALUE1}

Severity: Not classified | Information | Warning | Average | High | Disaster

* Problem expression:
```
{$IFCONTROL:"{#IFNAME}"}=1 and ({Template Module
Interfaces Simple SNMPv1:net.if.status[ifOperStatus.
{#SNMPINDEX}].last()}=2 and {Template Module Interfaces
Simple SNMPv1:net.if.status[ifOperStatus.
{#SNMPINDEX}].diff()}=1)
```
Expression constructor

OK event generation: Expression | Recovery expression | None

* Recovery expression:
```
{Template Module Interfaces Simple
SNMPv1:net.if.status[ifOperStatus.{#SNMPINDEX}].last()}<>2
```
Expression constructor

PROBLEM event generation mode: Single | Multiple

OK event closes: All problems | All problems if tag values match

Allow manual close: ☑

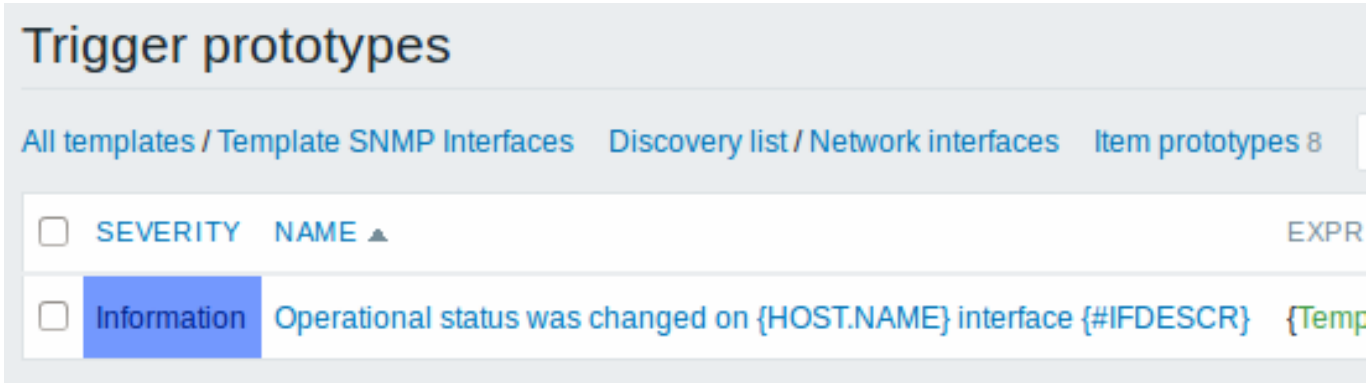
URL:

Description:
```
This trigger expression works as follows:
1. Can be triggered if operations status is down.
2. {$IFCONTROL:"{#IFNAME}"}=1 - user can redefine Context macro to value -
0. That marks this interface as not important. No new trigger will be fired if this
interface is down.
3. {TEMPLATE_NAME:METRIC.diff()}=1) - trigger fires only if operational
status was up(1) sometime before. (So, do not fire 'ethernal off' interfaces.)

WARNING: if closed manually - won't fire again on next poll, because of .diff.
```

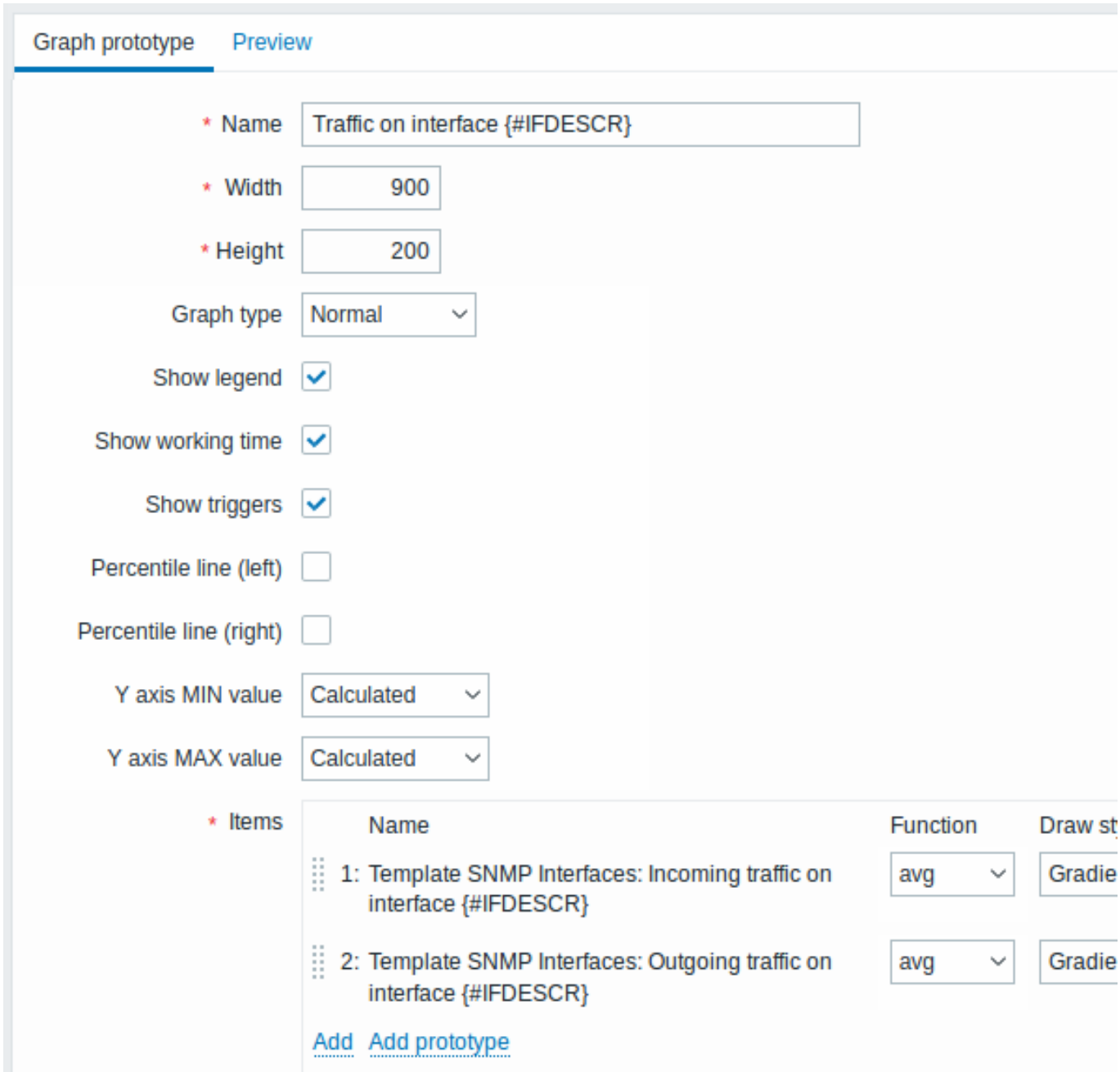
Create enabled: ☑

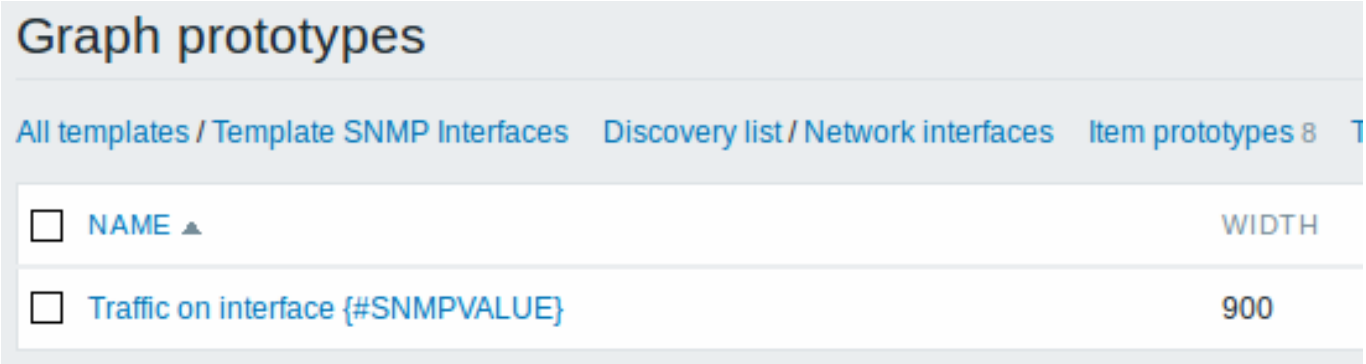
Discover: ☑



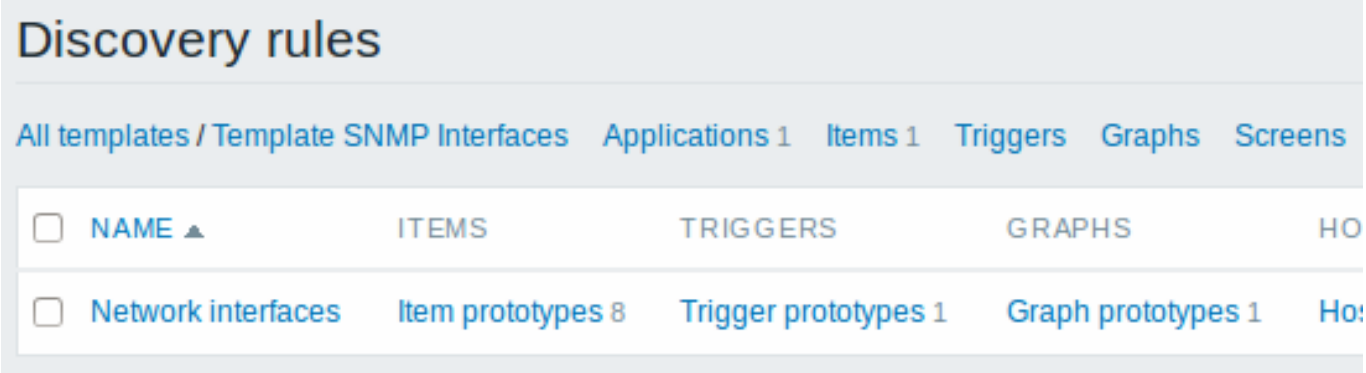
Graph prototypes

The following screenshot illustrates how we can use these macros in graph prototypes:



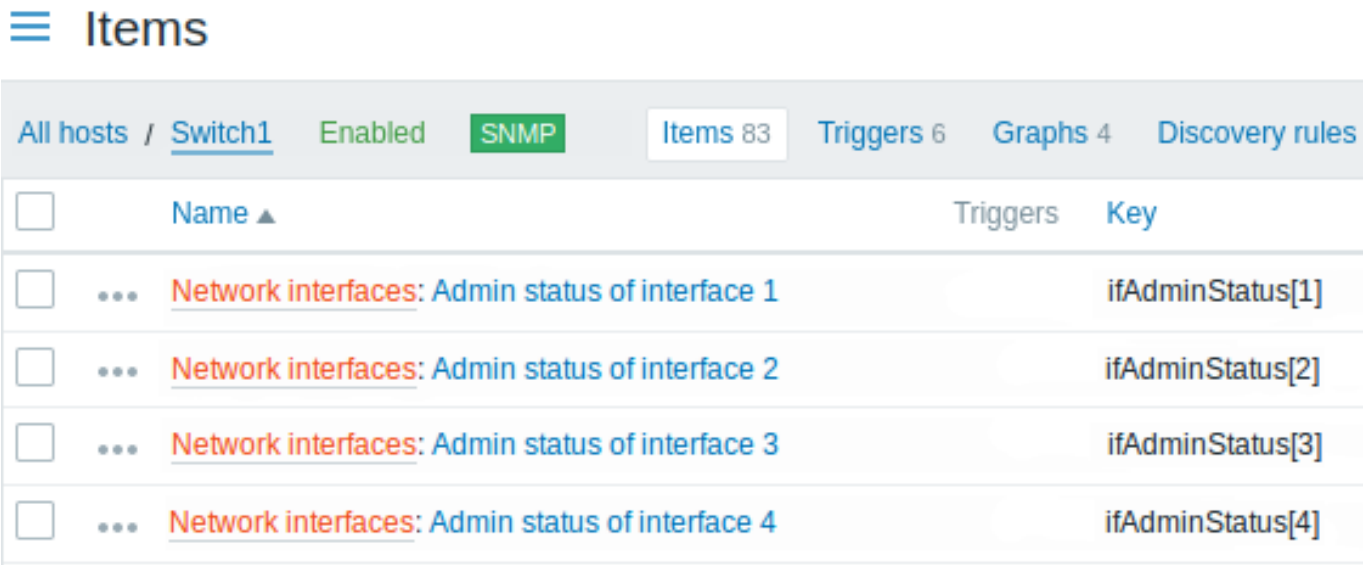


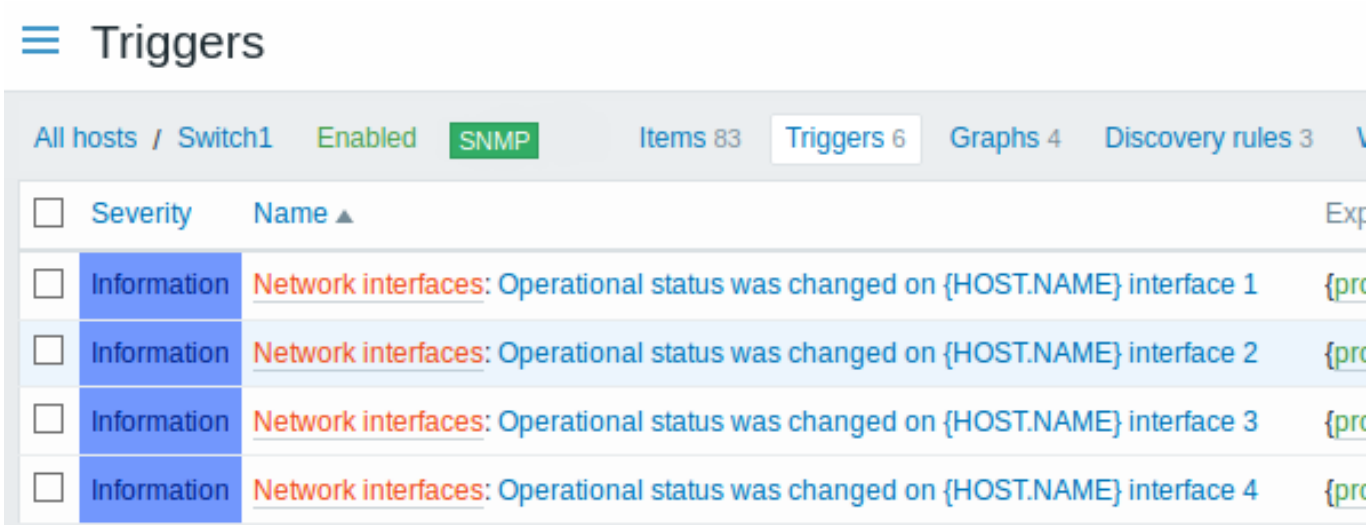
A summary of our discovery rule:

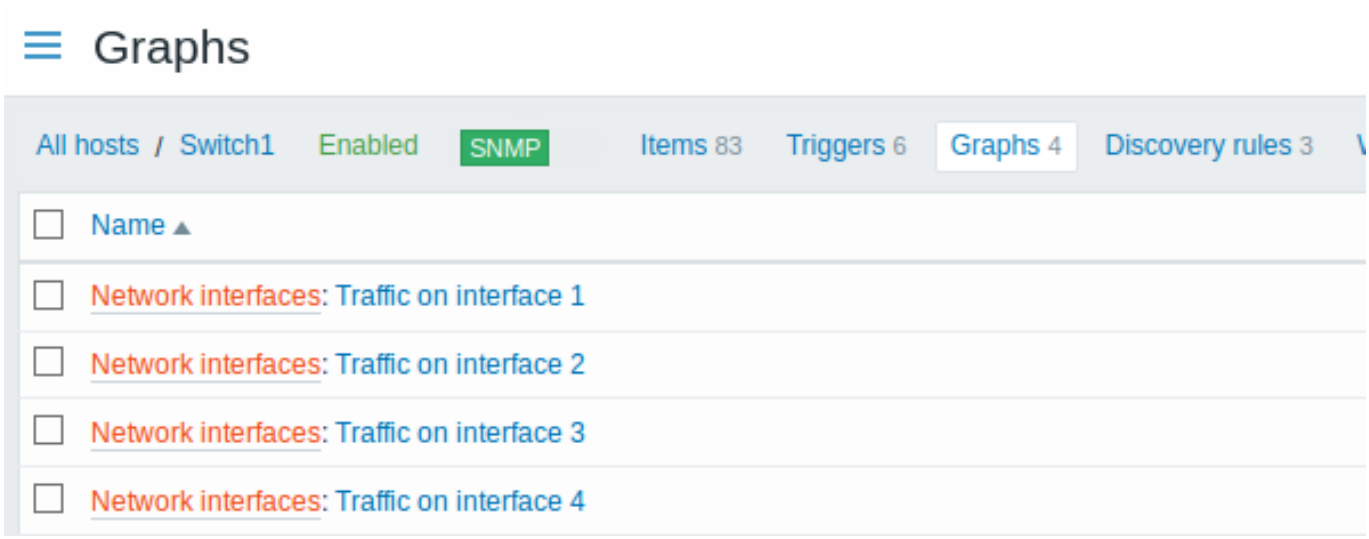


Discovered entities

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.







5 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>]			

Item key			
	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	Examples: → jmx.discovery - retrieve all JMX MBean attributes → jmx.discovery[beans] - retrieve all JMX MBeans → jmx.discovery[attribu - retrieve all garbage collector attributes → jmx.discovery[beans, - 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

Item key
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	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.

Item key

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:

```
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 a LLD rule creation with the use of Mbean. To understand the difference between a LLD rule collecting Mbeans and a LLD rule collecting Mbean attributes better please take a look at following table:

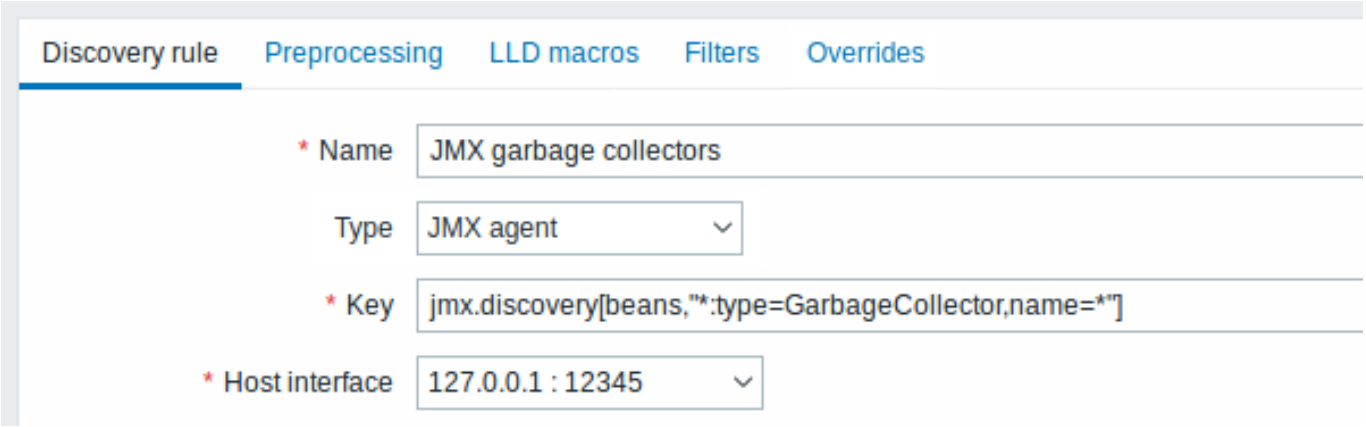
MBean1	**MBean2**	**MBean3**
MBean1Attribute1	MBean2Attribute1	MBean3Attribute1
MBean1Attribute2	MBean2Attribute2	MBean3Attribute2
MBean1Attribute3	MBean2Attribute3	MBean3Attribute3

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.

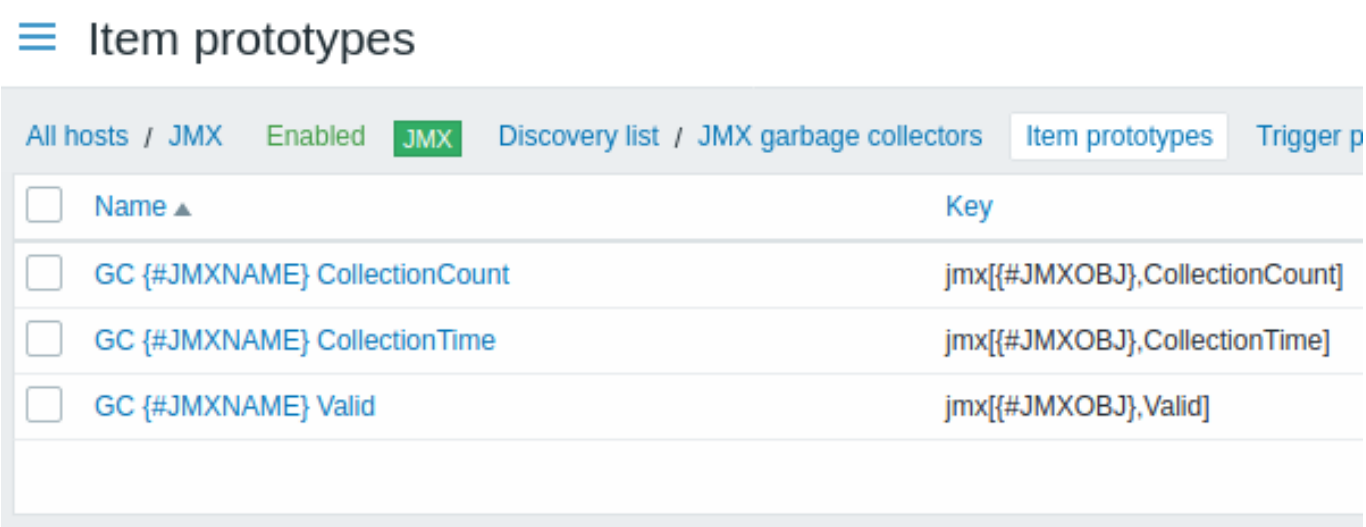
Discovery rule configuration collecting Mbeans (without the attributes) looks like the following:



The key used here:

```
jmx.discovery[beans,"*:type=GarbageCollector,name=*"]
```

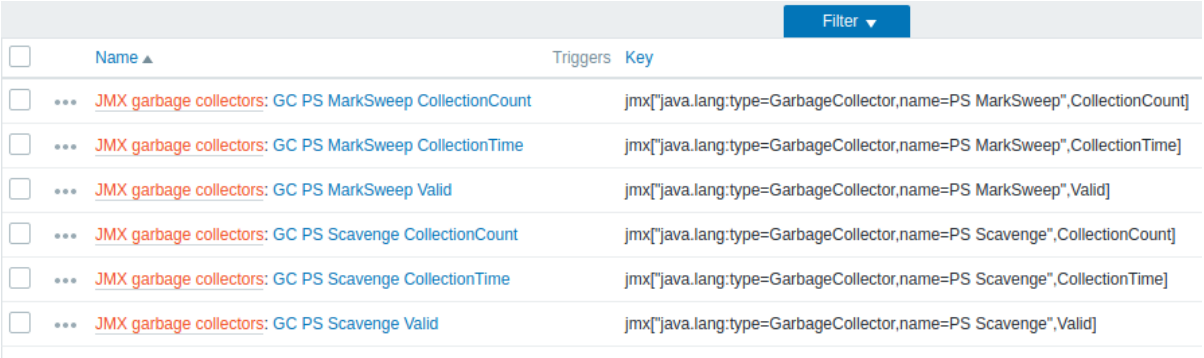
All the garbage collectors without attributes will be discovered. As Garbage collectors have the same attribute set, we can use desired attributes in item prototypes the following way:



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):

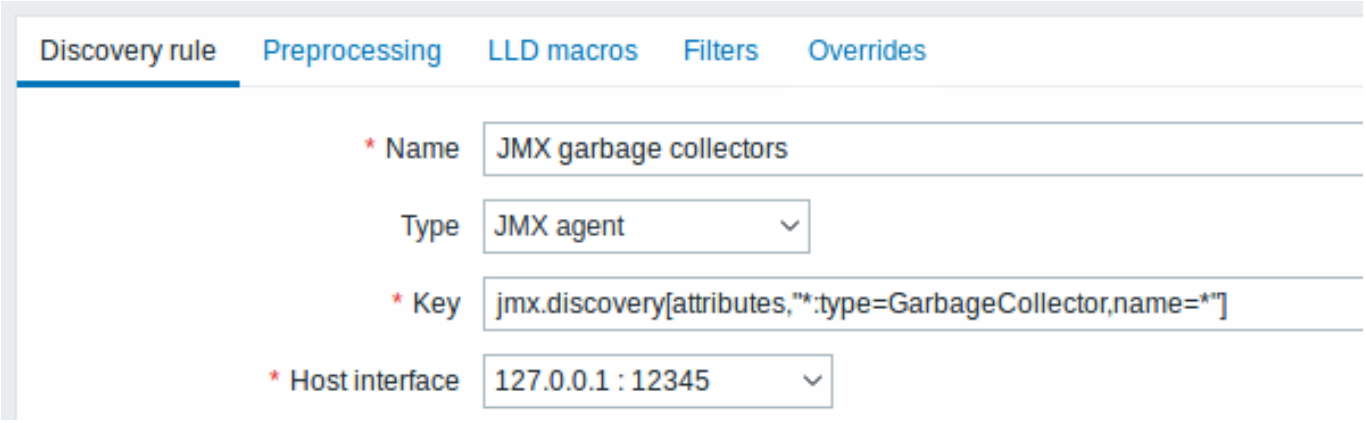


Example 2: Discovering Mbean attributes

This rule will return 9 objects with the following fields: MBean1Attribute1, MBean2Attribute1, Mbean3Attribute1,MBean1Attribute2,MBean2Attr Mbean3Attribute2, MBean1Attribute3, MBean2Attribute3, Mbean3Attribute3.

For more information about objects please refer to supported macros table, *Discovery of MBean attributes* section.

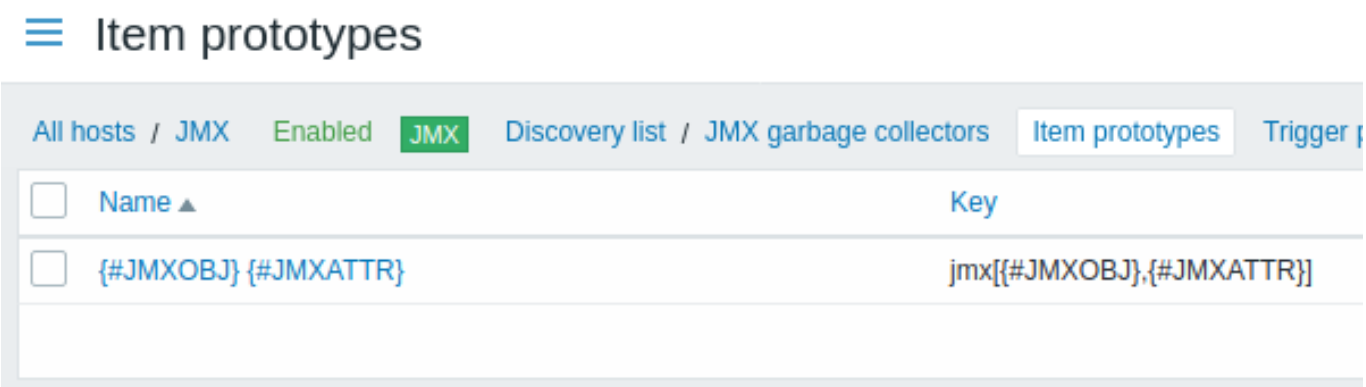
Discovery rule configuration collecting Mbean attributes looks like the following:



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 is 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
        }
    }
]
```

6 Discovery of IPMI sensors

Overview

It is possible to automatically discover IPMI sensors.

To do that, you may use a combination of:

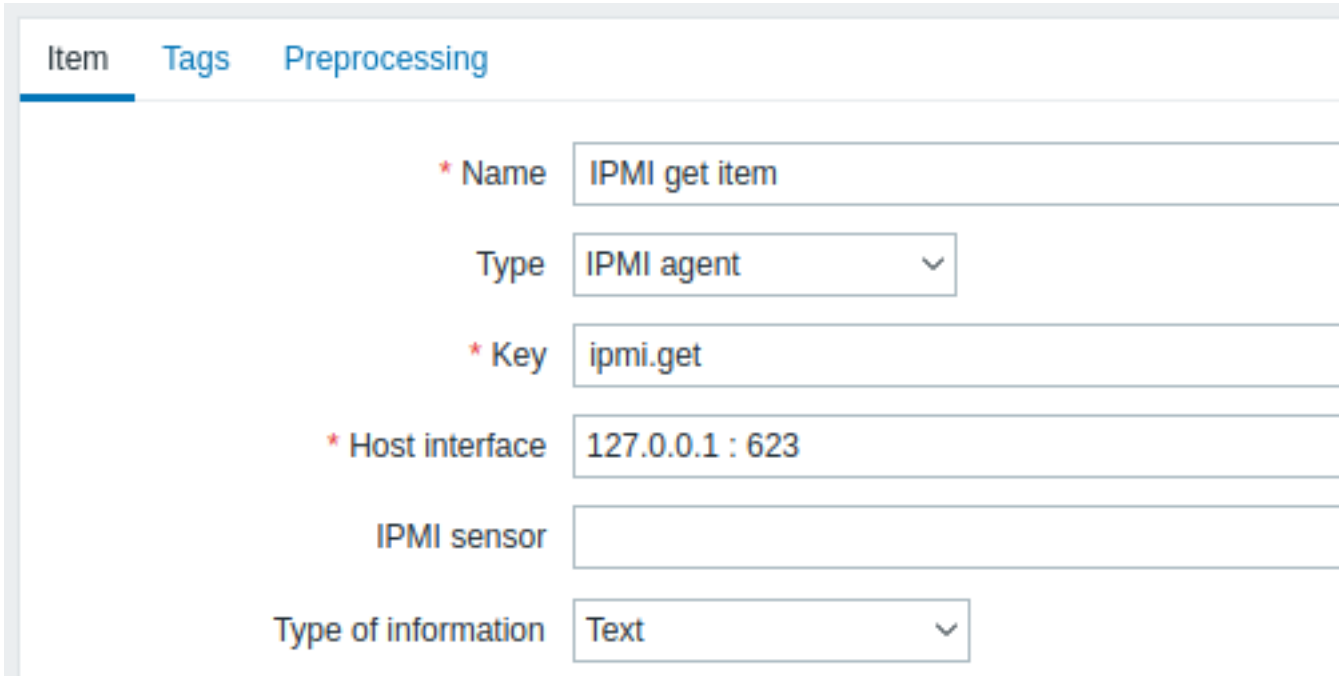
- the `ipmi.get` IPMI item (supported since Zabbix **5.0.0**) as the master item
- dependent low-level discovery rule and item prototypes

Configuration

Master item

Create an IPMI item using the following key:

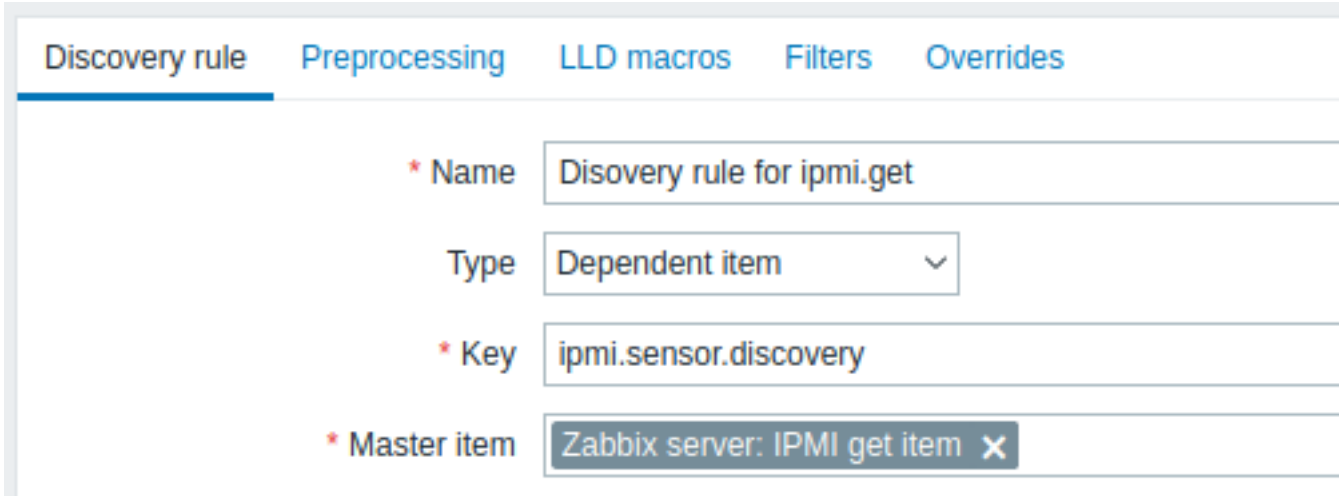
`ipmi.get`



Set the type of information to ”Text” for possibly big JSON data.

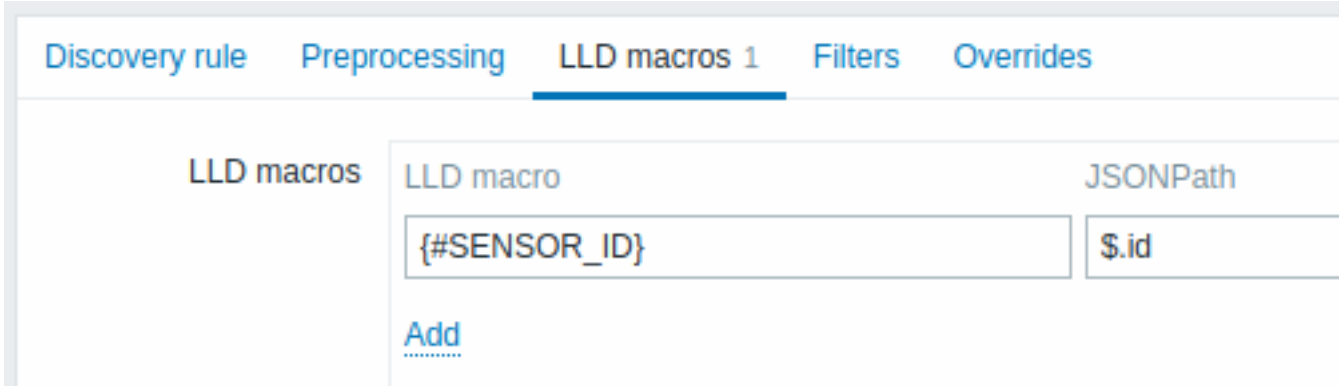
Dependent LLD rule

Create a low-level discovery rule as ”Dependent item” type:



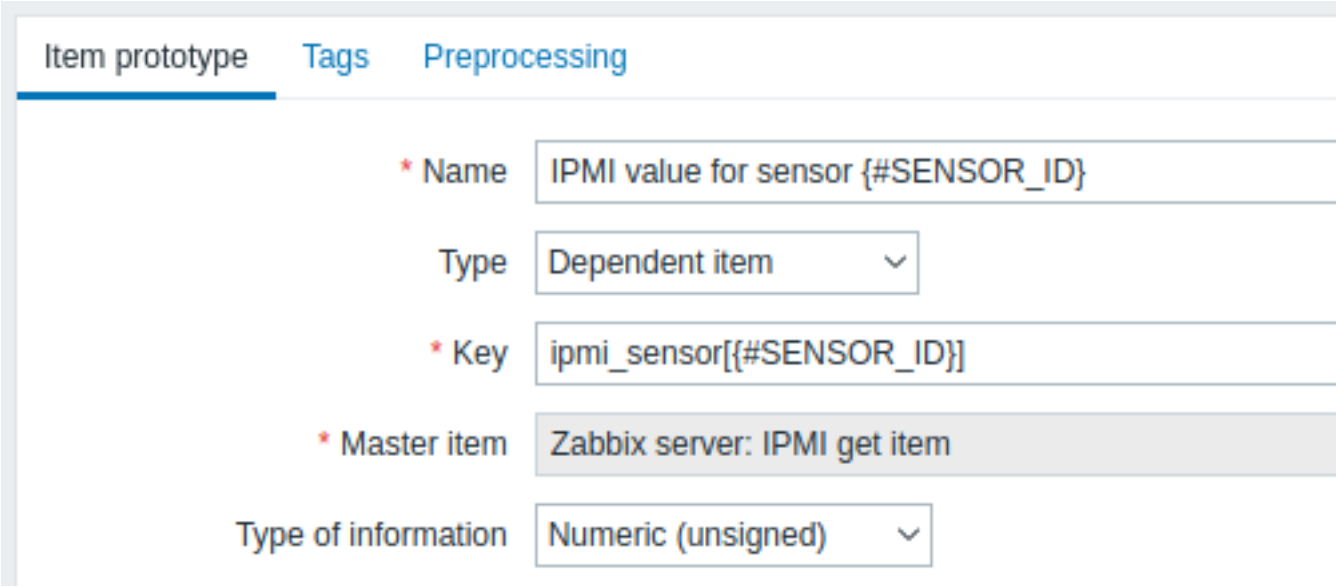
As master item select the `ipmi.get` item we created.

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.



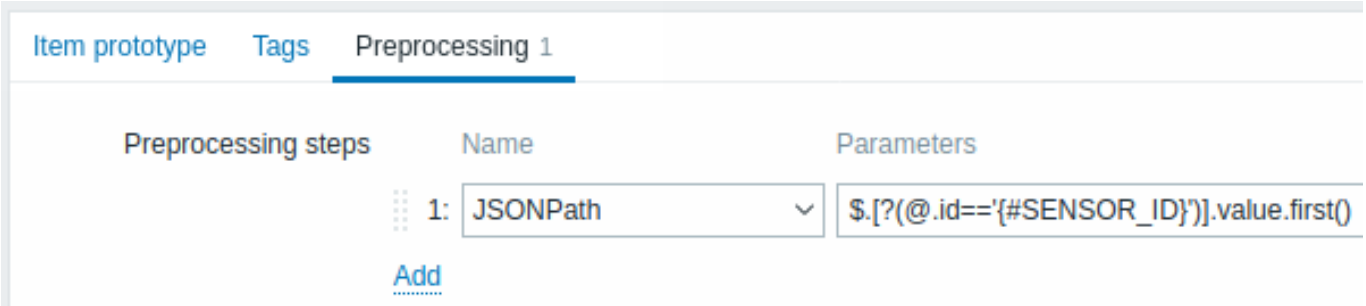
Note the use of the {#SENSOR_ID} macro in the item prototype name and key:

- *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.

7 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}; item key: `systemd.unit.info["{#UNIT.NAME}"]`
- Item name: {#UNIT.DESCRIPTION}; item key: `systemd.unit.info["{#UNIT.NAME}",LoadState]`

`systemd.unit.info` agent items are supported since Zabbix 4.4.

8 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: 0 - Running 1 - Paused 2 - Start pending 3 - Pause pending 4 - Continue pending 5 - Stop pending 6 - Stopped 7 - Unknown
{#SERVICE.STATENAME}	Name of the service state (*Running*, *Paused*, *Start pending*, *Pause pending*, *Continue pending*, *Stop pending*, *Stopped* or *Unknown*).
{#SERVICE.PATH}	Service path.
{#SERVICE.USER}	Service user.
{#SERVICE.STARTUP}	Numerical value of the service startup type: 0 - Automatic 1 - Automatic delayed 2 - Manual 3 - Disabled 4 - Unknown
{#SERVICE.STARTUPNAME}	Name of the service startup type (*Automatic*, *Automatic delayed*, *Manual*, *Disabled*, *Unknown*).
{#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.

9 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

The item to use in the discovery rule is

`perf_instance.discovery[object]`

or, to be able to provide the object name in English only, independently of OS localization:

`perf_instance_en.discovery[object]`

For example:

```
perf_instance.discovery[Processador]
perf_instance_en.discovery[Processor]
```

These items are supported since Zabbix Windows agent 5.0.1.

Supported macros

The discovery will return all instances of the specified object in the {#INSTANCE} macro, which may be used in the prototypes of perf_count and perf_count_en items.

```
[
    {"{#INSTANCE}":"0"},
    {"{#INSTANCE}":"1"},
    {"{#INSTANCE}":"_Total"}
]
```

For example, if the item key used in the discovery rule is:

`perf_instance.discovery[Processor]`

you may create an item prototype:

`perf_counter["\Processor({#INSTANCE})\% Processor Time"]`

Notes:

- If the specified object is not found or does not support variable instances then the discovery item will become NOTSUPPORTED.
- If the specified object supports variable instances, but currently does not have any instances, then an empty JSON array will be returned.
- In case of duplicate instances they will be skipped.

10 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.

11 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.

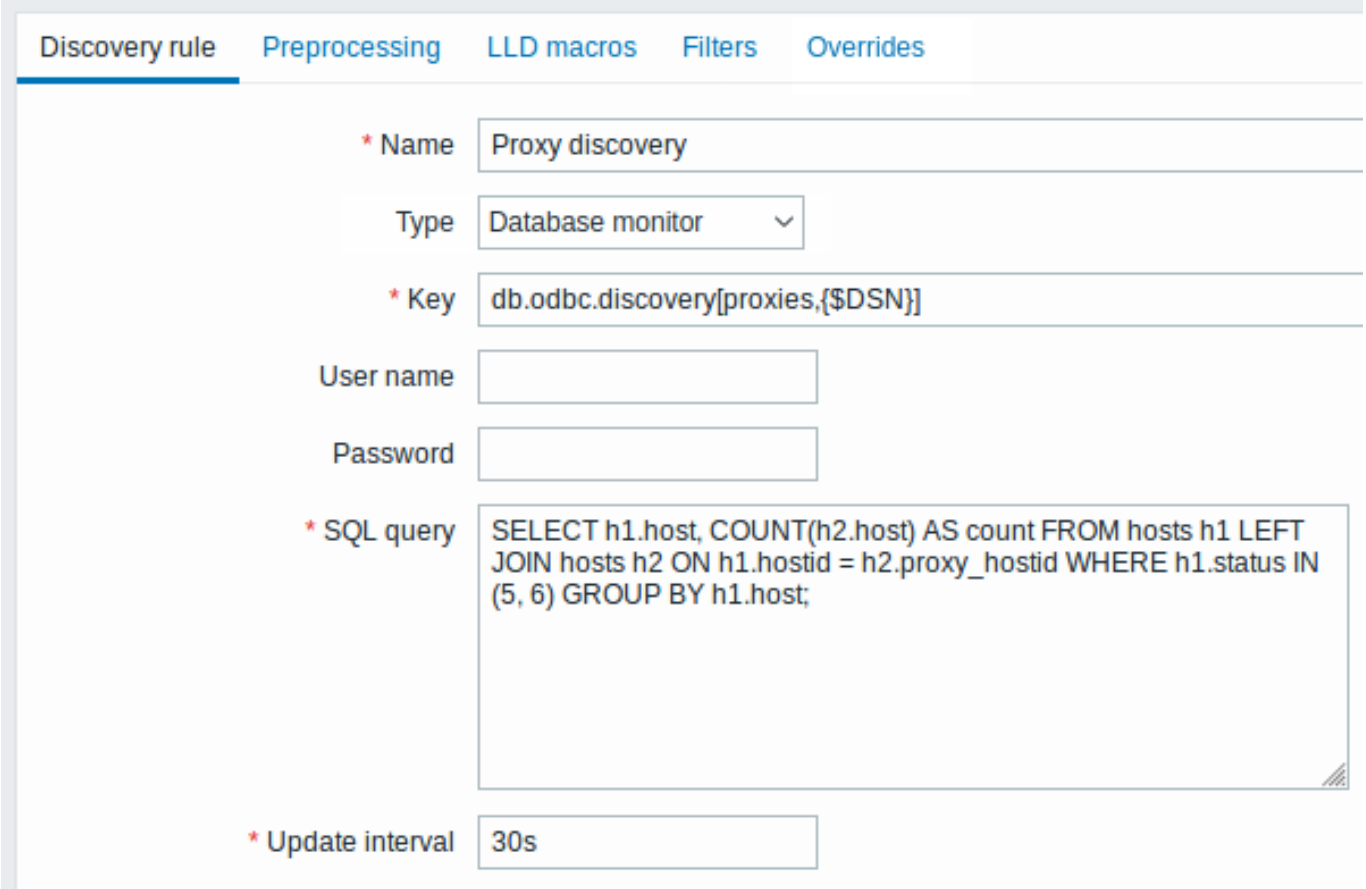
Two item keys may be used in "Database monitor" discovery rules:

- **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"
    }
```

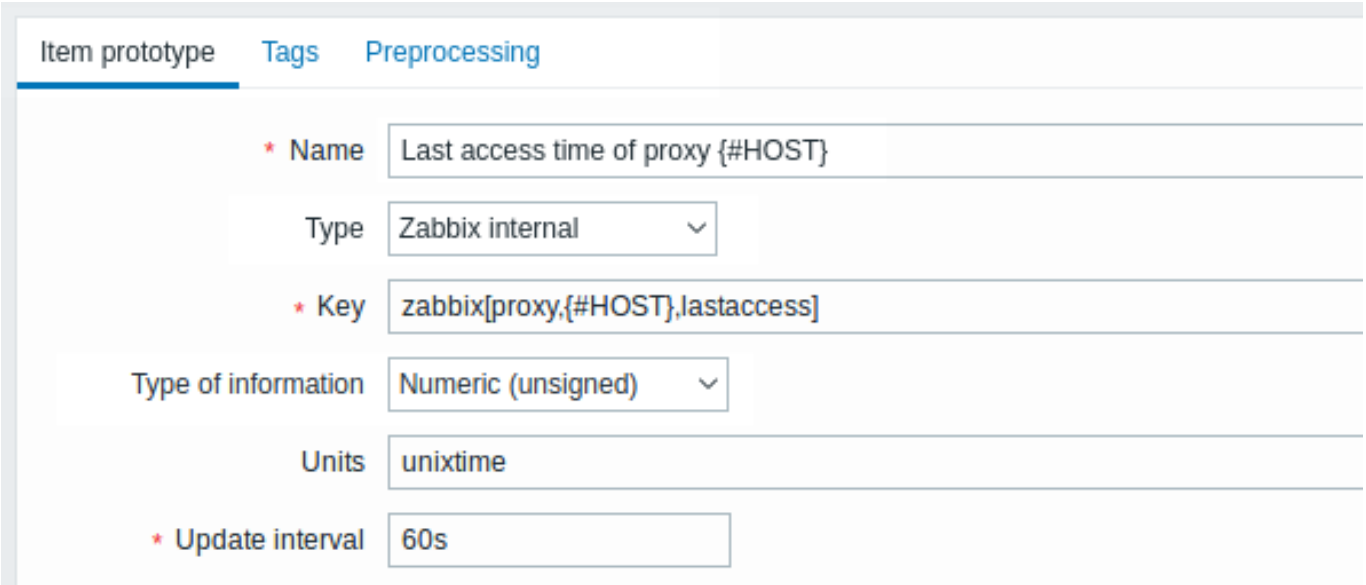
```
]
```

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)
```

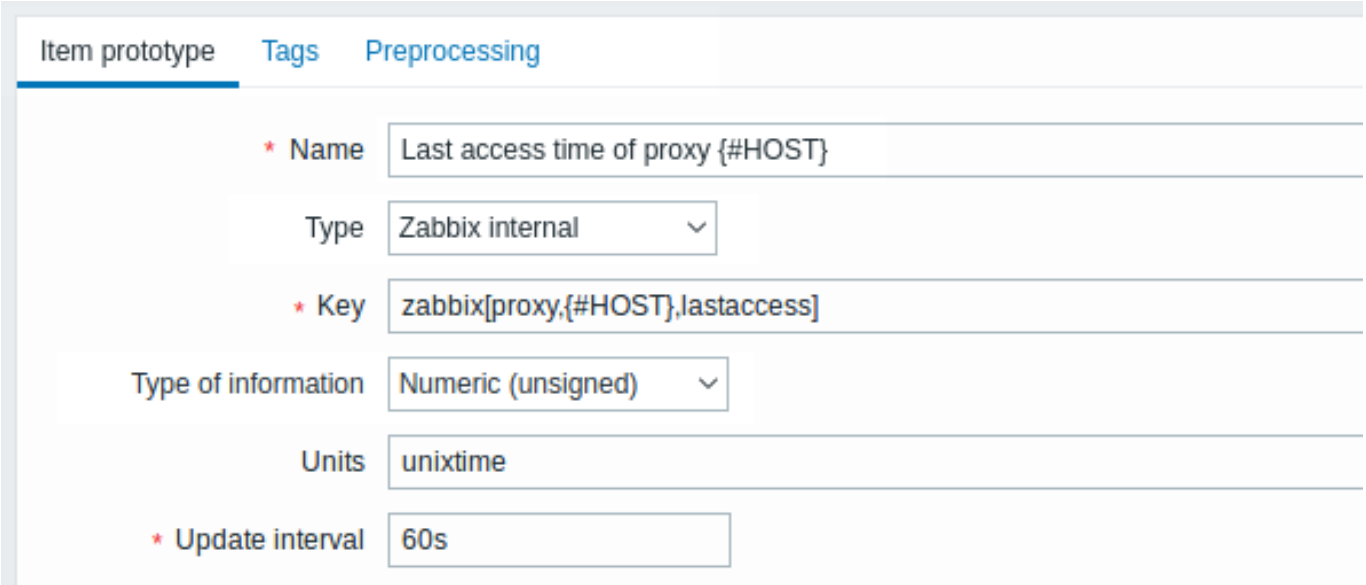
this JSON will be returned:

```
[
    {
        "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:



12 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

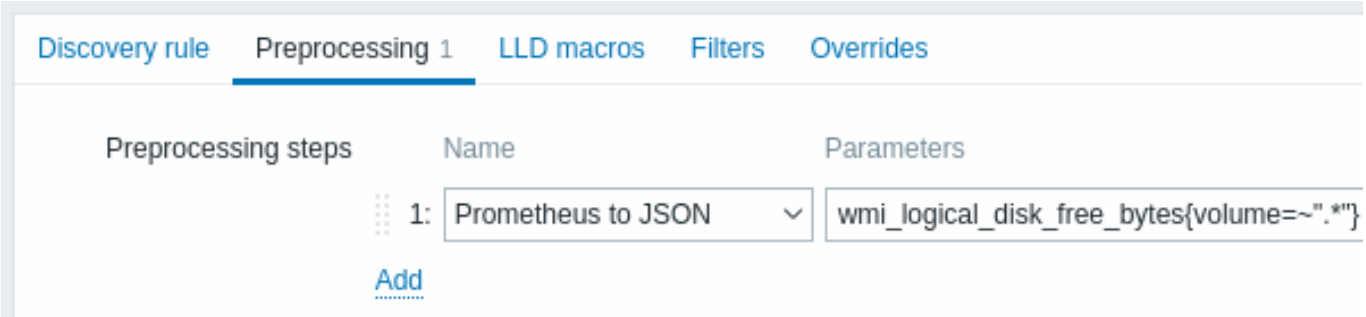
The low-level discovery rule should be created as a dependent item to the HTTP master item that collects Prometheus data.

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"
    }
]
```

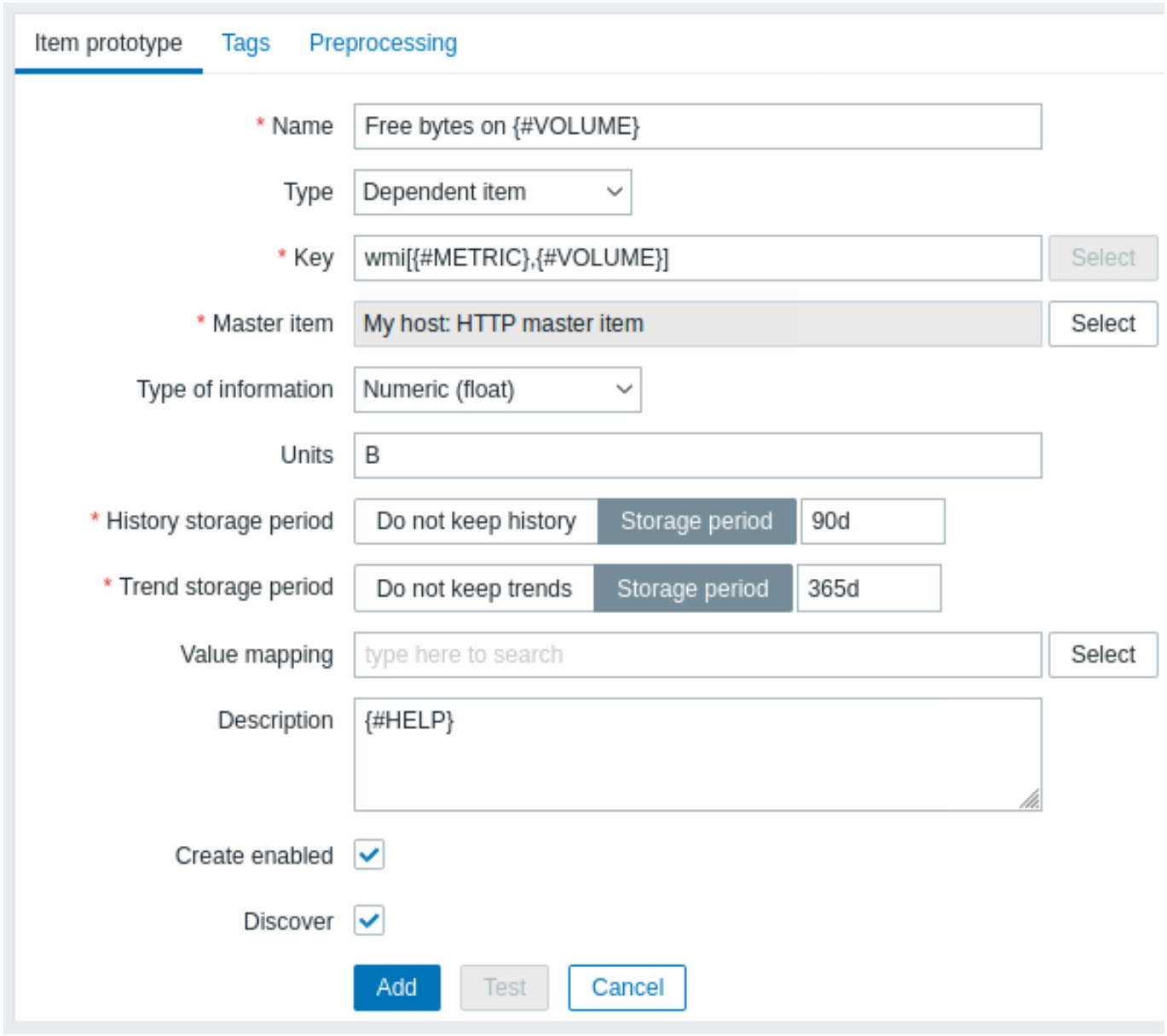
Mapping LLD macros

Next you have to go to the LLD macros tab and make the following mappings:

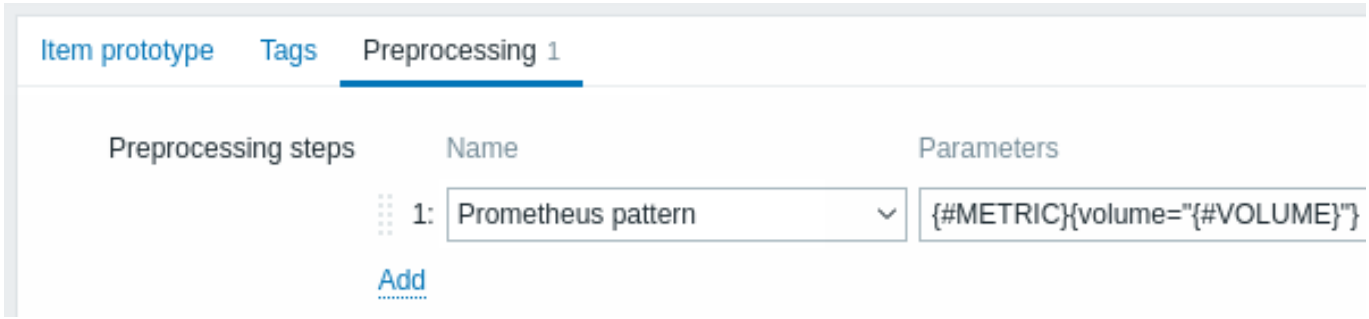
```
{#VOLUME}=$.labels['volume']
{#METRIC}=$['name']
{#HELP}=$['help']
```

Item prototype

You may want to create an item prototype like this:



with preprocessing options:



13 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.

14 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[1]	**Yes**
Local administration	No
Ready for embedded hardware	**Yes**
One way TCP connections	**Yes**
Centralized configuration	**Yes**
Generates notifications	No

Note:
[1] Automatic DB creation feature only works with SQLite. Other databases require a manual setup.

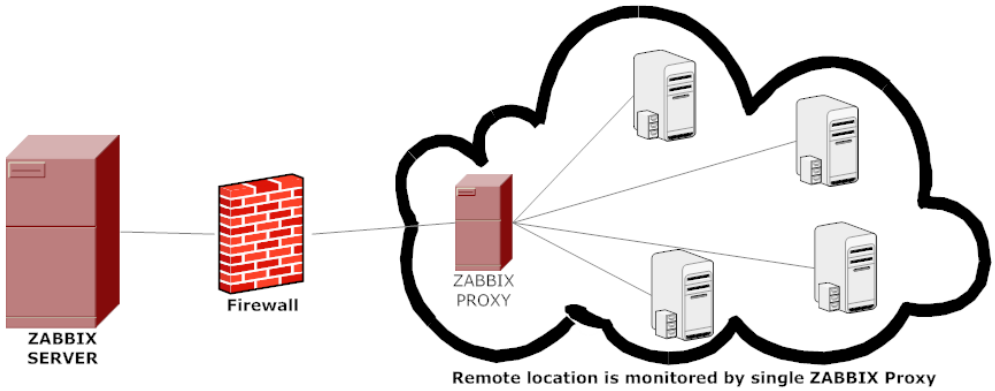
1 Proxies

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 throttling list, but still not accepting proxy data.
Drops to 20%	Normal	Drops the throttling list and starts accepting proxy data normally.

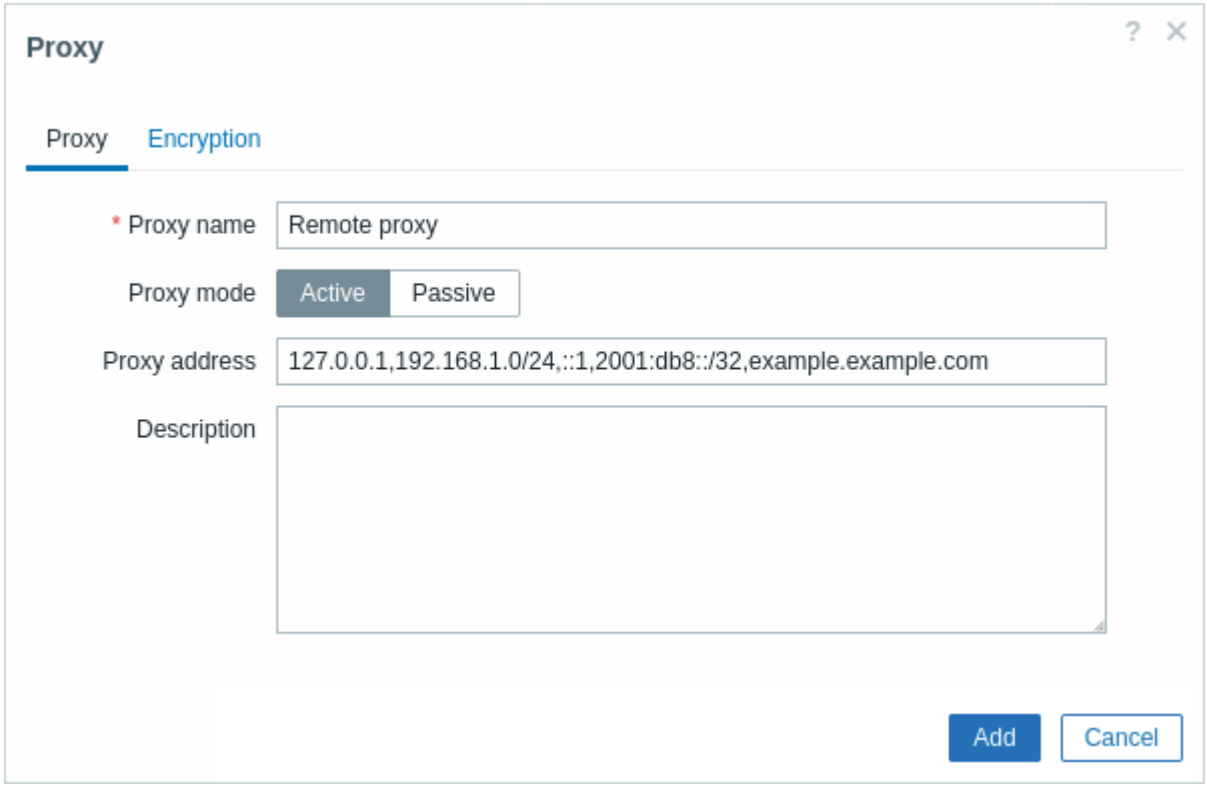
You may use the `zabbix[wcache,history,pused]` internal item to correlate this behavior of Zabbix server with a metric.

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/UDP 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”.

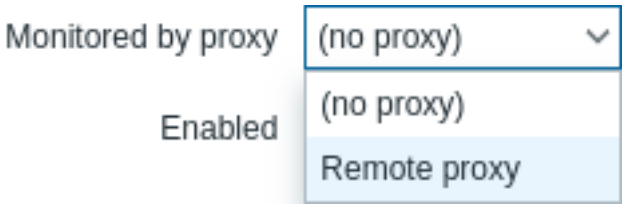
Parameter	Description
Issuer	Allowed issuer of certificate. Certificate is first validated with CA (certificate authority). If it is valid, signed by the CA, then the *Issuer* field can be used to further restrict allowed CA. This field is optional, intended to use if your Zabbix installation uses certificates 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.

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_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.

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 *Configuration* → *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 *Configuration* → *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 *Configuration* → *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 *Configuration* → *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

Library	Certificate ciphersuites	PSK ciphersuites
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 and TLSCipherPSK
- In case of zabbix_get and zabbix_sender utilities the command-line parameters `--tls-cipher13` and `--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

Parameter	OpenSSL 1.1.1d [1]
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 in-house certificate authority (CA). Certificate verification is done 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 how to set up and operate internal CA, how to generate certificate requests and sign them, how to revoke certificates you can find numerous online how-tos, for example, OpenSSL PKI Tutorial v1.1 .

Carefully consider and test your certificate extensions - see Limitations on using X.509 v3 certificate extensions.

Certificate configuration parameters

Parameter	Mandatory	Description
TLSCAFile	yes	Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification. In case of certificate chain with several members they must be ordered: lower level CA certificates first followed by certificates of higher level CA(s). Certificates from multiple CA(s) can be included in a single file.
TLSCRLFile	no	Full pathname of a file containing Certificate Revocation Lists. See notes in Certificate Revocation Lists (CRL).
TLSCertFile	yes	Full pathname of a file containing certificate (certificate chain). In case of certificate chain with several members they must be ordered: server, proxy, or agent certificate first, followed by lower level CA certificates then certificates of higher level CA(s).
TLSKeyFile	yes	Full pathname of a file containing private key. Set access rights to this file - it must be readable only by Zabbix user.
TLSServerCertIssuer	no	Allowed server certificate issuer.
TLSServerCertSubject	no	Allowed server certificate subject.

Configuring certificate on Zabbix server

1. In order to verify peer certificates, Zabbix server must have access to file with their top-level self-signed root CA certificates. For example, if we expect certificates from two independent root CAs, we can put their certificates into file `/home/zabbix/zabbix_ca_file` like this:

```
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. Put Zabbix server certificate chain into file, for example, `/home/zabbix/zabbix_server.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=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-----
```

Here the first is Zabbix server certificate, followed by intermediate CA certificate.

Note:
Use of any attributes except of the ones mentioned above is discouraged for both client and server certificates, because it may affect certificate verification process. For example, OpenSSL might fail to establish encrypted connection if *X509v3 Extended Key Usage* or *Netscape Cert Type* are set. See also: Limitations on using X.509 v3 certificate extensions.

3. Put Zabbix server private key into file, for example, `/home/zabbix/zabbix_server.key`:

```
-----BEGIN PRIVATE KEY-----
MIIEwAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQC9tIXIJoVnNXDl
...
IJLkhbybBYEf47MLhffWa7XvZTY=
-----END PRIVATE KEY-----
```

4. Edit TLS parameters in Zabbix server configuration file like this:

```
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSCertFile=/home/zabbix/zabbix_server.crt
TLSKeyFile=/home/zabbix/zabbix_server.key
```

Configuring certificate-based encryption for Zabbix proxy

1. Prepare files with top-level CA certificates, proxy certificate (chain) and private key as described in Configuring certificate on Zabbix server. Edit parameters `TLSCAFile`, `TLSCertFile`, `TLSKeyFile` in proxy configuration accordingly.

2. For active proxy edit `TLSConnect` parameter:

```
TLSConnect=cert
```

For passive proxy edit `TLSAccept` parameter:

```
TLSAccept=cert
```

3. Now you have a minimal certificate-based proxy configuration. You may prefer to improve proxy security by setting `TLSServerCertIssuer` and `TLSServerCertSubject` parameters (see Restricting allowed certificate Issuer and Subject).

4. In final proxy configuration file TLS parameters may look like:

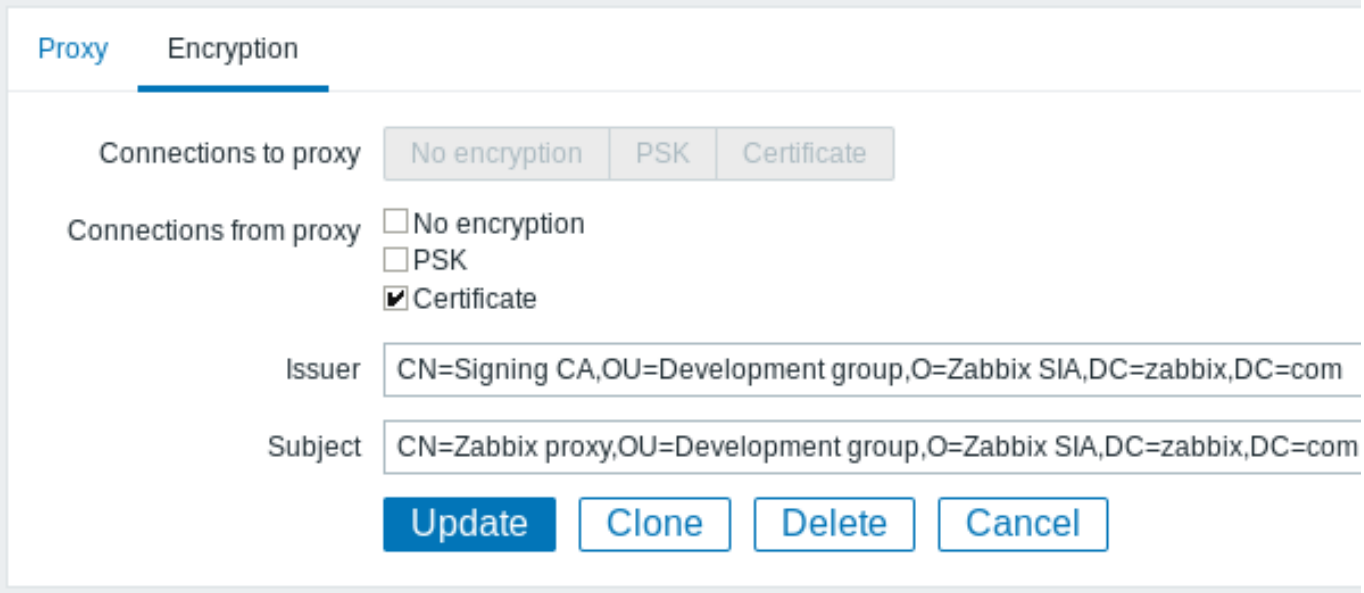
```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix server,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_proxy.crt
TLSKeyFile=/home/zabbix/zabbix_proxy.key
```

5. Configure encryption for this proxy in Zabbix frontend:

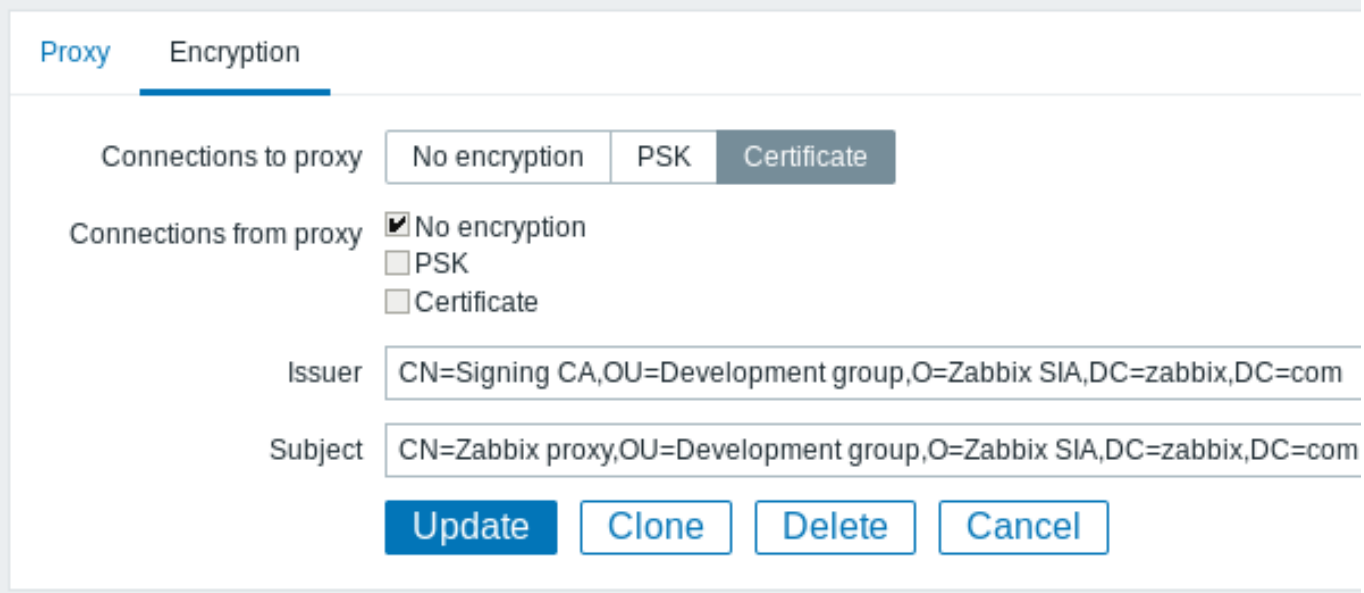
- Go to: *Administration* → *Proxies*
- Select proxy and click on **Encryption** tab

In examples below Issuer and Subject fields are filled in - see Restricting allowed certificate Issuer and Subject why and how to use these fields.

For active proxy



For passive proxy



Configuring certificate-based encryption for Zabbix agent

1. Prepare files with top-level CA certificates, agent certificate (chain) and private key as described in Configuring certificate on Zabbix server. Edit parameters `TLSCAFile`, `TLSCertFile`, `TLSKeyFile` in agent configuration accordingly.

2. For active checks edit `TLSConnect` parameter:

```
TLSConnect=cert
```

For passive checks edit `TLSAccept` parameter:

```
TLSAccept=cert
```

3. Now you have a minimal certificate-based agent configuration. You may prefer to improve agent security by setting `TLSServerCertIssuer` and `TLSServerCertSubject` parameters. (see Restricting allowed certificate Issuer and Subject).

4. In final agent configuration file TLS parameters may look like:

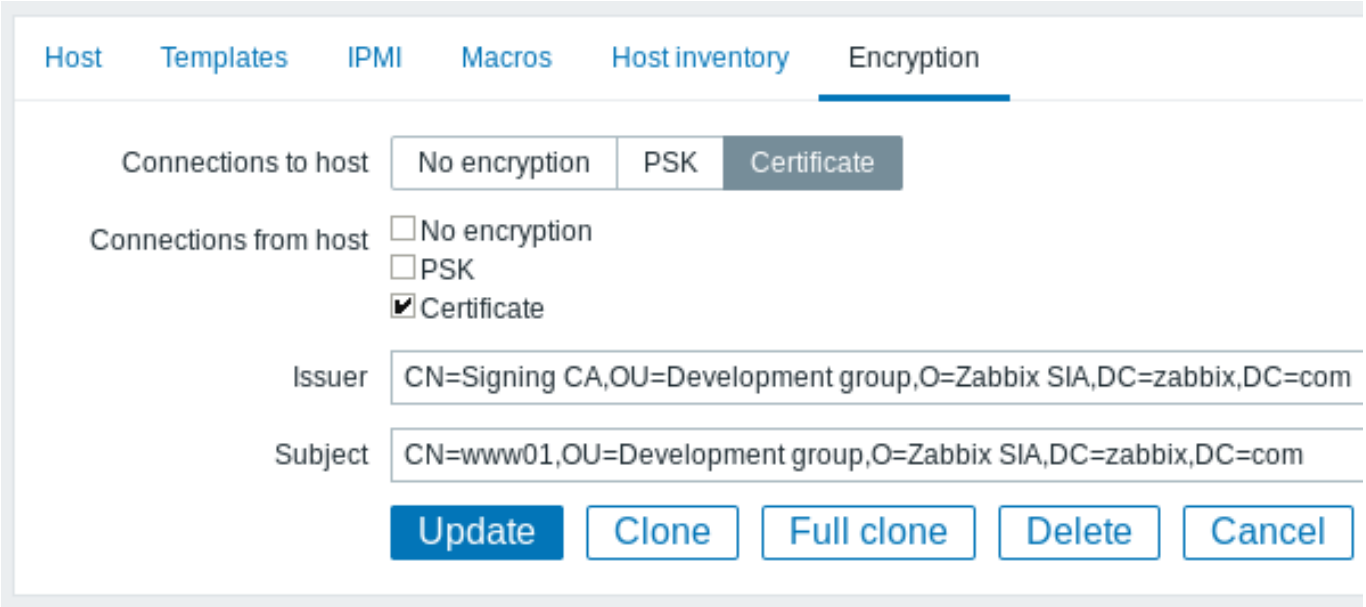
```
TLSConnect=cert
TLSAccept=cert
TLSCAFile=/home/zabbix/zabbix_ca_file
TLSServerCertIssuer=CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSServerCertSubject=CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com
TLSCertFile=/home/zabbix/zabbix_agentd.crt
TLSKeyFile=/home/zabbix/zabbix_agentd.key
```

(Example assumes that host is monitored via proxy, hence proxy certificate Subject.)

5. Configure encryption for this agent in Zabbix frontend:

- Go to: *Configuration* → *Hosts*
- Select host and click on **Encryption** tab

In example below Issuer and Subject fields are filled in - see Restricting allowed certificate Issuer and Subject why and how to use these fields.



Restricting allowed certificate Issuer and Subject

When two Zabbix components (e.g. server and agent) establish a TLS connection they both check each others certificates. If a peer certificate is signed by a trusted CA (with pre-configured top-level certificate in `TLSCAFile`), is valid, has not expired and passes some other checks then communication can proceed. Certificate issuer and subject are not checked in this simplest case.

Here is a risk - anybody with a valid certificate can impersonate anybody else (e.g. a host certificate can be used to impersonate server). This may be acceptable in small environments where certificates are signed by a dedicated in-house CA and risk of impersonating is low.

If your top-level CA is used for issuing other certificates which should not be accepted by Zabbix or you want to reduce risk of impersonating you can restrict allowed certificates by specifying their Issuer and Subject strings.

For example, you can write in Zabbix proxy configuration 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
```

With these settings, an active proxy will not talk to Zabbix server with different Issuer or Subject string in certificate, a passive proxy will not accept requests from such server.

A few notes about Issuer or Subject string matching:

1. Issuer and Subject strings are checked independently. Both are optional.
2. UTF-8 characters are allowed.
3. Unspecified string means any string is accepted.
4. Strings are compared "as-is", they must be exactly the same to match.
5. Wildcards and regexp's are not supported in matching.
6. Only some requirements from RFC 4514 Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names are implemented:
    1. escape characters '"' (U+0022), '+' U+002B, ',' U+002C, ';' U+003B, '<' U+003C, '>' U+003E, '\' U+005C anywhere in string.
    2. escape characters space (' ' U+0020) or number sign ('#' U+0023) at the beginning of string.
    3. escape character space (' ' U+0020) at the end of string.
7. Match fails if a null character (U+0000) is encountered (RFC 4514 allows it).
8. Requirements of RFC 4517 Lightweight Directory Access Protocol (LDAP): Syntaxes and Matching Rules and RFC 4518 Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation are not supported due to amount of work required.

Order of fields in Issuer and Subject strings and formatting are important! Zabbix follows RFC 4514 recommendation and uses "reverse" order of fields.

The reverse order can be illustrated by example:

```
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
```

Note that it starts with low level (CN), proceeds to mid-level (OU, O) and ends with top-level (DC) fields.

OpenSSL by default shows certificate Issuer and Subject fields in "normal" order, depending on additional options used:

```
$ 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
```

Here Issuer and Subject strings start with top-level (DC) and end with low-level (CN) field, spaces and field separators depend on options used. None of these values will match in Zabbix Issuer and Subject fields!

Attention:
To get proper Issuer and Subject strings usable in Zabbix invoke OpenSSL with special options `-nameopt esc_2253,esc_ctrl,utf8,dump_nostr,dump_unknown,dump_der,sep_comma_plus,dn_rev,sname`:

```
$ 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
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
```

Now string fields are in reverse order, fields are comma-separated, can be used in Zabbix configuration files and frontend.

Limitations on using X.509 v3 certificate extensions

- **Subject Alternative Name (*subjectAltName*)** extension.
  Alternative subject names from *subjectAltName* extension (like IP address, e-mail address) are not supported by Zabbix. Only value of "Subject" field can be checked in Zabbix (see Restricting allowed certificate Issuer and Subject).
  If certificate uses the *subjectAltName* extension then result depends on particular combination of crypto toolkits Zabbix components are compiled with (it may or may not work, Zabbix may refuse to accept such certificates from peers).
- **Extended Key Usage** extension.
  If used then generally both *clientAuth* (TLS WWW client authentication) and *serverAuth* (TLS WWW server authentication) are necessary.
  For example, in passive checks Zabbix agent acts in a TLS server role, so *serverAuth* must be set in agent certificate. For active checks agent certificate needs *clientAuth* to be set.
  GnuTLS issues a warning in case of key usage violation but allows communication to proceed.

- **Name Constraints** extension.
  Not all crypto toolkits support it. This extension may prevent Zabbix from loading CA certificates where this section is marked as *critical* (depends on particular crypto toolkit).

Certificate Revocation Lists (CRL)

If a certificate is compromised CA can revoke it by including in CRL. CRLs can be configured in server, proxy and agent configuration file using parameter `TLSCRLFile`. For example:

```
TLSCRLFile=/home/zabbix/zabbix_crl_file
```

where `zabbix_crl_file` may contain CRLs from several CAs and look like:

```
-----BEGIN X509 CRL-----
MIIB/DCB5QIBATANBgkqhkiG9w0BAQUFADCBgTETMBEGCgmSJomT8ixkARkWA2Nv
...
treZeUPjb7LSmZ3K2hpbZN7SoOZcAoHQ3GWd9npuctg=
-----END X509 CRL-----
-----BEGIN X509 CRL-----
MIIB+TCB4gIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLGQBGRYDY29t
...
CAEebS2CND3ShBedZ8YSil59O6JvaDP61lR5lNs=
-----END X509 CRL-----
```

CRL file is loaded only on Zabbix start. CRL update requires restart.

Attention:
If Zabbix component is compiled with *OpenSSL* and CRLs are used then each top and intermediate level CA in certificate chains must have a corresponding CRL (it can be empty) in `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:

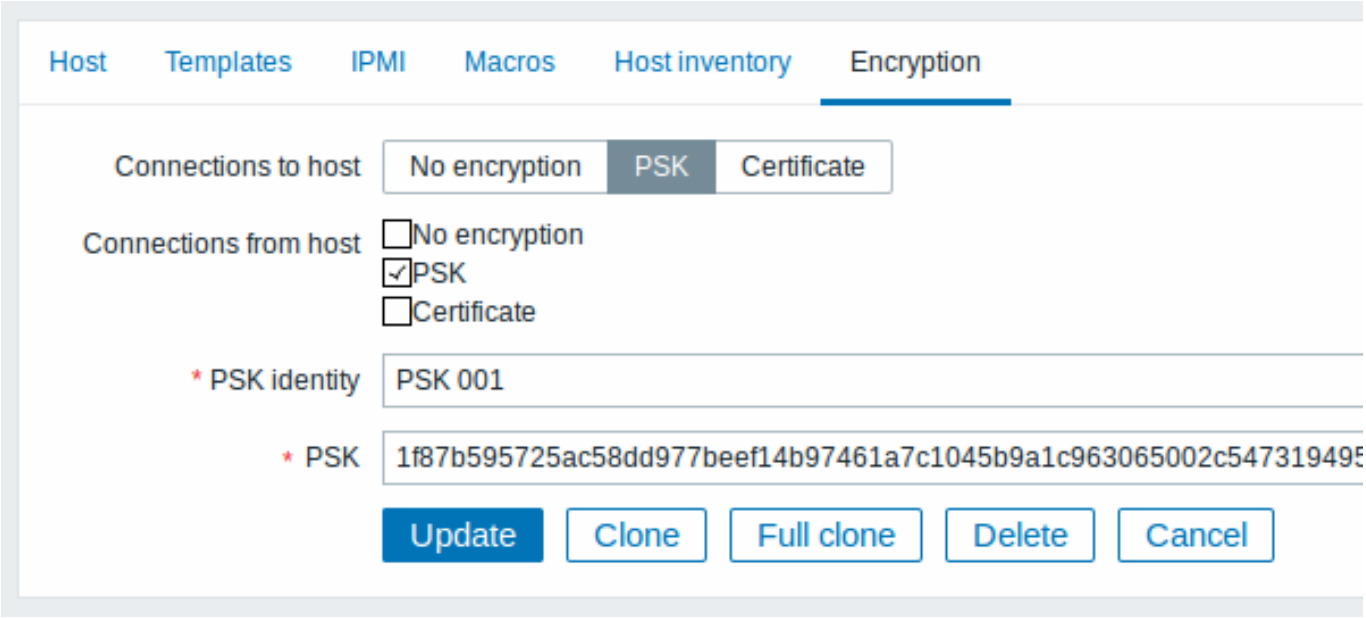
```
$ zabbix_get -s 127.0.0.1 -k "system.cpu.load[all,avg1]" --tls-connect=psk \
            --tls-psk-identity="PSK 001" --tls-psk-file=/home/zabbix/zabbix_agentd.psk
```

(To minimize downtime see how to change connection type in Connection encryption management).

Configure PSK encryption for this agent in Zabbix frontend:

- Go to: *Configuration* → *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.)

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.

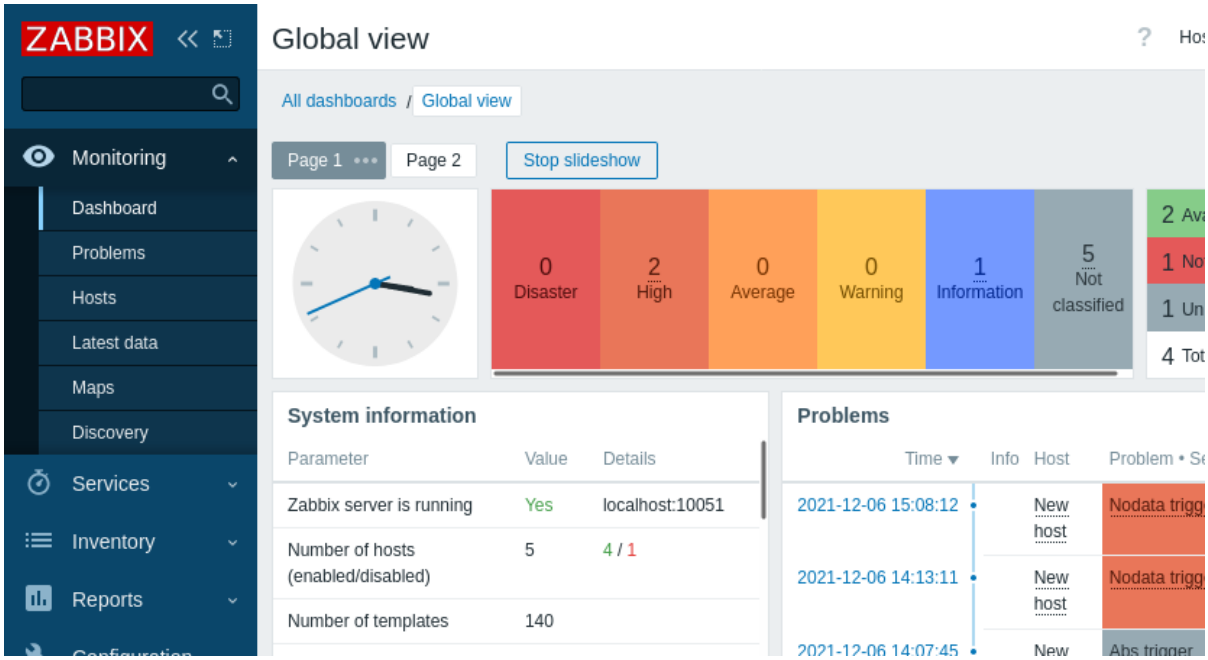
Frontend help A help link is provided in Zabbix frontend forms with direct links to the corresponding parts of the documentation.

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.

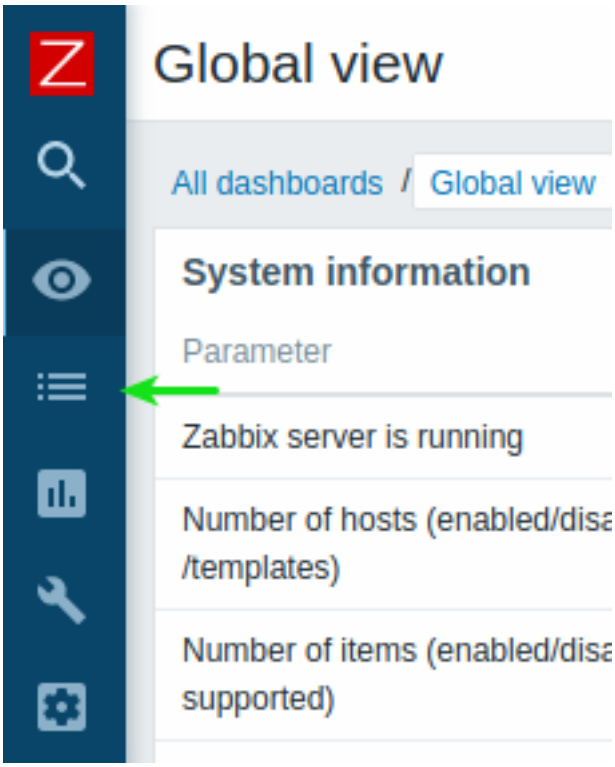


Working with the menu

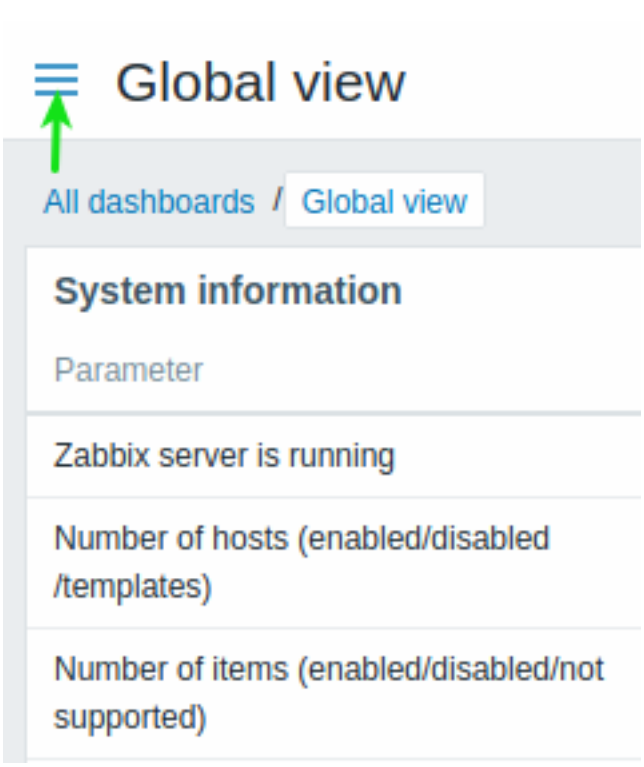
A global search box is located below the Zabbix logo.

The menu can be collapsed or hidden completely:

- To collapse, click on  next to Zabbix logo
- To hide, click on  next to Zabbix logo



Collapsed menu with only the icons visible.



Hidden menu.

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.

2 Frontend sections

1 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 Dashboard

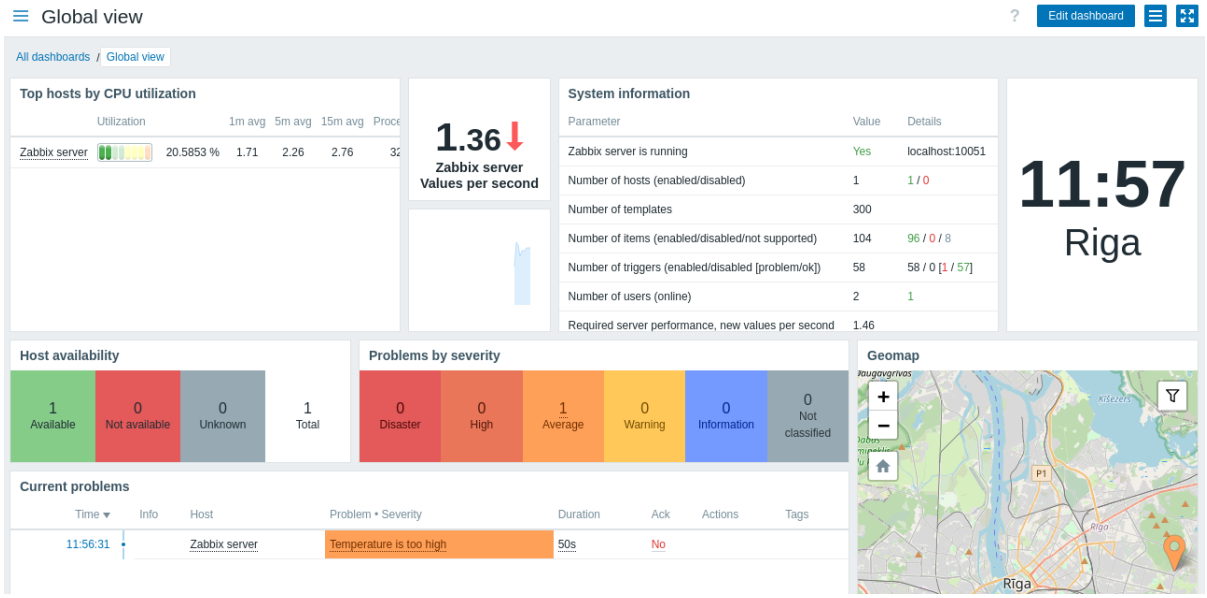
Overview

The *Monitoring → Dashboard* 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 stratched 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.

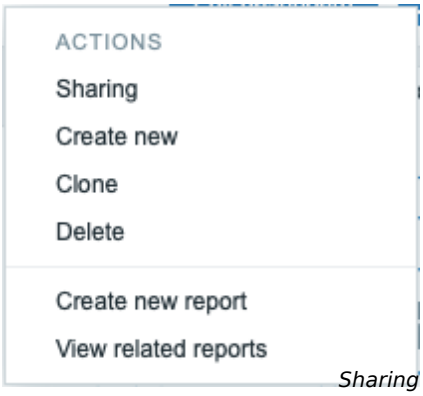
To delete one or several dashboards, mark the checkboxes of the respective dashboards and click on *Delete* below the list.

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.

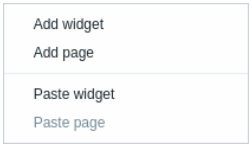
The sharing status of a dashboard is displayed in the list of all dashboards. To edit the sharing status of a dashboard, click on the *Sharing* option in the action menu when viewing a single dashboard:

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).

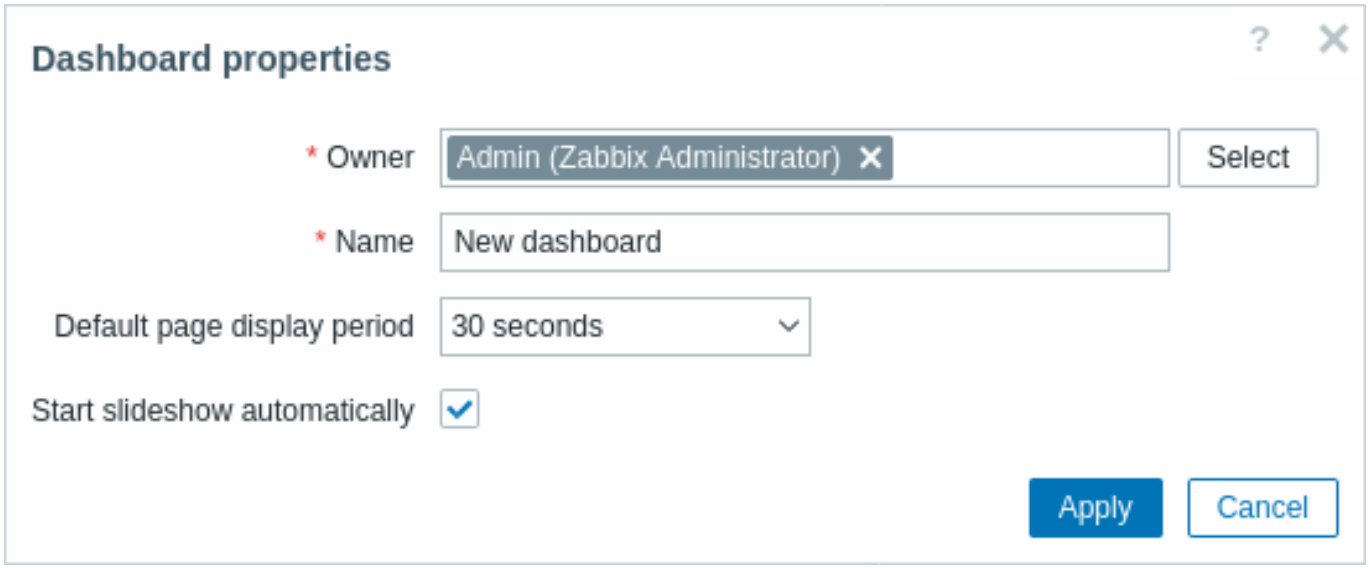
	*Add widget* - add a new widget *Add page* - add a new page *Paste widget* - paste a copied widget. This option is grayed out if no widget has been copied. Only one entity (widget or page) can be copied at one time. *Paste page* - paste a copied page. This option is grayed out if no page has been copied.
	Save dashboard changes. Cancel dashboard changes.

Creating a dashboard

It is possible to create a new dashboard in two ways:

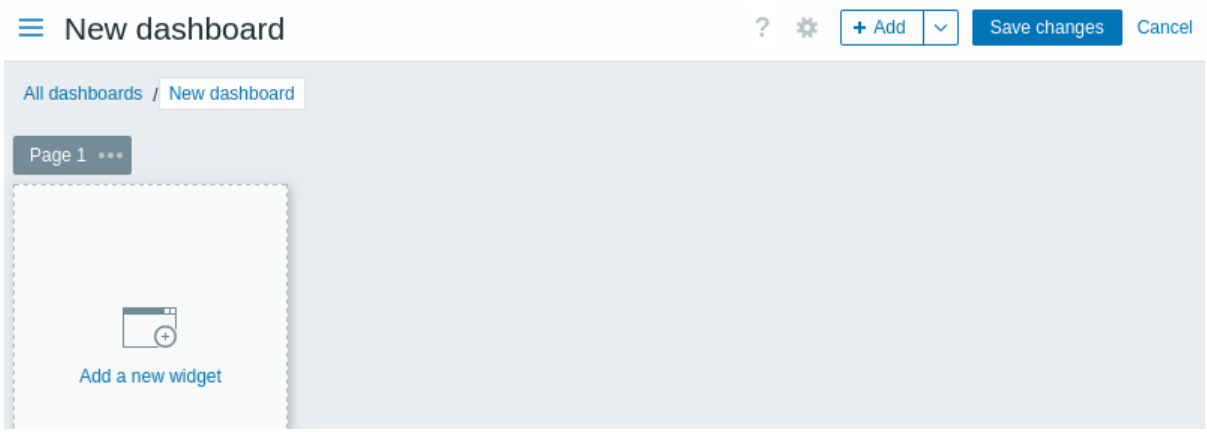
- Click on *Create dashboard*, when viewing all dashboards
- Select *Create new* from the action menu, when viewing a single dashboard

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:

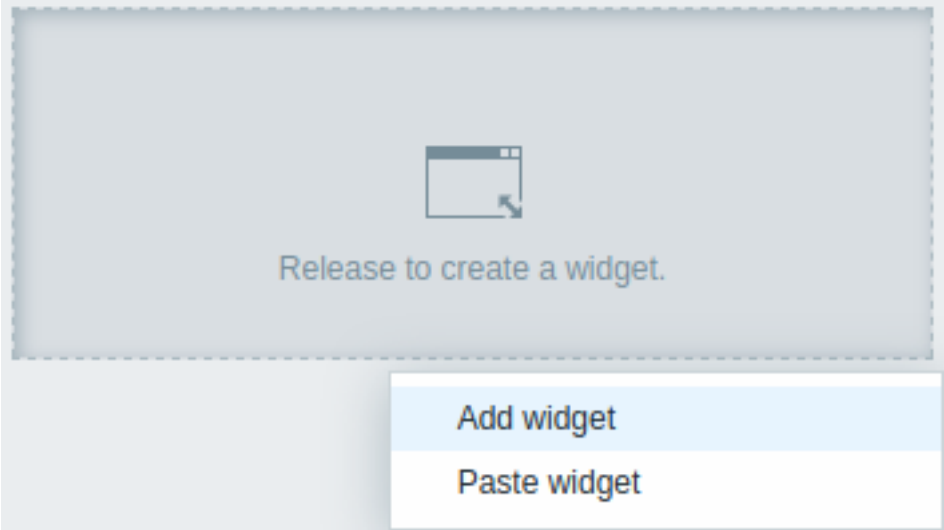
- Click on the 

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

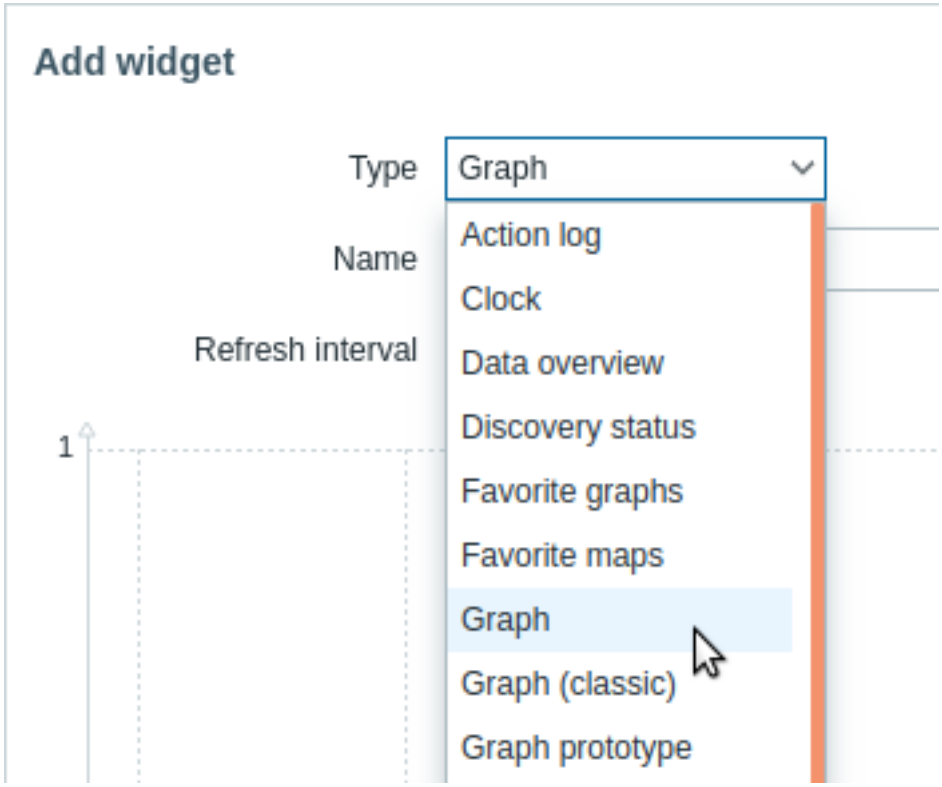
- Move your mouse to the desired empty spot for the new widget. Notice how a placeholder appears, on mouseover, on any empty slot on the dashboard. Then click to open the widget configuration form. After filling the form the widget will be created in its default size or, if its default size is bigger 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

The following widgets can be added to a dashboard:

- 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
- System information
- Problems by severity
- Top hosts
- Trigger overview
- URL
- Web monitoring

In dashboard editing mode widgets can be resized and moved around the dashboard 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



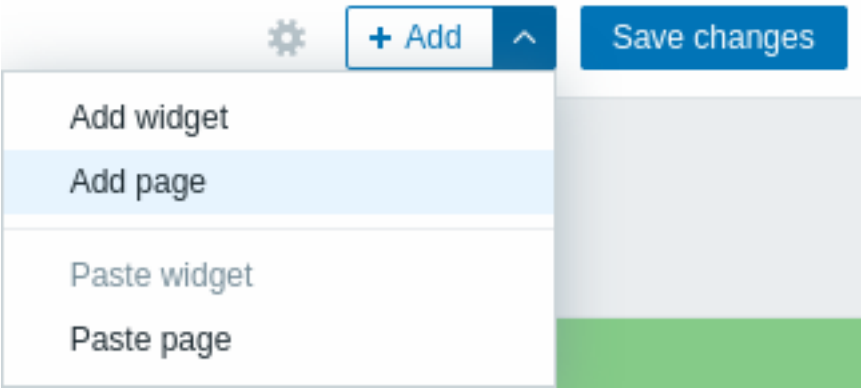
Slideshow-related controls are also available in kiosk mode (where only the page content is shown):

-  - stop slideshow
-  - start slideshow
-  - go back one page
-  - go to the next page

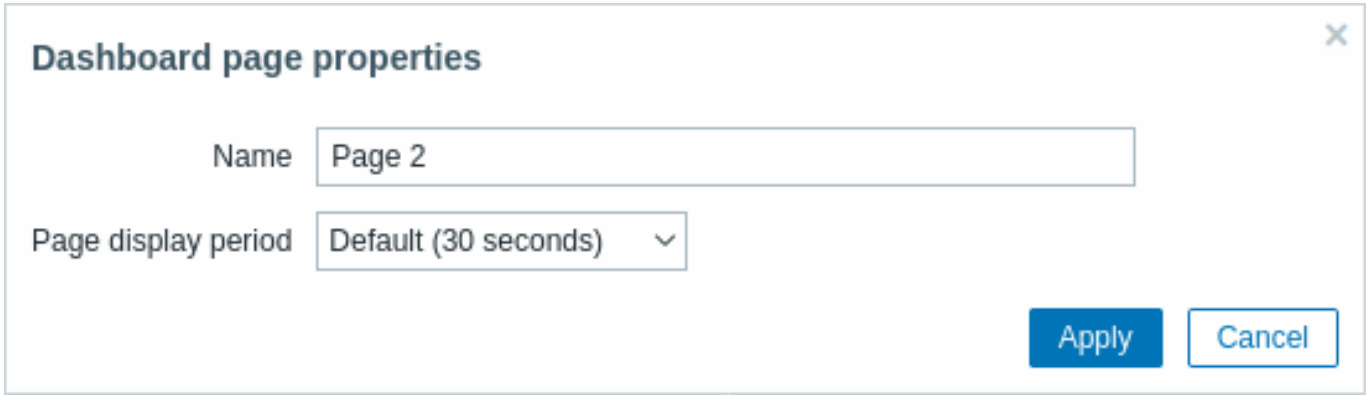
Adding pages

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*).



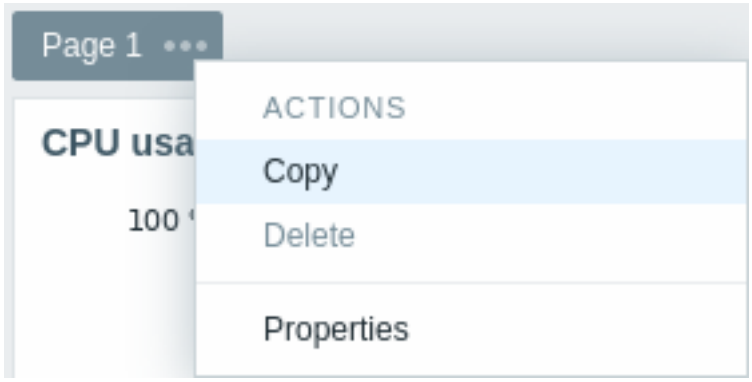
The pages can be reordered by dragging-and-dropping the page tabs. Reordering maintains the original page naming. It is always possible to go to each page by clicking on its tab.

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:

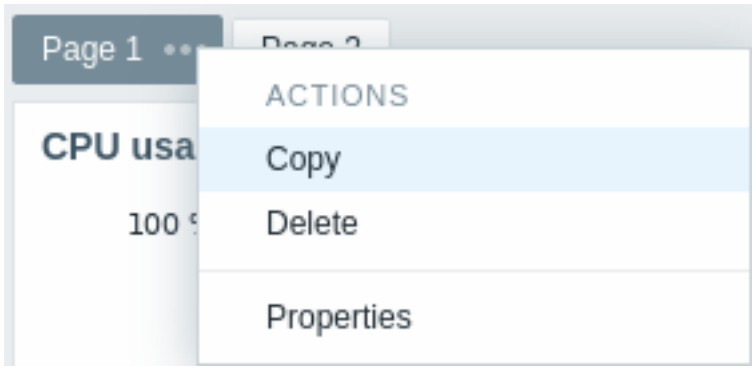


To paste the copied page:

- 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:



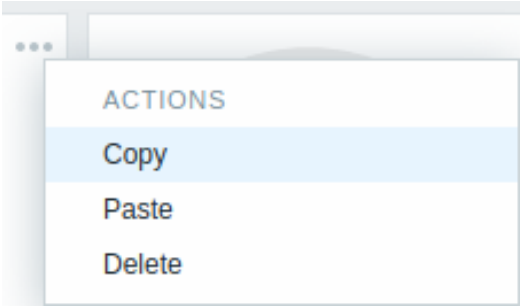
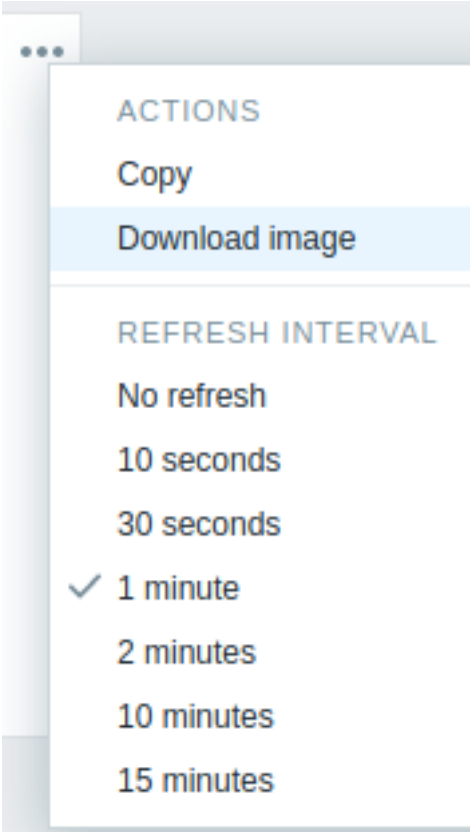
It contains the following options:

- *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 *Dynamic item*. 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 graphs containing data from various hosts.

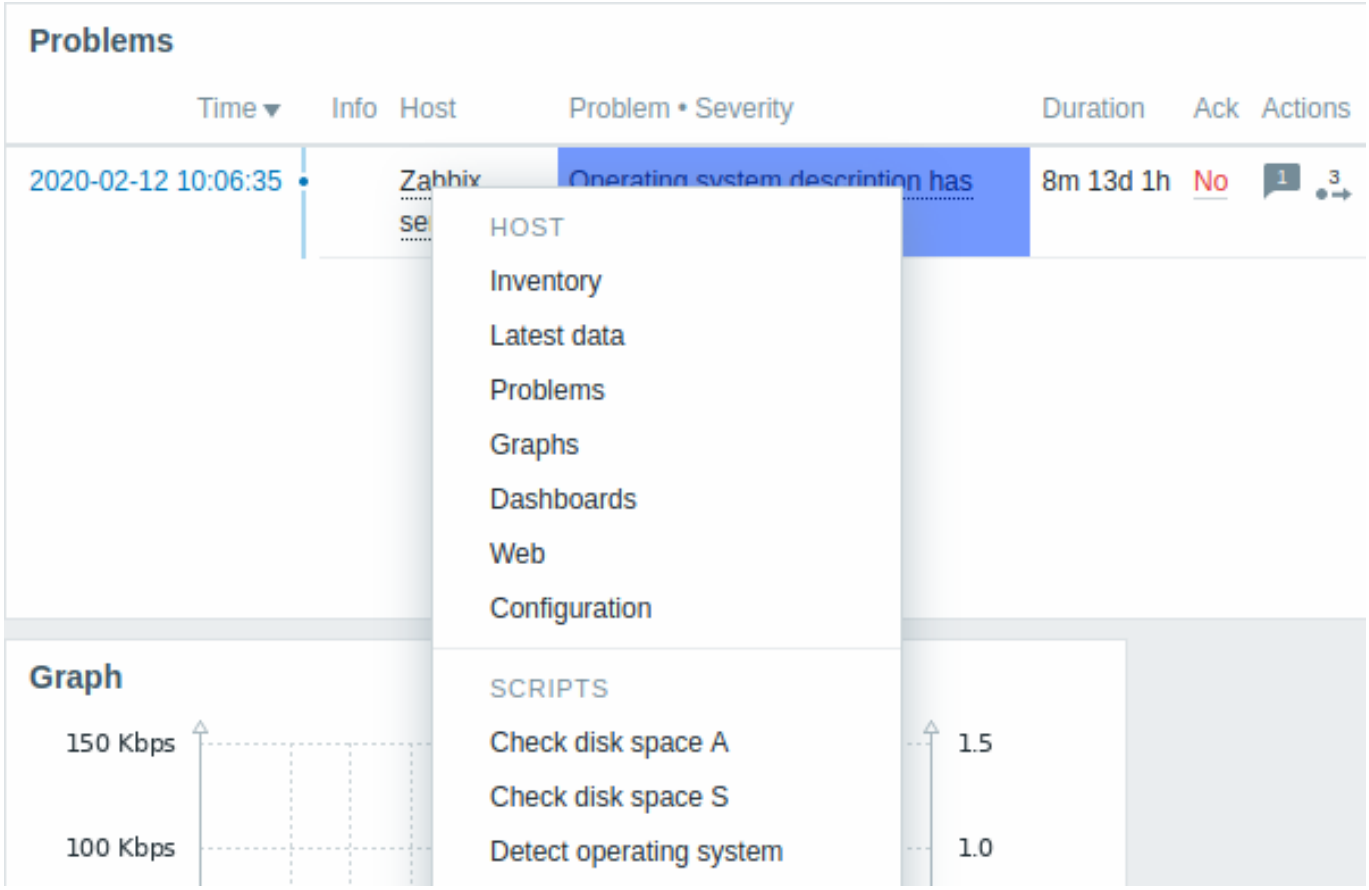
Permissions to dashboards

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.

Host menu

Clicking on a host in the *Problems* widget brings up the host menu. It includes links to host inventory, latest data, problems, graphs, dashboards, web scenarios and configuration. Note that host configuration is available for Admin and Superadmin users only.



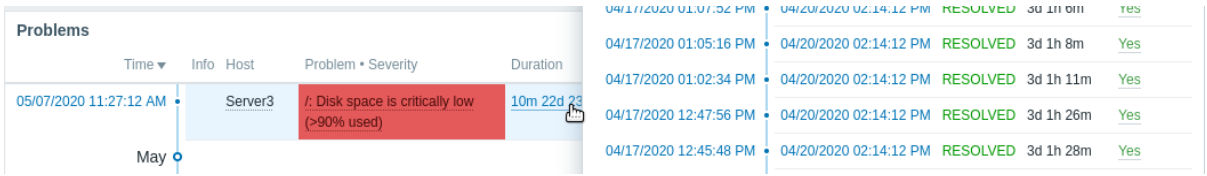
Global scripts can also be run from the host menu. These scripts need to have their scope defined as 'Manual host action' to be available in the host menu.

The host menu is accessible by clicking on a host in several other frontend sections:

- Monitoring → Problems
- Monitoring → Problems → Event details
- Monitoring → Hosts
- Monitoring → Hosts → Web Monitoring
- Monitoring → Latest data
- Monitoring → Maps
- Reports → Triggers top 100

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.

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
- 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 upcoming major release.

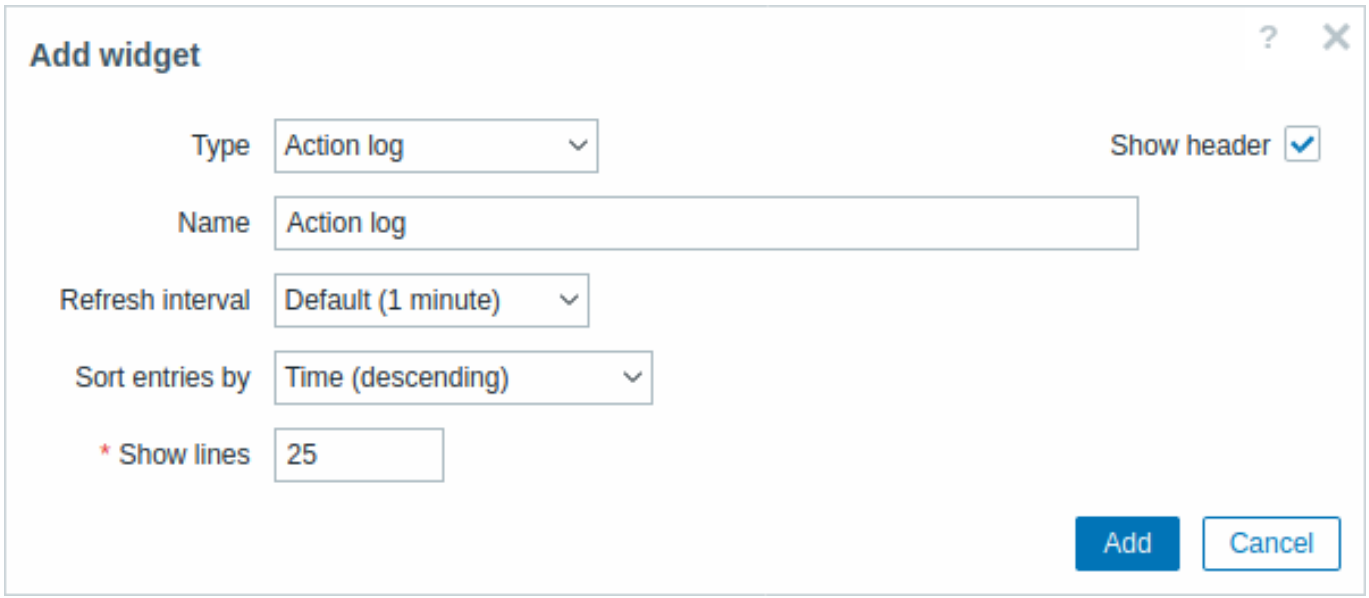
1 Action log

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:

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.

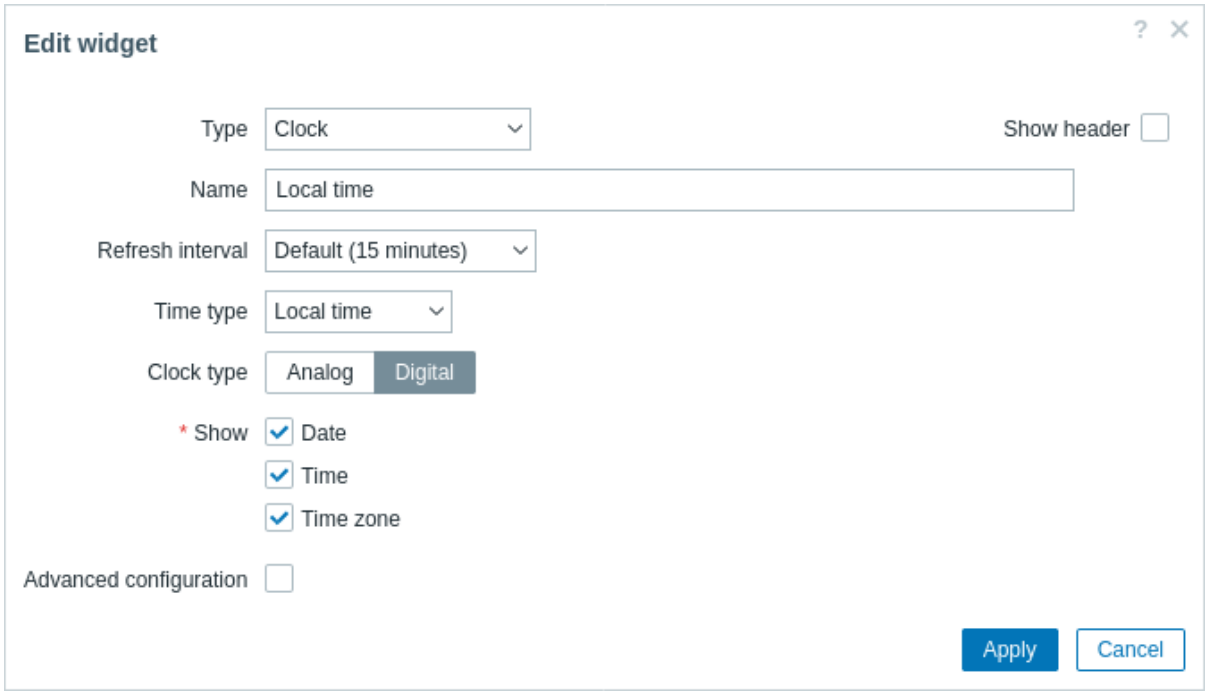
Both analog and digital clocks can be displayed:





Configuration

To configure, select *Clock* as type:



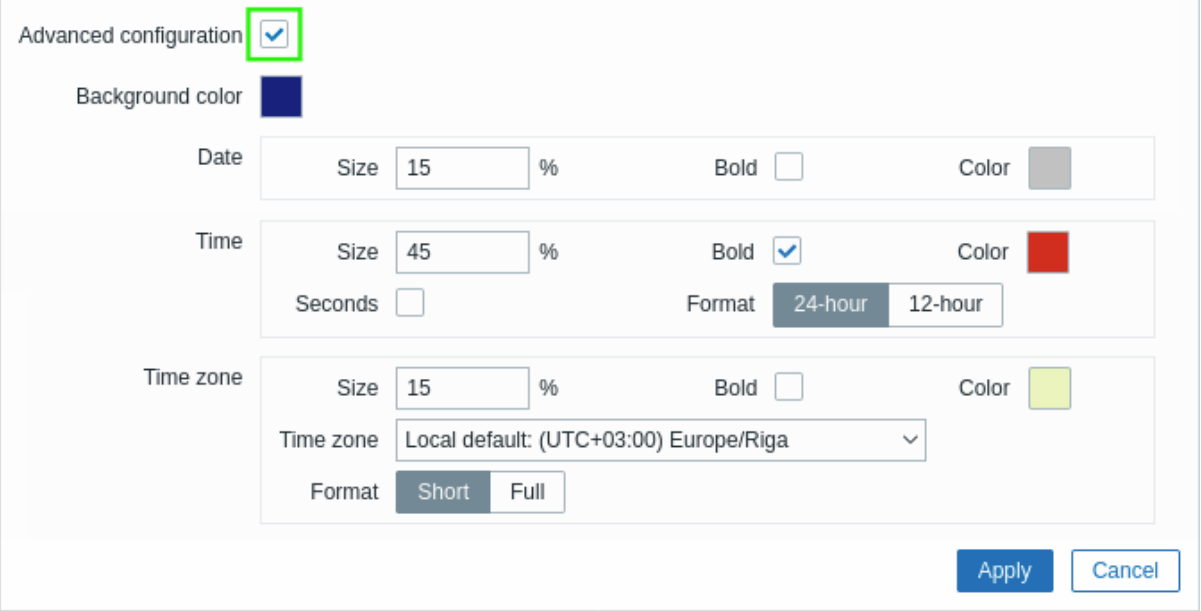
In addition to the parameters that are common for all widgets, you may set the following specific options:

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.

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.

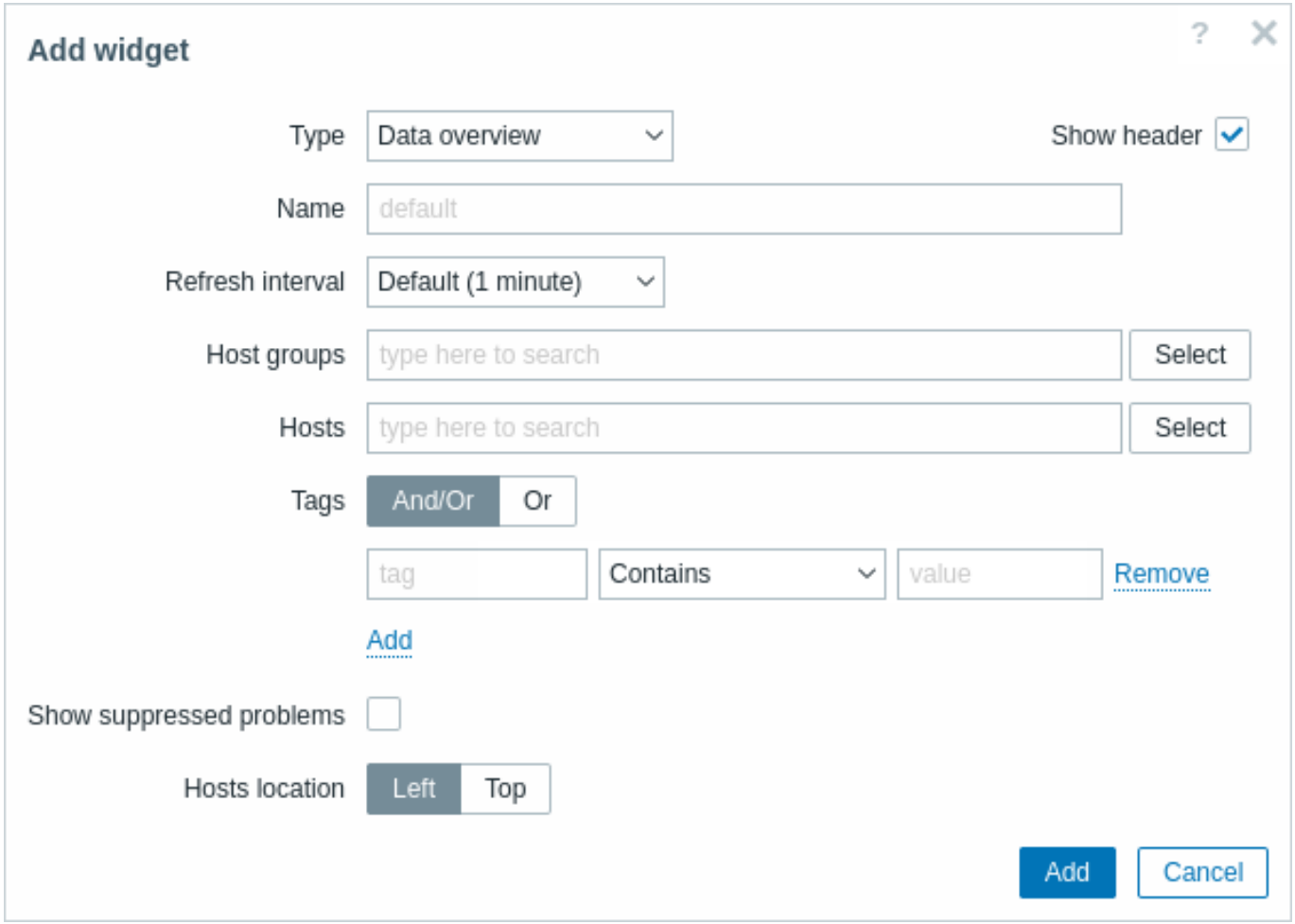
By default, only values that fall within the last 24 hours are displayed. This limit has been introduced with the aim of improving initial loading times for large pages of latest data. This limit is configurable in *Administration* → *General* → *GUI*, using the *Max history display period* option.

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:



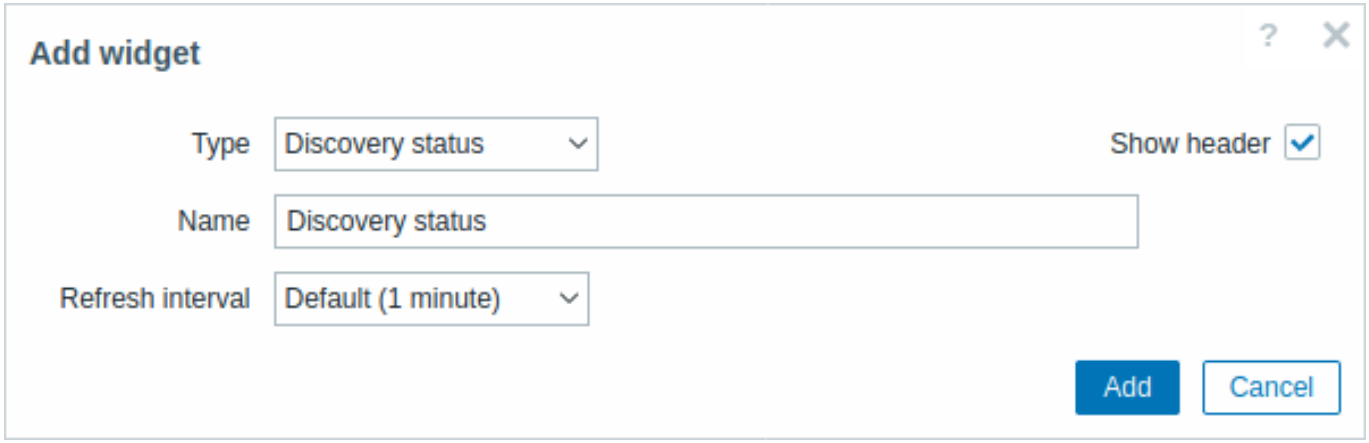
In addition to the parameters that are common for all widgets, you may set the following specific options:

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.



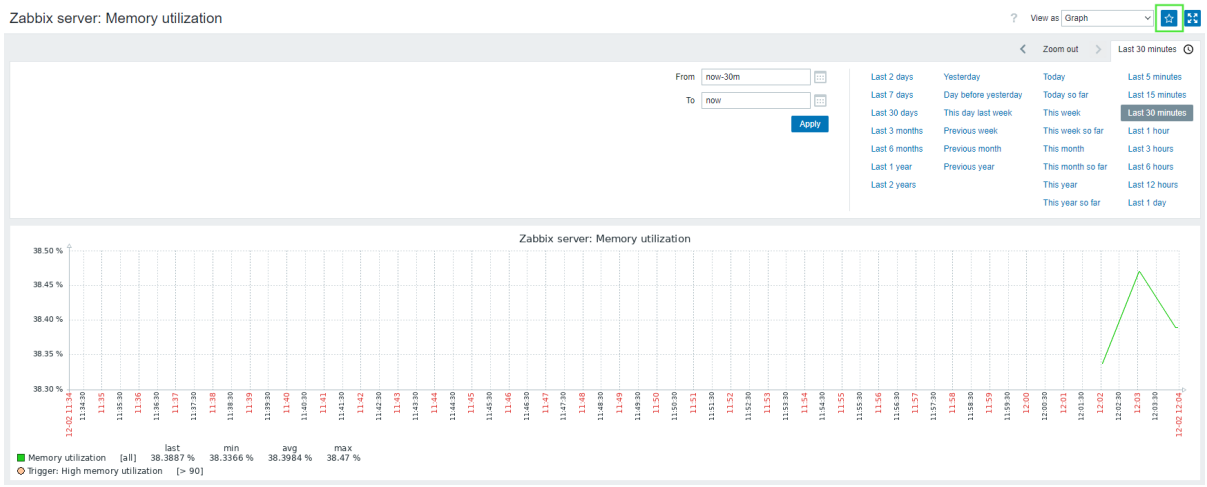
All configuration parameters are common for all widgets.

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).

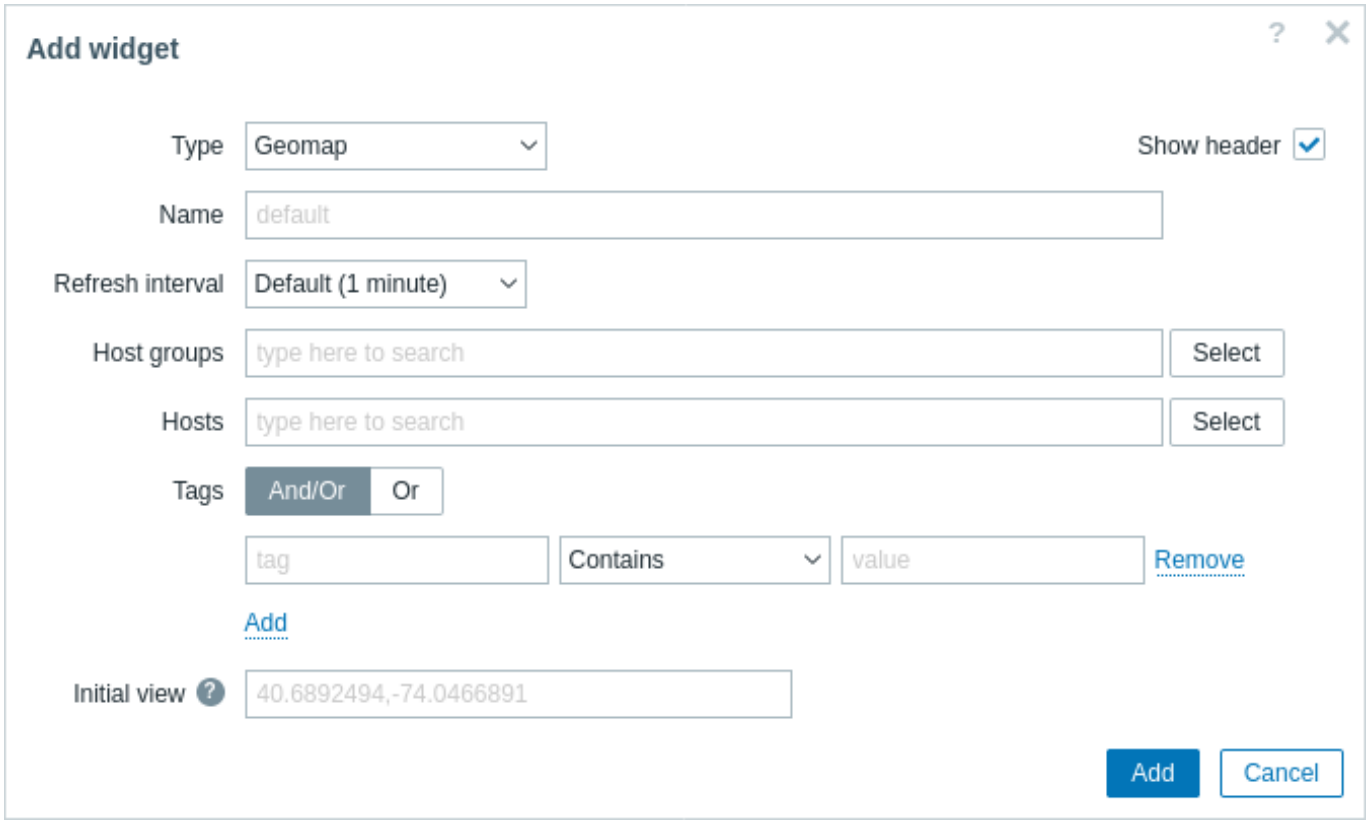
By default, the widget displays all enabled hosts with valid geographical coordinates defined in the host configuration. It is possible to configure host filtering in the widget parameters.

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.



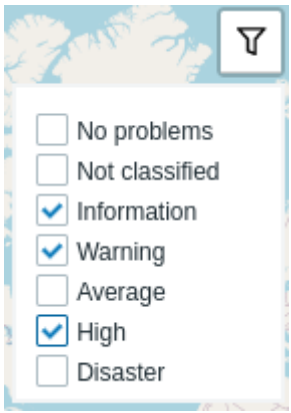
In addition to the parameters that are common for all widgets, you may set the following specific options:

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

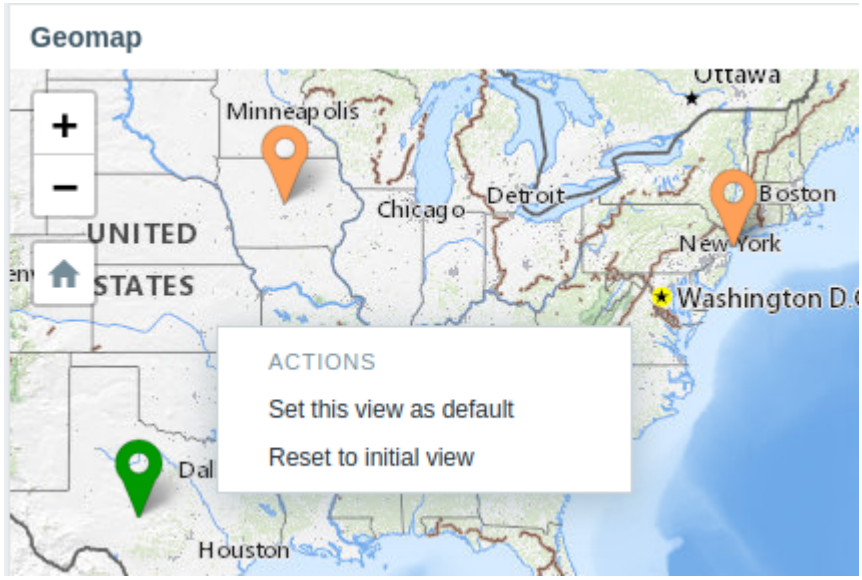
Host markers displayed on the map have the color of the host's most serious problem and green color if a host has no problems. Clicking on a host marker allows viewing the host's visible name and the number of unresolved problems grouped by severity. Clicking on the visible name will open host menu.

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.



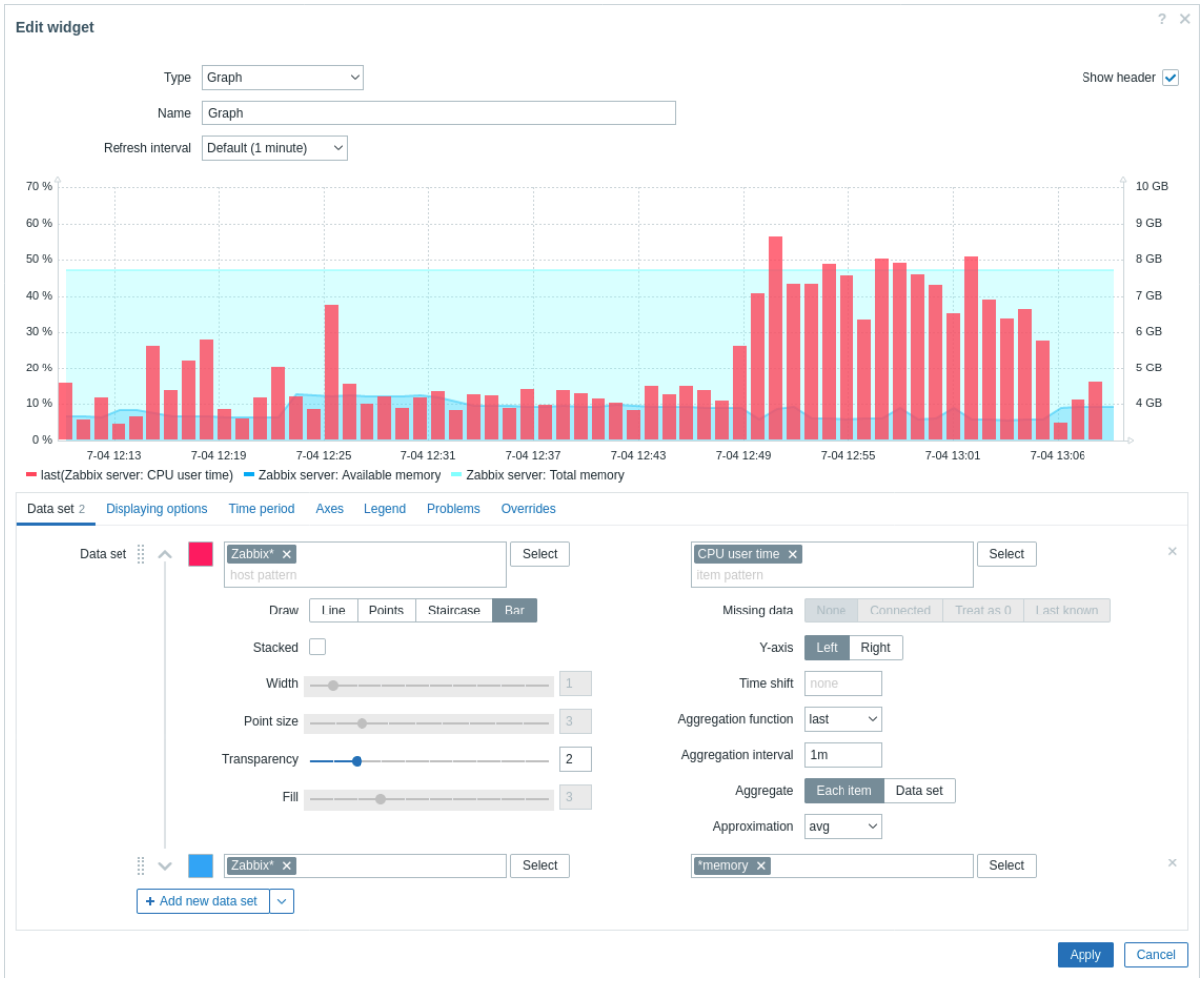
8 Graph

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).

Configuration

To configure, select *Graph* as type:



The **Data set** tab allows to add data sets and define their visual representation:

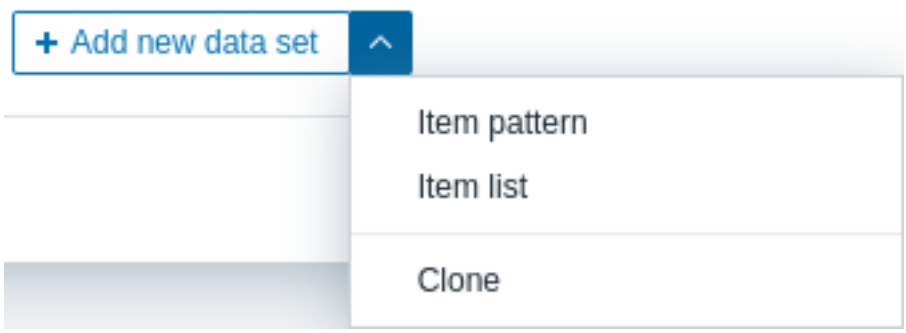
Data set	Enter the host and item patterns; data of items that match the entered patterns will be displayed on the graph. Wildcard patterns may be used (for example, * will return results that match zero or more characters). To specify a wildcard pattern, just enter the string manually and press *Enter*. While you are typing, note how all matching hosts are displayed in the dropdown. Up to 50 items may be displayed in the graph. Host pattern and item pattern fields are mandatory. 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). **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 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).

Existing data sets are displayed in a list. You may:

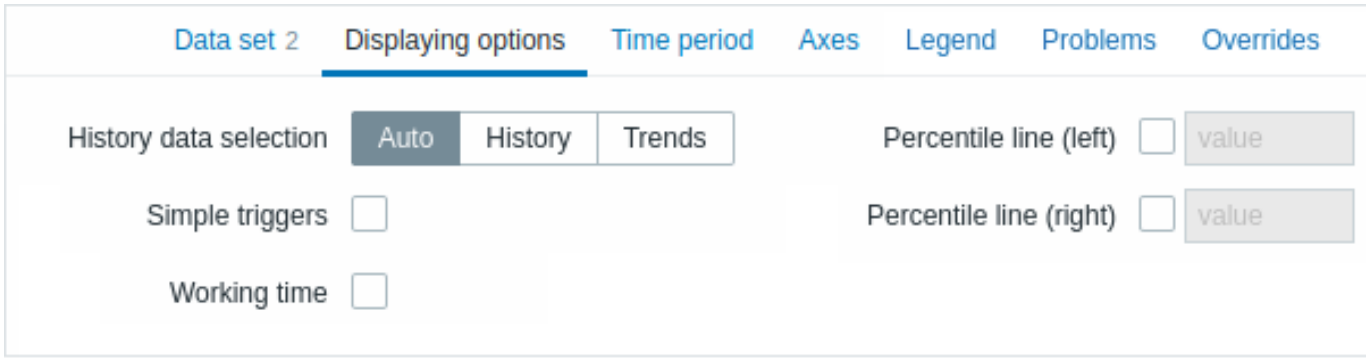
-  - click on the move icon and drag a data set to a new place in the list
-  - click on the expand icon to expand data set details. When expanded, this icon turns into a collapse icon.
- - click on the color icon to change the base color, either from the color picker or manually. The base color is used 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.

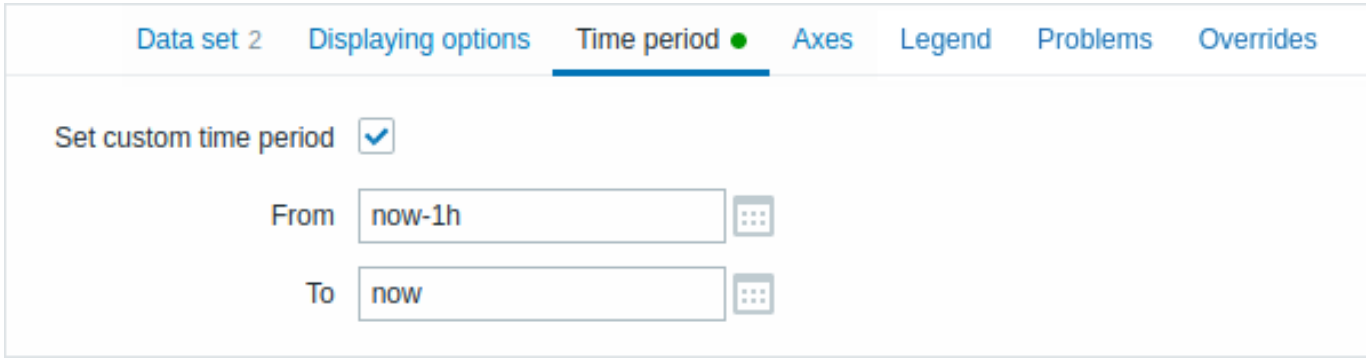


The **Displaying options** tab allows to define history data selection:



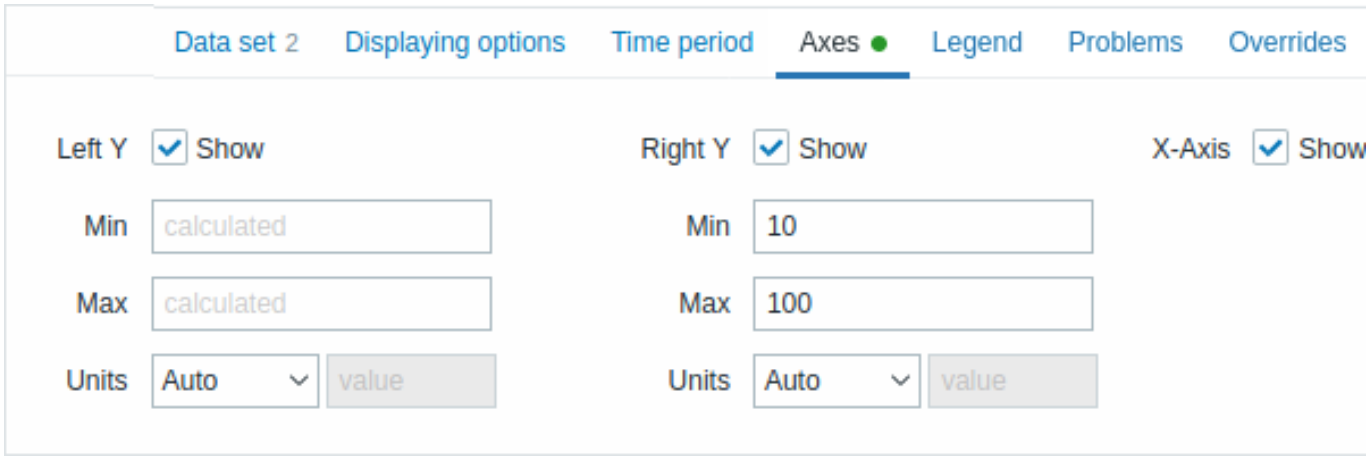
History data selection	Set the source of graph data: **Auto** - data are sourced according to the classic graph algorithm (default) **History** - data from history **Trends** - data from trends
Simple triggers	Mark the checkbox to show the trigger thresholds for simple triggers. The thresholds will be 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.

The **Time period** tab allows to set a custom time period:



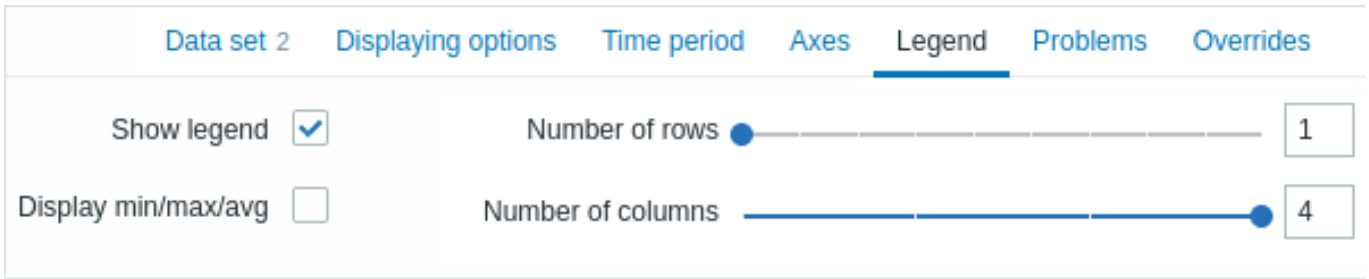
Set custom time period	Mark this checkbox to set the custom time period for the graph (unmarked by default).
From	Set the start time of the custom time period for the graph.
To	Set the end time of the custom time period for the graph.

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.

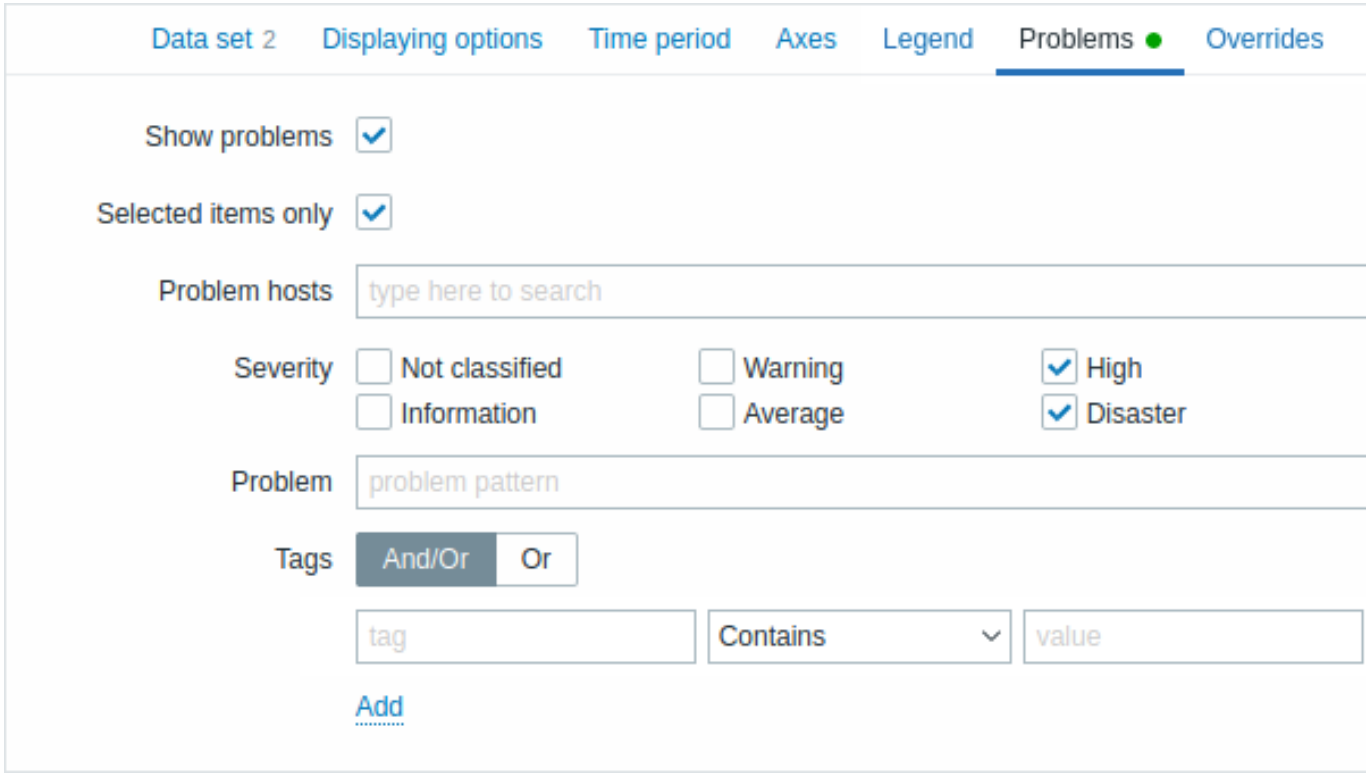
The **Legend** tab allows to customize the graph legend:



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.

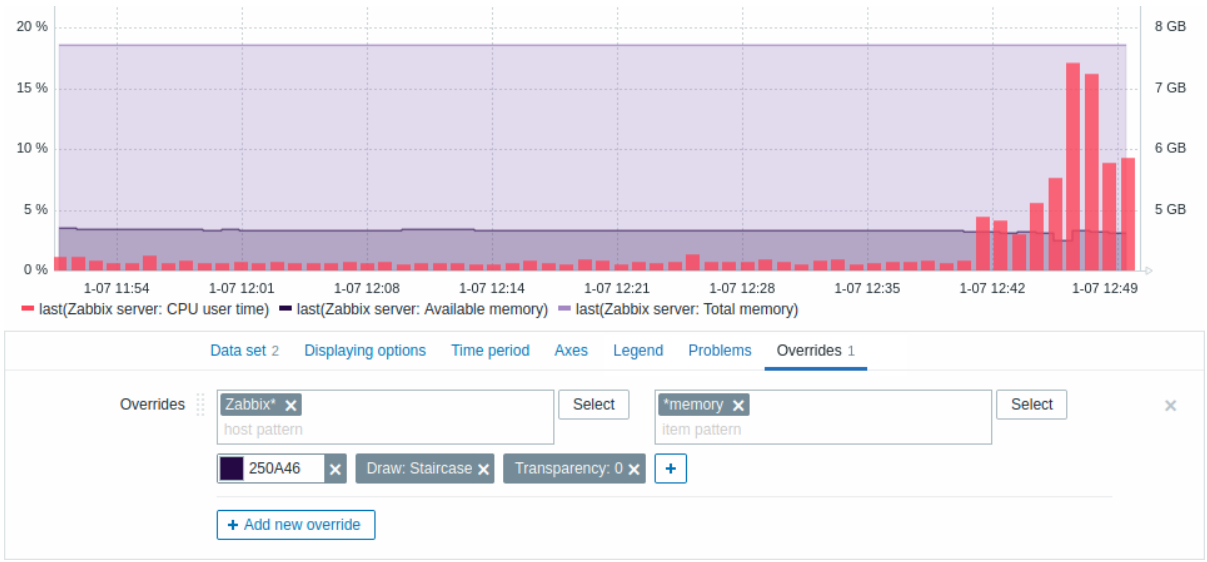
Number of rows	Set the number of legend rows to be displayed.
Number of columns	Set the number of legend columns to be displayed.

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 the problem severities to be displayed on the graph.
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

The **Overrides** tab allows to add custom overrides for data sets:

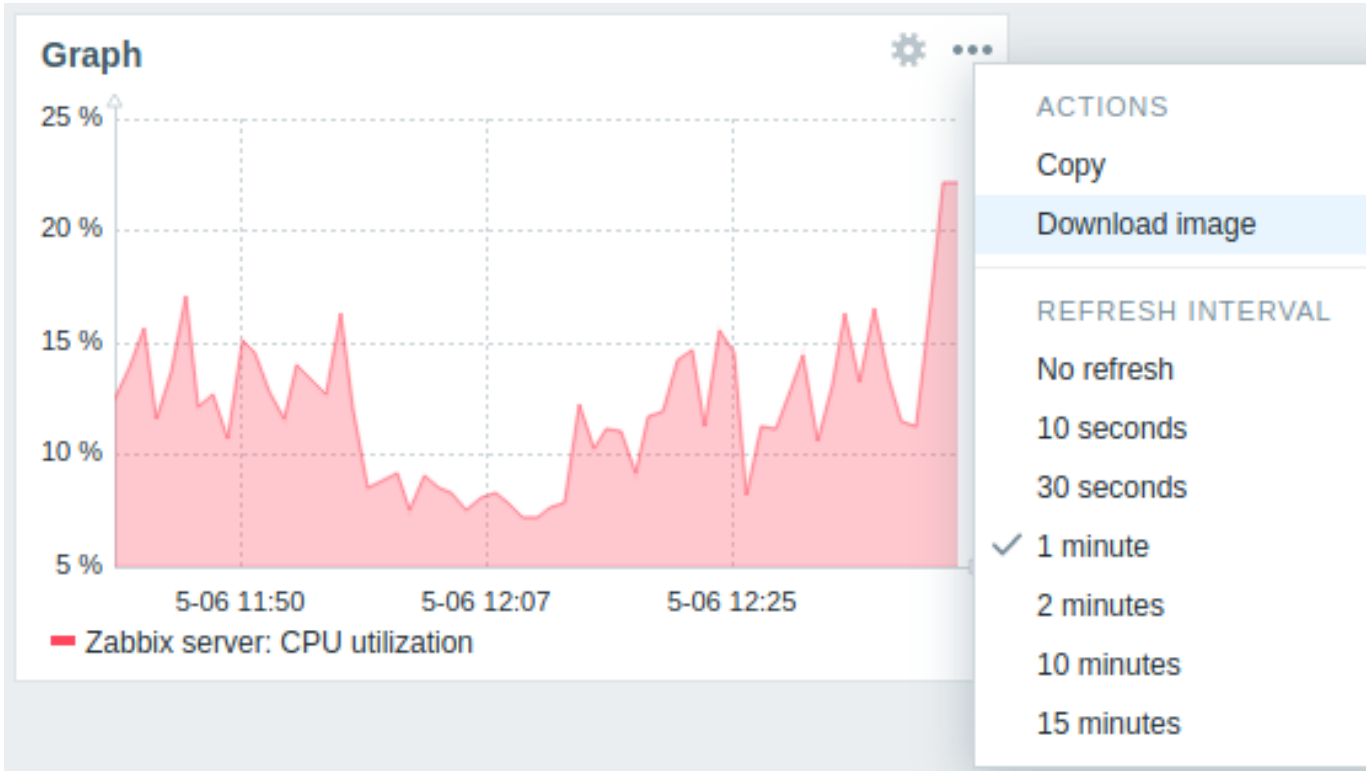


Overrides are useful when several items are selected for a data set using the * wildcard and you want to change how the items are displayed by default (e.g. default base color or any other property).

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.

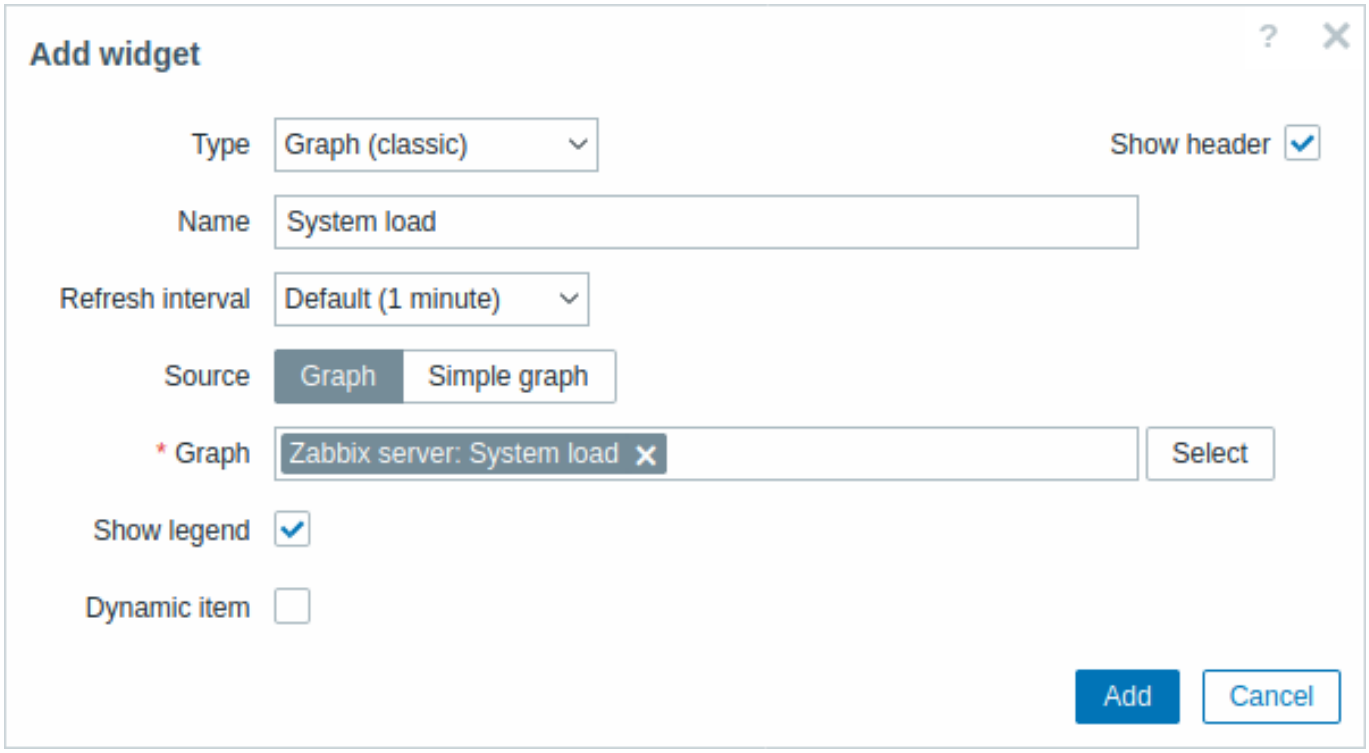
9 Graph (classic)

Overview

In the classic graph widget, you can display a single custom graph or simple graph.

Configuration

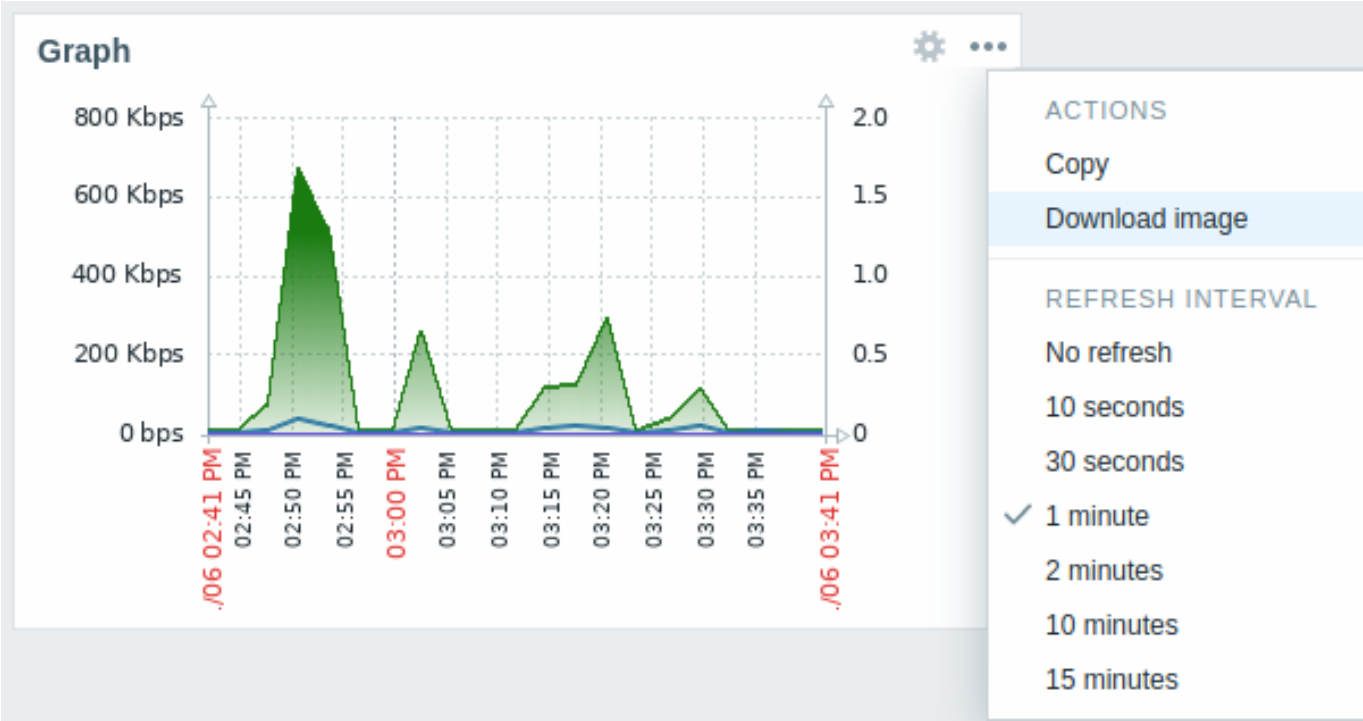
To configure, select *Graph (classic)* as type:



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).
Dynamic item	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.

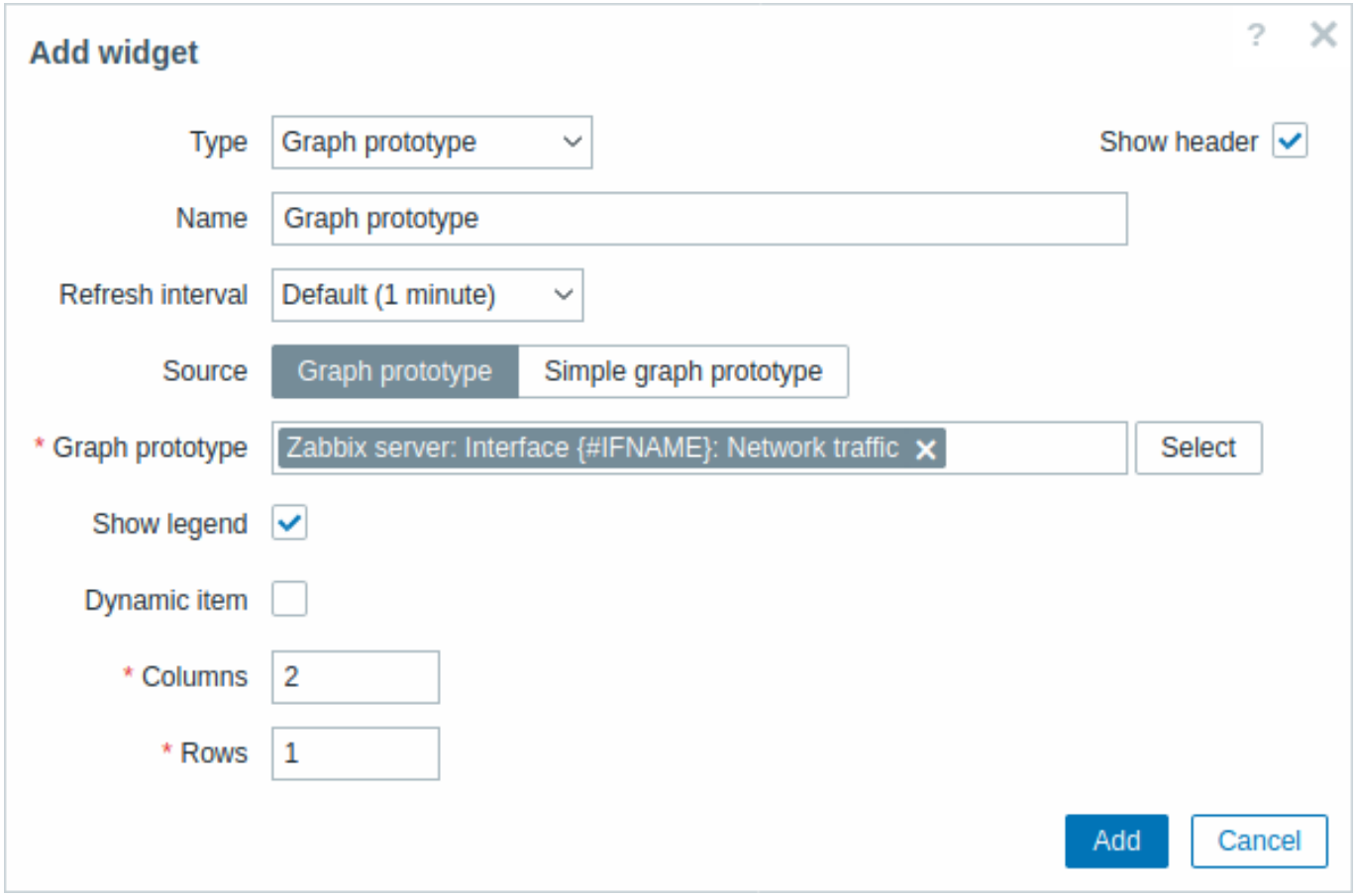
10 Graph prototype

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

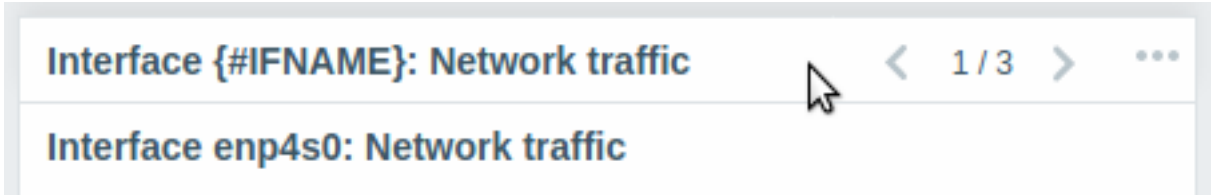
To configure, select *Graph prototype* as widget type:



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).
Dynamic item	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).



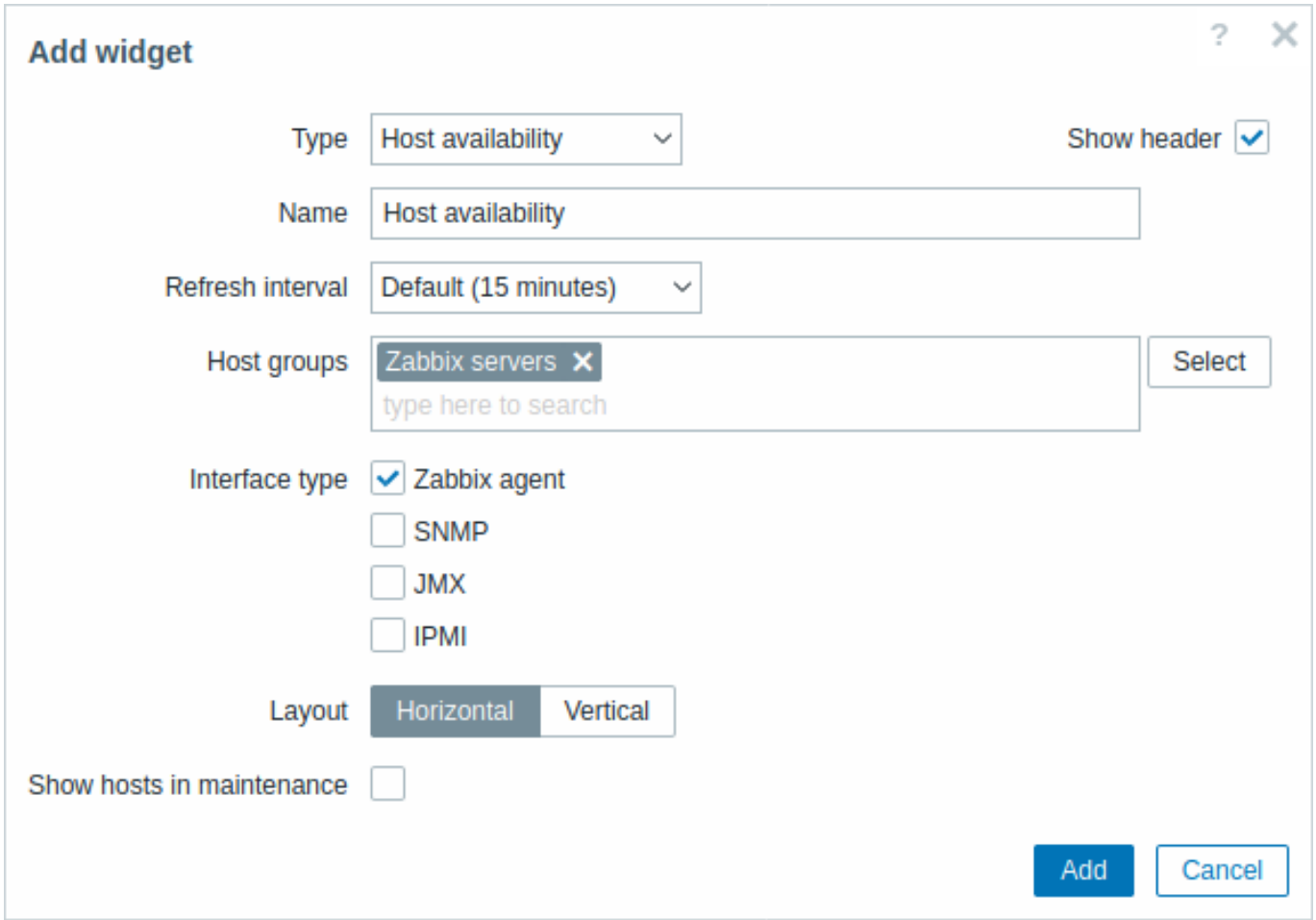
Vertical display (lines).

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:



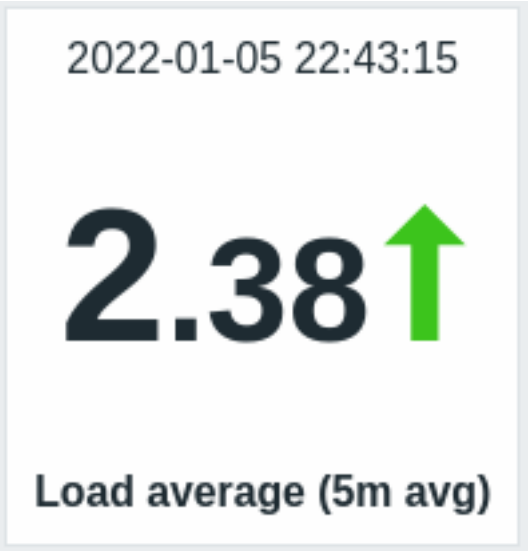
In addition to the parameters that are common for all widgets, you may set the following specific options:

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.



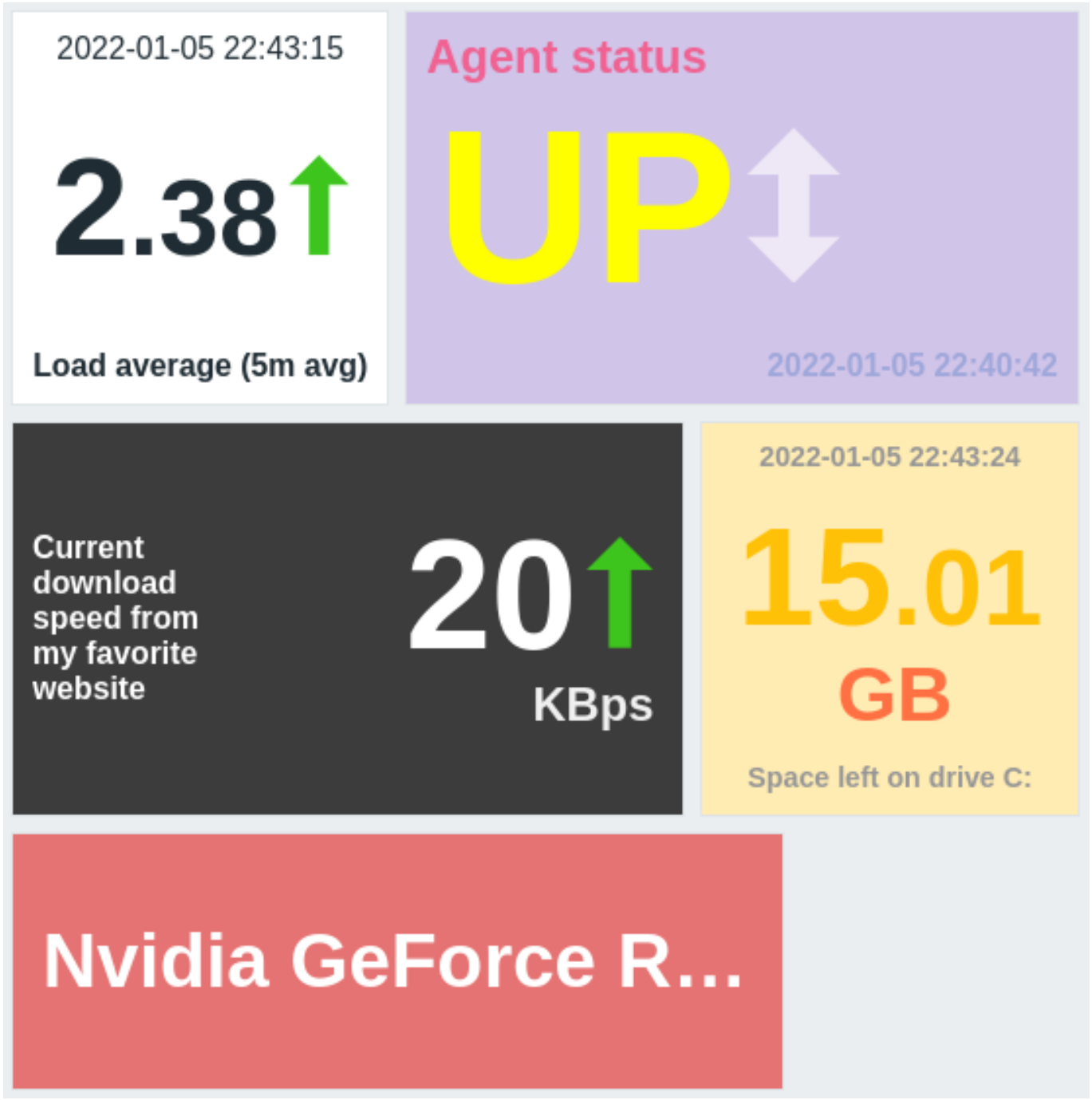
Besides the value itself, additional elements can be displayed, if desired:

- time of the metric
- item description
- change indicator for the value
- 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.

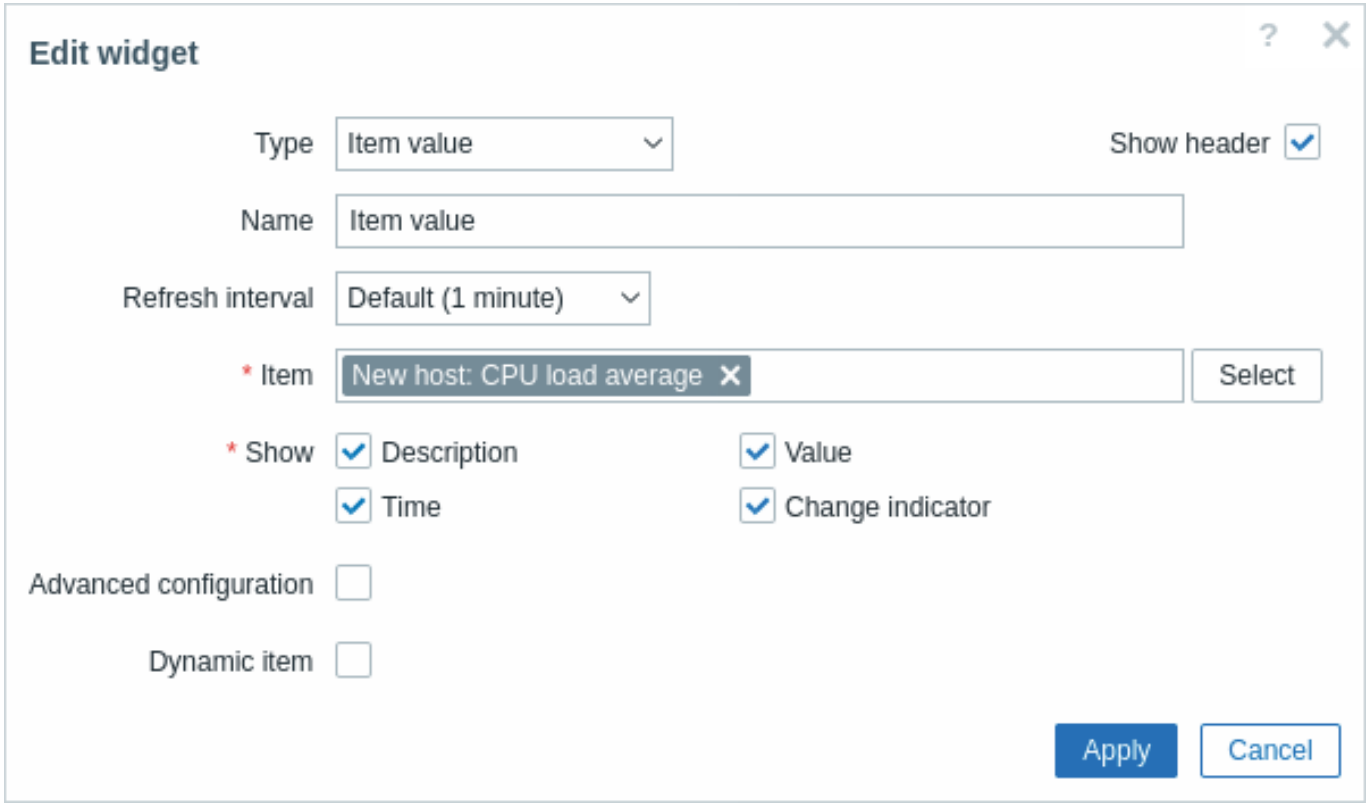
Clicking on the value leads to an ad-hoc graph for numeric items or latest data for string items.

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.
Dynamic item	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 for the whole widget.



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 prioritised; 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	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	Time is the 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.

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

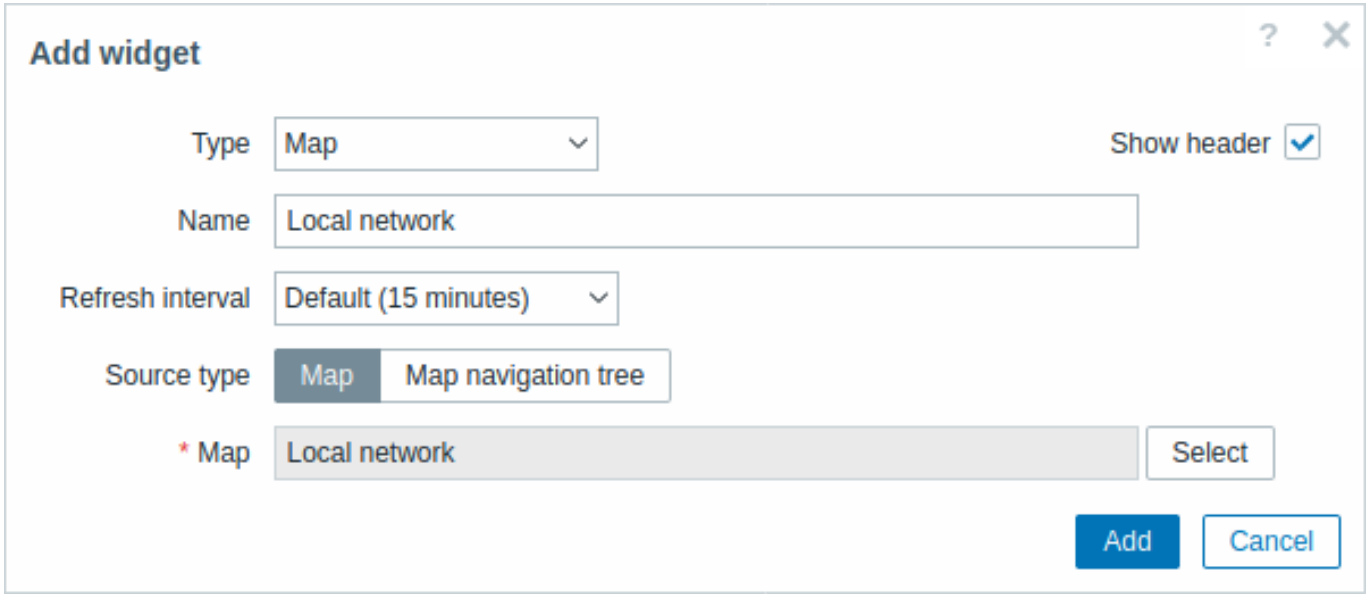
Overview

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 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*.

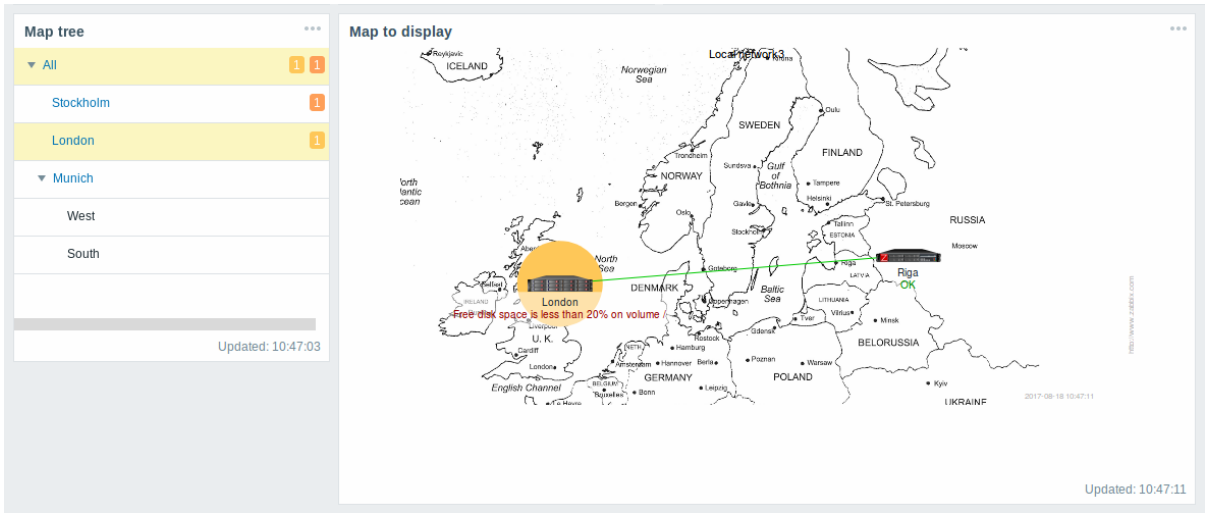
See also: known issue with IE11

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.

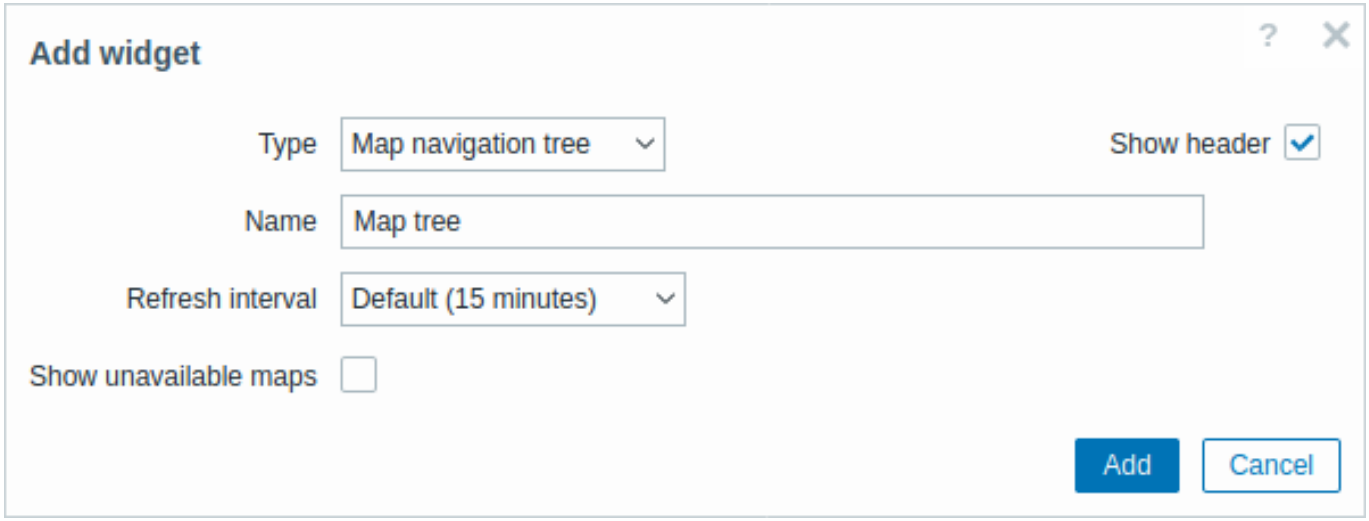
It becomes even more powerful if you link the *Map* widget to the navigation tree. In this case, clicking on a map name in the navigation tree displays the map in full in the *Map* widget.



Statistics with the top-level map in the hierarchy display a sum of problems of all sub-maps and their own problems.

Configuration

To configure the navigation tree widget, 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 sub-maps are displayed even if the parent level map is unavailable. If unmarked, available sub-maps to an unavailable parent map will not be displayed at all. Problem count is calculated based on available maps and available map elements.

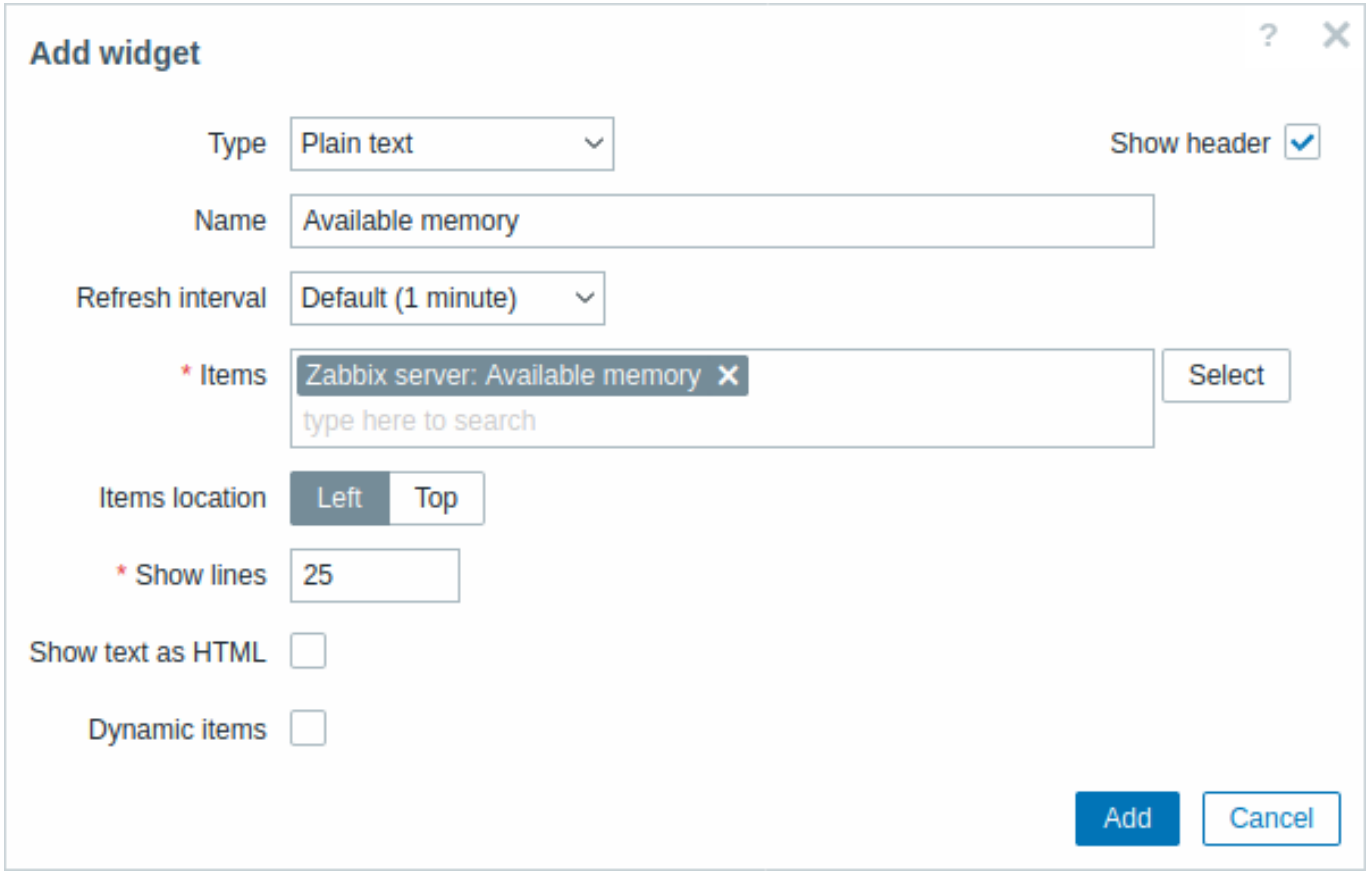
15 Plain text

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.
Dynamic item	Set to display different data depending on the selected host.

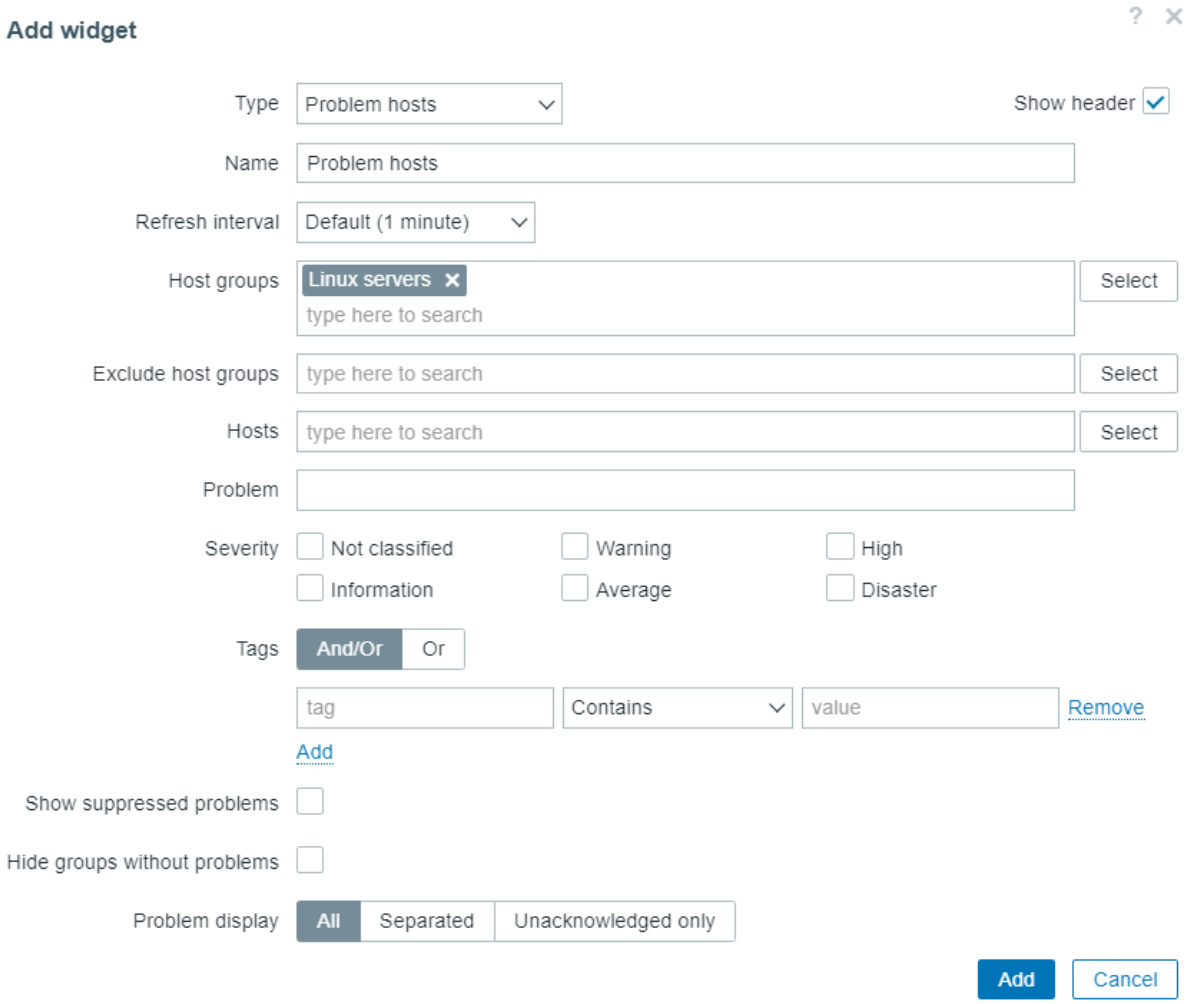
16 Problem hosts

Overview

In the problem host widget, you can display high-level information about host availability.

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:

Parameter	Description
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 the problem severities to be displayed in the widget.

Parameter	Description
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.

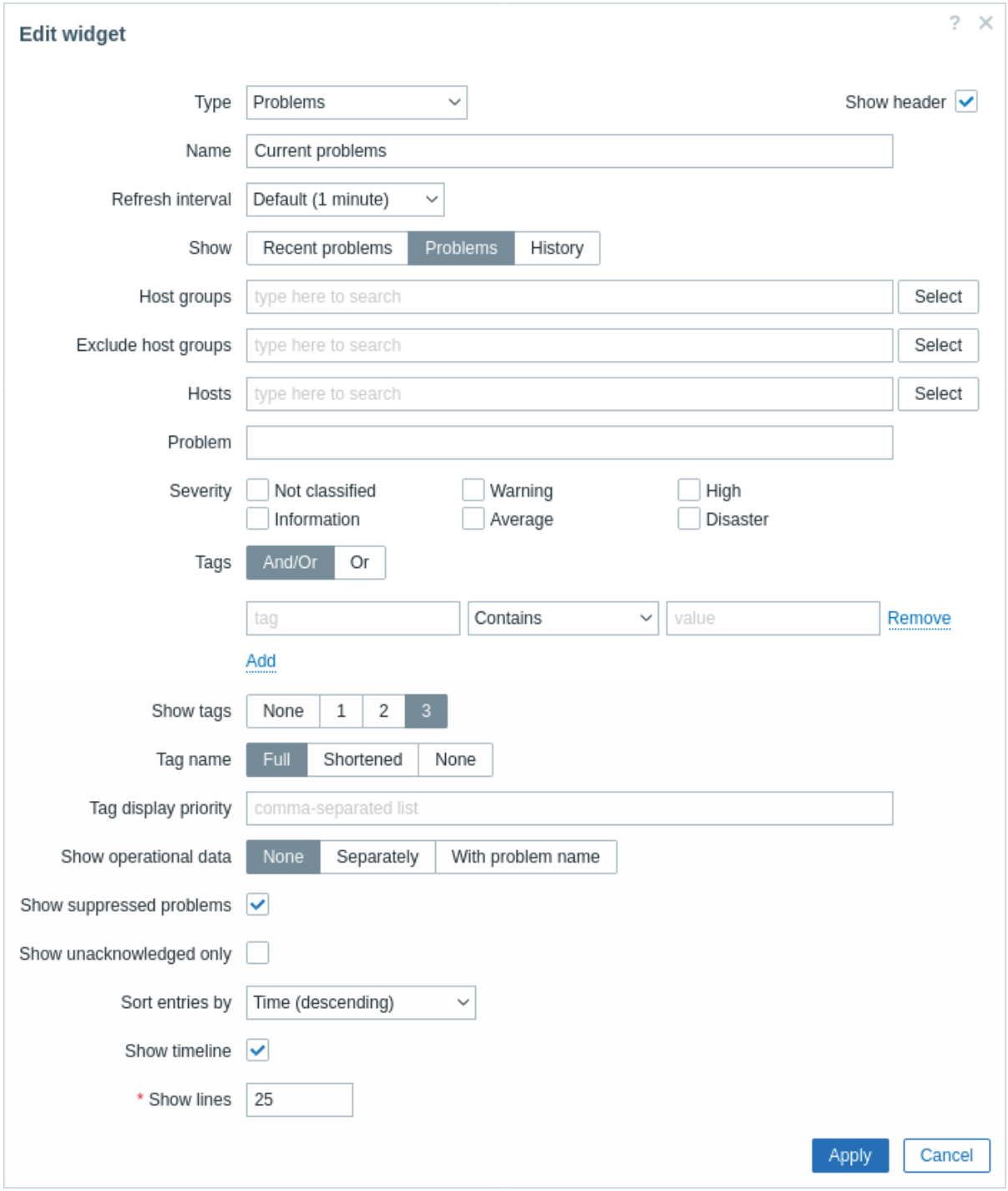
17 Problems

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.

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	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.

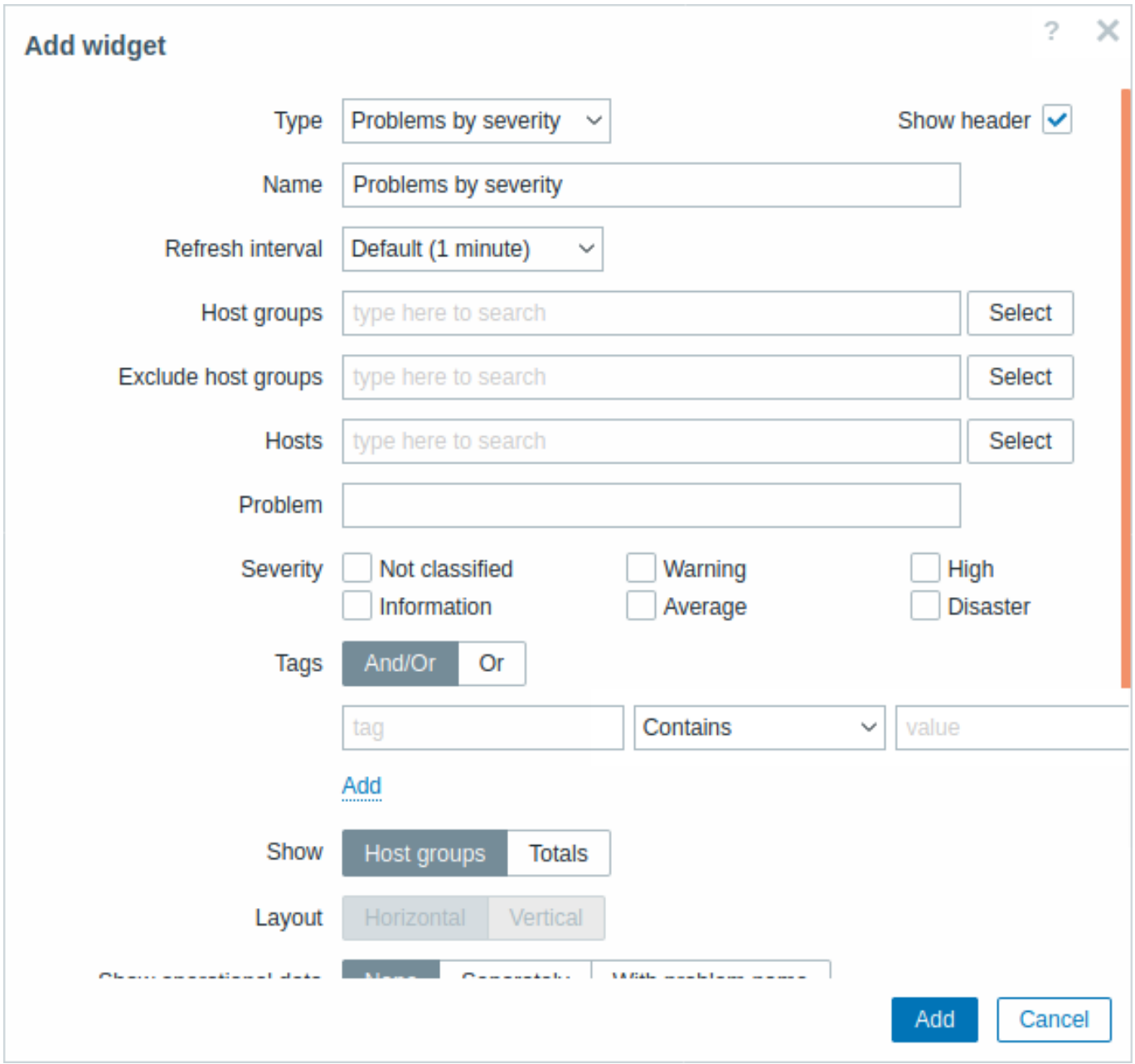
Parameter	Description
Exclude host groups	Enter host groups to hide problems of from the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Problems from these 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 the problem severities to be displayed in the widget.
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 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.

Overview

In this widget, you can display problems by severity. You can limit what hosts and triggers are displayed in the widget and define how the problem count is displayed.

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:

Parameter	Description
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.

Parameter	Description
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 the problem severities to be displayed in the widget.
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.

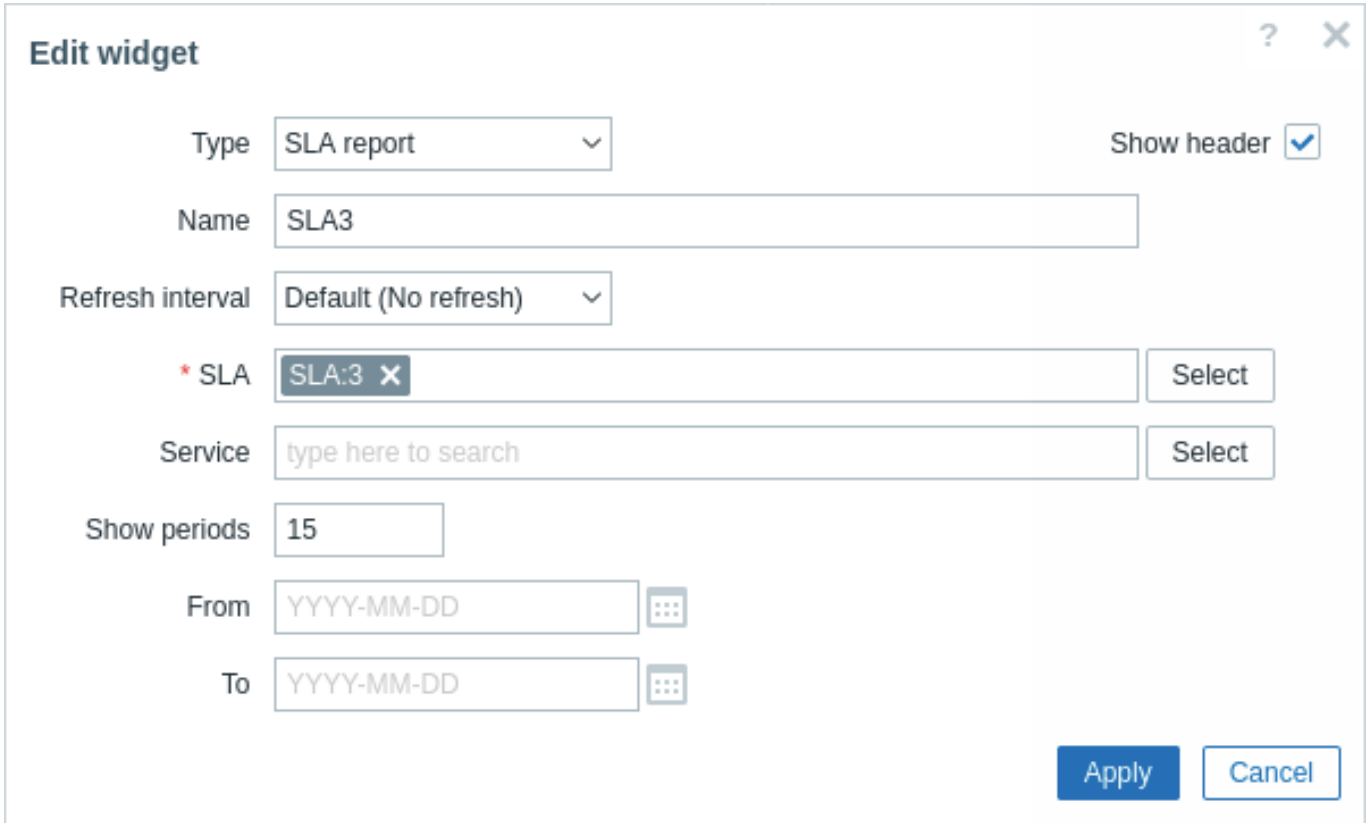
19 SLA report

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.

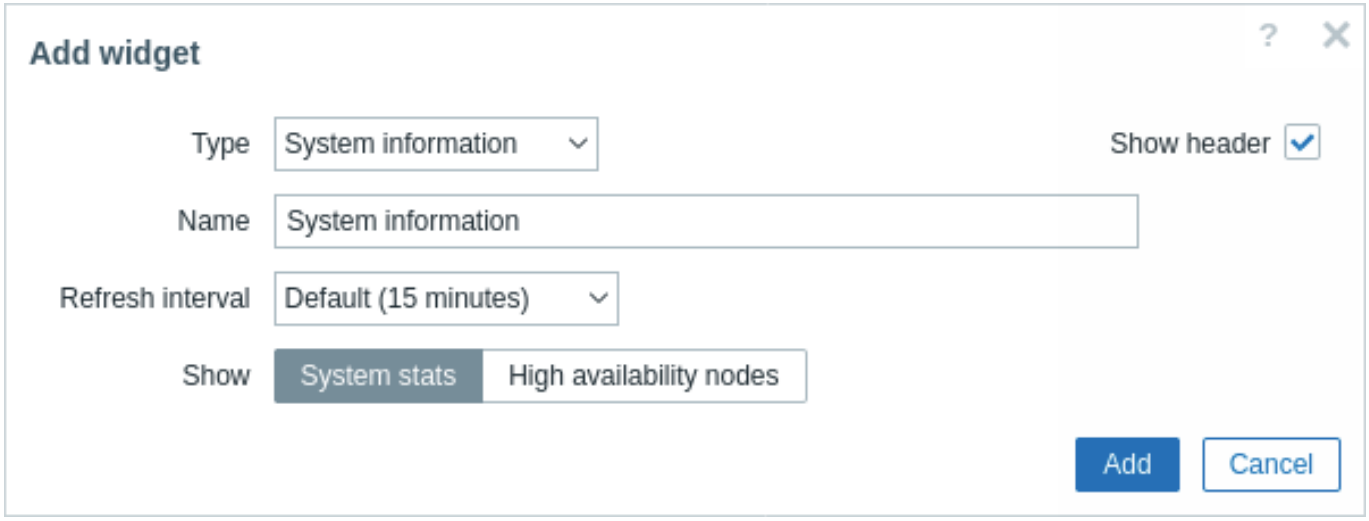
20 System information

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:



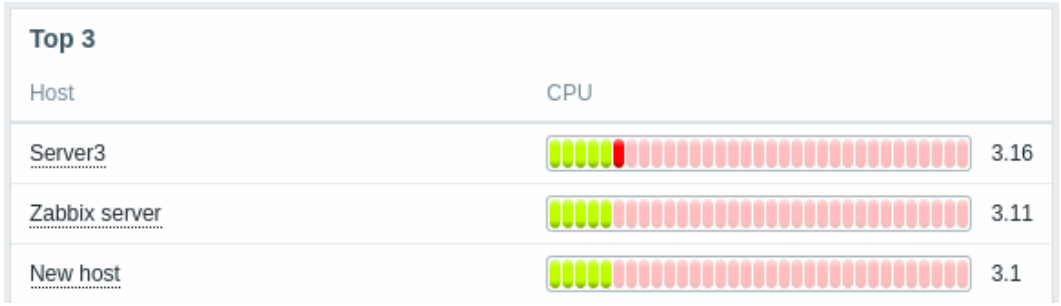
All configuration parameters are common for all widgets.

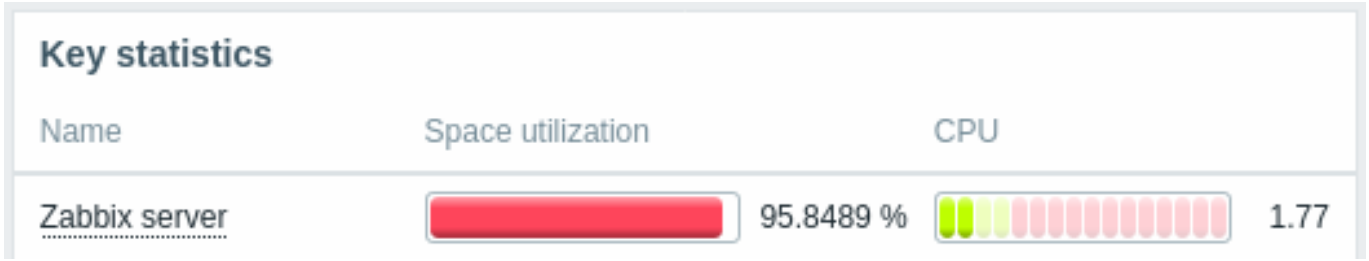
21 Top hosts

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.

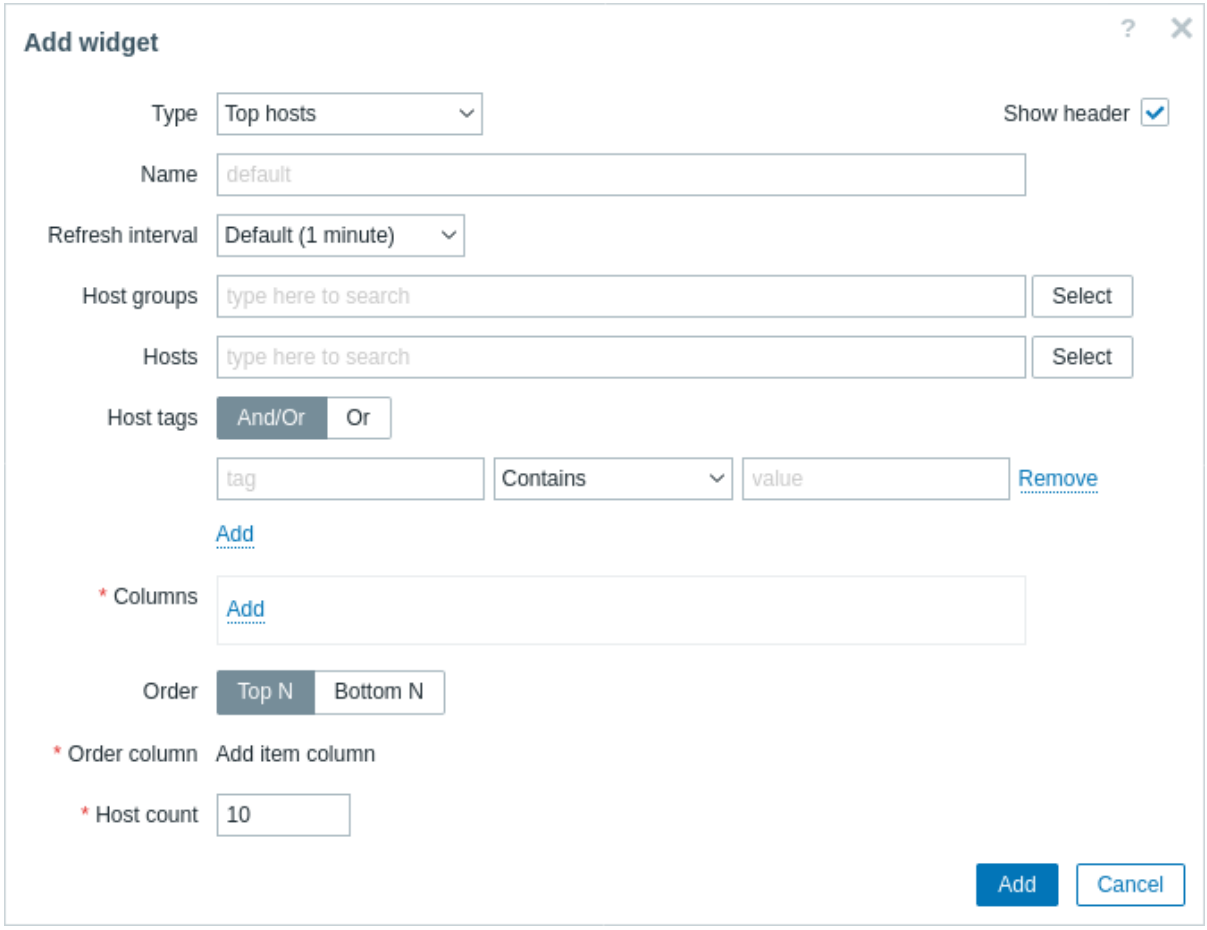
The maximum number of hosts that can be displayed is 100.





Configuration

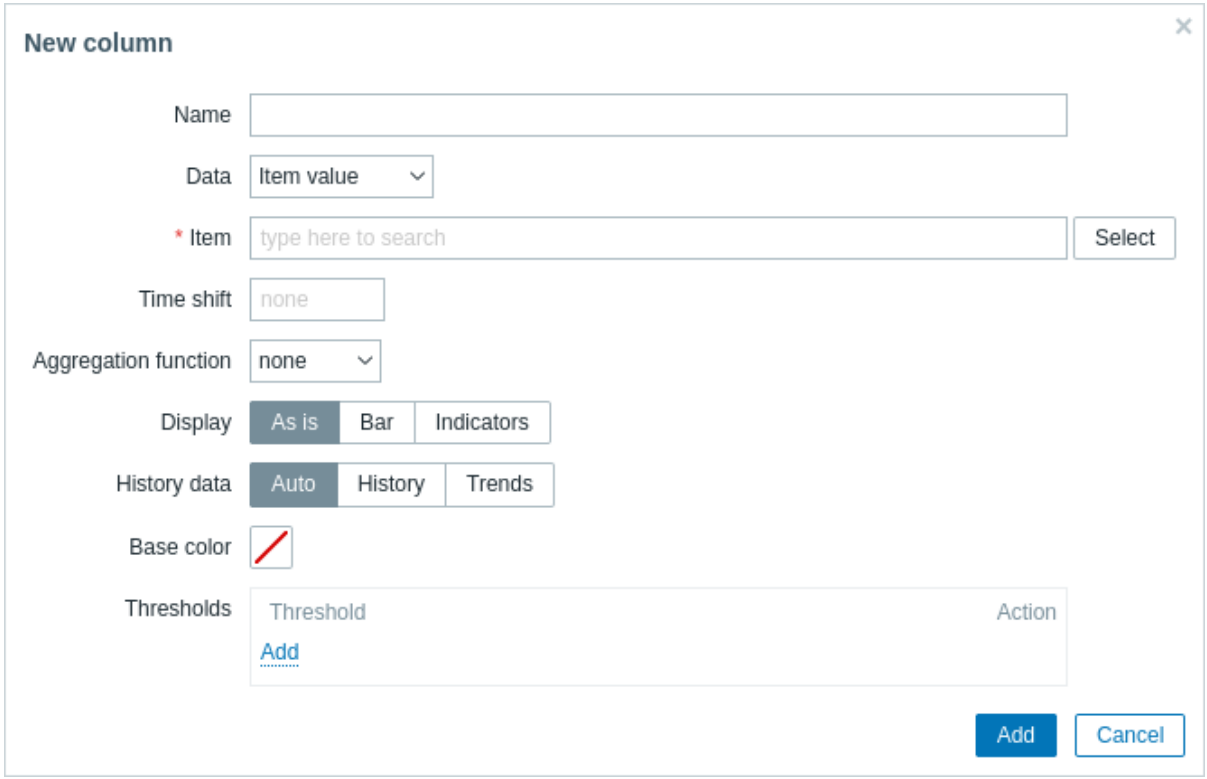
To configure, select *Top hosts* as type:



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.
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.

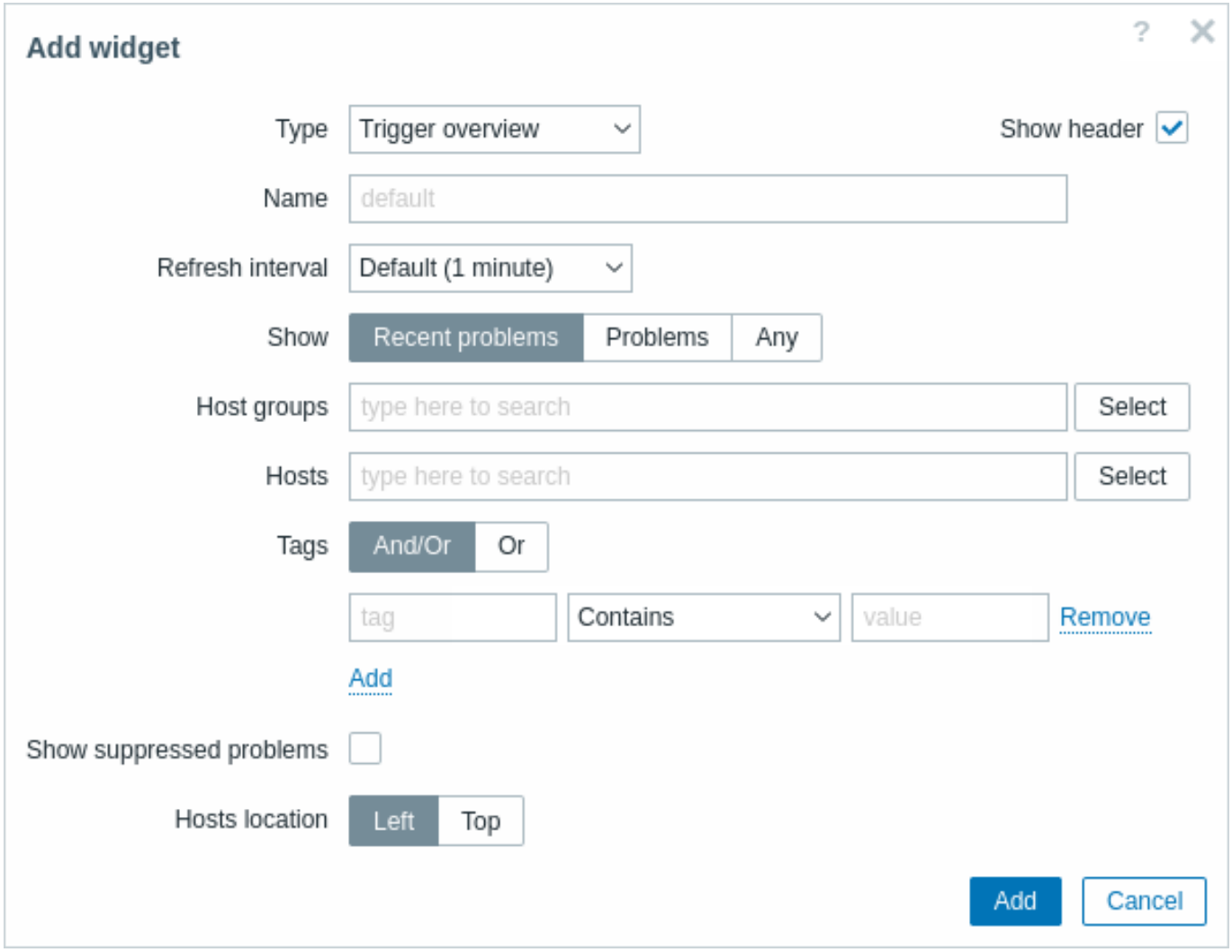
- The trigger states are displayed as colored blocks (the color of the blocks for PROBLEM triggers depends on the problem severity color, which can be adjusted in the problem update screen). Note that recent trigger state changes (within the last 2 minutes) will be displayed as blinking blocks.
- Gray 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.

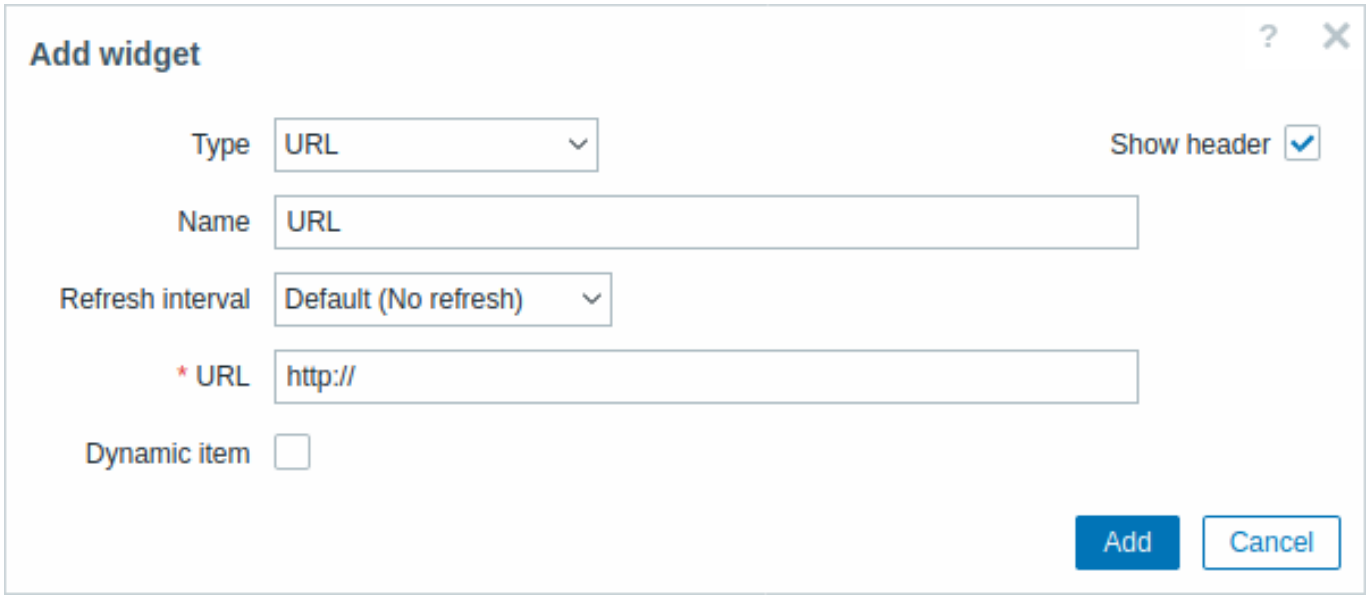
23 URL

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. Relative paths are allowed since Zabbix 4.4.8. {HOST.*} macros are supported.

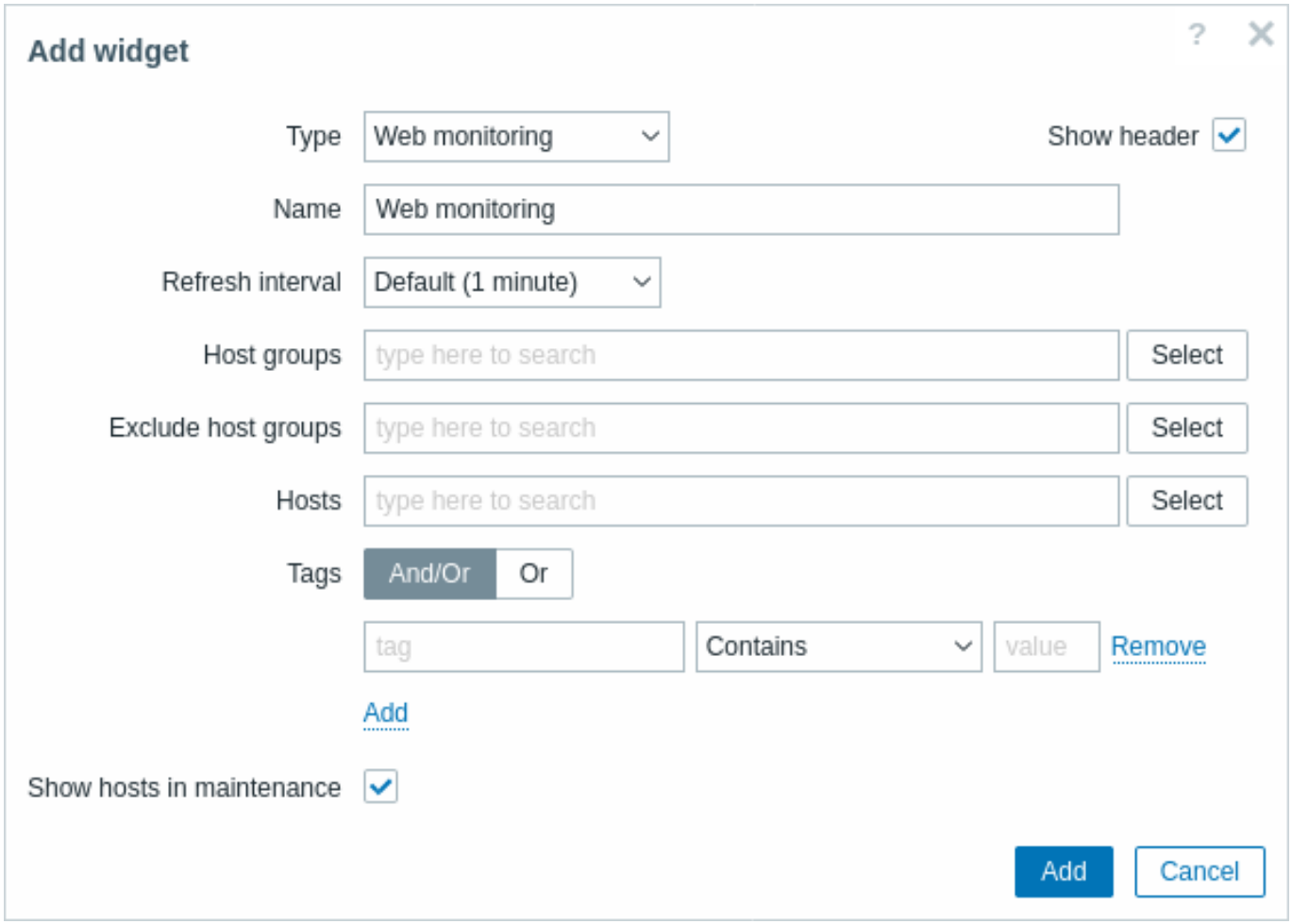
Dynamic item	Set to display different URL content depending on the selected host. This can work if {HOST.*} macros are used in the URL.

Attention:
Browsers might not load an HTTP page included in the widget if Zabbix frontend is accessed over HTTPS.

Overview

This widget displays a status summary of the active web monitoring scenarios.

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:

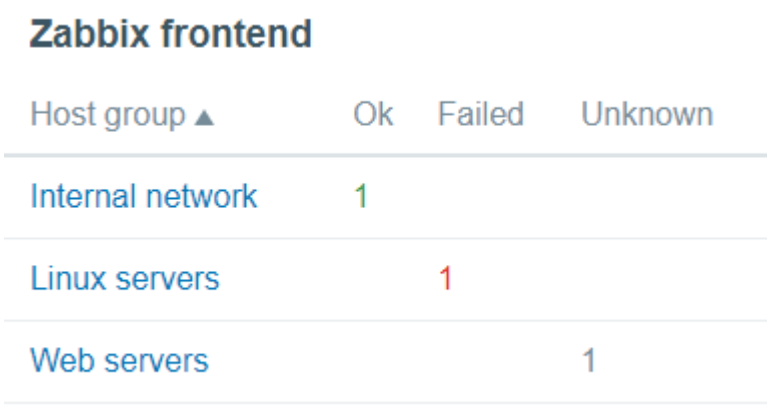
Parameter	Description
Host groups	Enter host groups to display in the widget. This field is auto-complete so starting to type the name of a group will offer a dropdown of matching groups. Specifying a parent host group implicitly selects all nested host groups. Host data from these host groups will be displayed in the widget. If no host groups are entered, all host groups will be displayed.

Parameter	Description
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

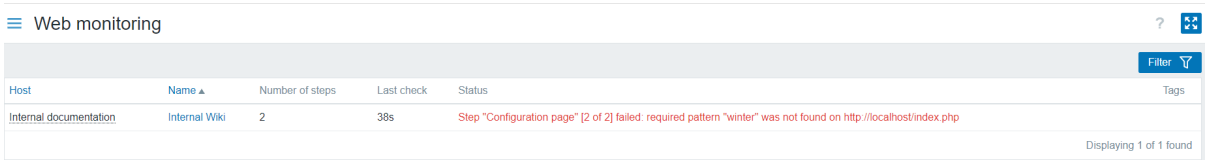
Once you have completed the configuration, you might like to see the widget with the data it displays. To do it, go to *Monitoring* → *Dashboards*, click on the name of a dashboard where you created the widget.

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.”

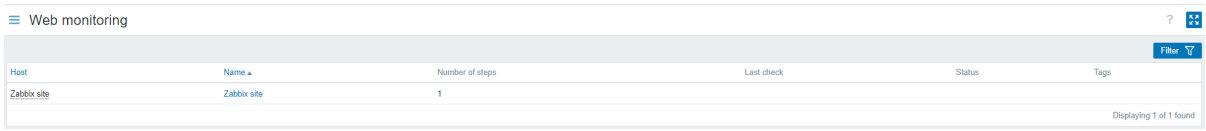


A web monitoring widget displays the following information:

- 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.



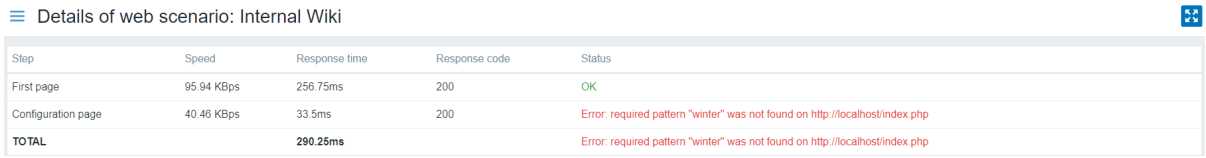
- *Unknown* - displays a number of web scenarios (in grey color) for which Zabbix has neither collected data, nor has an information about the failed steps.



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.

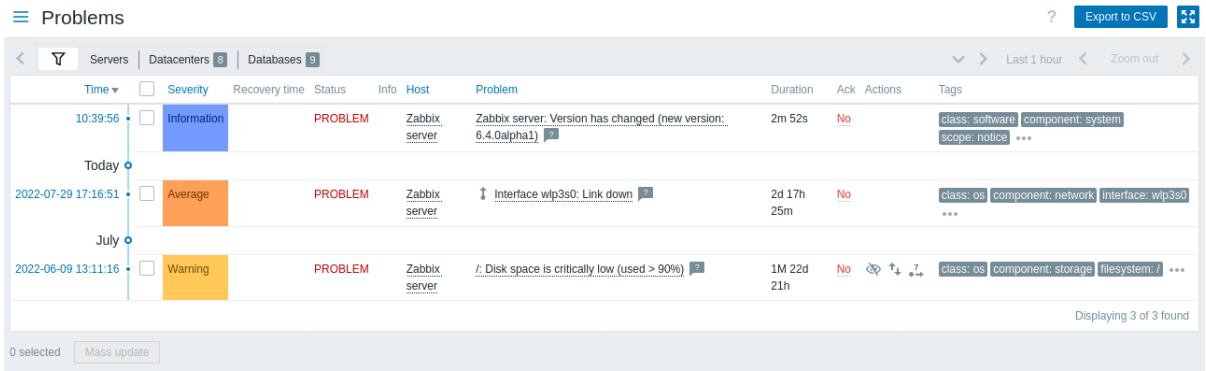
To view the details in the case of *Unknown* status, you can repeat the same steps as explained for *Failed*.

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.

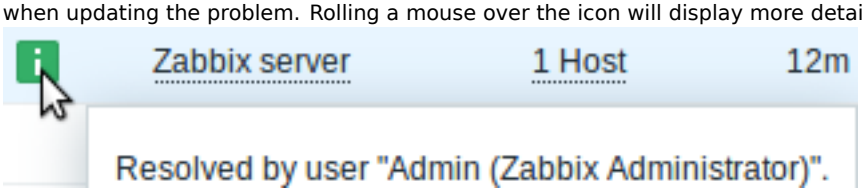
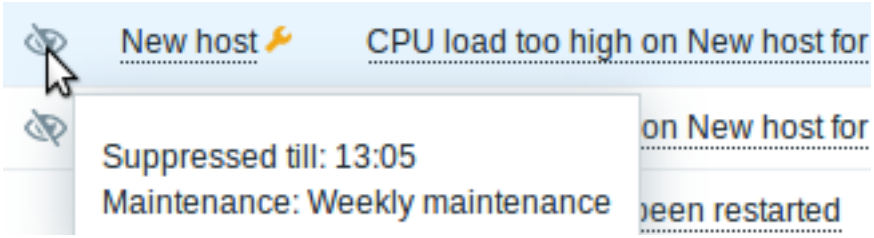
2 Problems

Overview

In *Monitoring* → *Problems* you can see what problems you currently have. Problems are those triggers that are in the ”Problem” state.



Column	Description
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.

Column	Description
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.
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
Ack	The acknowledgment status of the problem is displayed: **Yes** - green text indicating that the problem is acknowledged. A problem is considered to be acknowledged if all events for it are acknowledged. **No** - a red link indicating unacknowledged events. If you click on the link you will be taken to the problem update screen where various actions can be taken on the problem, including commenting and acknowledging the problem.

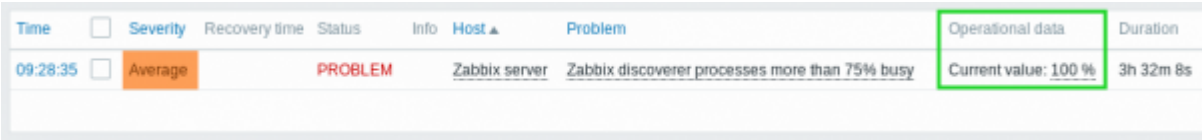
Column	Description
Actions	History of activities about the problem is displayed using symbolic icons:  - comments have been made. The number of comments is also displayed.  - problem severity has been increased (e.g. Information → Warning)  - problem severity has been decreased (e.g. Warning → Information)  - 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

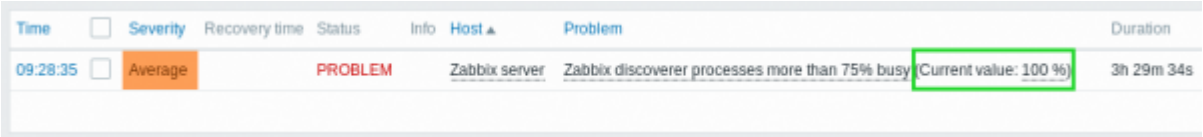
It is possible to display operational data for current problems, i.e. the latest item values as opposed to the item values at the time of the problem.

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).

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:

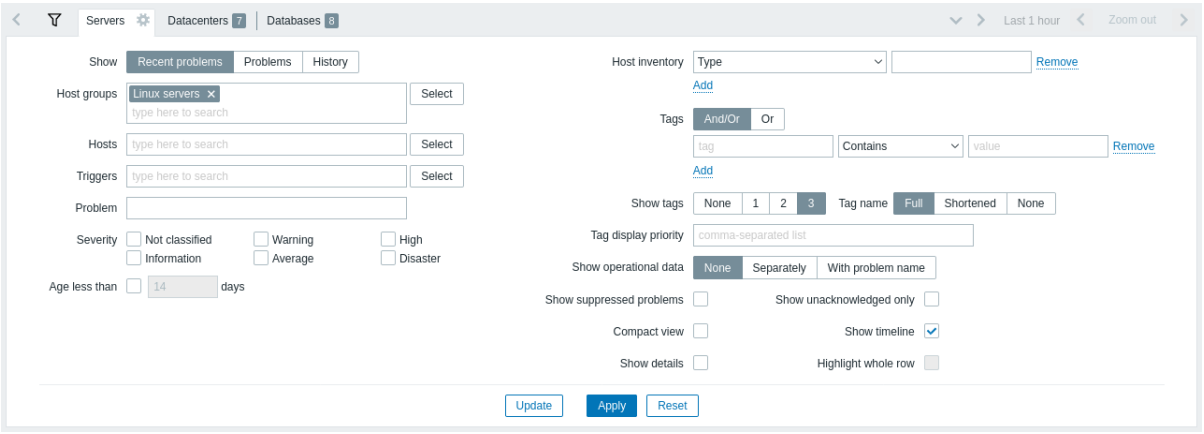
Export to CSV	Export content from all pages to a CSV file.

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.



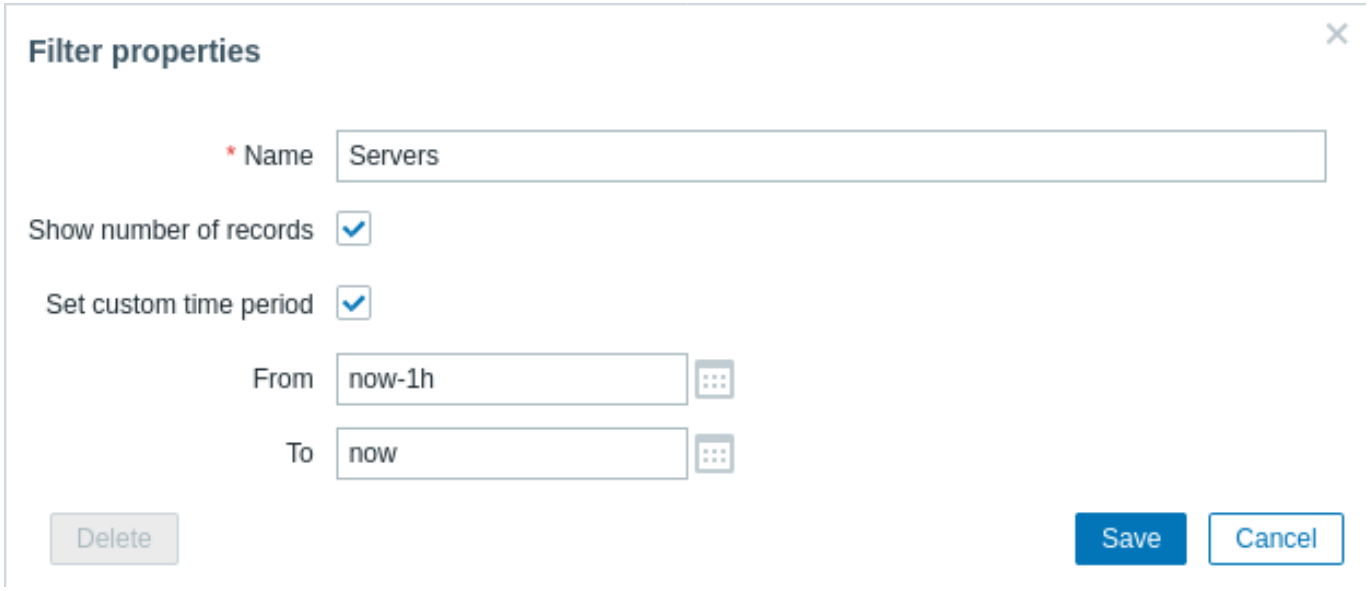
Parameter	Description
Show	Filter by problem status: **Recent problems** - unresolved and recently resolved problems are displayed (default) **Problems** - unresolved problems are displayed **History** - history of all events is displayed
Host groups	Filter by one or more host groups. Specifying a parent 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.
Host inventory	Filter by inventory type and value.

Parameter	Description
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.
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 suppressed problems	Mark the checkbox to display problems that would otherwise be suppressed (not shown) because of host maintenance or single problem suppression.
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 unacknowledged only	Mark the checkbox to display unacknowledged problems only.
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.

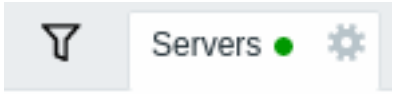
To save a new set of filter parameters, open the main tab, and configure the filter settings, then press the *Save as* button. In a new popup window, define *Filter properties*.



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.

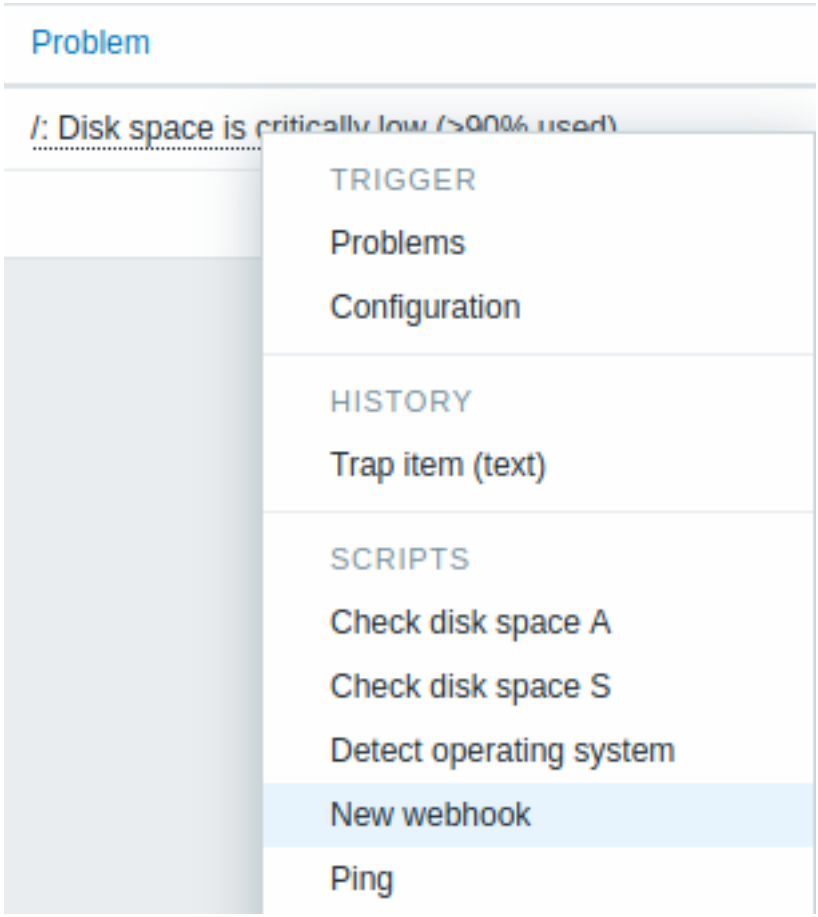
Filter buttons

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.

Event menu

Clicking on the problem name brings up the event menu:

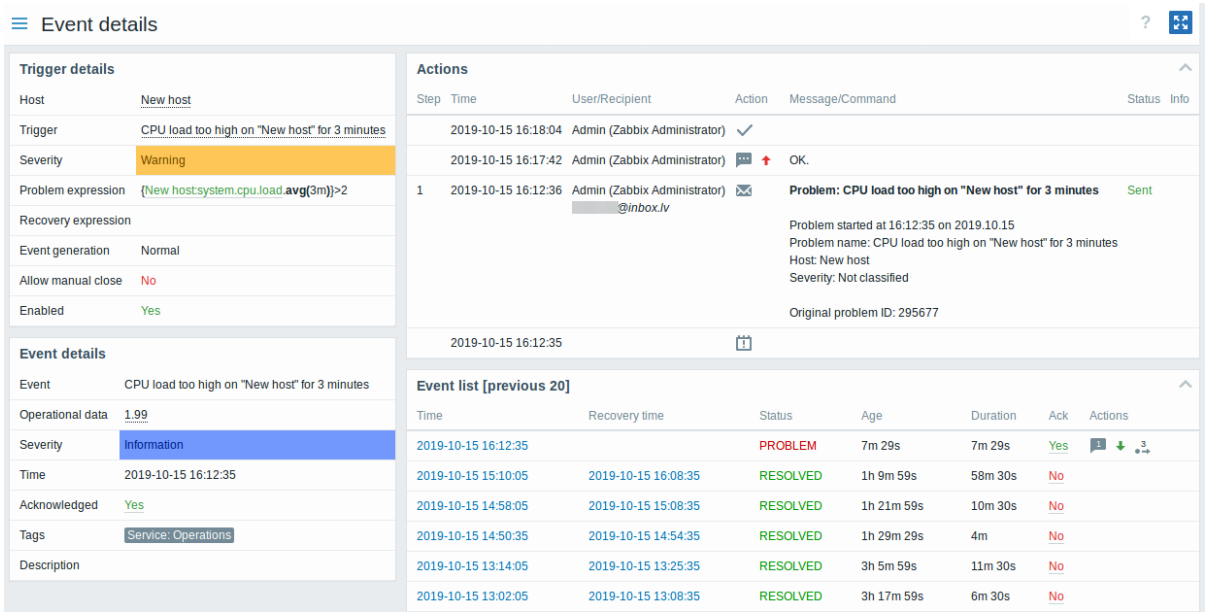


The event menu allows to:

- filter the problems of the trigger
- access the trigger configuration
- access a simple graph/item history of the underlying item(s)
- access an external ticket of the problem (if configured, see the *Include event menu entry* option when configuring webhooks)
- execute global scripts (these scripts need to have their scope defined as 'Manual event action'). This feature may be handy for running scripts used for managing problem tickets in external systems.

Viewing details

The times for problem start and recovery in *Monitoring* → *Problems* are links. Clicking on them opens more details of the event.



Note how the problem severity differs for the trigger and the problem event - for the problem event it has been updated using the *Update problem* screen.

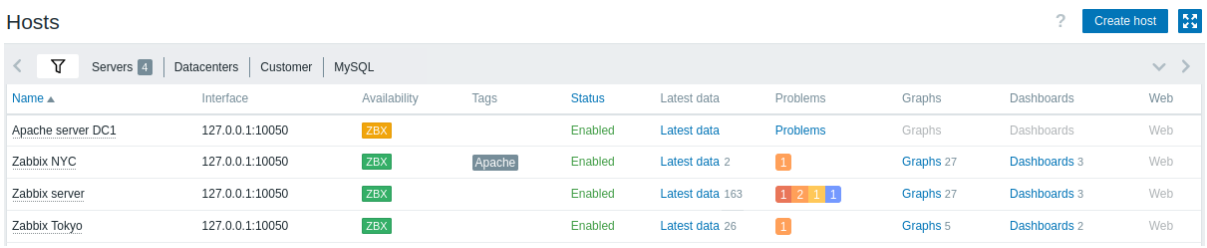
In the action list, the following icons are used to denote the activity type:

- - 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

3 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.
Availability	Host availability per configured interface is displayed. Icons represent only those interface types (Zabbix agent, SNMP, IPMI, JMX) that are configured. If you position the mouse on the icon, a popup list appears listing all interfaces of this type with details, status and errors (for the agent interface, availability of active checks is also listed). The column is empty for hosts with no interfaces. The current status of all interfaces of one type is displayed by the respective icon color: **Green** - all interfaces available **Yellow** - at least one interface available and at least one unavailable; others can have any value including 'unknown' **Red** - no interfaces available **Gray** - at least one interface unknown (none unavailable) **Active check availability** Since Zabbix 6.2 active checks also affect host availability, if there is at least one enabled active check on the host. To determine active check availability heartbeat messages are sent in the agent active check thread. The frequency of the heartbeat messages is set by the `HeartbeatFrequency` parameter in Zabbix agent and agent 2 configurations (60 seconds by default, 0-3600 range). Active checks are considered unavailable when the active check heartbeat is older than 2 x HeartbeatFrequency seconds. *Note* that if Zabbix agents older than 6.2.x are used, they are not sending any active check heartbeats, so the availability of their hosts will remain unknown. Active agent availability is counted towards the total Zabbix agent availability in the same way as a passive interface is (for example, if a passive interface is available, while the active checks are unknown, the total agent availability is set to gray(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.
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.

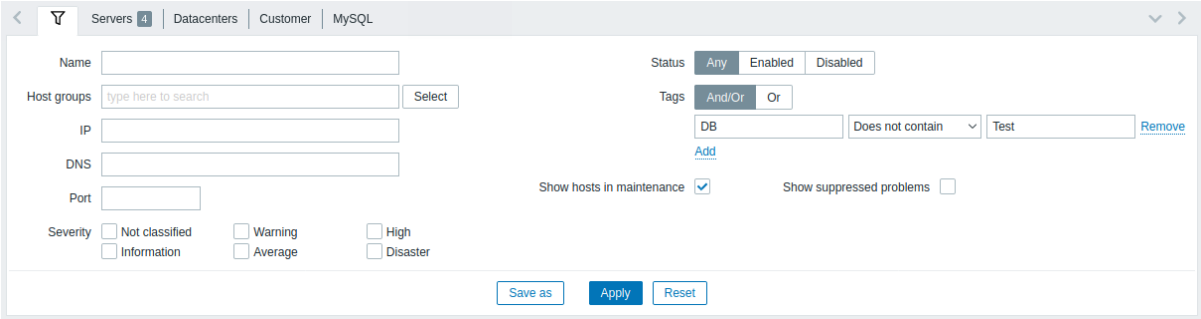
View mode buttons being common for all sections are described on the Monitoring page.

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 parent 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.

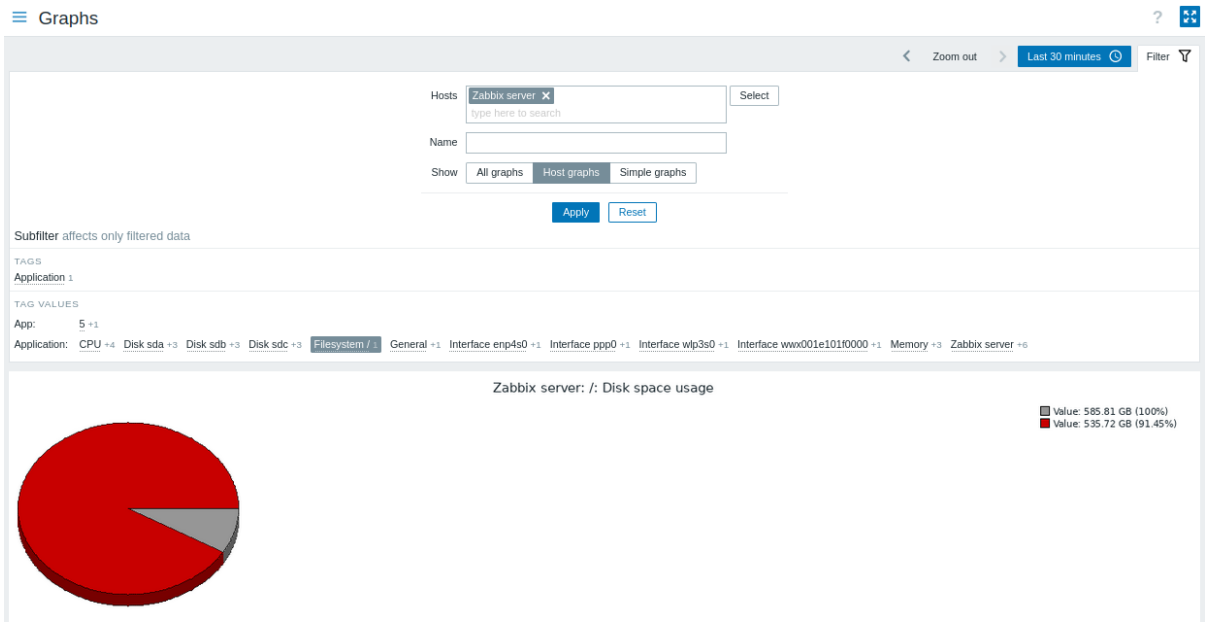
See more details about saving filters.

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.

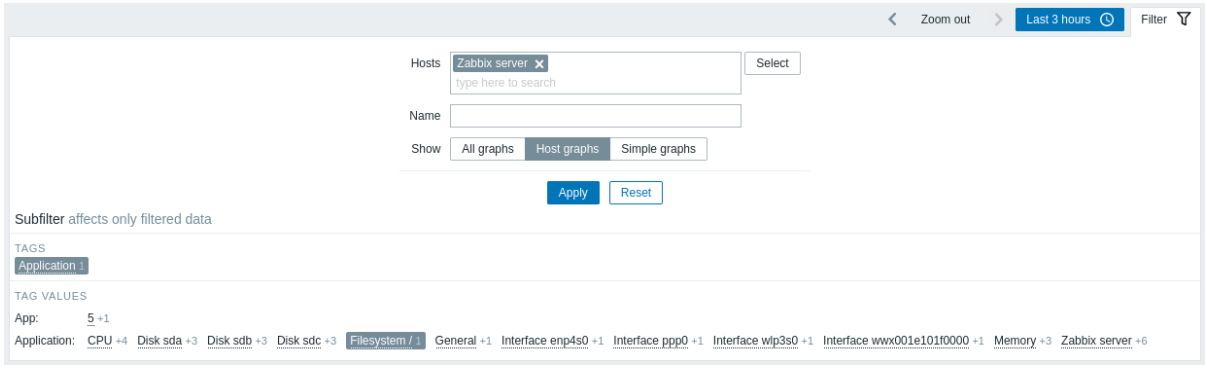
Time period selector

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 is useful for a quick one-click access to related graphs. The subfilter operates autonomously from the main filter - results are filtered immediately, no need to click on *Apply* in the main filter.

Note that the subfilter only allows to further modify the filtering from the main filter.

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 subfilter shows **clickable links** allowing to filter graphs based on a common entity - the tag name or tag value. As soon as the entity is clicked, graphs are immediately filtered; the selected entity is highlighted with gray background. To remove the filtering, click on the entity again. To add another entity to the filtered results, click on another entity.

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.

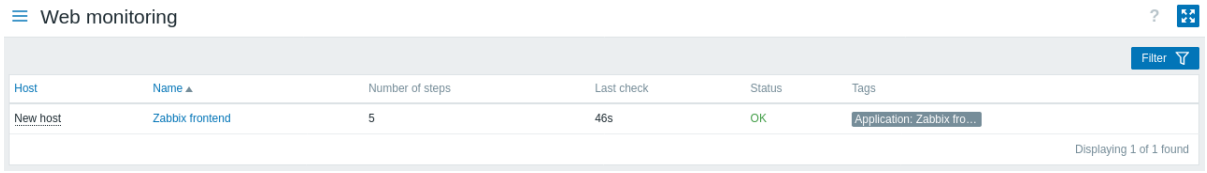
Buttons

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.



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* 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

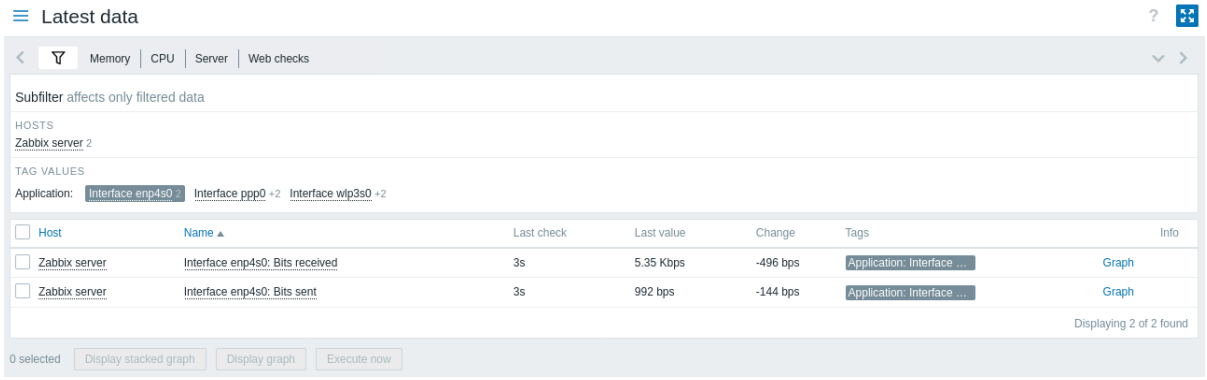
View mode buttons being common for all sections are described on the Monitoring page.

4 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.

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*(/manual/web_interface/frontend_sections/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.

Item menu

Clicking on the item name opens the item menu with links to:

- simple graphs
- list of latest values
- list of 500 latest values
- item configuration
- the option to execute a check for new item value immediately (see also mass actions)

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

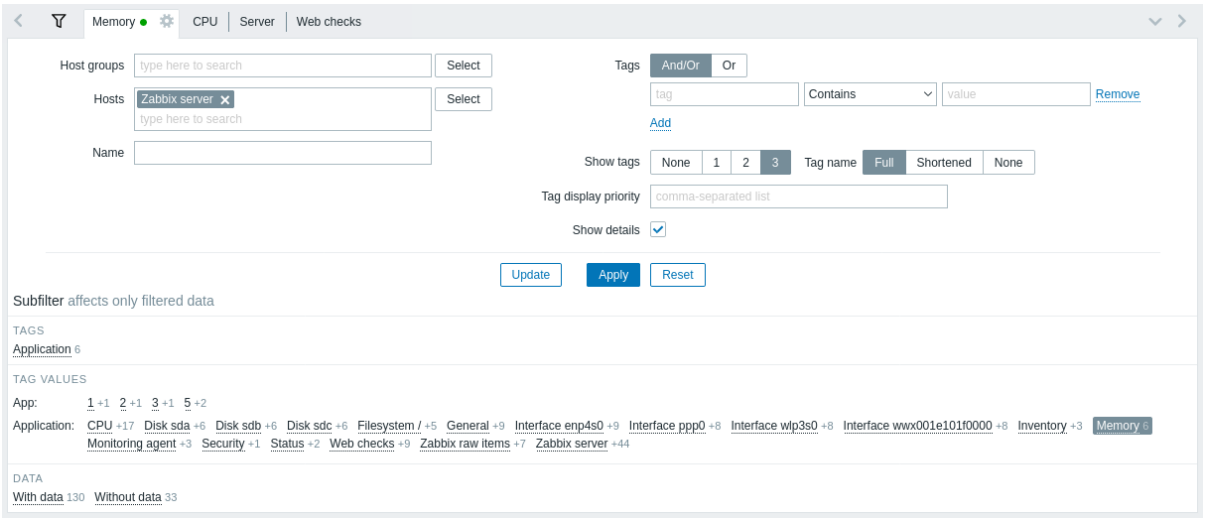
- *Execute now* - execute a check for new item values immediately. Supported for **passive** checks only (see more details). This option is available only for hosts with read-write access. Accessing this option for hosts with read-only permissions depends on the user role option called *Invoke "Execute 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 is useful for a quick one-click access to groups of related items. The subfilter operates autonomously from the main filter - results are filtered immediately, no need to click on *Apply* in the main filter.

Note that the subfilter only allows to further modify the filtering from the main filter.

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 subfilter shows **clickable links** allowing to filter items based on a common entity group - the host, tag name or tag value. As soon as the entity is clicked, items are immediately filtered; the selected entity is highlighted with gray background. To remove the filtering, click on the entity again. To add another entity to the filtered results, click on another entity.

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.

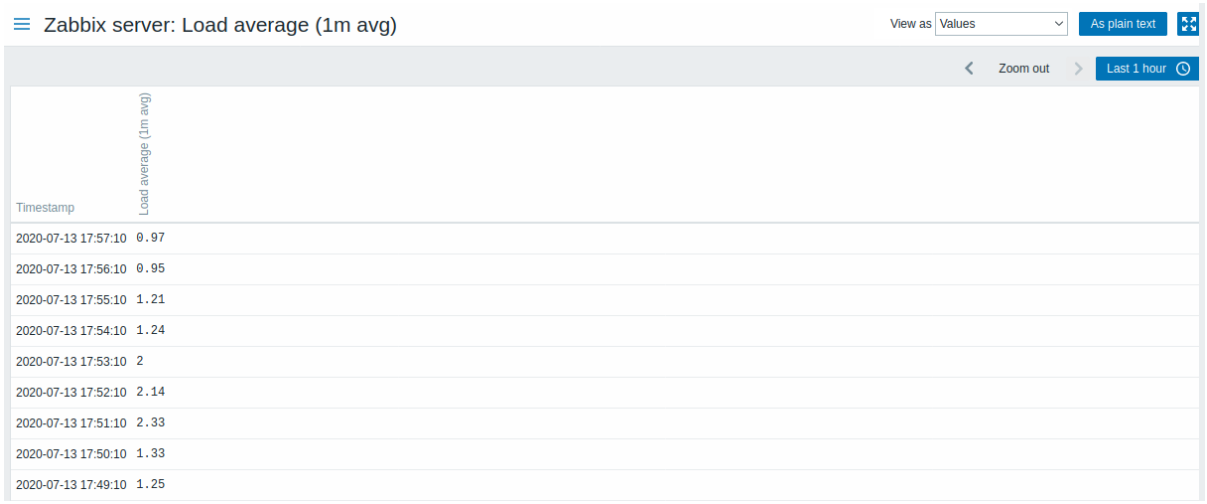
Once one entity is selected, the numbers with other available entities are displayed with a plus sign indicating how many items may be added to the current selection.

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.

5 Maps

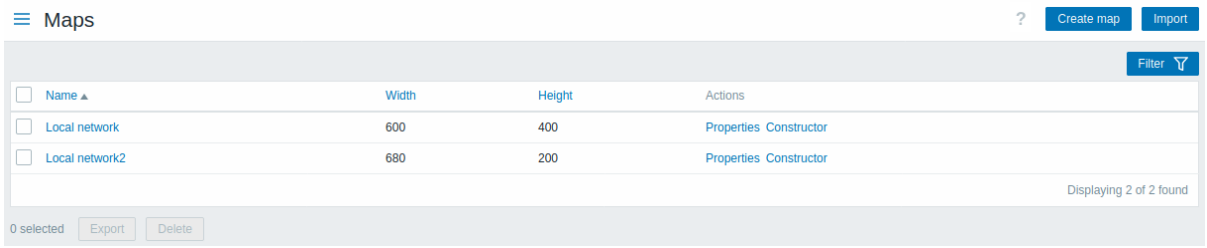
Overview

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.

Column	Description
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

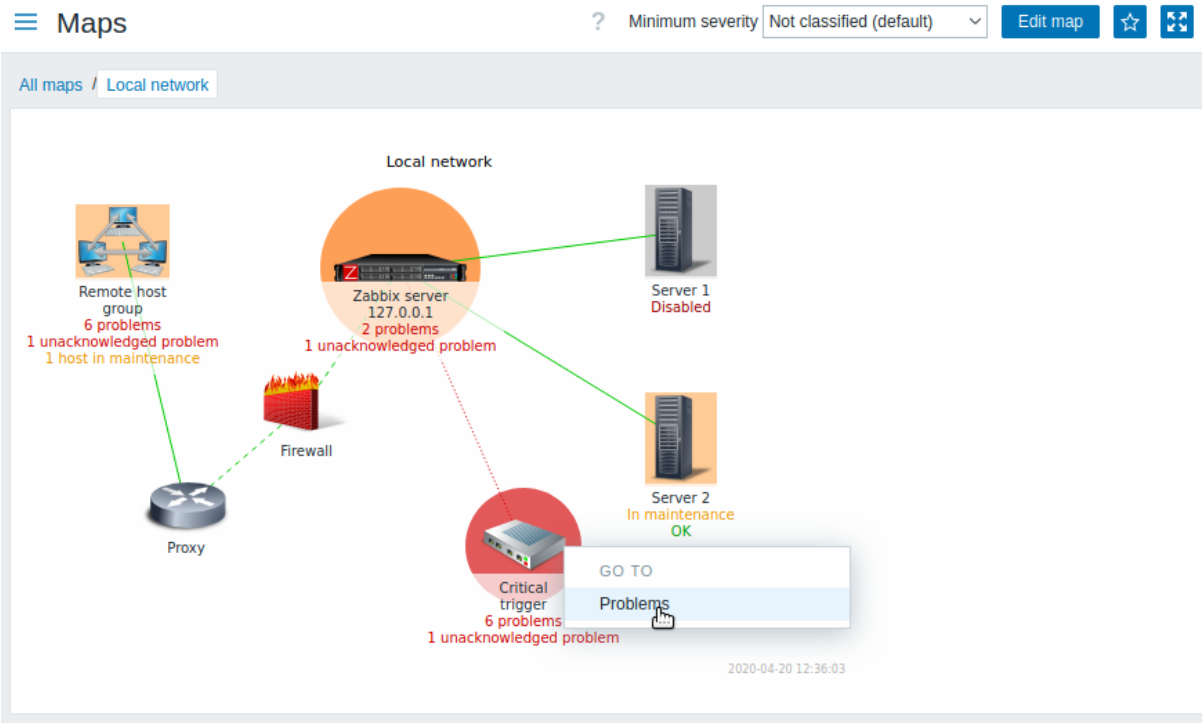
To use these options, mark the checkboxes before the respective maps, then click on the required button.

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.

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 the Dashboard.
★	The map is in the favorites widget in the Dashboard. 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

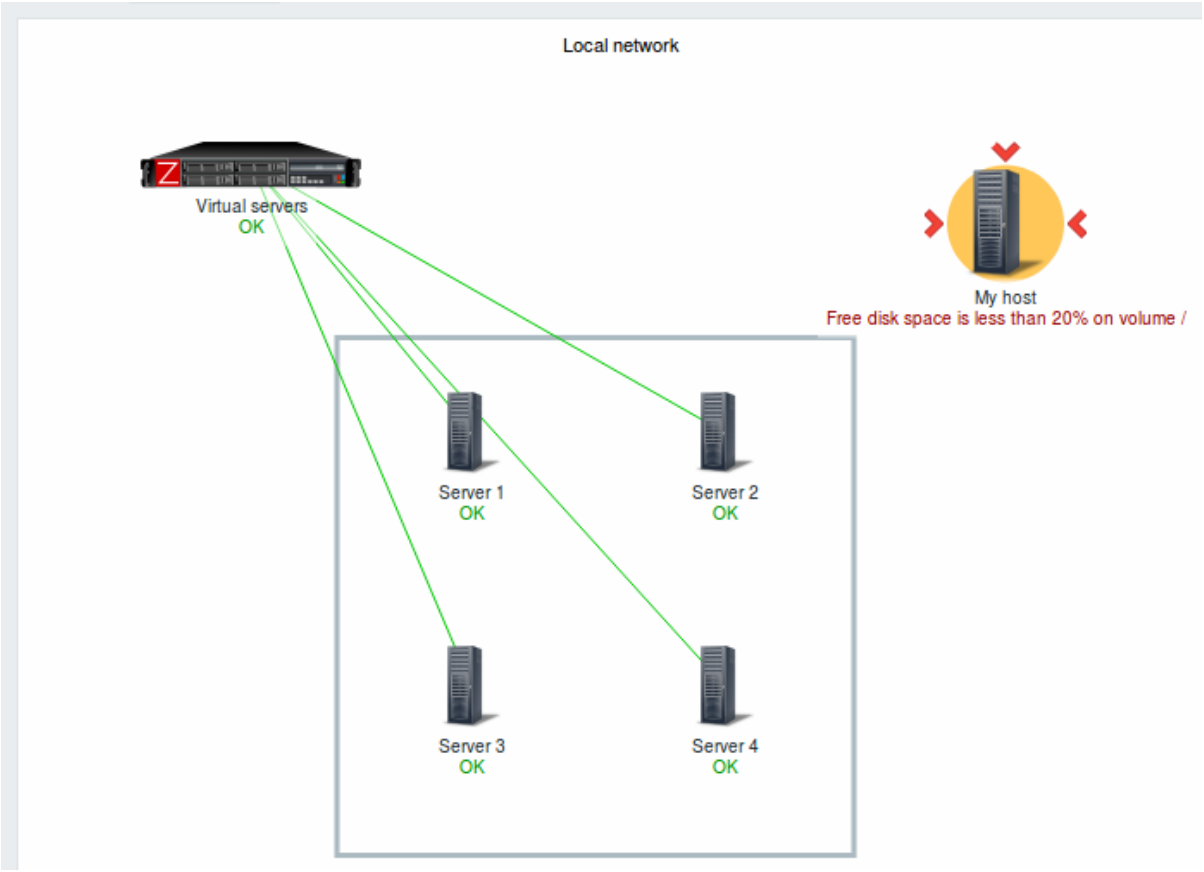
A hidden "aria-label" property is available allowing map information to be read with a screen reader. Both general map description and individual element description is available, in the following format:

- 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`

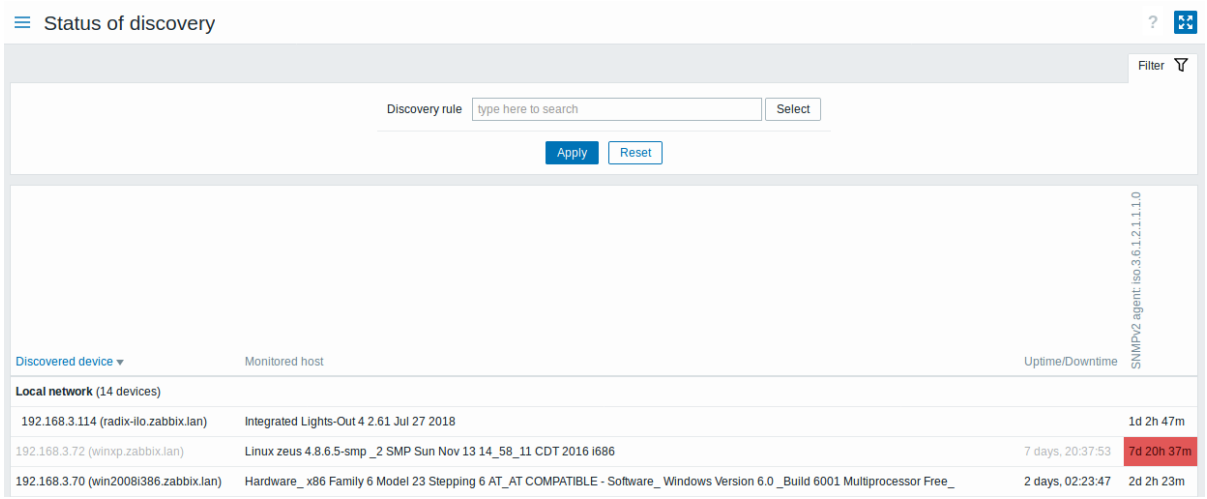
will open the map with that name (Local network).

If both `sysmapid` (map ID) and `mapname` (map name) are specified, `mapname` has higher priority.

6 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.

2 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.

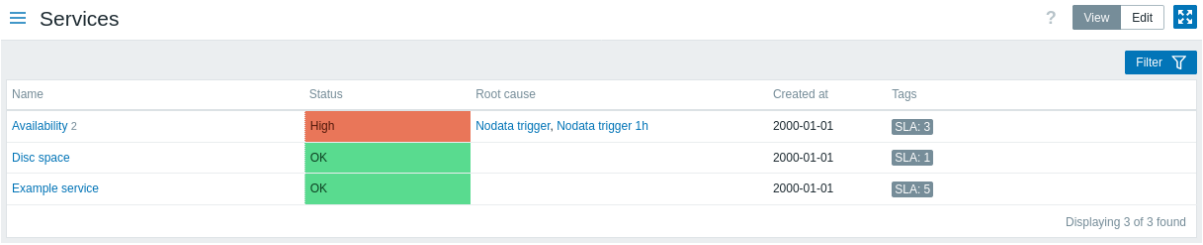
While the view mode allows to monitor services with their status and other details, you can also configure the service hierarchy in this section (add/edit services, child services) by switching to the edit mode.

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.

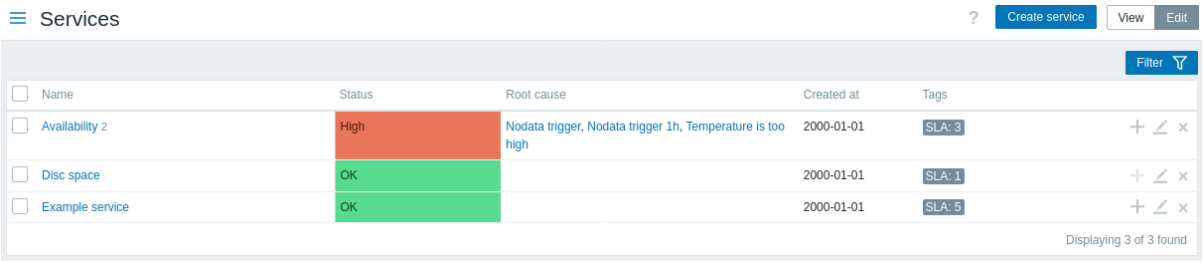
Using filter

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

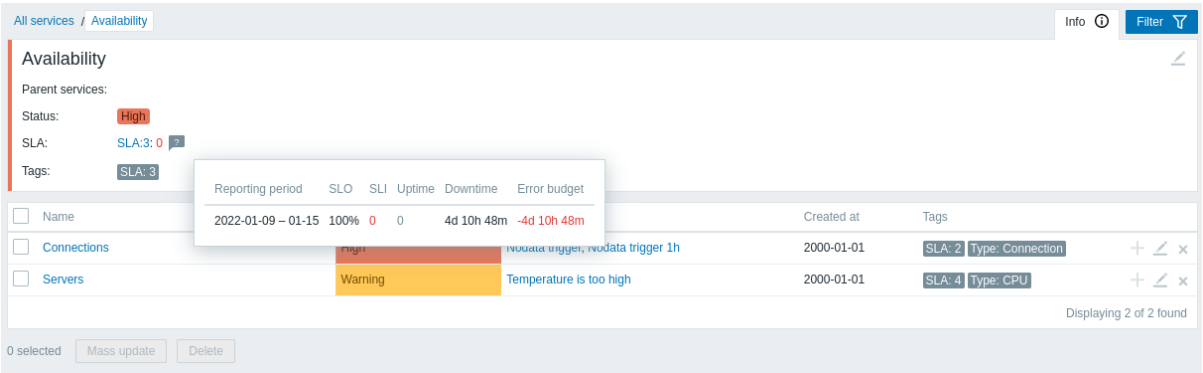


To configure a new service, click on the *Create service* button in the top right-hand corner.

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 Service actions

Overview

In the *Services* → *Service actions* section users can configure and maintain service actions.

Configured actions are displayed in the list with respect to the user role permissions. Users will only see actions for services their user role grants access to.

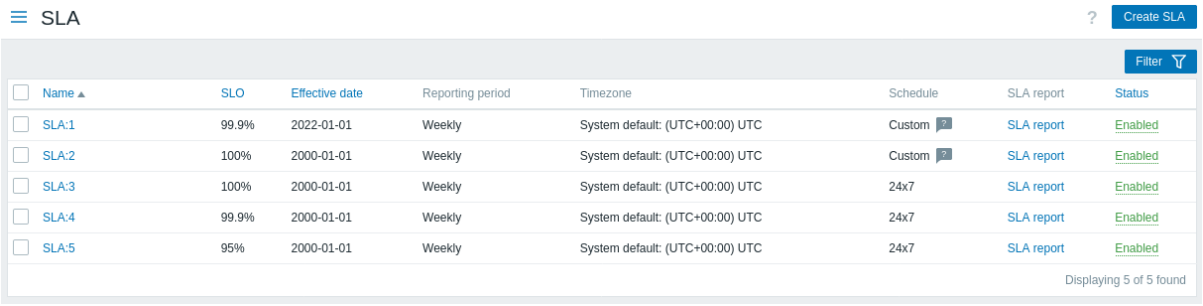
Displayed data, filter and available mass editing options are the same as for other types of actions.

3 SLA

Overview

This section allows to view and configure Service Level Agreements (SLA)s.

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.
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.

4 SLA report

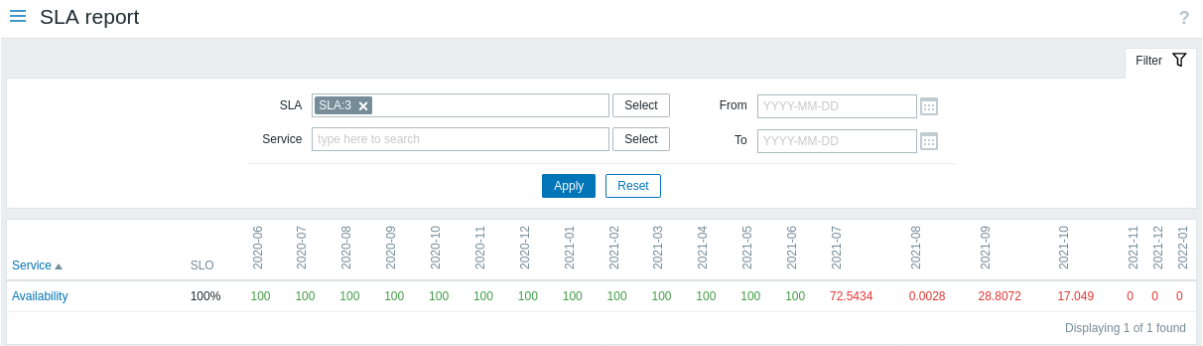
Overview

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.

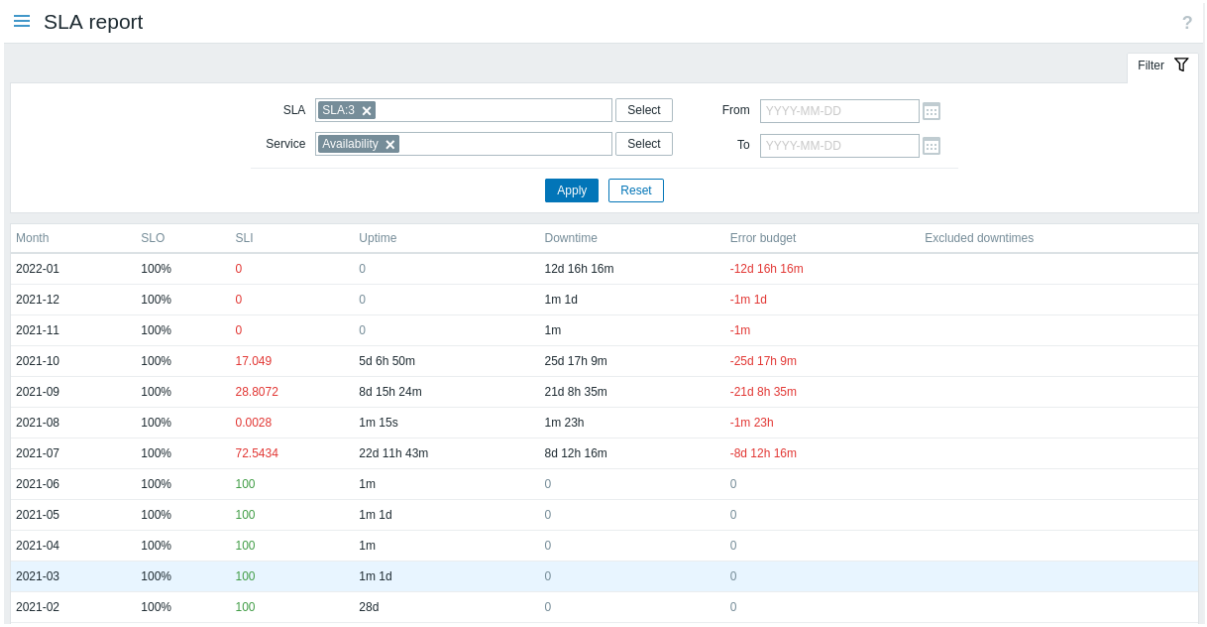


Each column (period) displays the SLI for that period. SLIs that are in breach of the set SLO are highlighted in red.

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.



3 Inventory

Overview

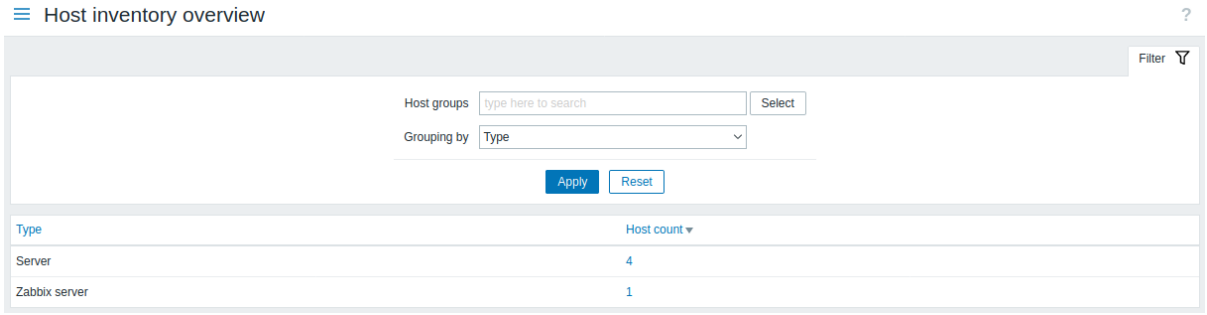
The Inventory menu features sections providing an overview of host inventory data by a chosen parameter as well as the ability to view host inventory details.

1 Overview

Overview

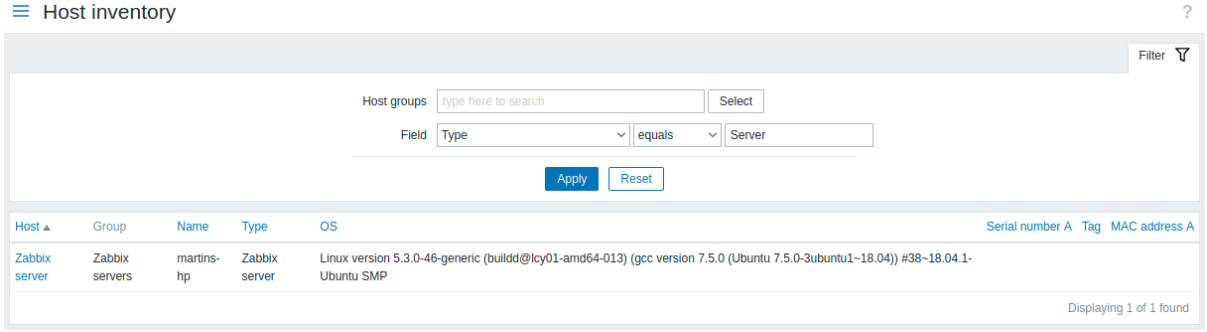
The *Inventory → Overview* section provides ways of having an overview of host inventory data.

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.

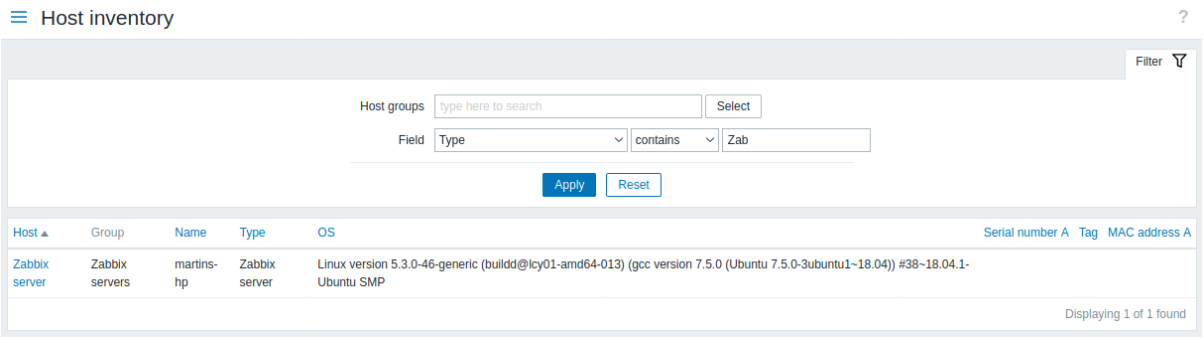


2 Hosts

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.

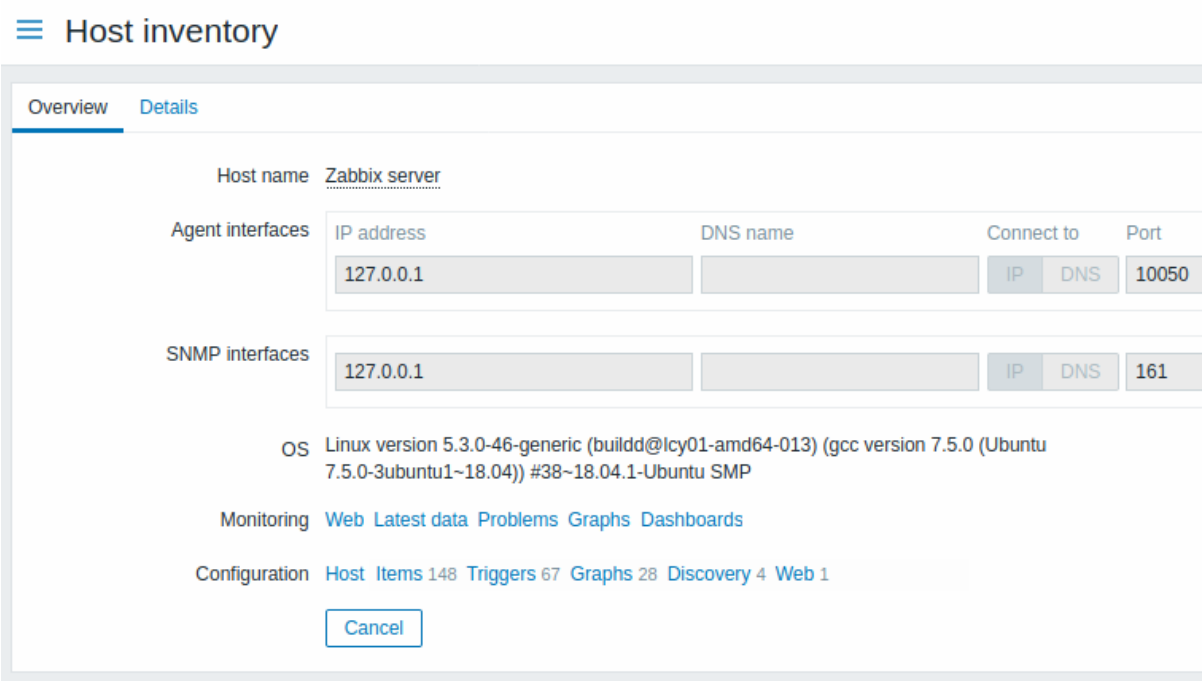


To display all host inventories, select no host group in the filter, clear the comparison field in the filter and press ”Filter”.

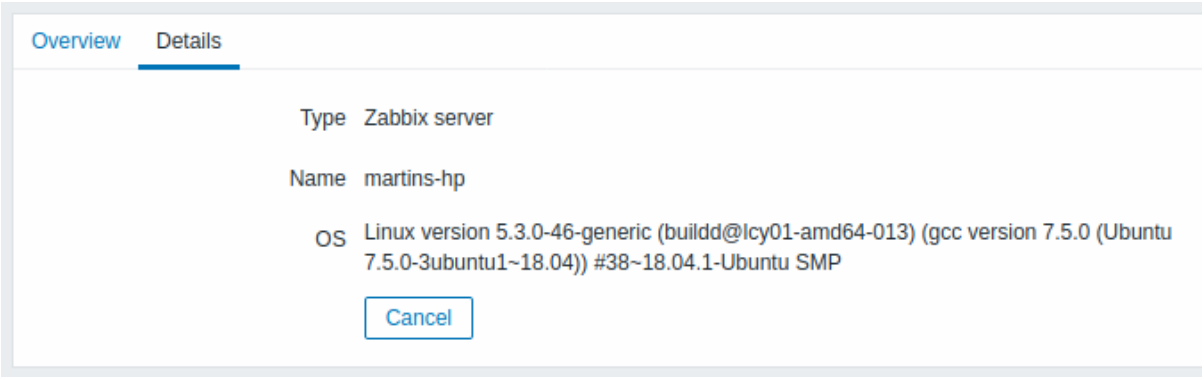
While only some key inventory fields are displayed in the table, you can also view all available inventory information for that host. To do that, click on the host name in the first column.

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.

4 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

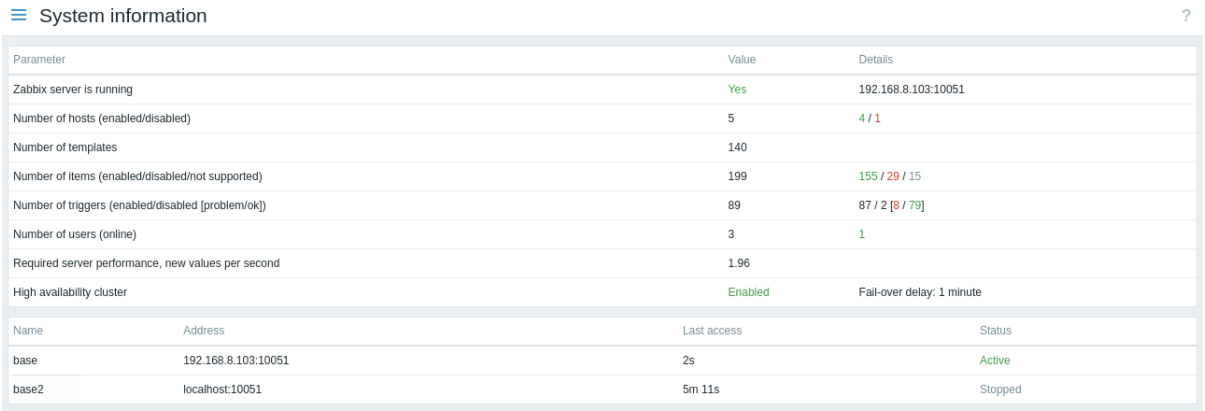
In *Reports* → *System information* a summary of key Zabbix server and system data is displayed.

Note that in a high availability setup, it is possible to redirect the system information source (server instance) by editing the ui/conf/zabbix.conf.php file - uncomment and set $ZBX_SERVER, $ZBX_SERVER_PORT to a server other than the one shown active.

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	Total number of hosts configured is displayed.	Number of monitored hosts/not monitored hosts.
Number of templates	Total number of templates is displayed.	
Number of items	Total number of items is displayed.	Number of monitored/disabled/unsupported items. Items on disabled hosts are counted as disabled.
Number of triggers	Total number of triggers is displayed.	Number of enabled/disabled triggers. [Triggers in problem/ok state.] Triggers assigned to disabled hosts or depending on disabled items are counted as disabled.
Number of users	Total number of users configured is displayed.	Number of users online.

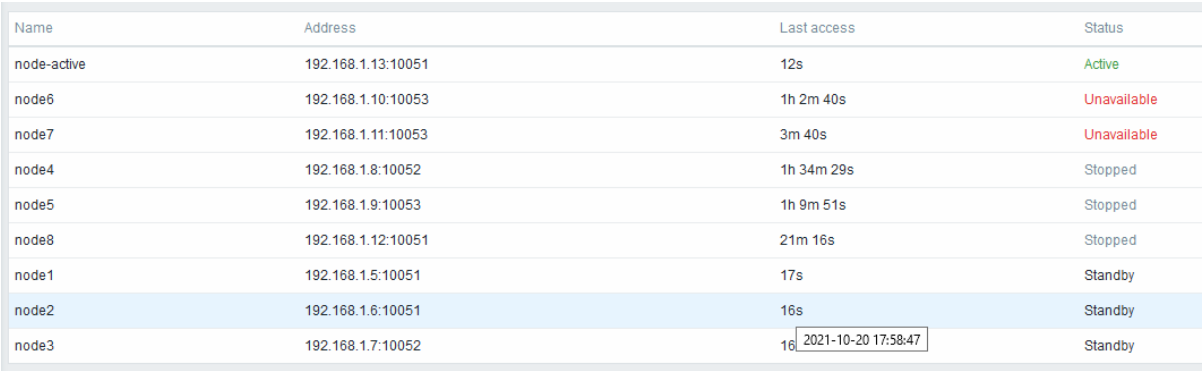
Parameter	Value	Details
Required server performance, new values per second	The expected number of new values processed by Zabbix server per second is displayed.	*Required server performance* is an estimate and can be useful as a guideline. For precise numbers of values processed, use the `zabbix[wcache,values,all]` 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	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.
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.



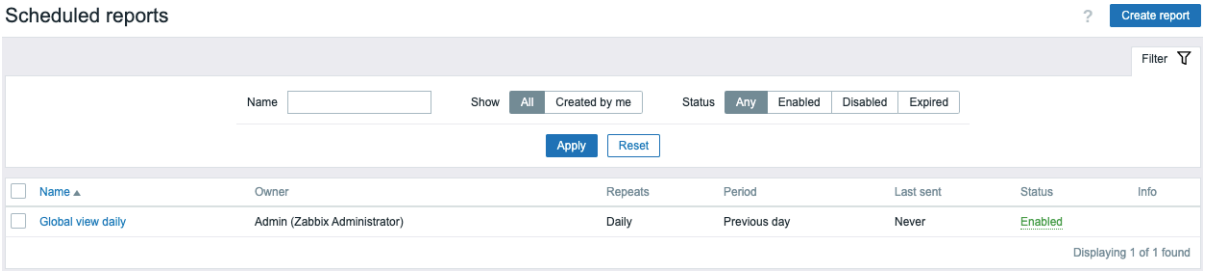
Displayed data:

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 the *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
Owner	User that 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 on it - from Enabled to Disabled (and back); from Expired to Disabled (and back). Displayed as a text for users with insufficient rights.
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 on the Filter tab in the upper right corner.

Mass update

Sometimes you may want to change status or delete 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:

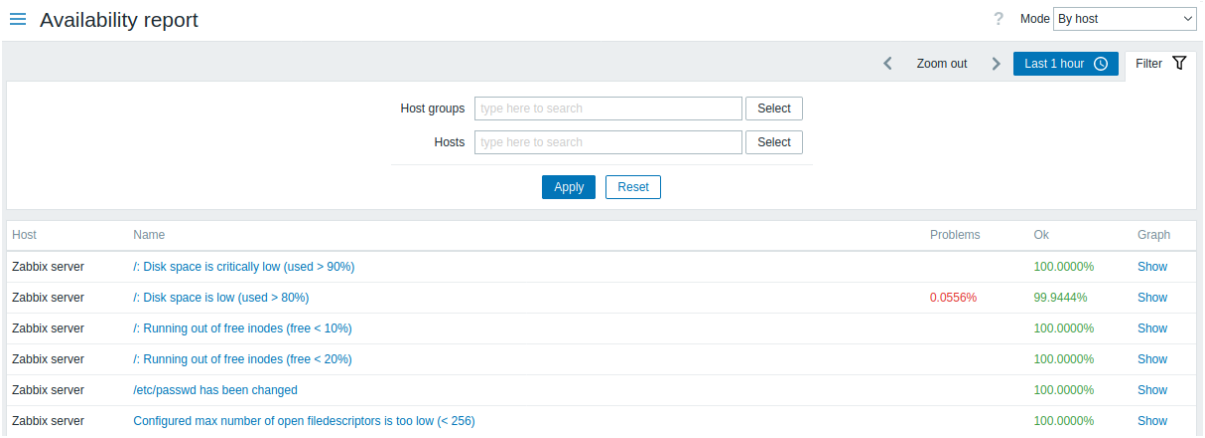
- Mark the checkboxes of the reports to update in the list
- Click on the required button below the list to make changes (Enable, Disable or Delete).

3 Availability report

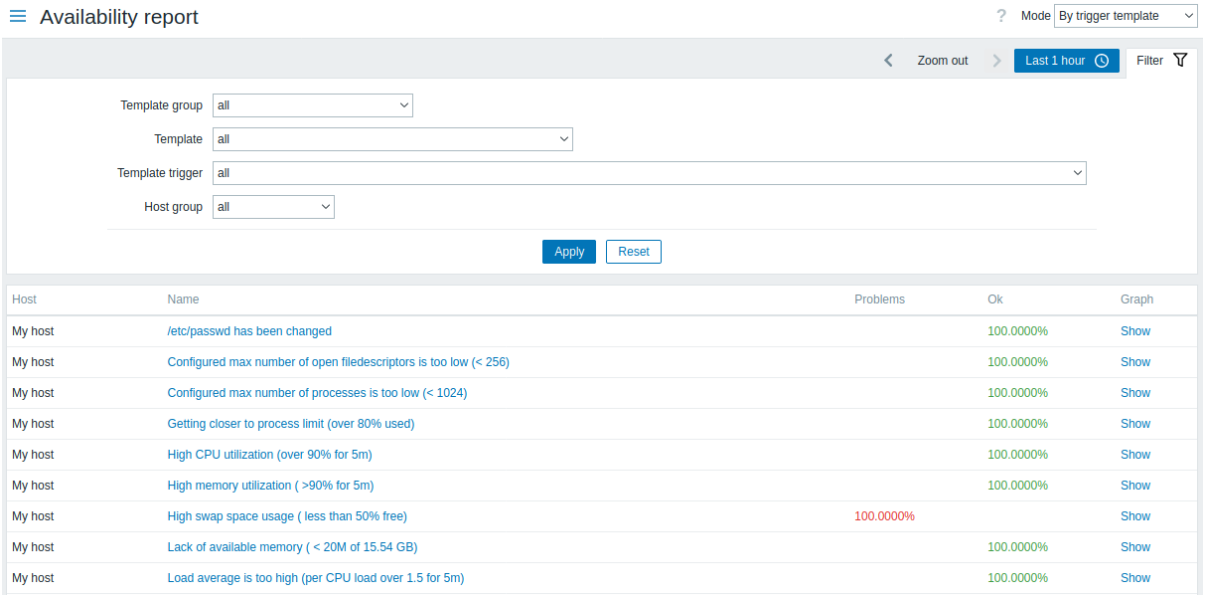
Overview

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 left.

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	Select all hosts with triggers from templates belonging to that group.
Template	Select hosts with triggers from the chosen template and all nested templates. Only triggers inherited from the selected template will be displayed. If a nested template has additional own triggers, those triggers will not be displayed.
Template trigger	Select hosts with chosen trigger. Other triggers of the selected hosts will not be displayed.
Host group	Select hosts belonging to the group.

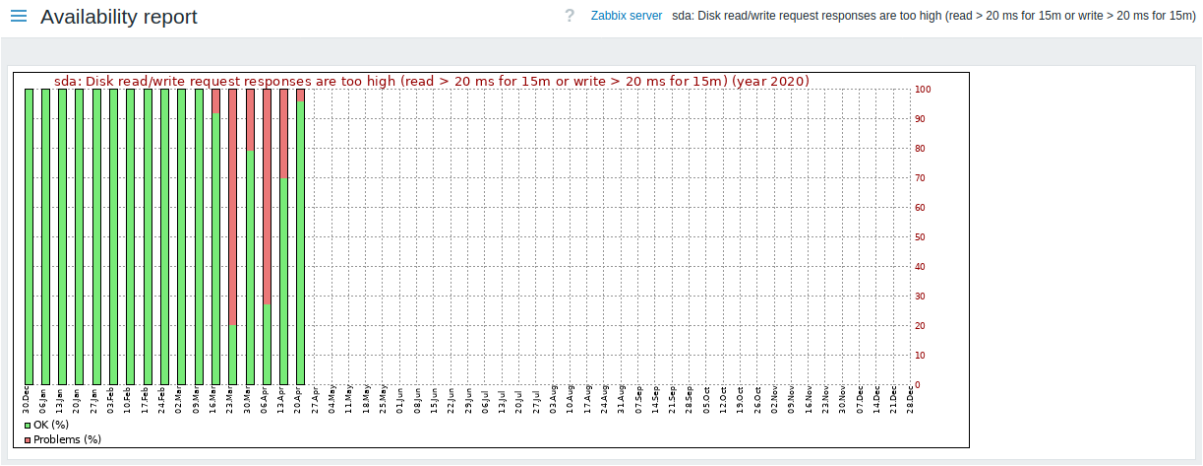
Filtering by host

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.

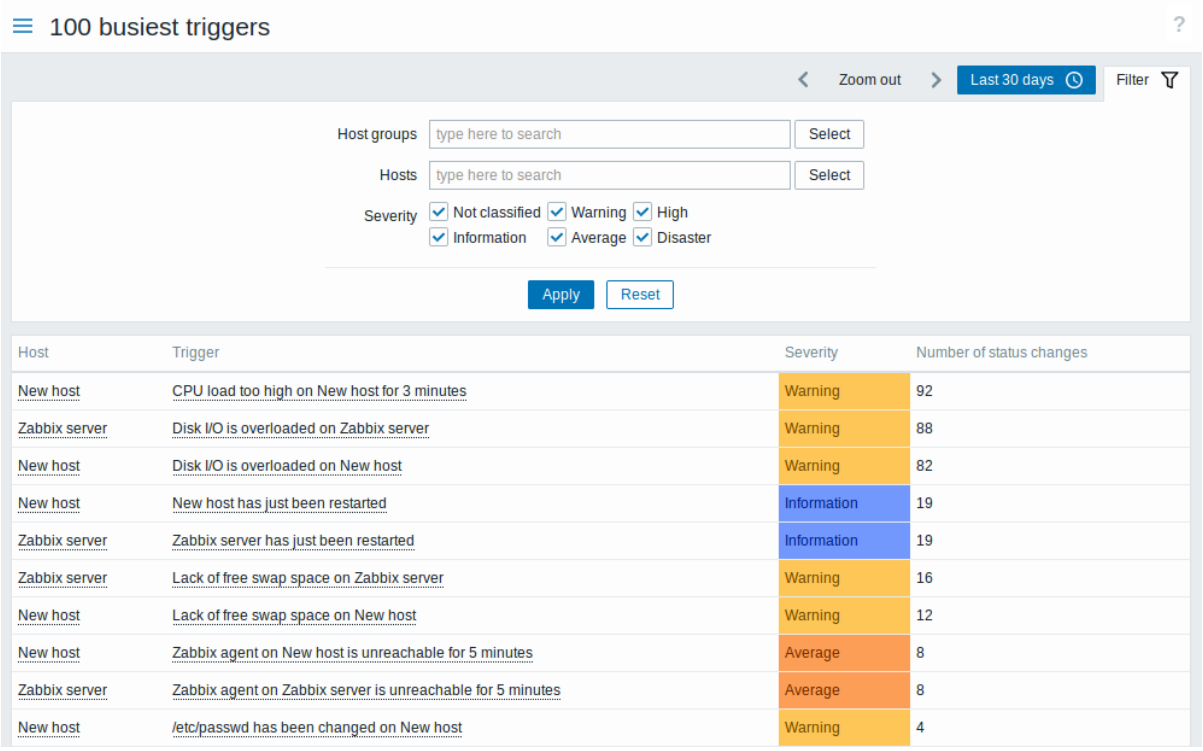


The green part of a bar stands for OK time and red for problem time.

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 - links to user-defined scripts, latest data, inventory, graphs, and dashboards for the host
- for trigger - 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

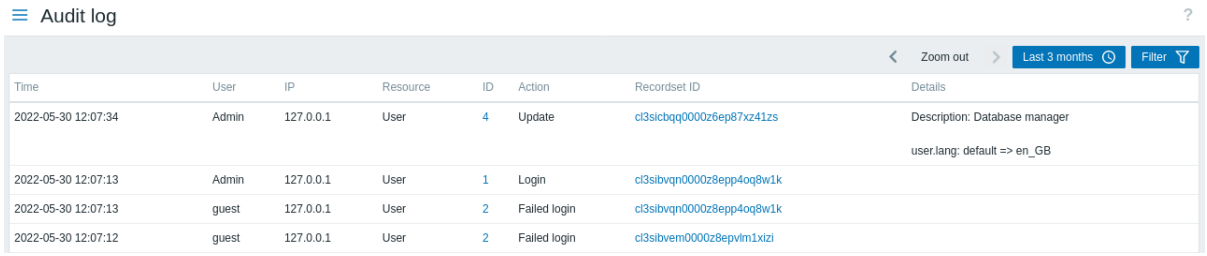
The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

5 Audit

Overview

In the *Reports* → *Audit* section users can view records of changes made in the frontend.

Note:
Audit logging should be enabled in the Administration settings to display audit records. If logging is disabled, history of frontend changes does not get recorded to the database and audit records cannot be viewed.

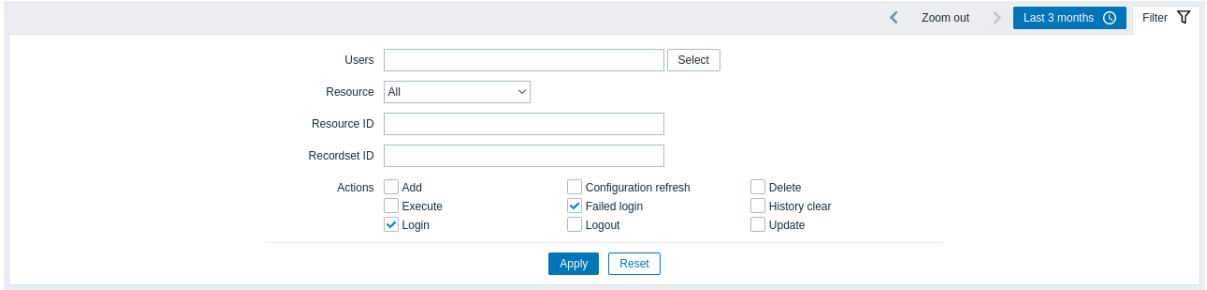


Audit log displays the following data:

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 (host, host group, etc.).
Action	Activity type: *Login, Logout, Added, Updated, Deleted, Enabled*, or *Disabled*.
ID	ID of the affected resource. Clicking on the hyperlink will result in filtering audit log records by this resource ID.
Recordset ID	Shared ID for all audit log records created as a result of the same frontend 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 frontend operation (Recordset ID). One or more actions (e. g., add, update, delete, etc) for the resource can be selected in the filter.

For better search performance, all data is searched with macros unresolved.

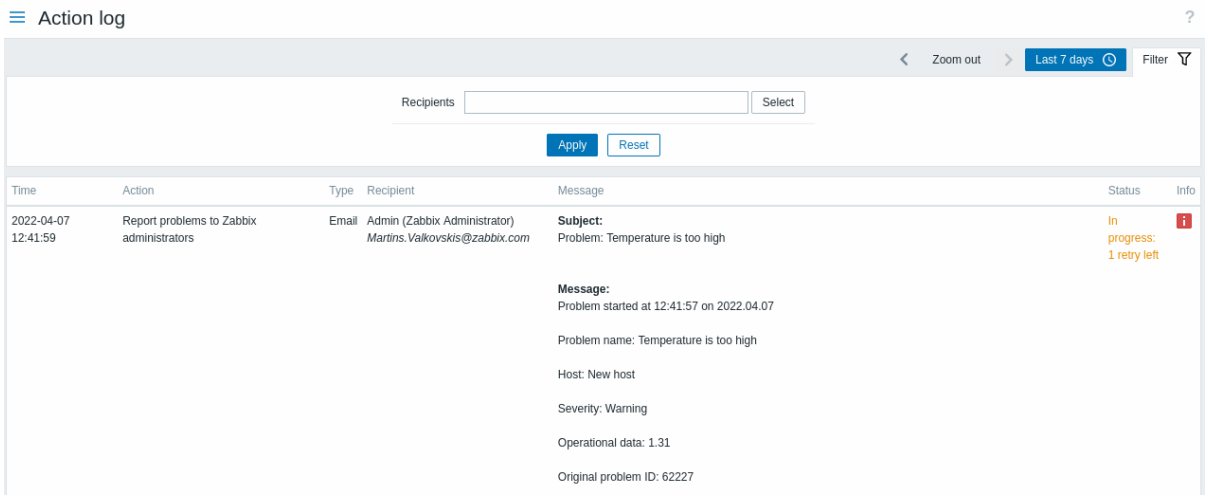
Time period selector

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 is displayed.
Type	Operation type is displayed - *Email* or *Command*.
Recipient(s)	Username, name, surname (in parentheses) and e-mail address of the notification recipient is displayed.
Message	The content of the message/remote command is displayed. A remote command is separated from the target host with a colon symbol: `<host>:<command>`. If the remote command is executed on Zabbix server, then the information has the following format: `Zabbix server:<command>`
Status	Operation status is displayed: *In progress* - action is in progress For actions in progress the number of retries left is displayed - the remaining number of times the server will try to send the notification. *Sent* - notification has been sent *Executed* - command has been executed *Not sent* - action has not been completed.
Info	Error information (if any) regarding the action execution is displayed.

Using filter

You may use the filter to narrow down the records by the message recipient(s). For better search performance, data is searched with macros unresolved.

The filter is located below the *Action log* bar. It can be opened and collapsed by clicking on the *Filter* tab on the left.

Time period selector

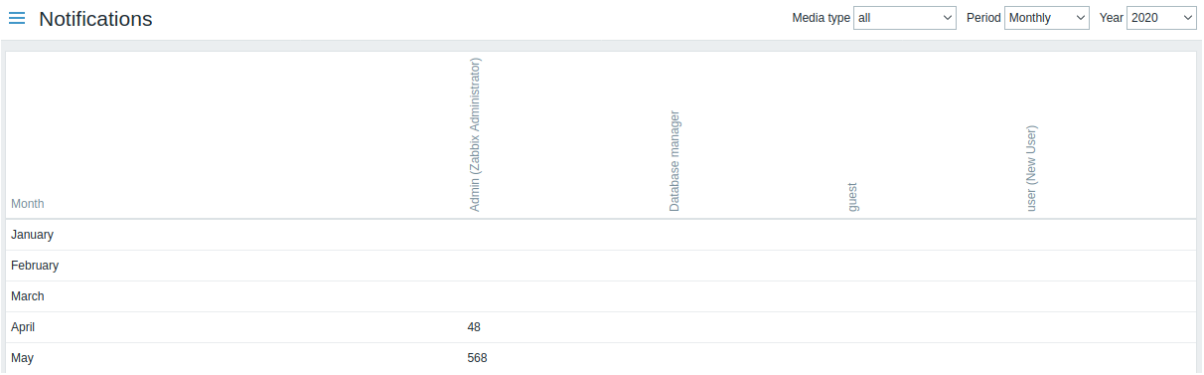
The time period selector allows to select often required periods with one mouse click. The time period selector can be opened by clicking on the time period tab next to the filter.

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.

5 Configuration

Overview

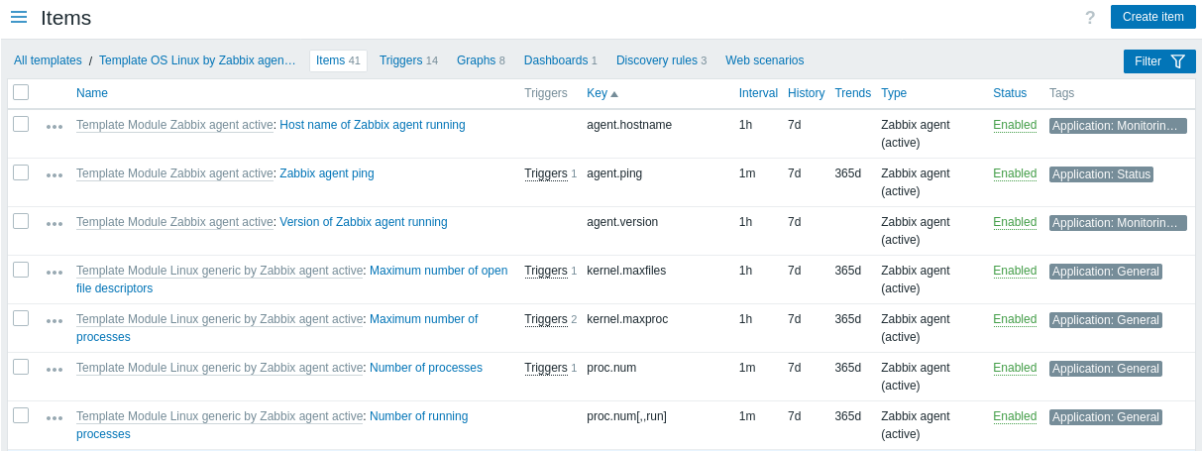
The Configuration menu contains sections for setting up major Zabbix functions, such as hosts and host groups, data gathering, data thresholds, sending problem notifications, creating data visualization and others.

1 Items

Overview

The item list for a template can be accessed from *Configuration* → *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 menu for this specific item with these options: **Create trigger** - create a trigger based on this item **Triggers** - click to see a list with links to already-configured trigger of this item **Create dependent item** - create a dependent item for this item **Create dependent discovery rule** - create a dependent discovery rule for this 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.

Column	Description
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:

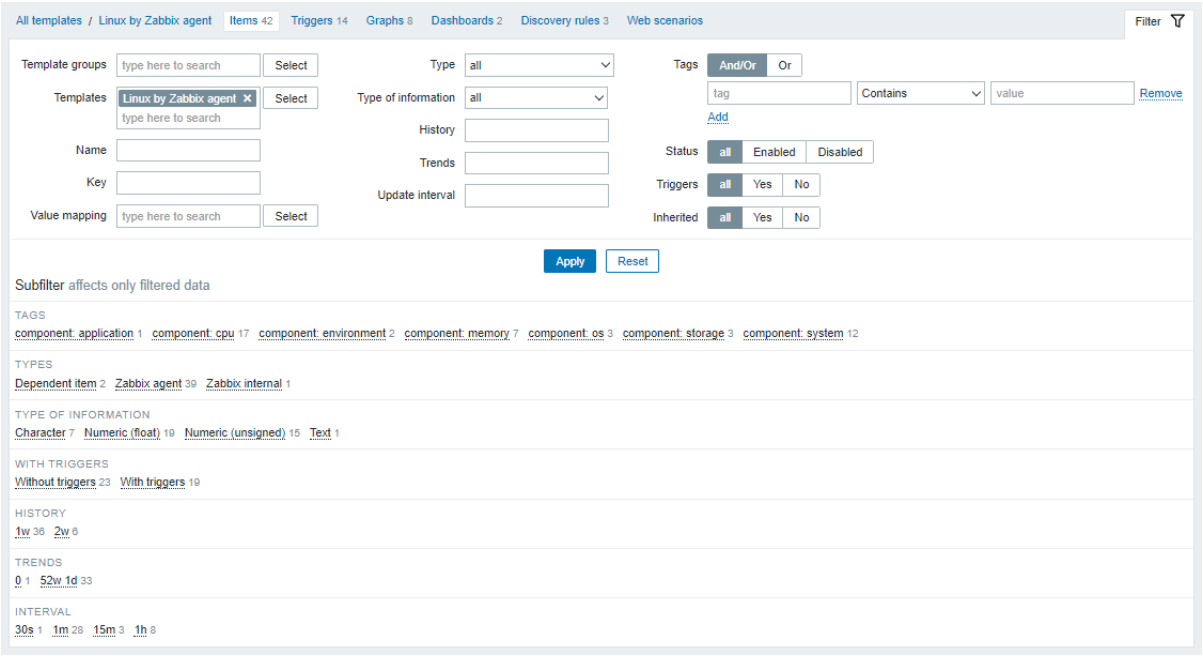
- *Enable* - change item status to *Enabled*.
- *Disable* - change item status to *Disabled*.
- *Copy* - copy the items to other hosts or templates.
- *Mass update* - update several properties for a number of items at once.
- *Delete* - delete the items.

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.

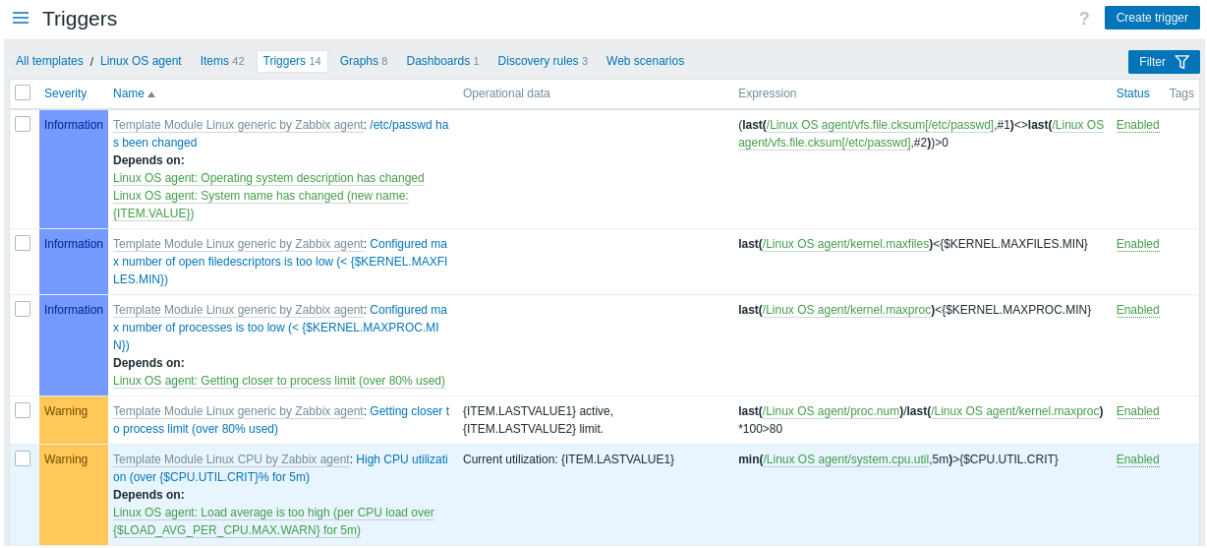
Parameter	Description
Trends	Filter by how long item trends are kept.
Update interval	Filter by item update interval.
Tags	Specify tags to limit the number of items displayed. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is 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.

The **Subfilter** below the filter offers further filtering options (for the data already filtered). You can select groups of items with a common parameter value. Upon clicking on a group, it gets highlighted and only the items with this parameter value remain in the list.

2 Triggers

Overview

The trigger list for a template can be accessed from *Configuration* → *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:

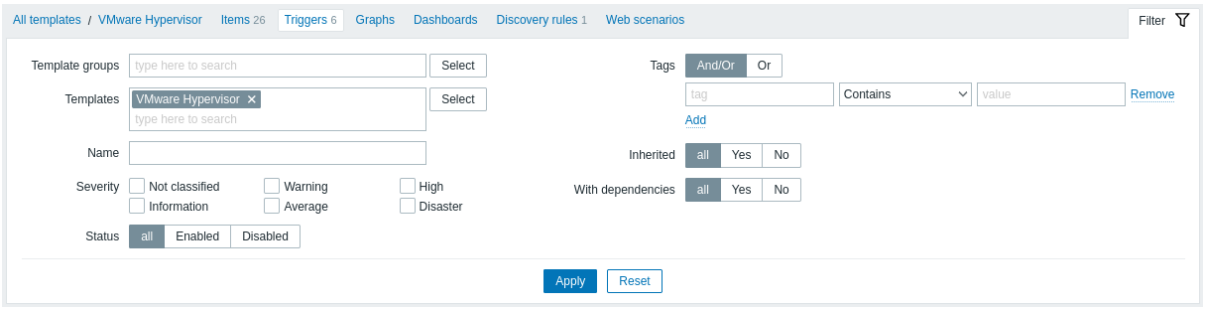
- *Enable* - change trigger status to *Enabled*
- *Disable* - change trigger status to *Disabled*
- *Copy* - copy the triggers to other hosts or templates
- *Mass update* - update several properties for a number of triggers at once
- *Delete* - delete the triggers

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.

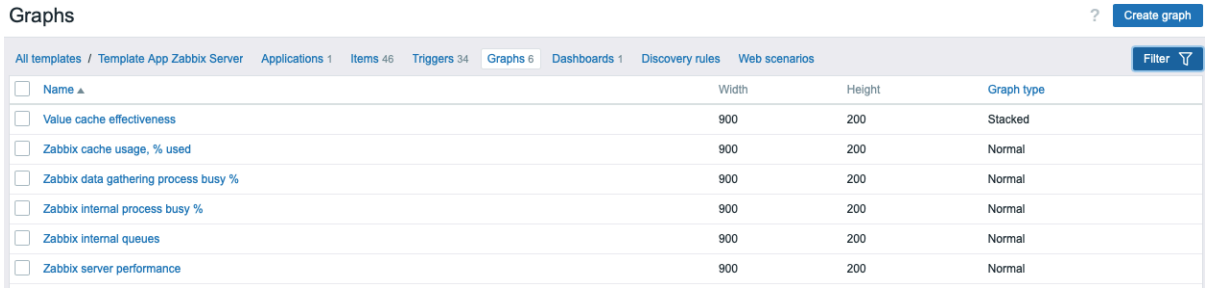
Parameter	Description
Tags	Filter by trigger tag name and value. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified 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 *Configuration* → *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

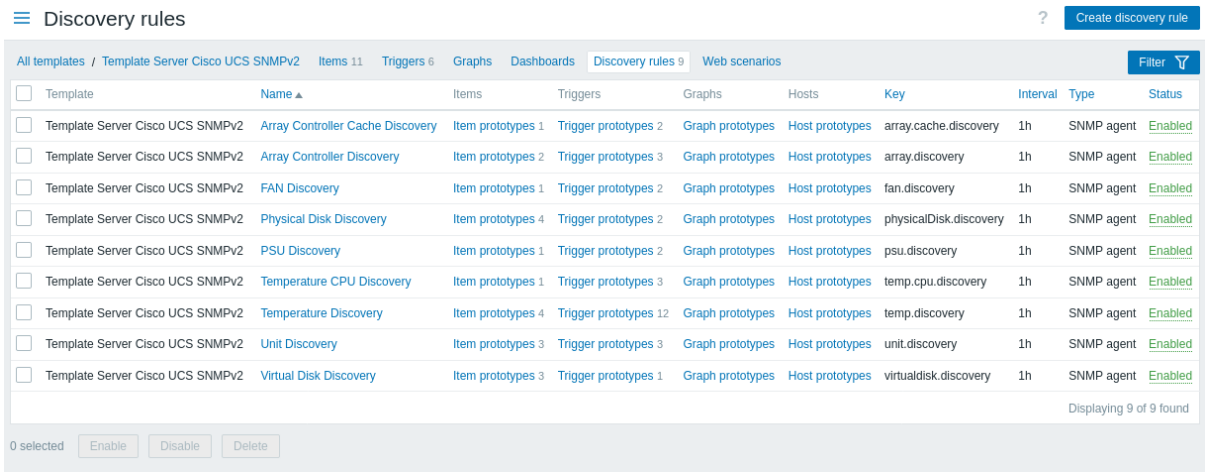
You can filter graphs by template group and template. For better search performance, data is searched with macros unresolved.

4 Discovery rules

Overview

The list of low-level discovery rules for a template can be accessed from *Configuration* → *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:

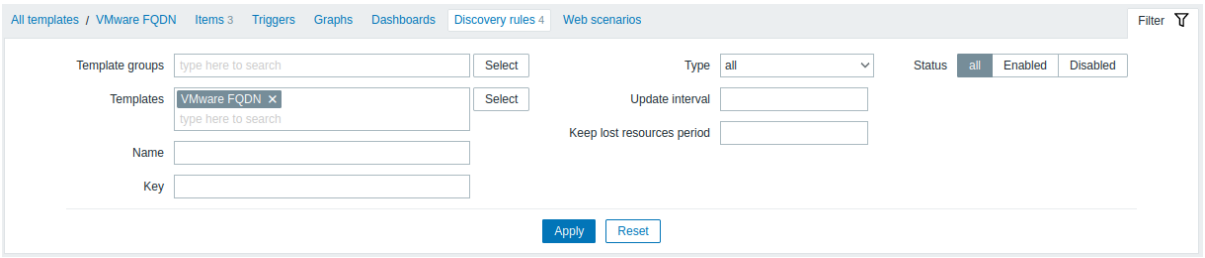
- *Enable* - change the low-level discovery rule status to *Enabled*
- *Disable* - change the low-level discovery rule status to *Disabled*
- *Delete* - delete the low-level discovery rules

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.



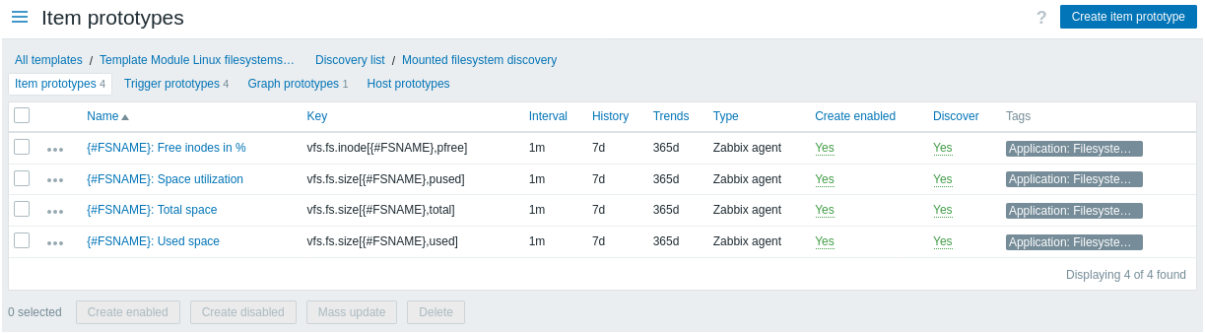
Parameter	Description
Template groups	Filter by one or more template groups. Specifying a parent template group implicitly selects all nested groups.
Templates	Filter by one or more templates.
Name	Filter by discovery rule name.
Key	Filter by discovery item key.
Type	Filter by discovery item type.
Update 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

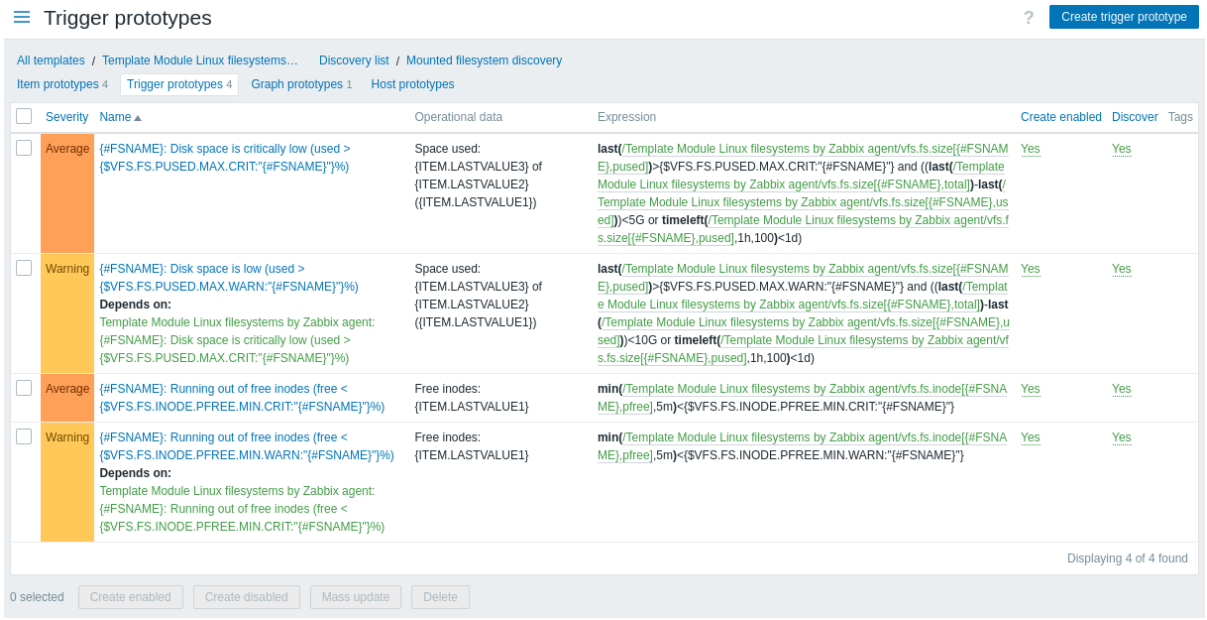
To use these options, mark the checkboxes before the respective item prototypes, then click on the required button.

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

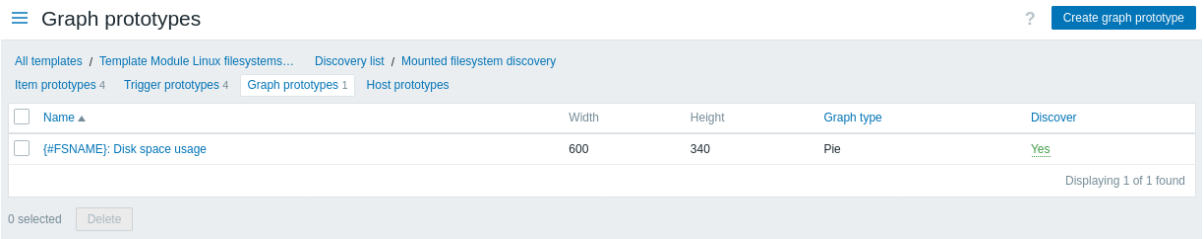
To use these options, mark the checkboxes before the respective trigger prototypes, then click on the required button.

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

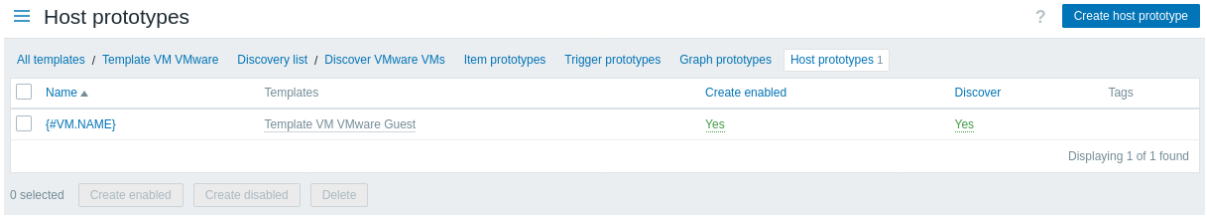
To use these options, mark the checkboxes before the respective graph prototypes, then click on the required button.

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

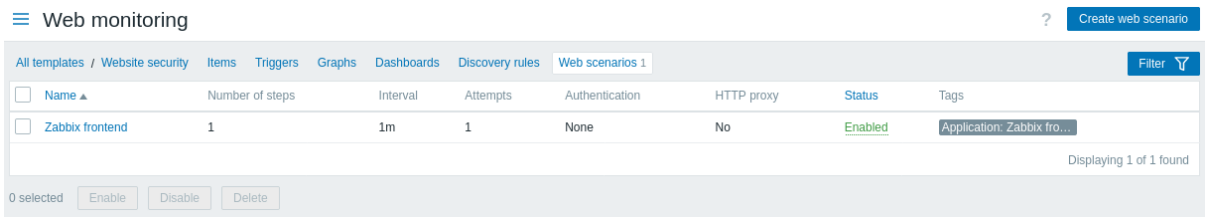
To use these options, mark the checkboxes before the respective host prototypes, then click on the required button.

5 Web scenarios

Overview

The web scenario list for a template can be accessed from *Configuration* → *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:

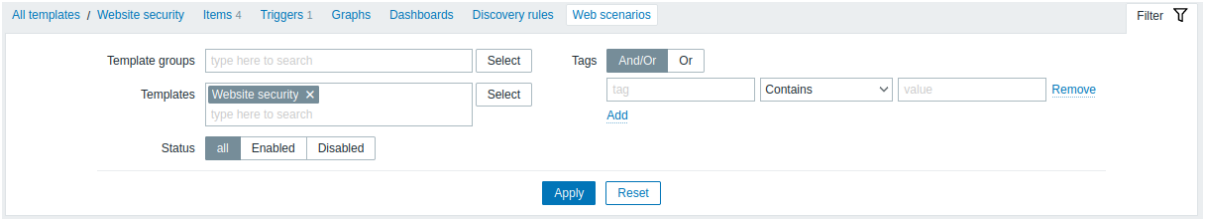
- *Enable* - change the scenario status to *Enabled*
- *Disable* - change the scenario status to *Disabled*
- *Delete* - delete the web scenarios

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.

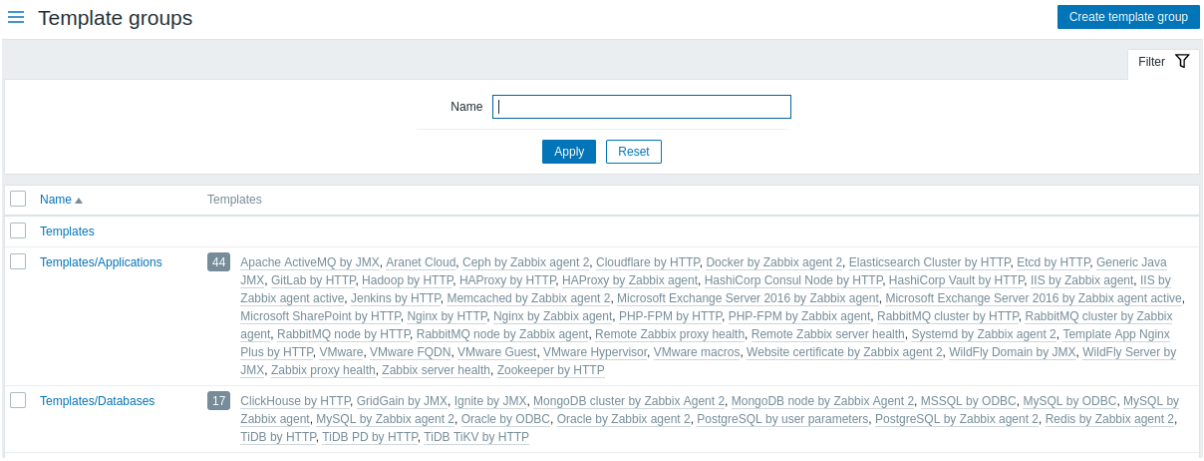


1 Template groups

Overview

In the *Configuration* → *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

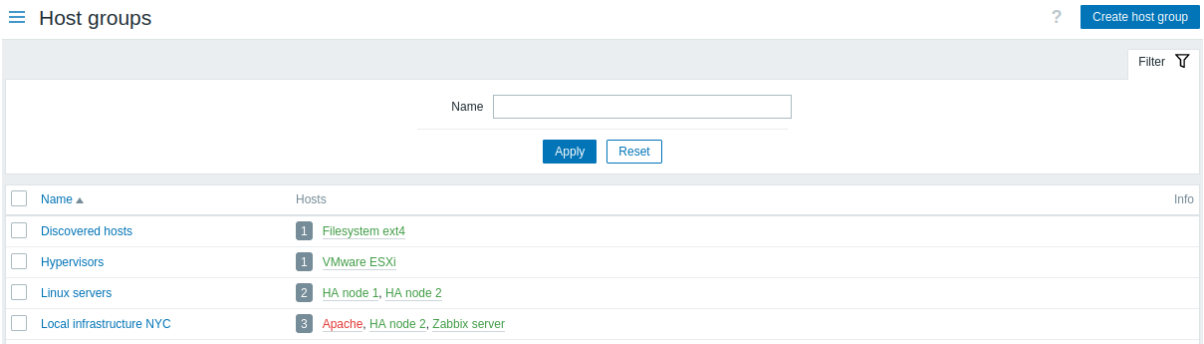
You can use the filter to display only the template groups you are interested in. For better search performance, data is searched with macros unresolved.

2 Host groups

Overview

In the *Configuration* → *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

To use these options, mark the checkboxes before the respective host groups, then click on the required button.

Using filter

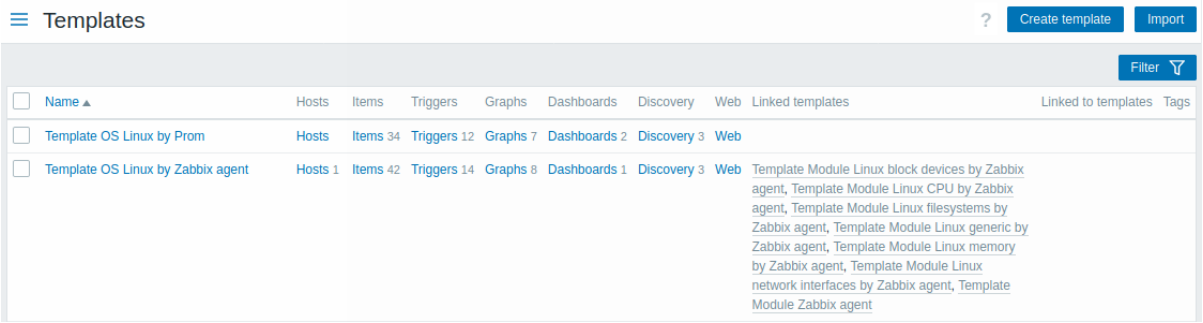
You can use the filter to display only the host groups you are interested in. For better search performance, data is searched with macros unresolved.

3 Templates

Overview

In the *Configuration* → *Templates* section users can configure and maintain templates.

A listing of existing templates with their details is displayed.



Displayed data:

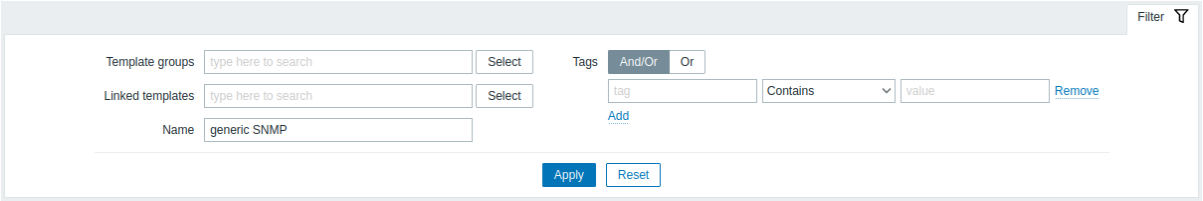
Column	Description
Name	Name of the template. Clicking on the template name opens the template configuration form.
Hosts	Number of editable hosts to which the template is linked; read-only hosts are not included. Clicking on *Hosts* will open the host list with only those hosts filtered that are linked 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, in a nested setup where the template will inherit all entities of the linked templates.
Linked to templates	The templates that the template is linked to ("children" templates that inherit all entities from this template). Since Zabbix 5.0.3, this column no longer includes hosts.
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.
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

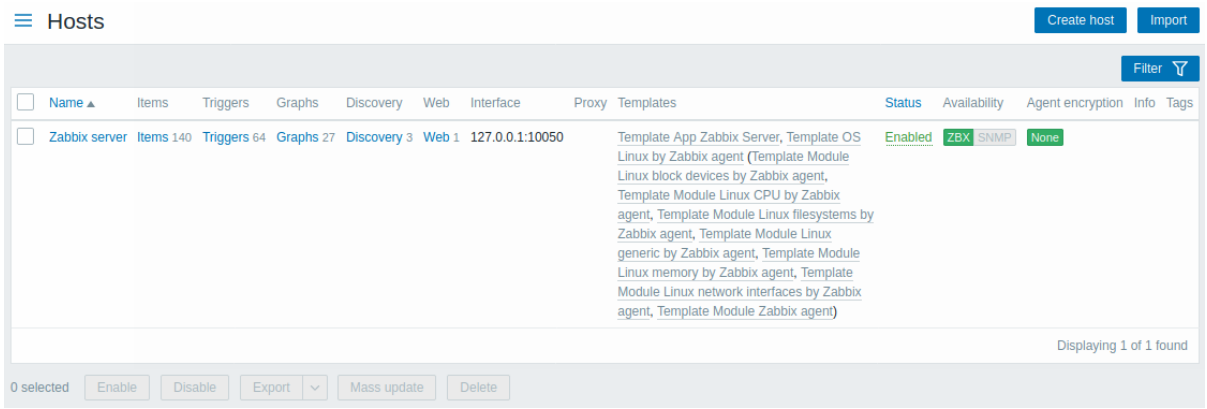
To use these options, mark the checkboxes before the respective templates, then click on the required button.

4 Hosts

Overview

In the *Configuration* → *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.

Column	Description
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 over the icon.
Availability	Host availability per configured interface is displayed. Icons represent only those interface types (Zabbix agent, SNMP, IPMI, JMX) that are configured. If you position the mouse on the icon, a popup list appears listing all interfaces of this type with details, status and errors (for the agent interface, availability of active checks is also listed). The column is empty for hosts with no interfaces. The current status of all interfaces of one type is displayed by the respective icon color: **Green** - all interfaces available **Yellow** - at least one interface available and at least one unavailable; others can have any value including 'unknown' **Red** - no interfaces available **Gray** - at least one interface unknown (none unavailable) **Active check availability** Since Zabbix 6.2 active checks also affect host availability, if there is at least one enabled active check on the host. To determine active check availability heartbeat messages are sent in the agent active check thread. The frequency of the heartbeat messages is set by the `HeartbeatFrequency` parameter in Zabbix agent and agent 2 configurations (60 seconds by default, 0-3600 range). Active checks are considered unavailable when the active check heartbeat is older than 2 x HeartbeatFrequency seconds. *Note* that if Zabbix agents older than 6.2.x are used, they are not sending any active check heartbeats, so the availability of their hosts will remain unknown. Active agent availability is counted towards the total Zabbix agent availability in the same way as a passive interface is (for example, if a passive interface is available, while the active checks are unknown, the total agent availability is set to gray(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:

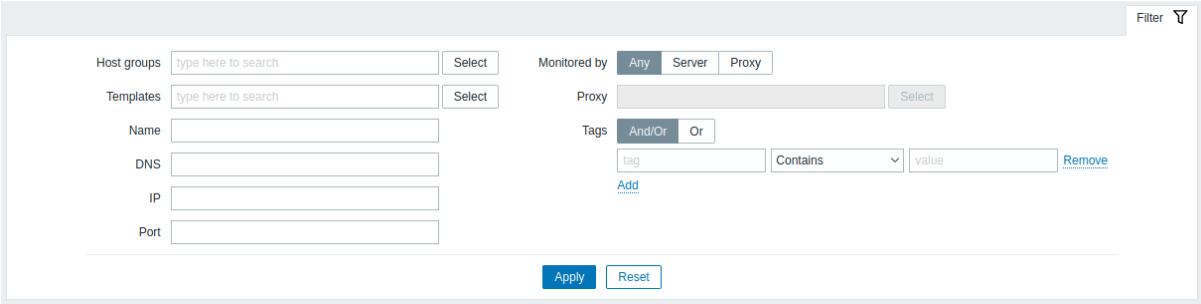
- *Enable* - change host status to *Monitored*
- *Disable* - change host status to *Not monitored*
- *Export* - export the hosts to a YAML, XML or JSON file
- *Mass update* - update several properties for a number of hosts at once
- *Delete* - delete the hosts

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* link is available above the list of hosts. If you click on it, a filter becomes available where you can filter hosts by host group, linked templates, name, DNS, IP, port number, if they are monitored by server or by proxy, proxy name and tags.



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.
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

Reading host availability

Host availability icons reflect the current host interface status on Zabbix server. Therefore, in the frontend:

- If you disable a host, availability icons will not immediately turn gray (unknown status), because the server has to synchronize the configuration changes first;
- If you enable a host, availability icons will not immediately turn green (available), because the server has to synchronize the configuration changes and start polling the host first.

Unknown interface status

Zabbix server determines an unknown status for the corresponding agent interface (Zabbix, SNMP, IPMI, JMX) if:

- there are no enabled items on the interface (they were removed or disabled);
- there are only active Zabbix agent items;
- there are no pollers for that type of the interface (e.g. StartPollers=0);
- host is disabled;
- host is set to be monitored by proxy, a different proxy or by server if it was 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).

Setting interface availability to unknown is done after server configuration cache synchronization. Restoring interface availability (available/unavailable) on hosts monitored by proxies is done after proxy configuration cache synchronization.

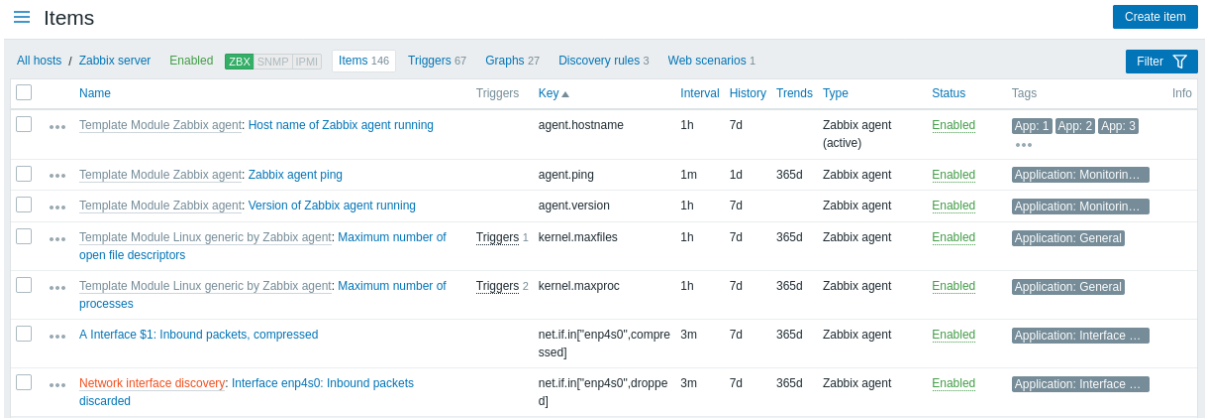
See also more details about host interface unreachability.

1 Items

Overview

The item list for a host can be accessed from *Configuration* → *Hosts* by clicking on *Items* for the respective host.

A list of existing items is displayed.



Displayed data:

Column	Description
Item menu	Click on the three-dot icon to open the menu for the specific item with these options: **Latest data** - see latest data of the item **Create trigger** - create a trigger based on this item **Triggers** - click to see a list with links to already-configured trigger of this item **Create dependent item** - create a dependent item for this item **Create dependent discovery rule** - create a dependent discovery rule for this item
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*

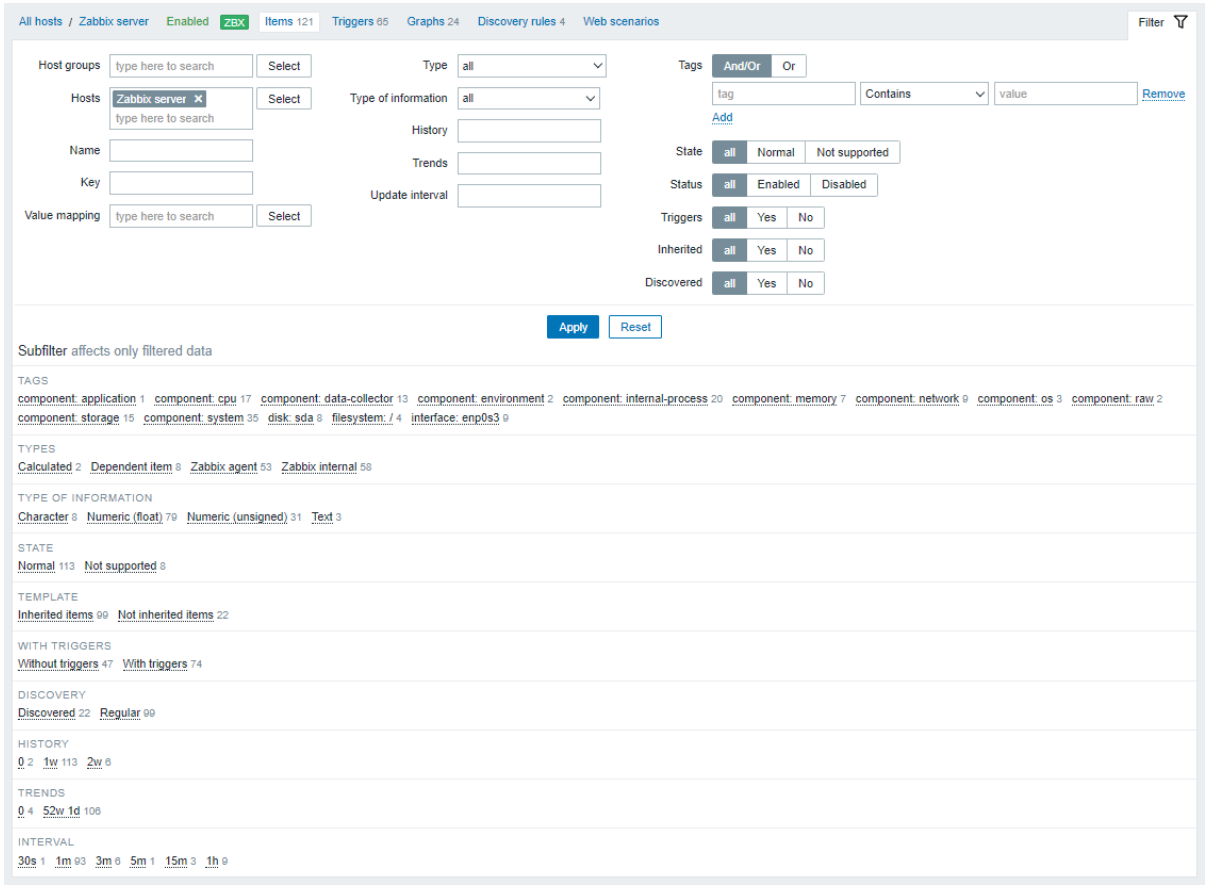
- *Disable* - change item status to *Disabled*
- *Execute now* - execute a check for new item values immediately. Supported for **passive** checks only (see more details). Note that when checking for values immediately, configuration cache is not updated, thus the values will not reflect very recent changes to item configuration.
- *Clear history* - 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.

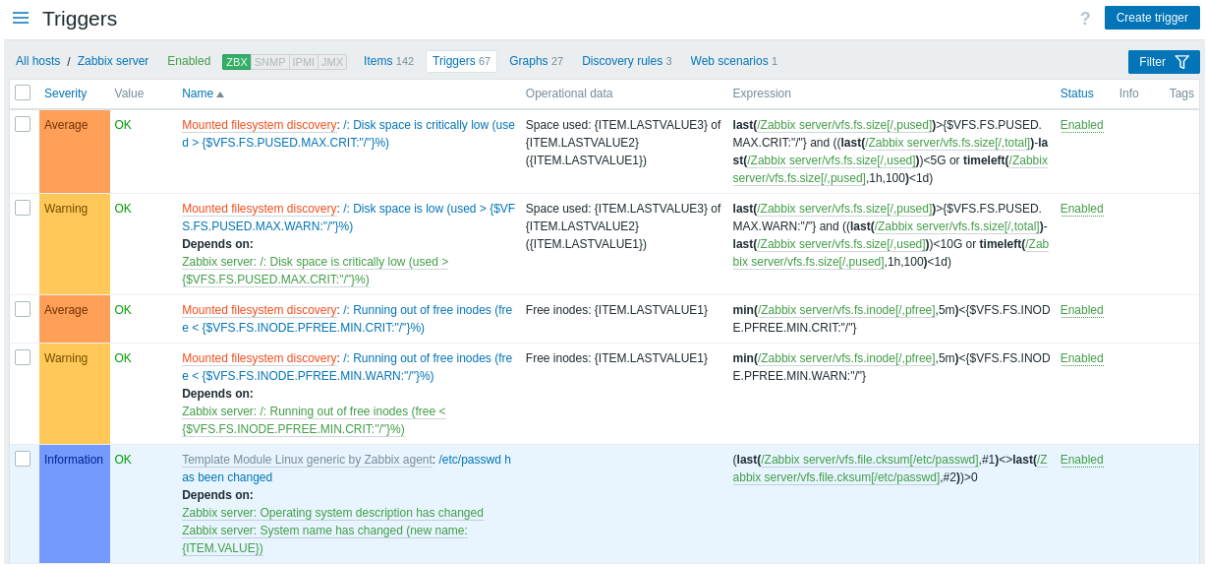
Parameter	Description
Tags	Specify tags to limit the number of items displayed. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - 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.

The **Subfilter** below the filter offers further filtering options (for the data already filtered). You can select groups of items with a common parameter value. Upon clicking on a group, it gets highlighted and only the items with this parameter value remain in the list.

2 Triggers

Overview

The trigger list for a host can be accessed from *Configuration* → *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.

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 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:

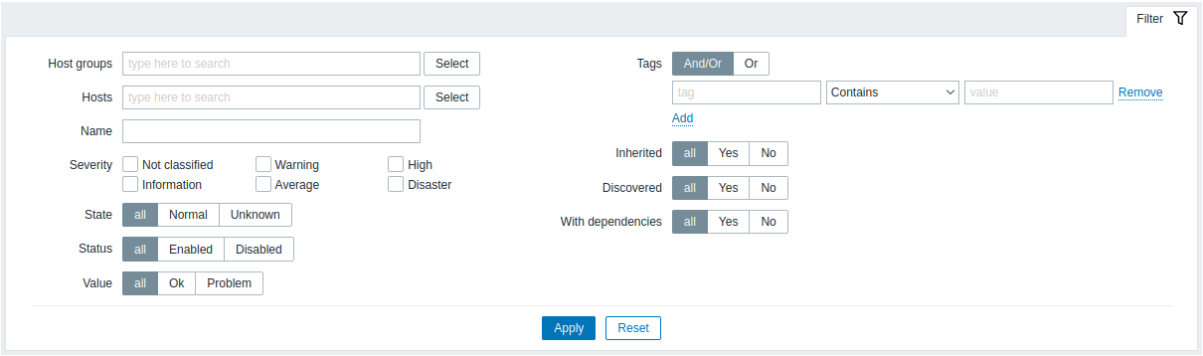
- *Enable* - change trigger status to *Enabled*.
- *Disable* - change trigger status to *Disabled*.
- *Copy* - copy the triggers to other hosts or templates.
- *Mass update* - update several properties for a number of triggers at once.
- *Delete* - delete the triggers.

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.

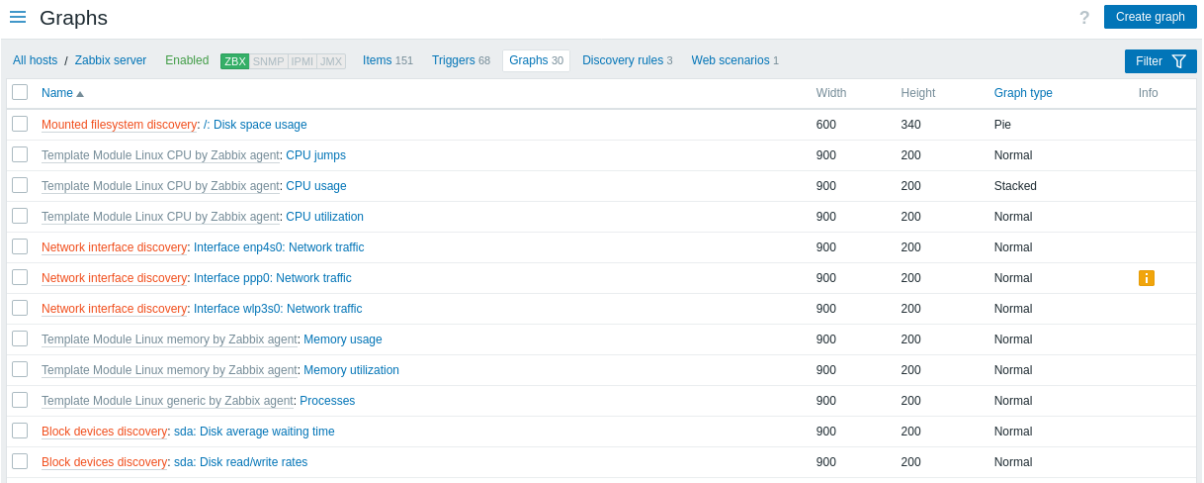
Parameter	Description
Tags	Filter by trigger tag name and value. It is possible to include as well as exclude specific tags and tag values. Several conditions can be set. Tag name matching is always case-sensitive. There are several operators available for each condition: **Exists** - include the specified 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 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 *Configuration* → *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

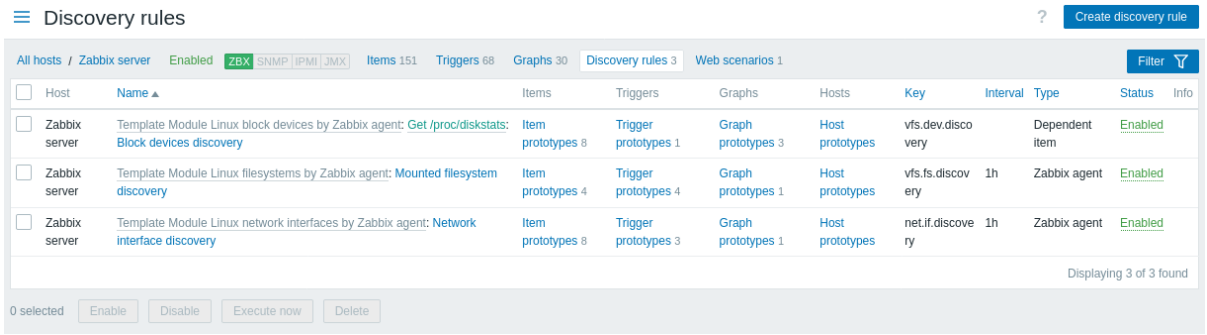
You can filter graphs by host group and host. For better search performance, data is searched with macros unresolved.

4 Discovery rules

Overview

The list of low-level discovery rules for a host can be accessed from *Configuration* → *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:

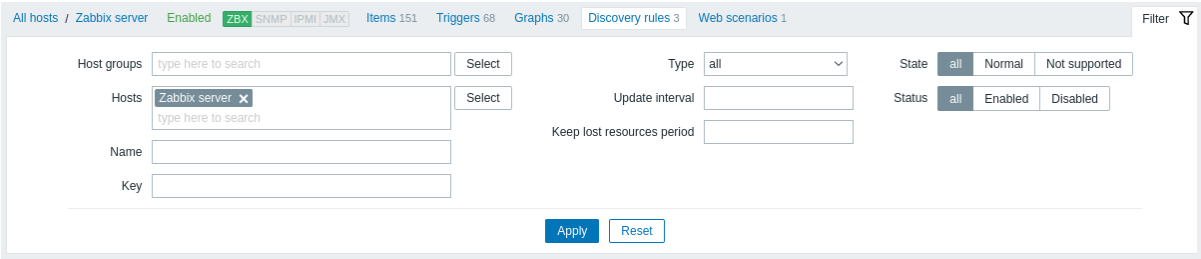
- *Enable* - change the low-level discovery rule status to *Enabled*.
- *Disable* - change the low-level discovery rule status to *Disabled*.
- *Execute now* - perform discovery based on the discovery rules immediately. See more details. Note that when performing discovery immediately, the configuration 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.

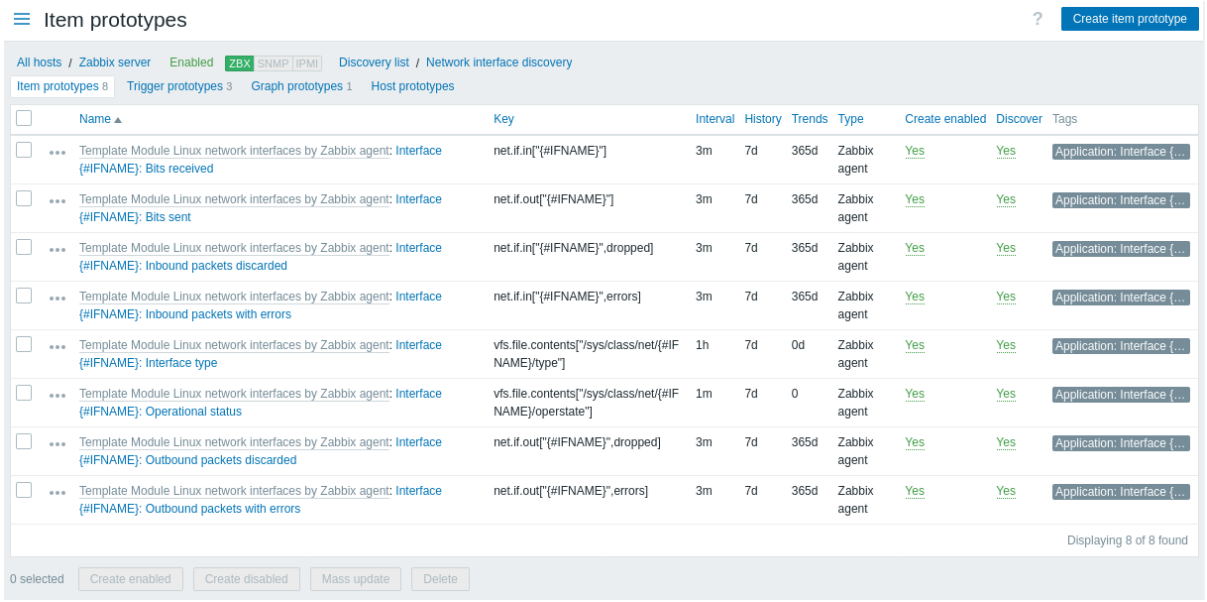


Parameter	Description
Host groups	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
Hosts	Filter by one or more hosts.
Name	Filter by discovery rule name.
Key	Filter by discovery item key.
Type	Filter by discovery item type.
Update interval	Filter 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.
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:

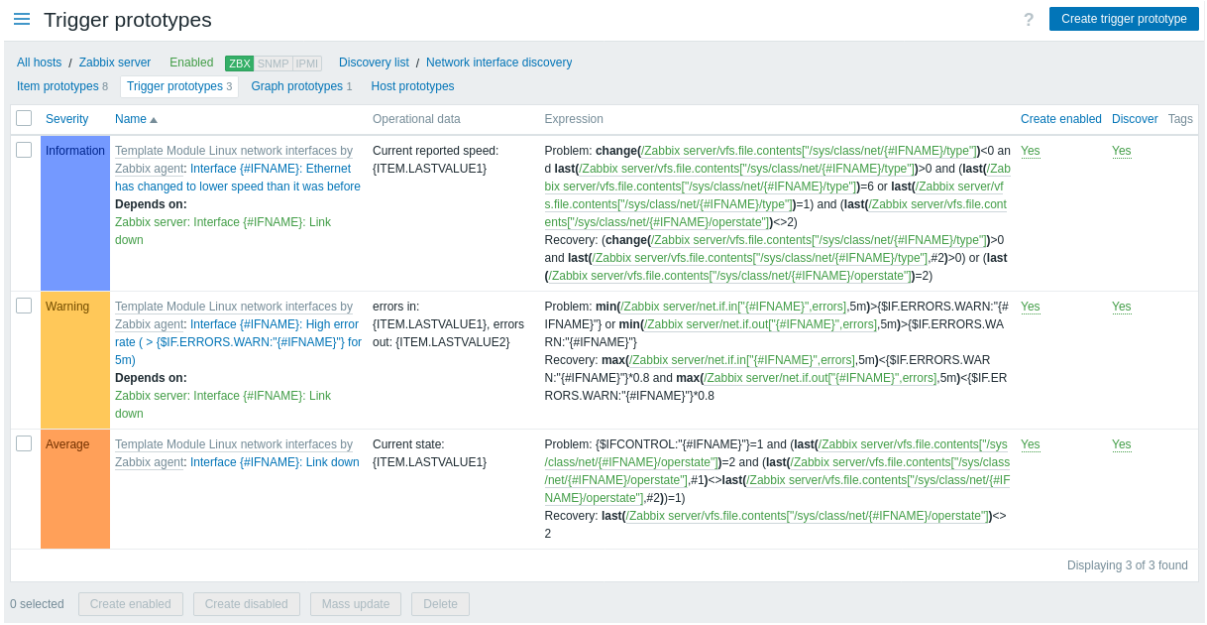
- *Create enabled* - create these items as *Enabled*
- *Create disabled* - create these items as *Disabled*
- *Mass update* - mass update these item prototypes
- *Delete* - delete these item prototypes

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.
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:

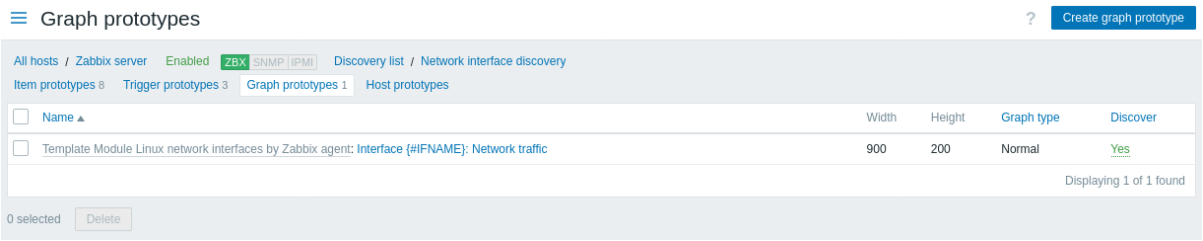
- *Create enabled* - create these triggers as *Enabled*
- *Create disabled* - create these triggers as *Disabled*
- *Mass update* - mass update these trigger prototypes
- *Delete* - delete these trigger prototypes

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:

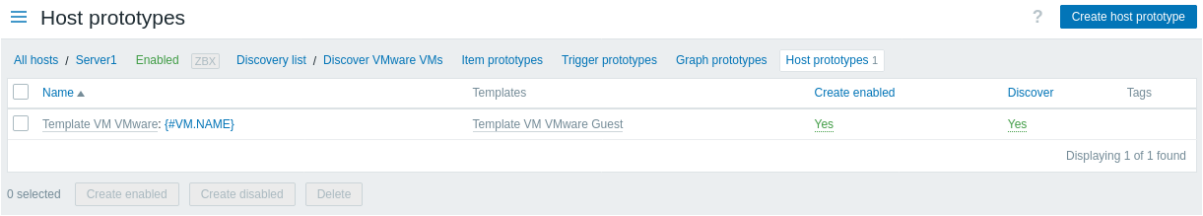
- *Delete* - delete these graph prototypes

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.

Column	Description
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

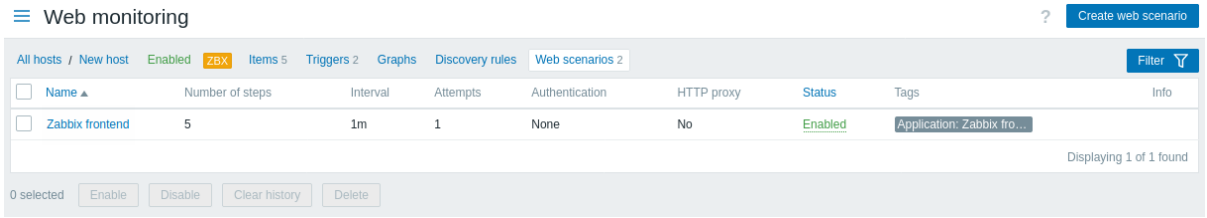
To use these options, mark the checkboxes before the respective host prototypes, then click on the required button.

5 Web scenarios

Overview

The web scenario list for a host can be accessed from *Configuration* → *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:

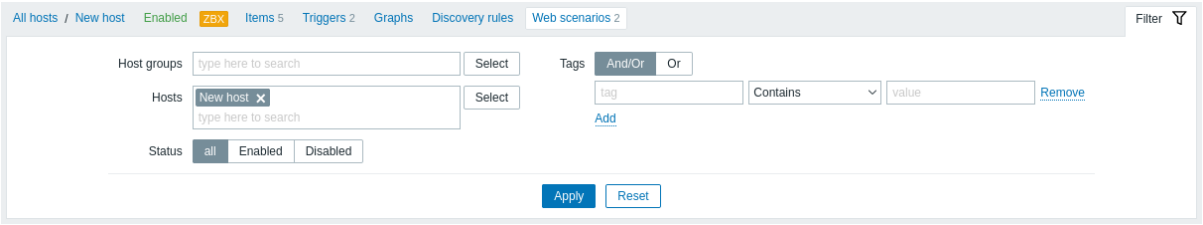
- *Enable* - change the scenario status to *Enabled*
- *Disable* - change the scenario status to *Disabled*
- *Clear history* - clear history and trend data for the scenarios
- *Delete* - delete the web scenarios

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.

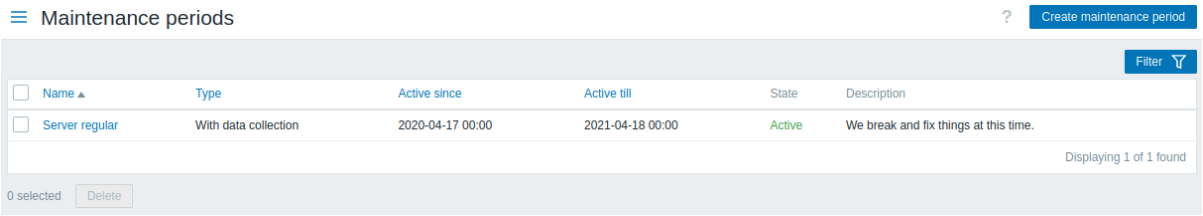


5 Maintenance

Overview

In the *Configuration* → *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:

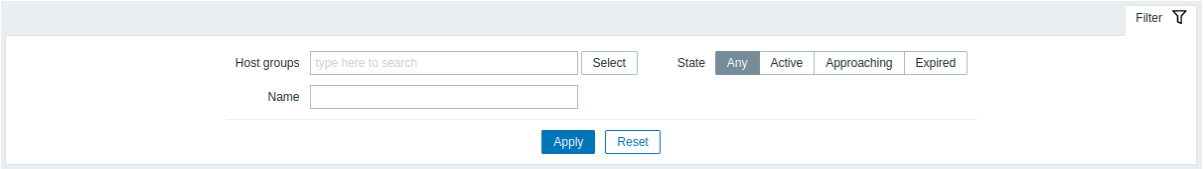
- *Delete* - delete the maintenance periods

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 Actions

Overview

In the *Configuration* → *Actions* section users can configure and maintain actions.

The actions displayed are actions assigned to the selected event source (trigger, discovery, autoregistration, internal actions).

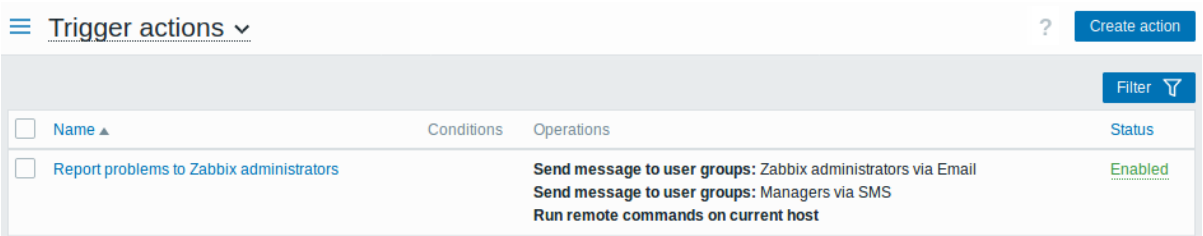
Actions are grouped into subsections by event source (trigger, service, discovery, autoregistration, internal actions). The list of available subsections appears upon pressing on *Actions* in the *Configuration* menu section. It is also possible to switch between subsections by using a title dropdown in the top left corner.

After selecting a subsection, a page with a list of existing actions with their details will be displayed.

For users without Super admin rights actions are displayed according to 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

Note:
Actions for services are maintained in a similar way in the *Services->Service actions* menu section. User's access to specific service actions depends on the user role permissions set in the *Access to services* menu section.



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 (e-mail, 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.

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*

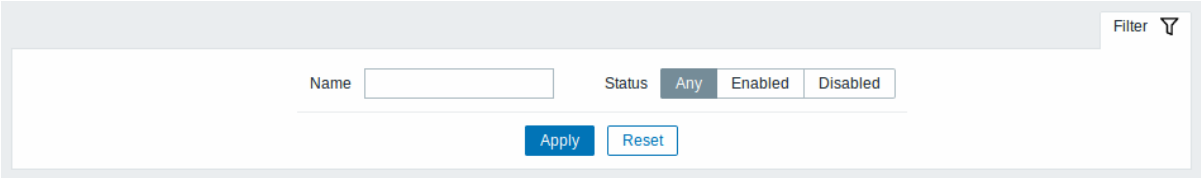
- *Delete* - delete the actions

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.

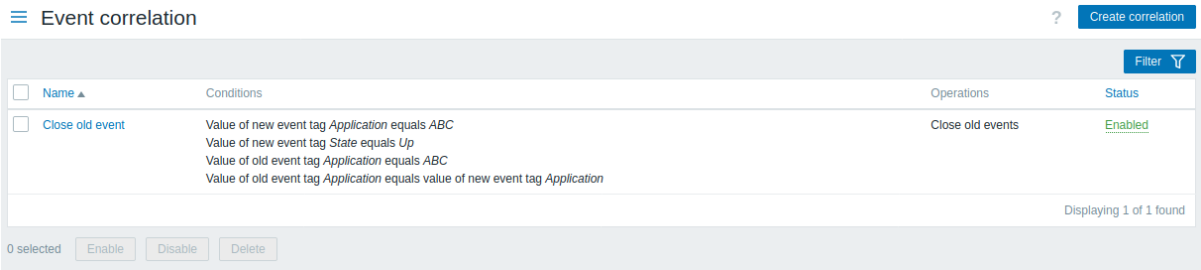
The *Filter* link is available above the list of actions. If you click on it, a filter becomes available where you can filter actions by name and status.



7 Event correlation

Overview

In the *Configuration* → *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:

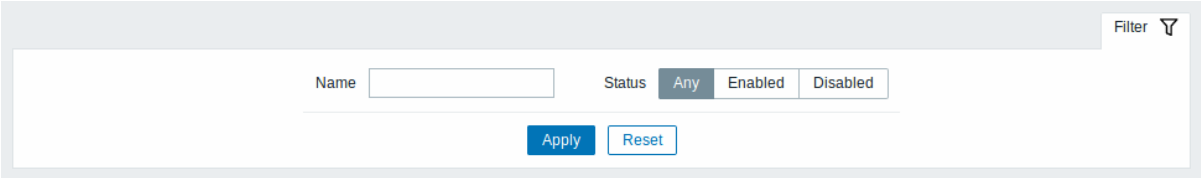
- *Enable* - change the correlation rule status to *Enabled*
- *Disable* - change the correlation rule status to *Disabled*
- *Delete* - delete the correlation rules

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.

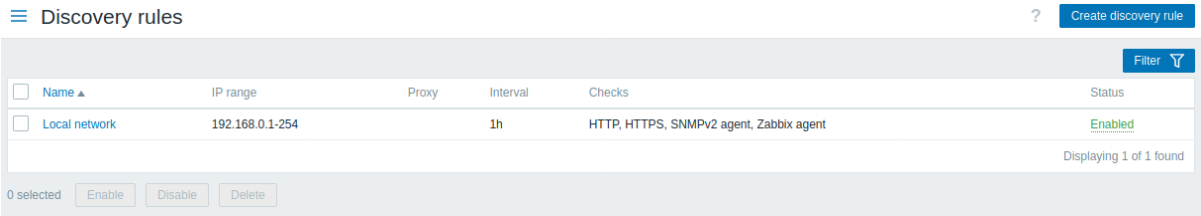


8 Discovery

Overview

In the *Configuration* → *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:

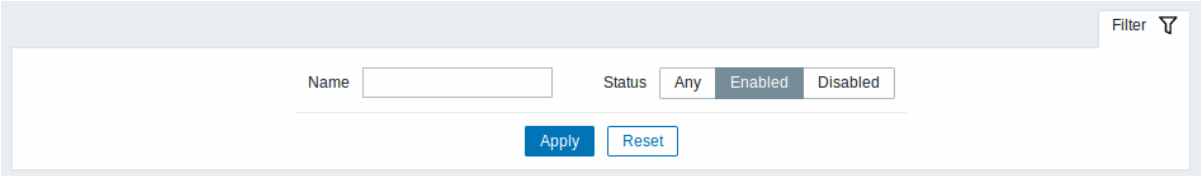
- *Enable* - change the discovery rule status to *Enabled*
- *Disable* - change the discovery rule status to *Disabled*
- *Delete* - delete the discovery rules

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.



6 Administration

Overview

The Administration menu is for administrative functions of Zabbix. This menu is available to users of Super Administrators type only.

1 General

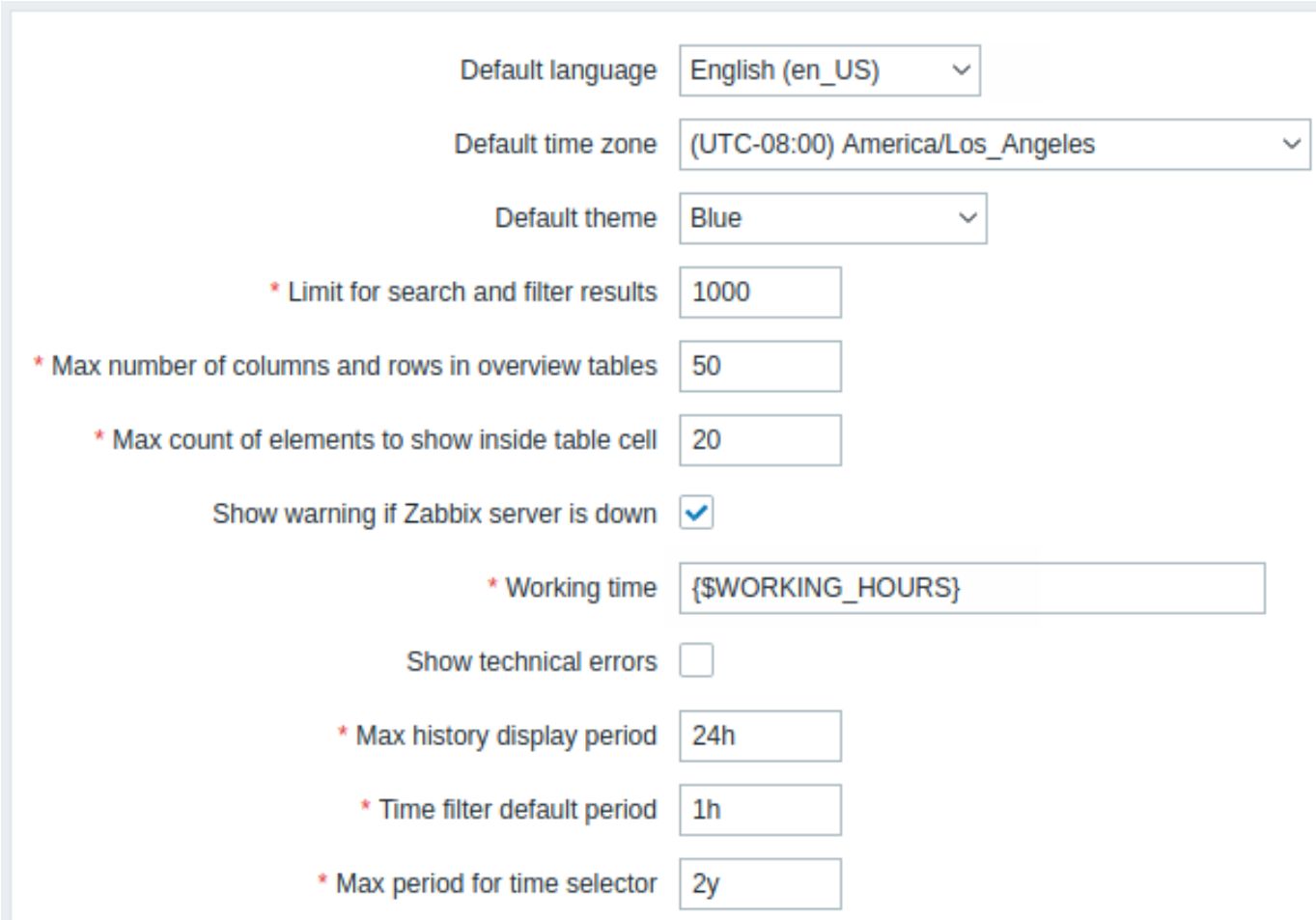
Overview

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.



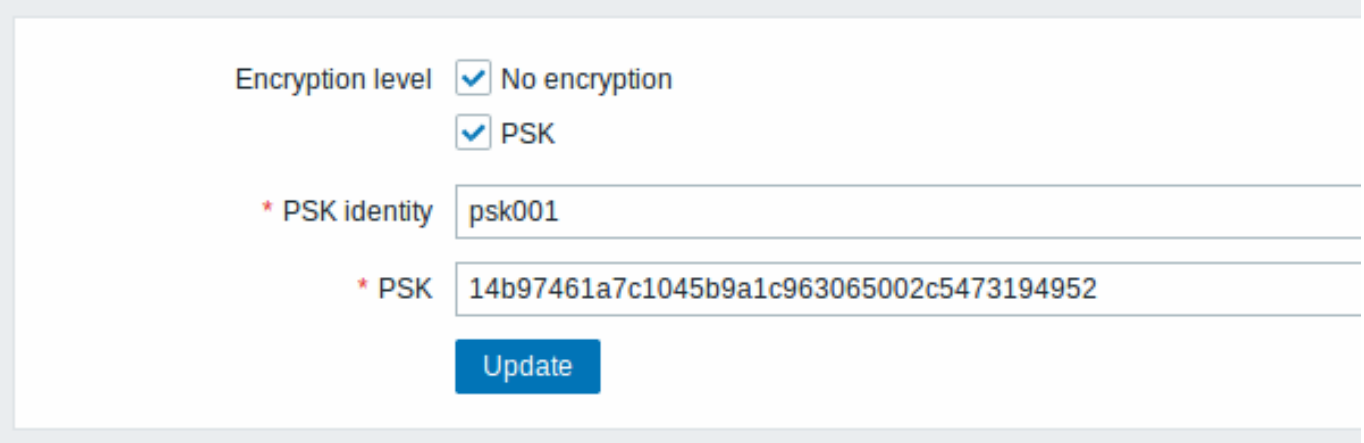
Configuration parameters:

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.
Limit for search and filter results	Maximum amount of elements (rows) that will be displayed in a web-interface list, for example, in *Configuration* → *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.
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.

Parameter	Description
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.

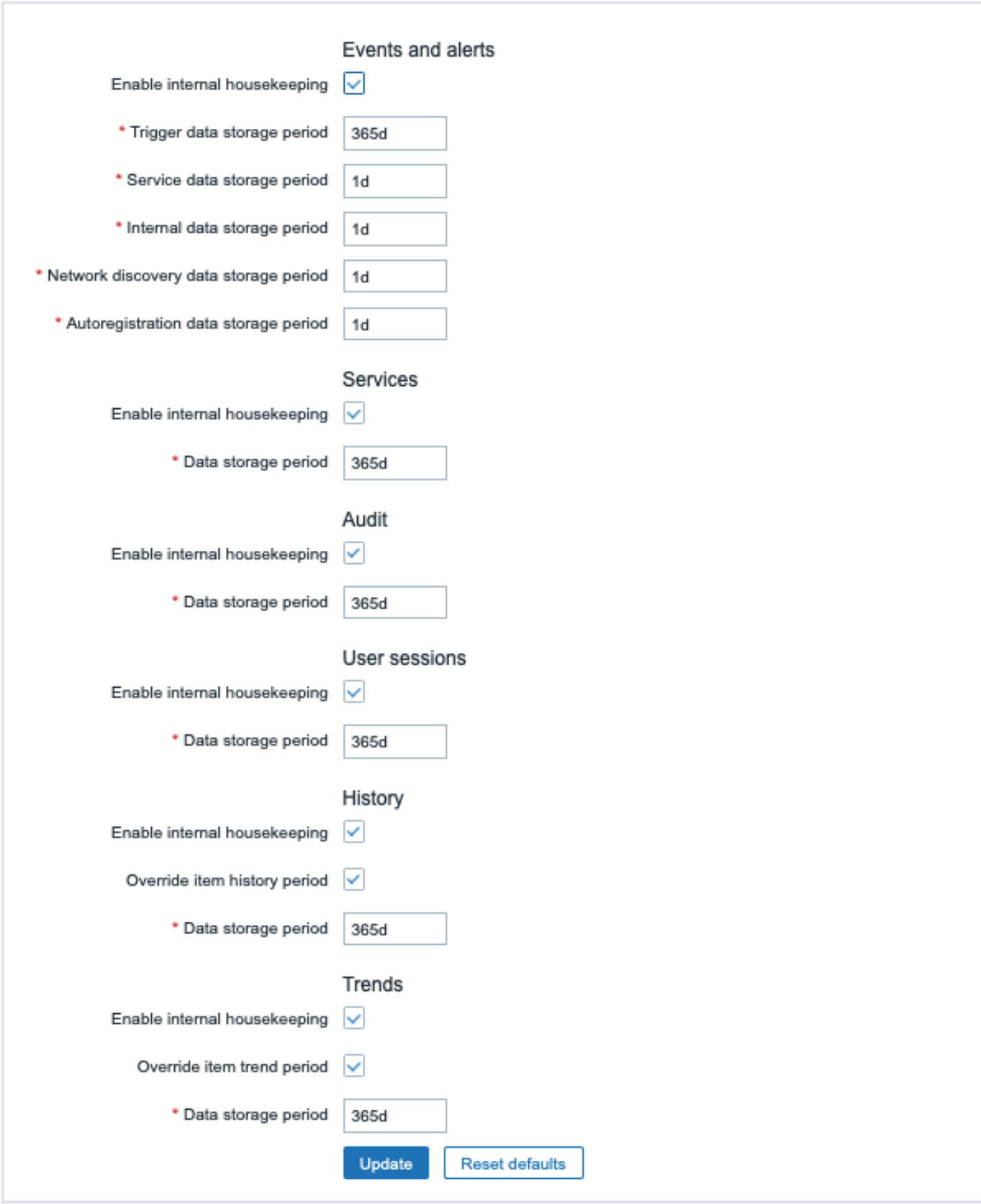
Configuration parameters:

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 Housekeeper

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.

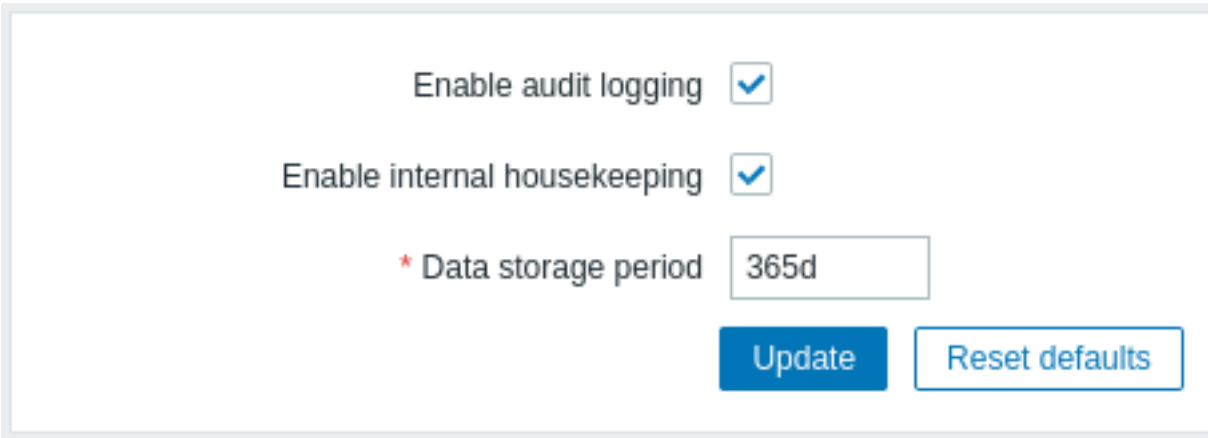
Attention:
If using TimescaleDB, in order to take full advantage of TimescaleDB automatic partitioning of history and trends tables, *Override item history period* and *Override item trend period* options must be enabled as well as *Enable internal housekeeping* option for history and trends. Otherwise, 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.

Reset defaults button allows to revert any changes made.

4 Audit log

This section allows configuring audit log settings.

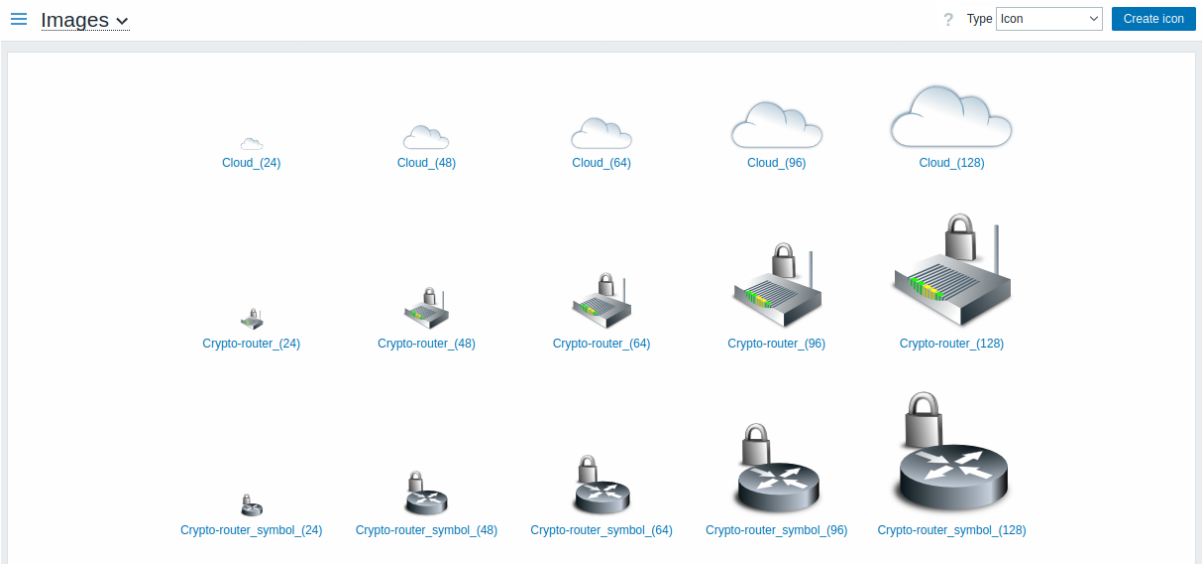


The following parameters are available:

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.

5 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.

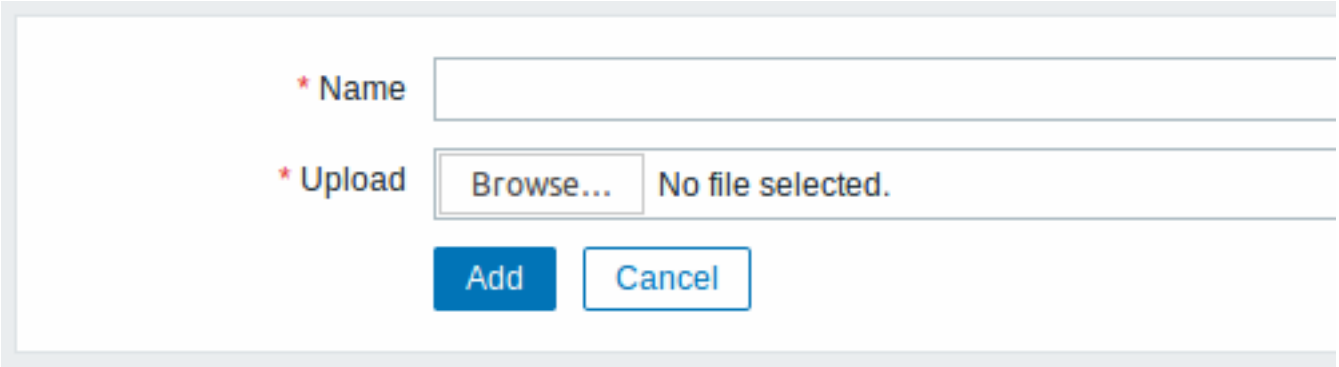


Image attributes:

Parameter	Description
Name	Unique name of an image.
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.

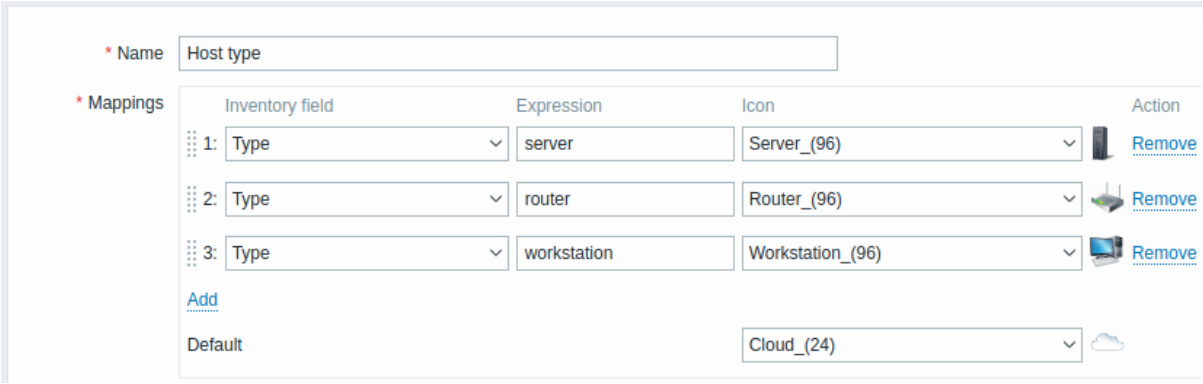
The upload of an image may fail if the image size is close to 1 MB and the `max_allowed_packet` MySQL configuration parameter is at a default of 1MB. In this case, increase the max_allowed_packet parameter.

6 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:

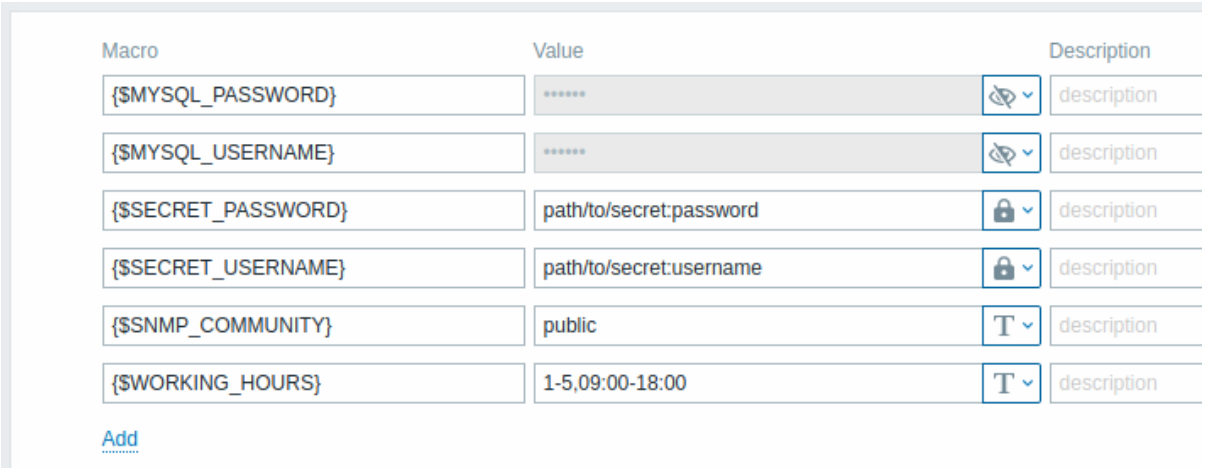
Parameter	Description
Name	Unique name of icon map.
Mappings	A list of mappings. The order of mappings determines which one will have priority. You can move mappings up and down the list with drag-and-drop.
Inventory field	Host inventory field that will be looked into to seek a match.
Expression	Regular expression describing the match.
Icon	Icon to use if a match for the expression is found.
Default	Default icon to use.

7 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.

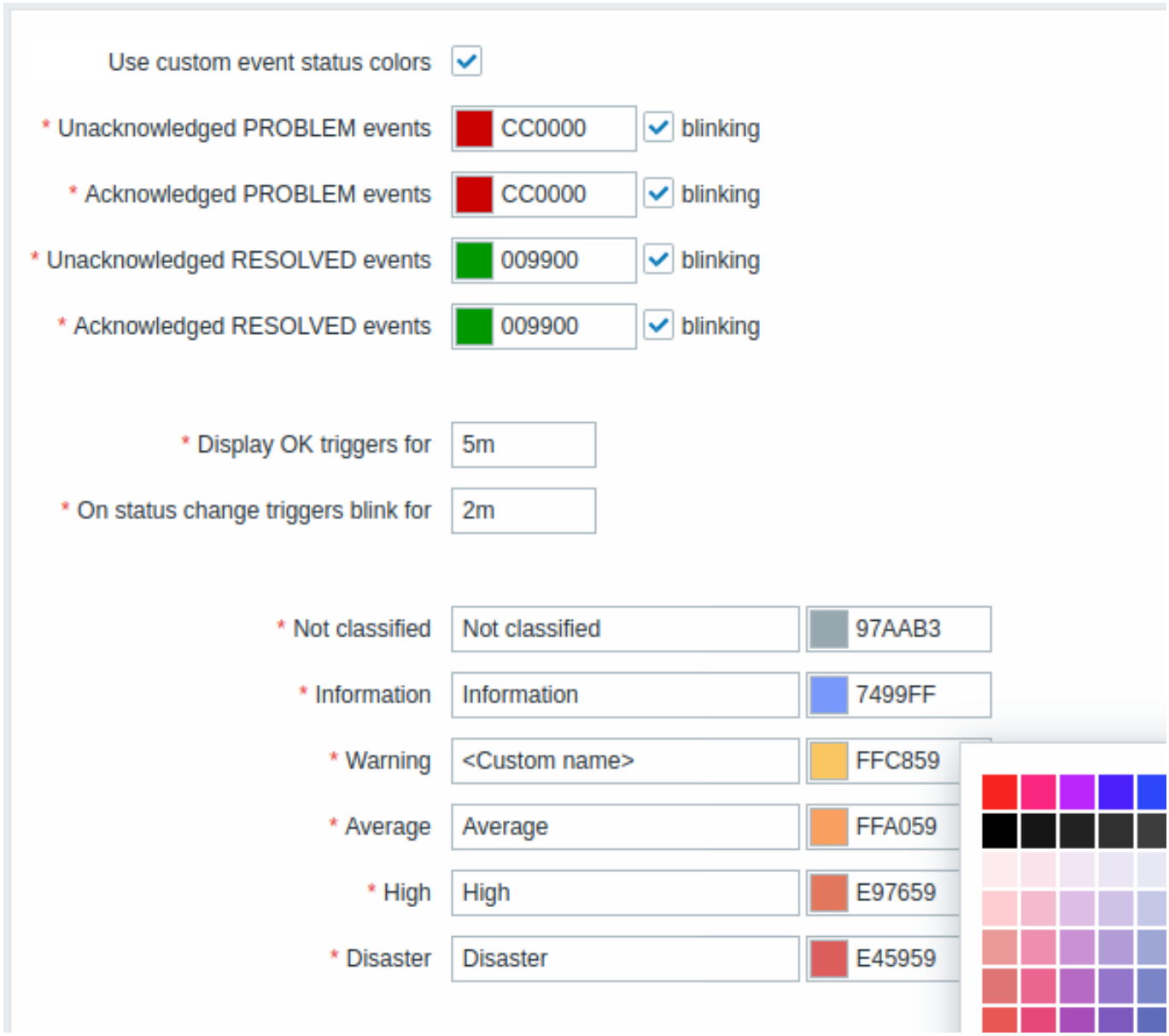
8 Macros

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.



9 Trigger displaying options

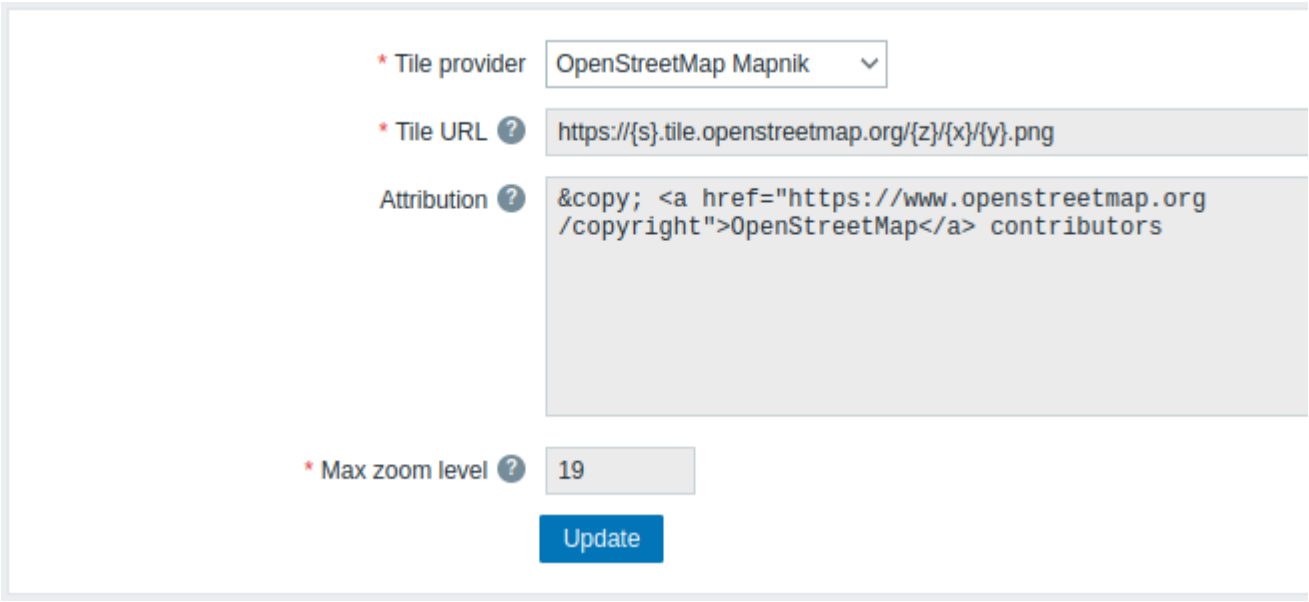
This section allows customizing how trigger status is displayed in the frontend and trigger severity names and colors.



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.

10 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.



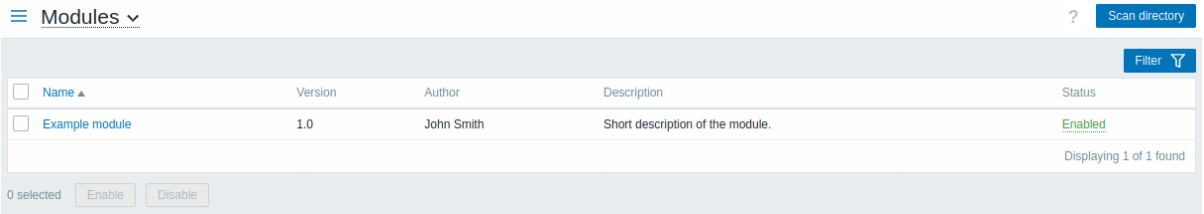
Parameter	Description
Tile provider	Select one of the available tile service providers or select *Other* to add another tile provider or self-hosted tiles (see Using a custom tile service provider).
Tile URL	The URL template for loading and displaying the tile layer on geographical maps. This field is 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	Tile provider attribution data to be displayed in a small text box on the map. This field is editable 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.

11 Modules

This section allows to administer custom frontend modules.

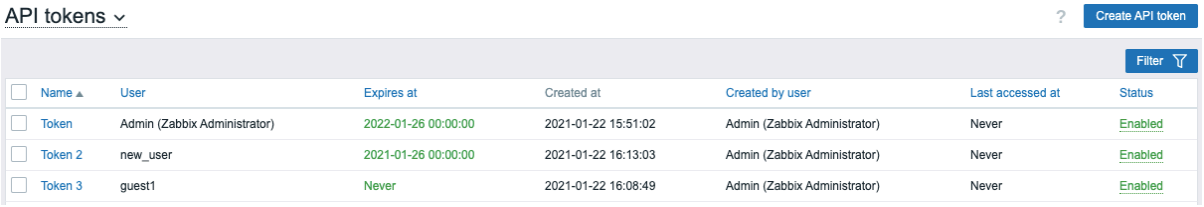


Click on *Scan directory* to register/unregister any custom modules. Registered modules will appear in the list, along with their details. Unregistered modules will be removed from the list.

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.

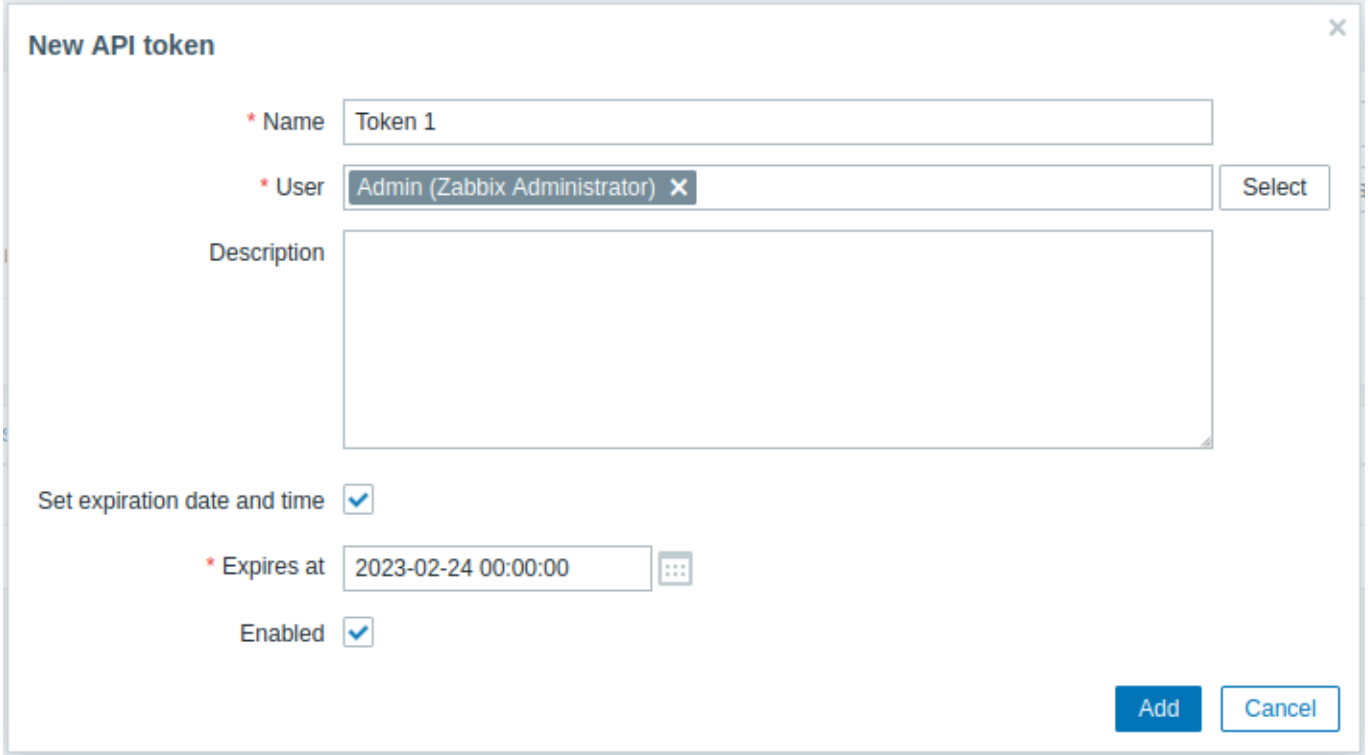
12 API tokens

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.

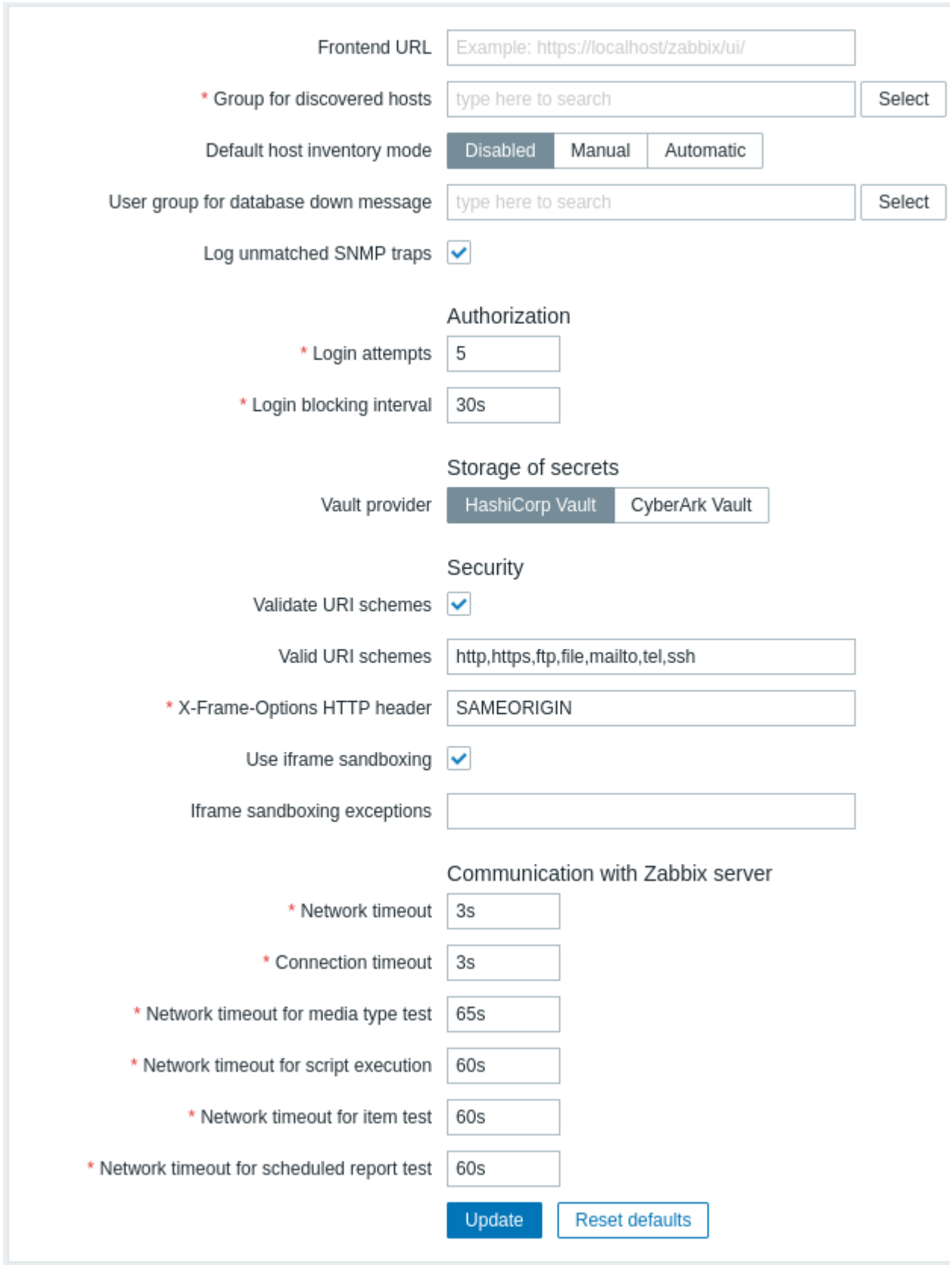
Warning:
Auth token value cannot be viewed again later. It is only available immediately after creating a token. If you lose a saved token you will have to regenerate it and doing so will create a new authorization string.

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.

13 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.
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	Uncheck the box to disable URI scheme validation against the whitelist defined in *Valid URI schemes*. (enabled by default).
Valid URI schemes	A comma-separated list of allowed URI schemes. Applies to all fields in the frontend where URIs are used (for example, map element URLs). this field is editable only if *Validate URI schemes* is selected.
X-Frame-Options HTTP header	Value of HTTP X-Frame-options header. Supported values: **SAMEORIGIN** (default) - the page can only be displayed in a frame on the same origin as the page itself. **DENY** - the page cannot be displayed in a frame, regardless of the site attempting to do so. **null** - disable X-Frame-options header (not recommended). Or a list (string) of comma-separated hostnames. If a listed hostname is not among allowed, the SAMEORIGIN option is used.
Use iframe sandboxing	This parameter determines whether retrieved URL content should be put into the sandbox or not. Note, that turning off sandboxing is not recommended.
Iframe sandboxing exceptions	If sandboxing is enabled and this field is empty, all sandbox attribute restrictions apply. To disable some of the restrictions, specified them in this field. This disables only restrictions listed here, other restrictions will still be applied. See sandbox attribute description for additional information.

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).

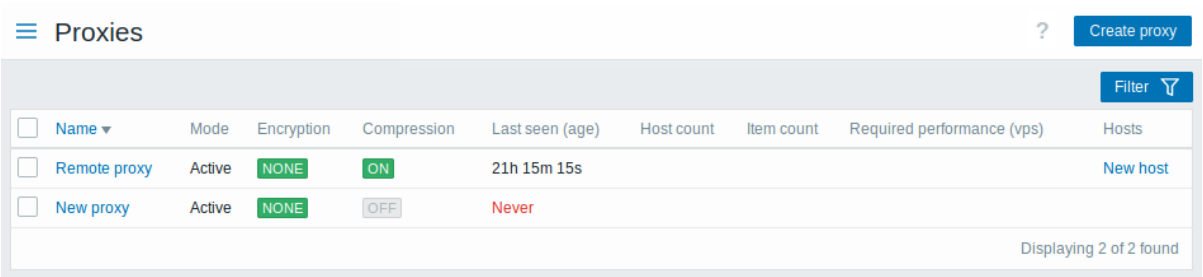
2 Proxies

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
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 performance (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:

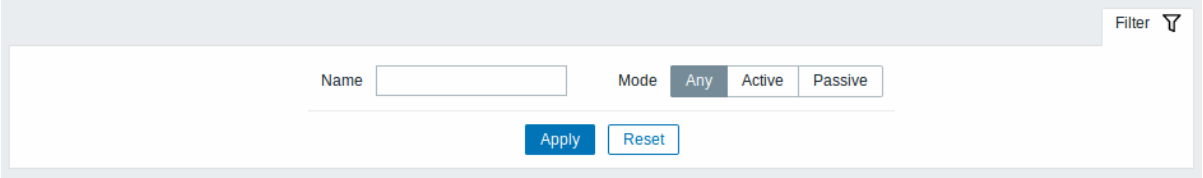
- *Refresh configuration* - refresh configuration of the proxies
- *Enable hosts* - change the status of hosts monitored by the proxy to *Monitored*
- *Disable hosts* - change the status of hosts monitored by the proxy to *Not monitored*
- *Delete* - delete the proxies

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 and mode.



3 Authentication

Overview

The *Administration → Authentication* section allows to specify the global user authentication method to Zabbix and internal password requirements. The available 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 method to LDAP system-wide or enable LDAP authentication only for specific user groups.

To set LDAP as default authentication method for all users, navigate to the *LDAP* tab and configure authentication parameters, then return to the *Authentication* tab and switch *Default authentication* selector to LDAP.

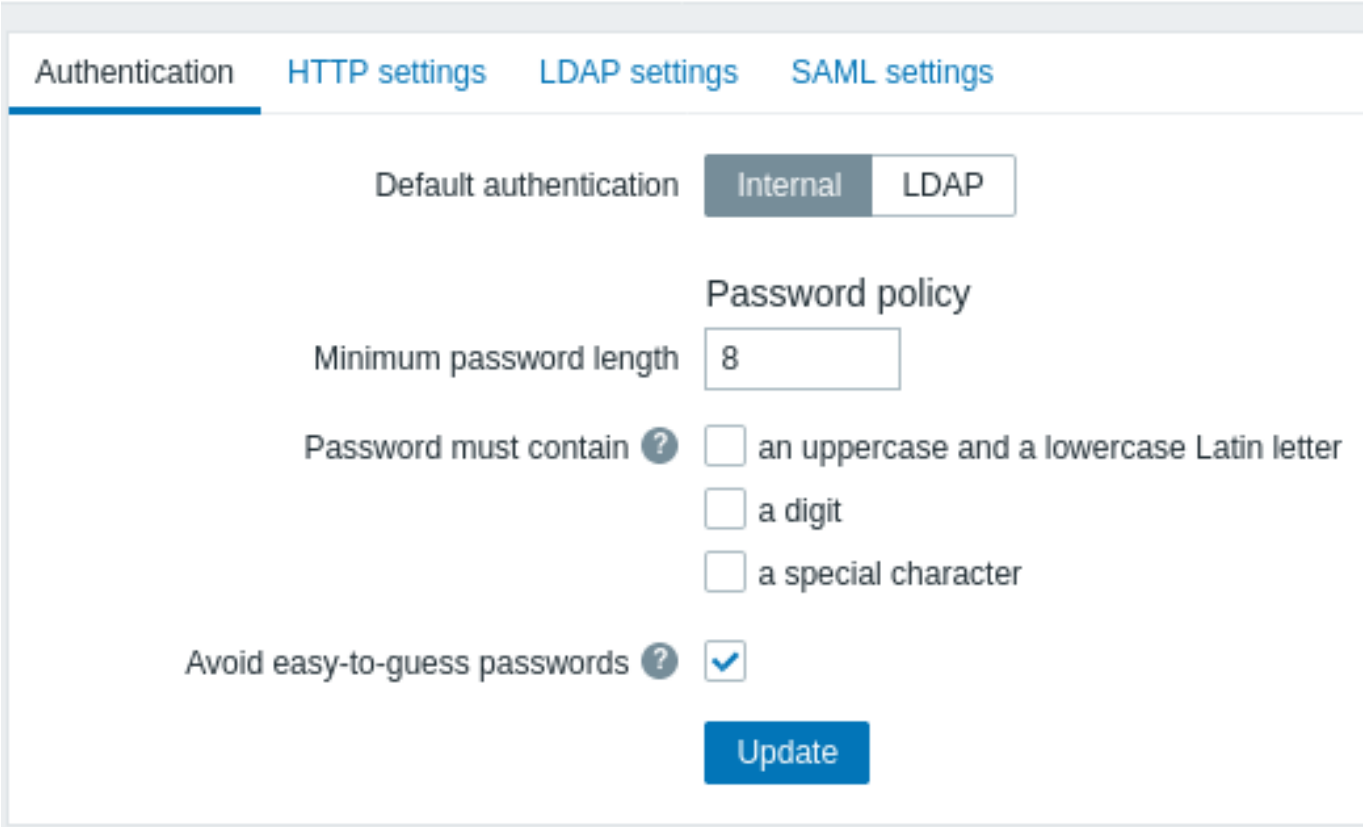
Note that the authentication method can be fine-tuned on the user group level. Even if LDAP authentication is set globally, some user groups can still be authenticated by Zabbix. These groups must have frontend access set to Internal. Vice versa, if internal authentication is used globally, LDAP authentication details can be specified and used for specific user groups whose frontend access is 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 internal authentication method.

HTTP and SAML 2.0 authentication methods can be used in addition to the default authentication method.

Internal authentication

The *Authentication* tab allows defining custom password complexity requirements for internal Zabbix users.

Authentication



The following password policy options can be configured:

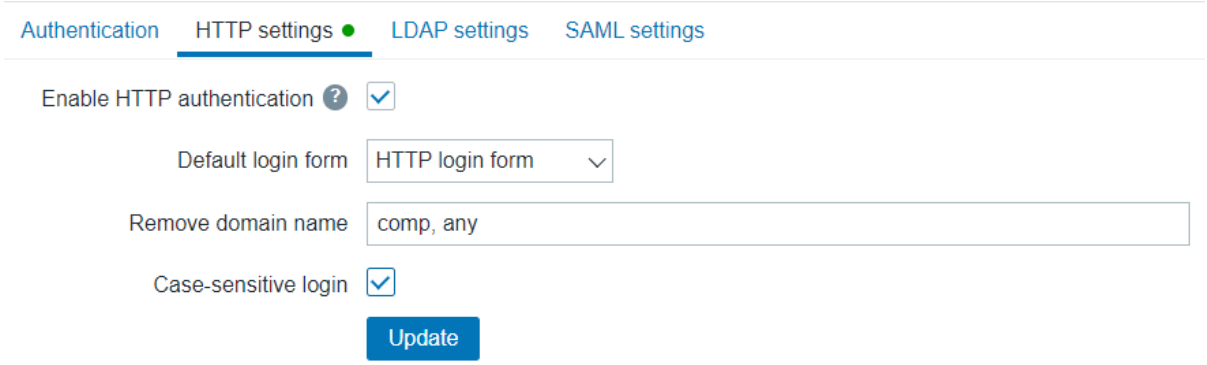
Parameter	Description
Minimum password length	By default, the minimum password length is set to 8. Supported range: 1-70. Note that passwords longer than 72 characters will be truncated.
Password must contain	Mark one or several checkboxes to require usage of specified characters in a password: -an uppercase 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.
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 *Administration→Users* menu.

HTTP authentication

HTTP or web server-based authentication (for example: Basic Authentication, 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 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.
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.

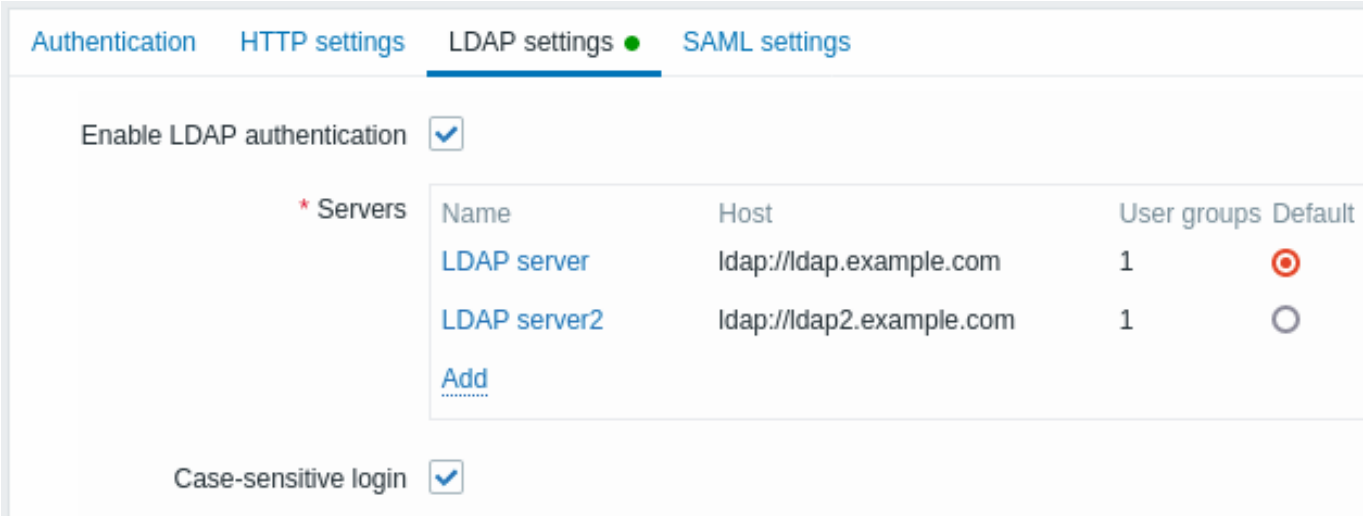
LDAP authentication

External LDAP authentication 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.

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.

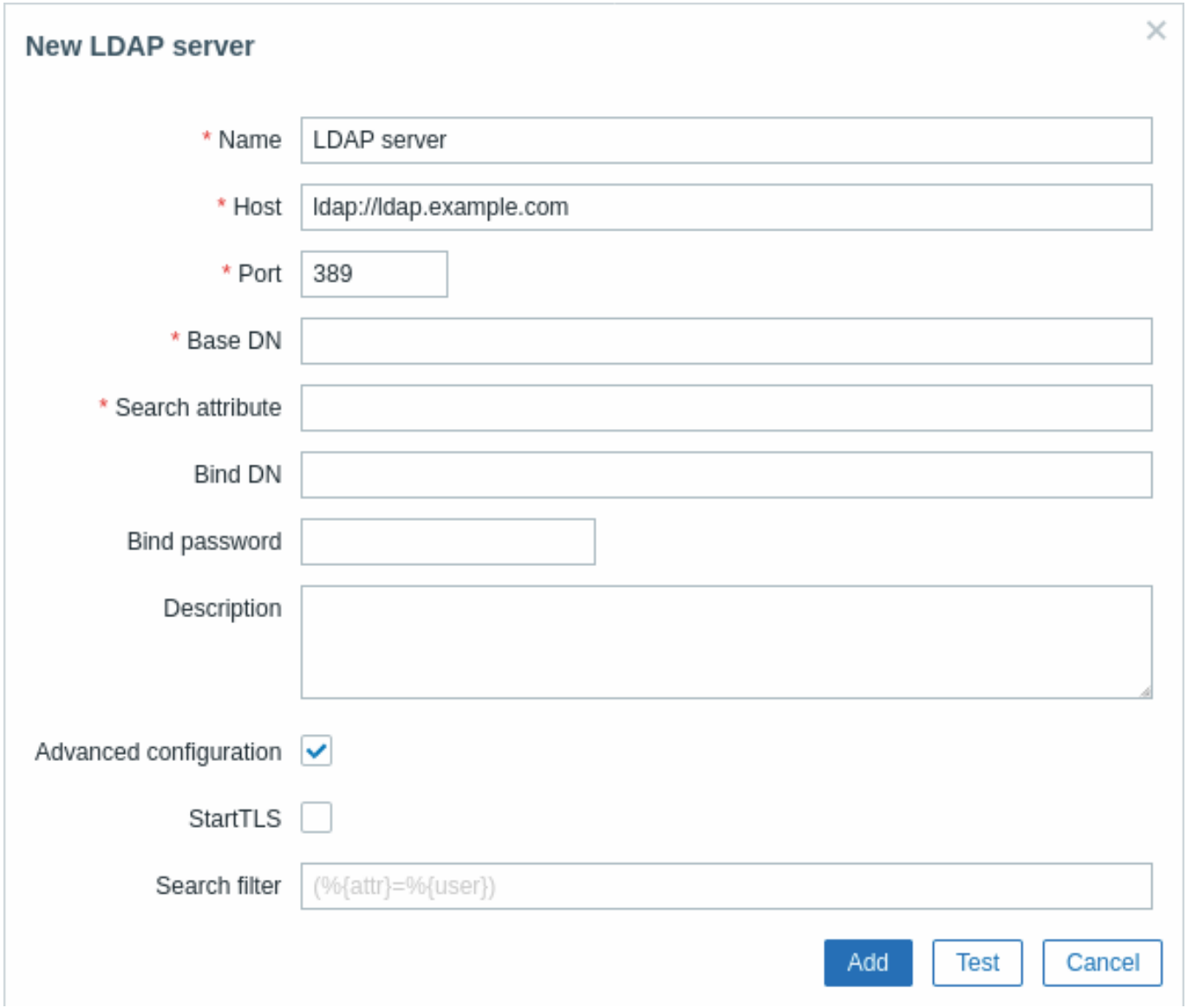
If a user is in multiple user groups and multiple LDAP servers, the first server in the list of LDAP servers sorted by name in ascending order will be used for authentication.

Zabbix LDAP authentication works at least with Microsoft Active Directory and OpenLDAP.



Configuration parameters:

Parameter	Description
Enable LDAP authentication	Mark the checkbox to enable LDAP authentication.
Servers	Click on *Add* to configure an LDAP server (see LDAP server configuration parameters 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).



LDAP server configuration parameters:

Parameter	Description
Name	Name of the LDAP server in Zabbix configuration.
Host	Host of the LDAP server. For example: ldap://ldap.example.com For secure LDAP server use *ldaps* protocol. 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.
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 search accounts: ou=Users,ou=system (for OpenLDAP), DC=company,DC=com (for Microsoft Active Directory)
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.

Parameter	Description
Bind password	LDAP password of the account for binding and searching over the LDAP server.
Description	Description of the LDAP server.
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 To access this option, mark the *Advanced configuration* checkbox first.
Search filter	Define a custom string when authenticating user in LDAP. The following placeholders are supported: `%{attr}` - search attribute name (uid, sAMAccountName) `%{user}` - user username value to authenticate. If omitted then LDAP will use the default filter: `(%{attr}=%{user})` To access this option, mark the *Advanced configuration* checkbox first.

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.

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 *ldap_search* account name.

SAML authentication

SAML 2.0 authentication can be used to sign in to Zabbix. Note that a user must 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.

Note:
If SAML authentication is enabled, users will be able to choose between logging in locally or via SAML Single Sign-On.

Setting up the 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.

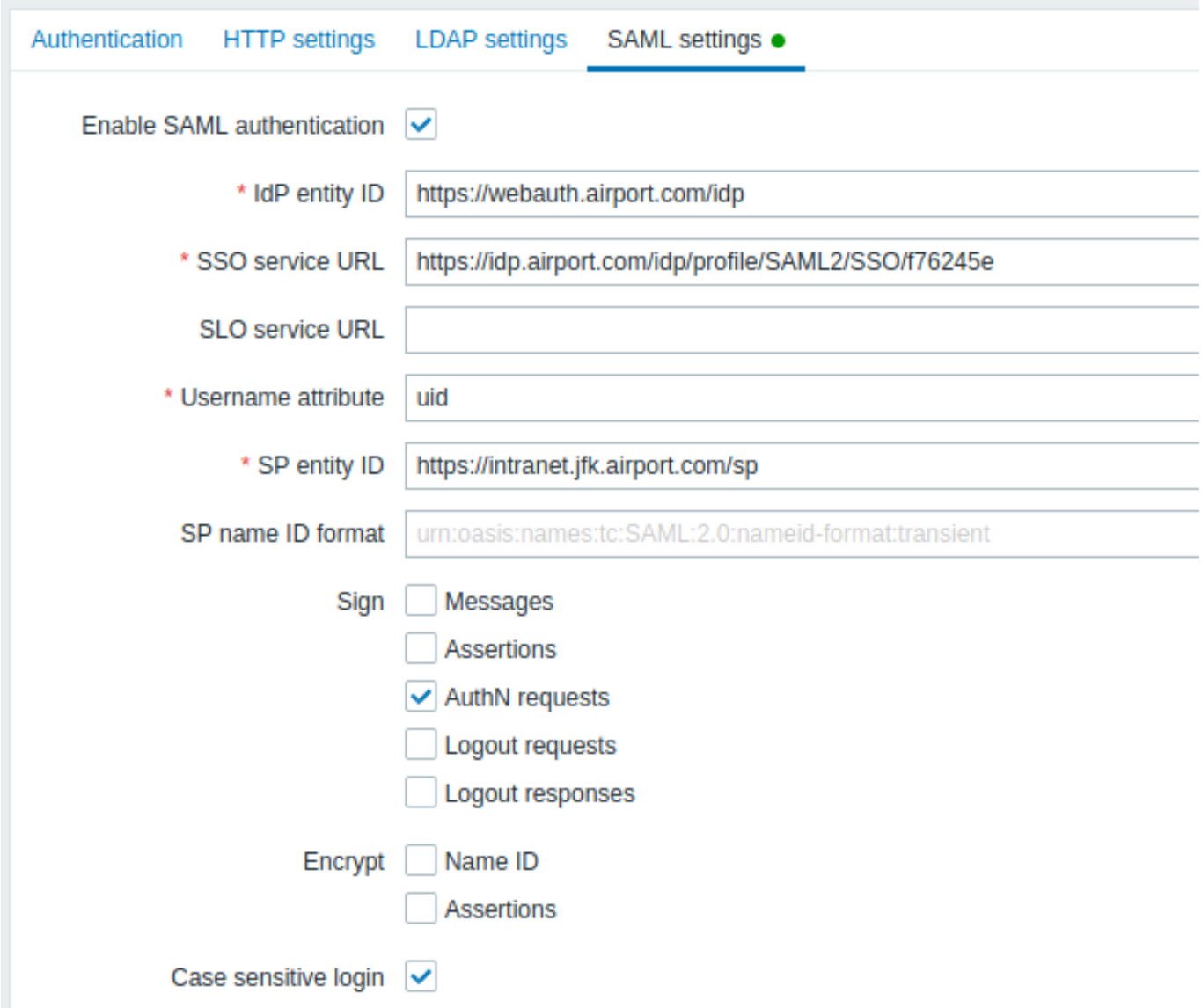
To use SAML authentication Zabbix should be configured in the following way:

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.
IDP entity ID	The unique identifier of SAML identity provider.
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.

Parameter	Description
// Username attribute//	SAML attribute to be used as a username when logging into Zabbix. 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 identifier of SAML service provider.
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: *Assertions* *Name ID*
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).

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',
        // ...
    ],
    // ...
];
```

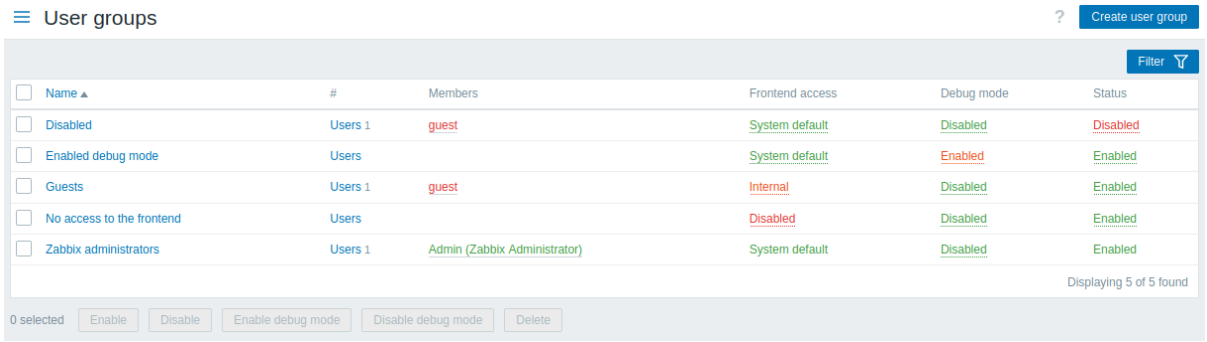
4 User groups

Overview

In the *Administration* → *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.

Column	Description
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** - Zabbix, LDAP or HTTP authentication; depending on the chosen authentication method **Internal** - the user is authenticated by Zabbix regardless of system settings **Disabled** - frontend access for this user is disabled. 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:

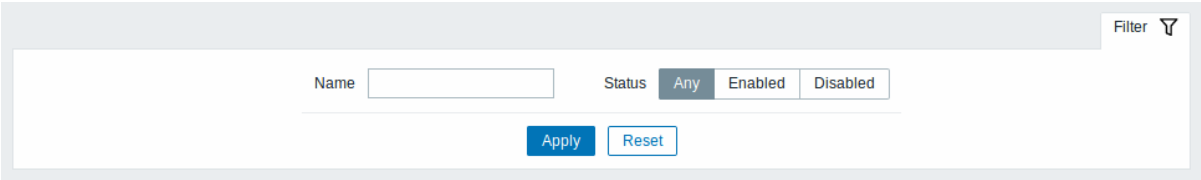
- *Enable* - change the user group status to *Enabled*
- *Disable* - change the user group status to *Disabled*
- *Enable debug mode* - enable debug mode for the user groups
- *Disable debug mode* - disable debug mode for the user groups
- *Delete* - delete the user groups

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.



5 User roles

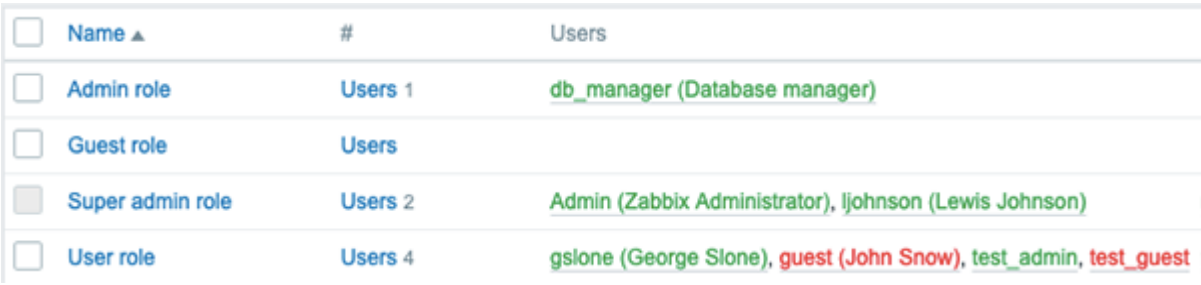
Overview

In the *Administration* → *User roles* section roles that can be assigned to system users and specific permissions for each role are maintained.

Default user roles

By default, Zabbix is configured with four user roles, which have a pre-defined set of permissions:

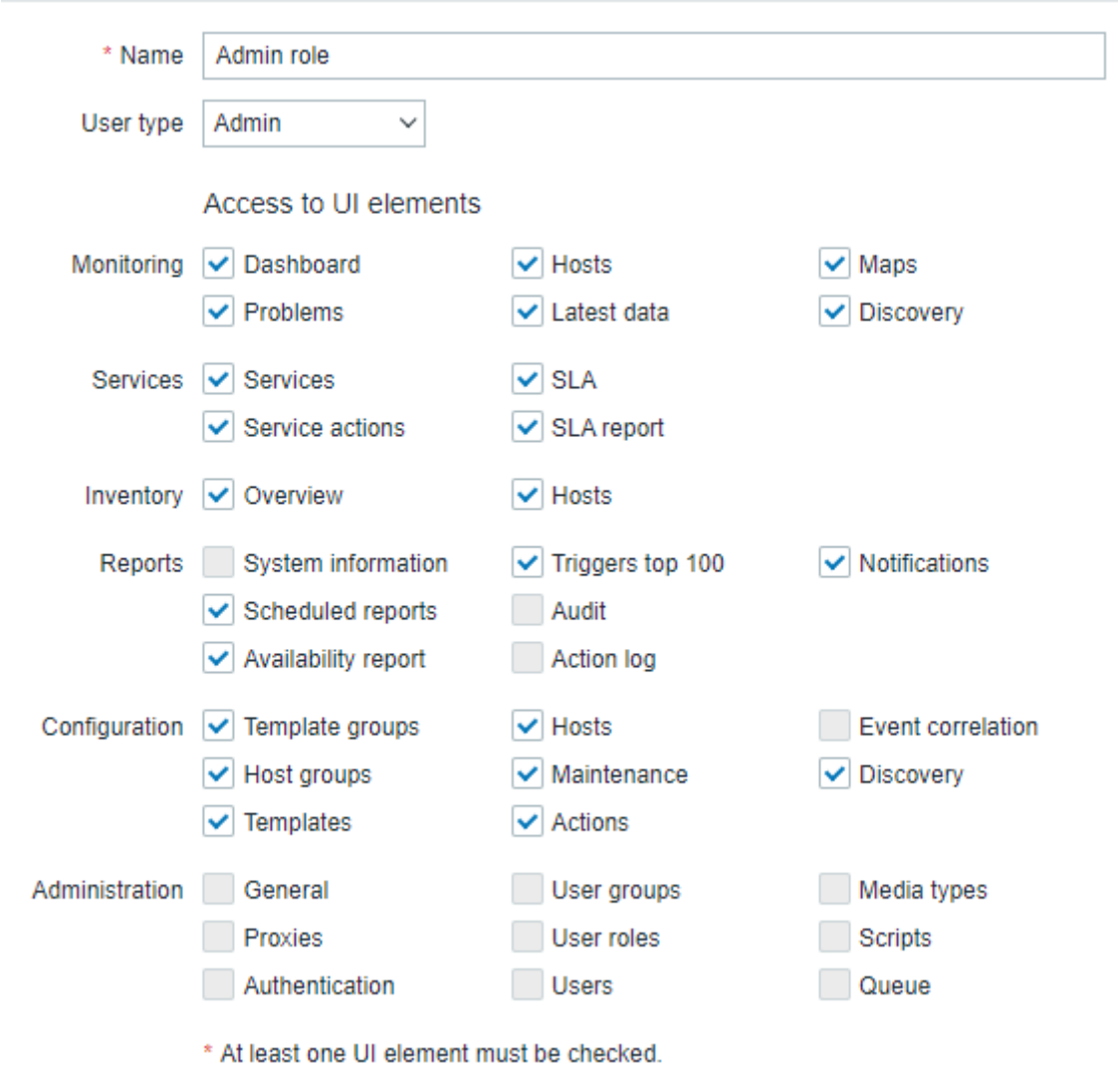
- Admin role
- Guest role
- Super admin role
- User role



Note:
Default *Super admin role* cannot be modified or deleted, because at least one Super admin user with unlimited privileges must exist in Zabbix.

Zabbix users with type Super admins and proper permissions can modify or delete existing roles or create new custom roles.

To create a new role, click on the *Create user role* button at the top right corner. To update an existing role, press on the role name to open the configuration form.



Available permission options along with default permission sets for pre-configured user roles in Zabbix are described below.

Parameter	Description	Default user roles			
		Super admin role	*Admin role*	*User role*	*Guest role*
User type	Selected user type determines the list of available permissions. Upon selecting a user type, all available permissions for this user type are granted by default. Uncheck the checkbox(es) to revoke certain permissions for the user role. Checkboxes for permissions not available for this user type are grayed out.	Super admin	Admin	User	User

Access to UI elements					
Monitoring					
Dashboard	Enable/disable access to a specific Monitoring menu section and underlying pages.	Yes	Yes	Yes	Yes
Problems					
Hosts					
Latest data					
Maps					
Discovery				No	No
Services					
Services	Enable/disable access to a specific Services menu section and underlying pages.	Yes	Yes	Yes	Yes
Service actions				No	No
SLA					
SLA report				Yes	Yes
Inventory					
Overview	Enable/disable access to a specific Inventory menu section and underlying pages.	Yes	Yes	Yes	Yes
Hosts					
Reports					
System information	Enable/disable access to a specific Reports menu section and underlying pages.	Yes	No	No	No
Scheduled reports			Yes		
Availability report				Yes	Yes
Triggers top 100					
Audit			No	No	No
Action log					
Notifications			Yes		
Configuration					
Template groups	Enable/disable access to a specific *Configuration* menu section and underlying pages.	Yes	Yes	No	No
Host groups					
Templates					
Hosts					
Maintenance					
Actions					
Event correlation			No		
Discovery			Yes		
Administration					
General	Enable/disable access to a specific *Administration* menu section and underlying pages.	Yes	No	No	No
Proxies					
Authentication					
User groups					
User roles					
Users					
Media types					
Scripts					
Queue					

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	
Default access to new actions	Enable/disable access to new actions.				

Notes:

- Each user may have only one role assigned.
- If an element is restricted, users will not be able to access it even by entering a direct URL to this element into the browser.
- Users of type *User* or *Admin* cannot change their own role settings.
- Users of type *Super admin* can modify settings of their own role (not available for the default *Super admin role*), but not the user type.
- Users of all levels cannot change their own user type.

See also:

- Configuring a user

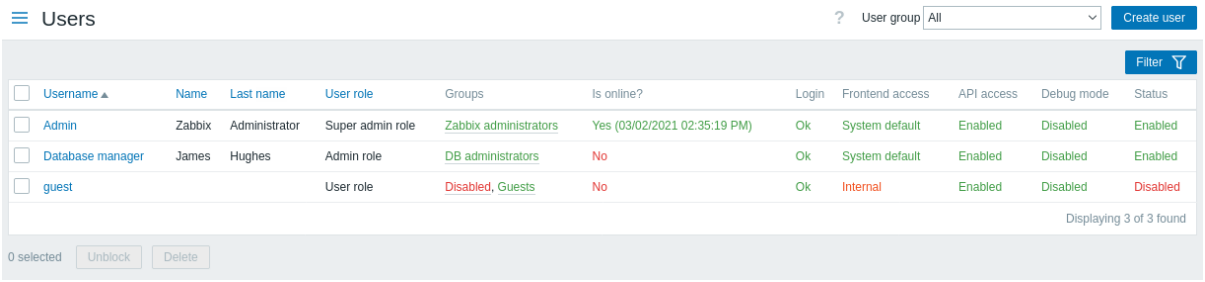
6 Users

Overview

In the *Administration* → *Users* section users of the system are maintained.

Users

A listing of existing users with their details is displayed.



From the dropdown to the right in the *Users* bar you can choose whether to display all users or those belonging to one particular group.

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.
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* section (five by default). By clicking on *Blocked* you can unblock the user.
Frontend access	Frontend access level is displayed - *System default*, *Internal* 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.

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:

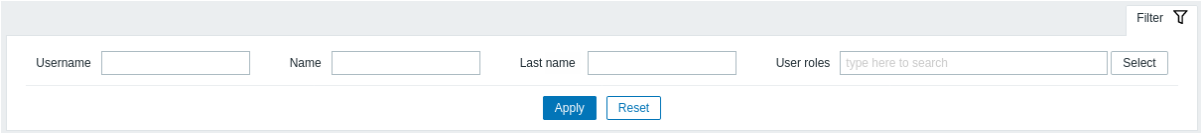
- *Unblock* - re-enable system access to blocked users
- *Delete* - delete the users

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 and user role.



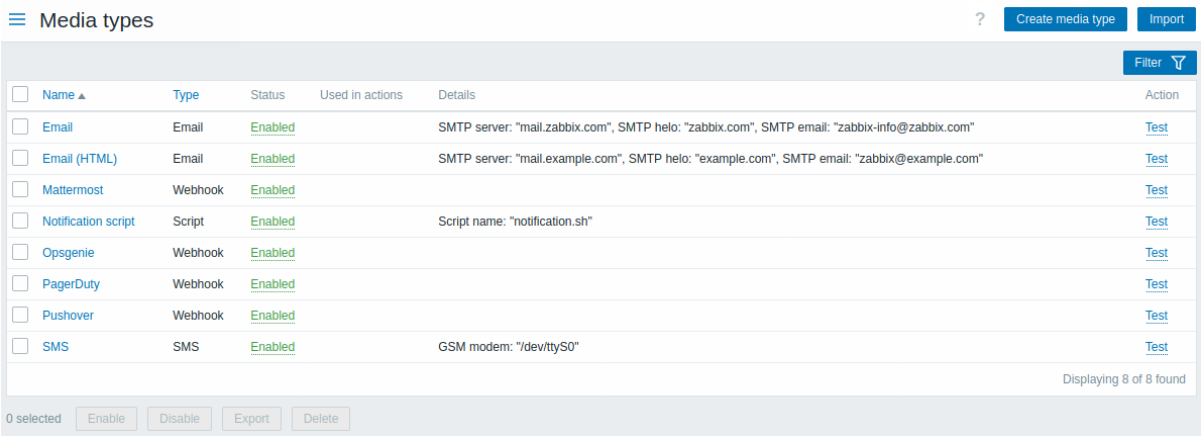
7 Media types

Overview

In the *Administration* → *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 e-mail 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 (e-mail, 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.

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:

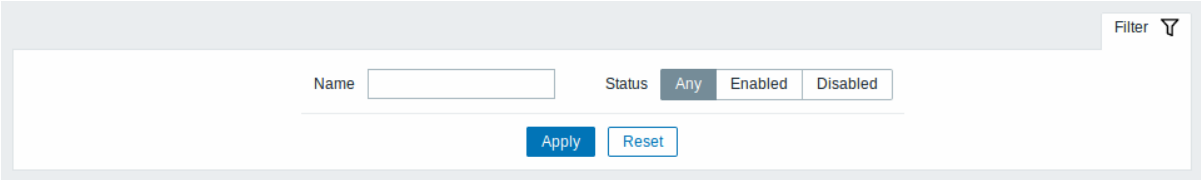
- *Enable* - change the media type status to *Enabled*
- *Disable* - change the media type status to *Disabled*
- *Export* - export the media types to a YAML, XML or JSON file
- *Delete* - delete the media types

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.



8 Scripts

Overview

In the *Administration* → *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:

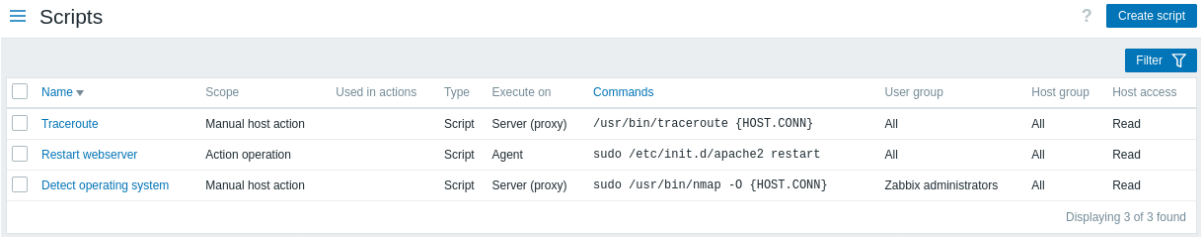
- from the host menu in various frontend locations (*Dashboard*, *Problems*, *Latest data*, *Maps*, etc)
- from the event menu
- can be run as an action operation

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.
Used in actions	Actions where the script is used are displayed.
Type	Script type is displayed - *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:

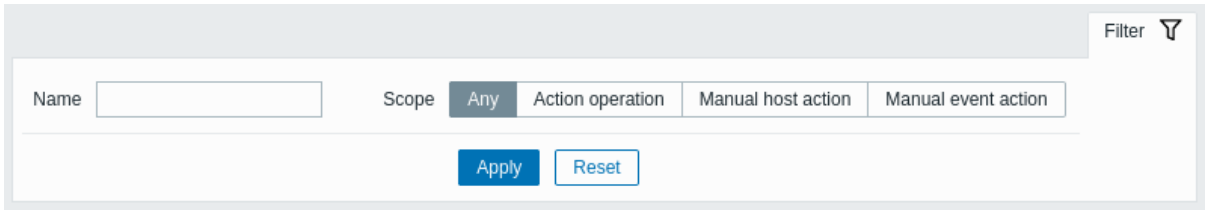
- *Delete* - delete the scripts

To use this option, mark the checkboxes before the respective scripts and click on *Delete*.

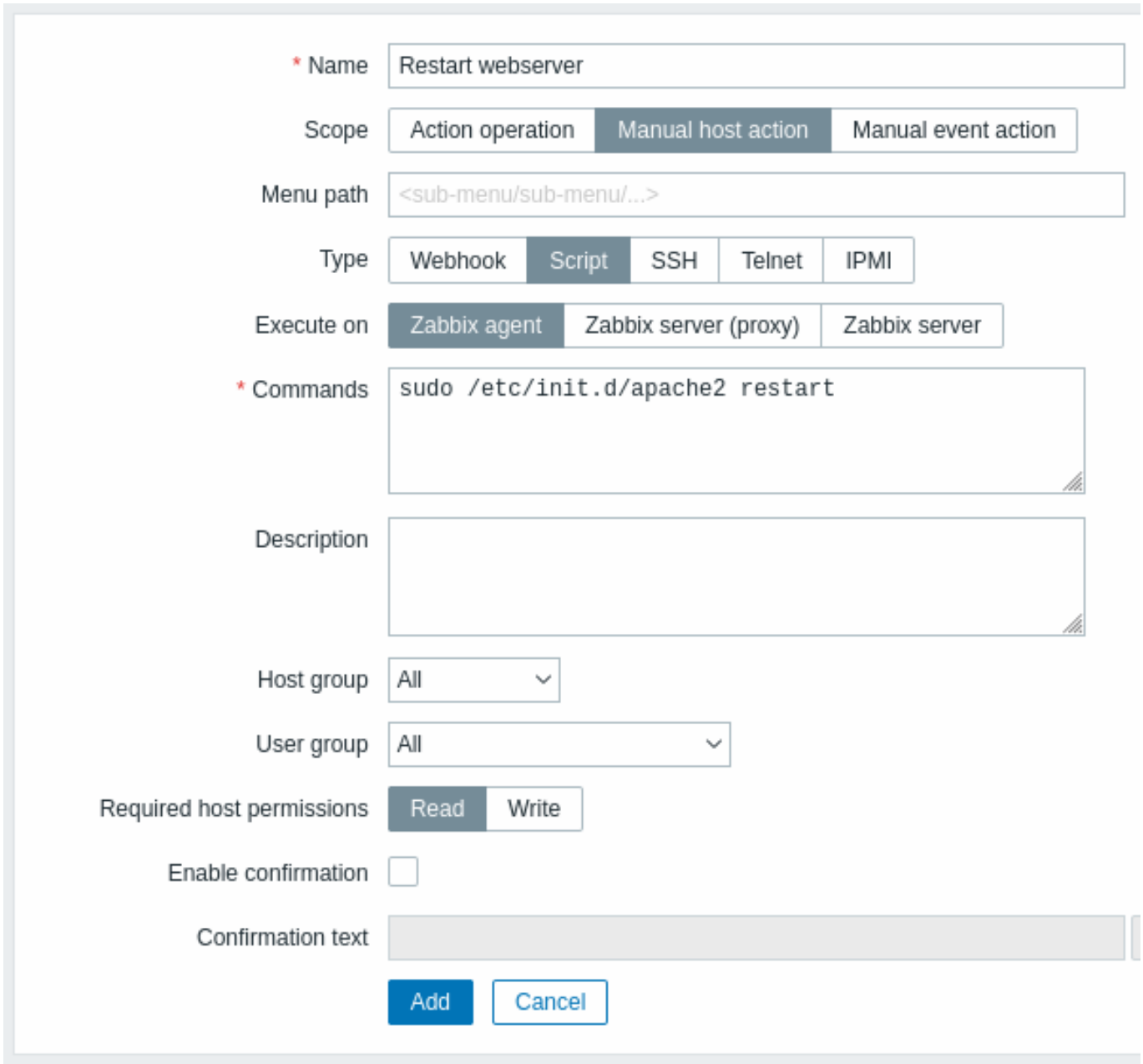
Using filter

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`

Parameter	Description
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 *Configuration* → *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.
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: **Webhook**, **Script**, **SSH**, **Telnet** or **IPMI** command.
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	

Parameter	Description
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	
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. {HOST.*} and {USER.*} macros are supported. 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*.

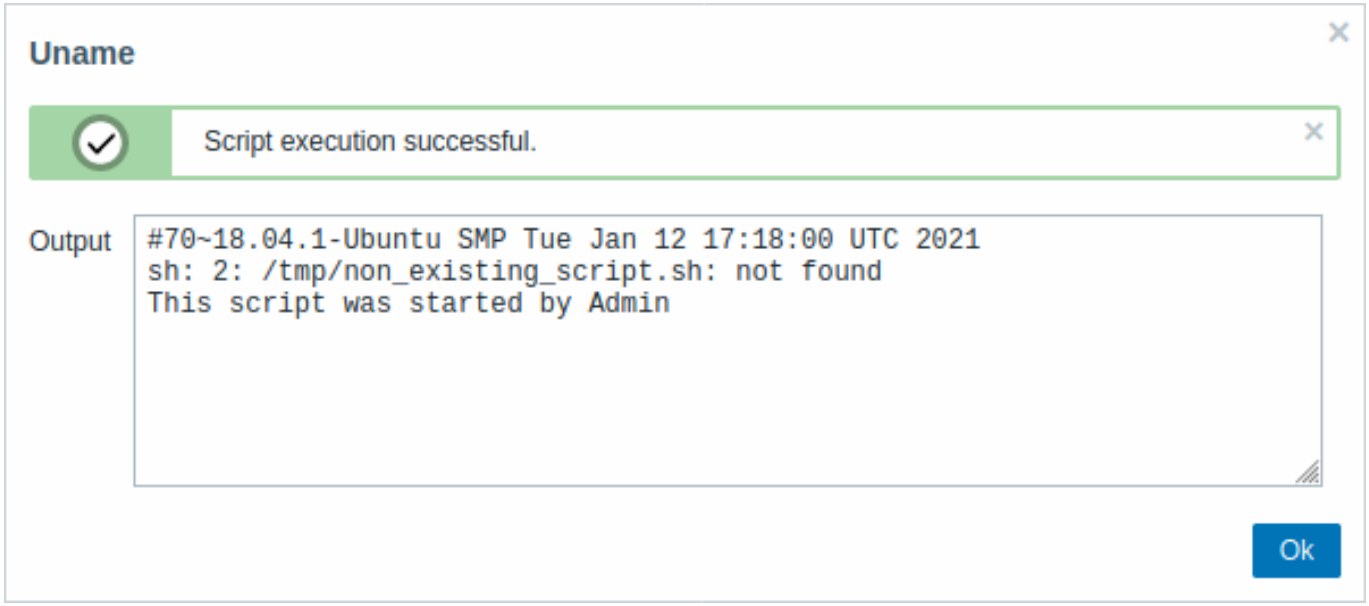
Script execution and result

Scripts run by Zabbix server are executed by the order described in Command execution section including exit code checking. The script result will be displayed in a pop-up window that will appear after the script is run.

Note: The return value of the script is standard output together with standard error.

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.

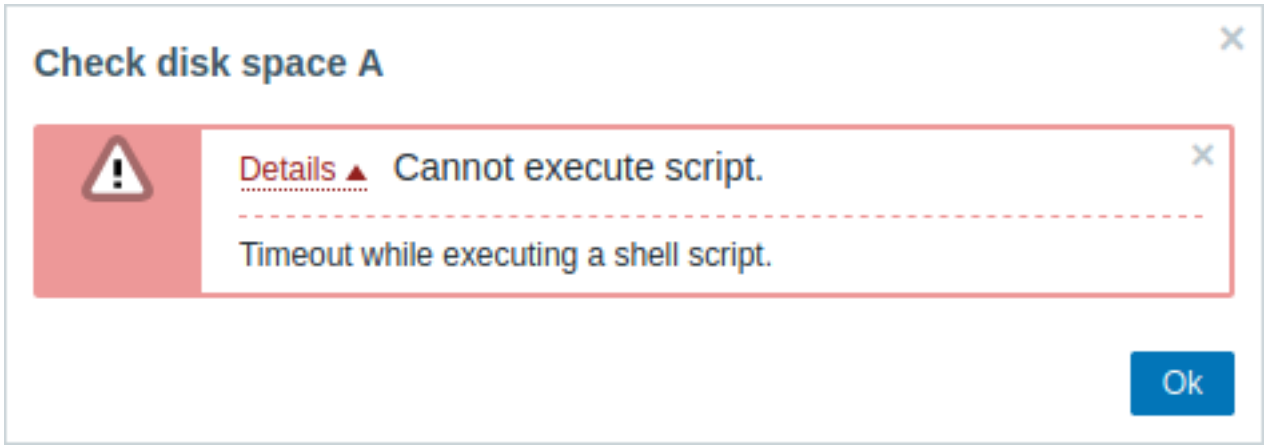
Script timeout

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 a situation, it is advised to optimize the script itself (instead of adjusting Timeout parameter to a corresponding value (in our case, > '5') by modifying the Zabbix agent configuration and Zabbix server configuration).

In case still the Timeout parameter is changed in Zabbix agent configuration following error message appears:

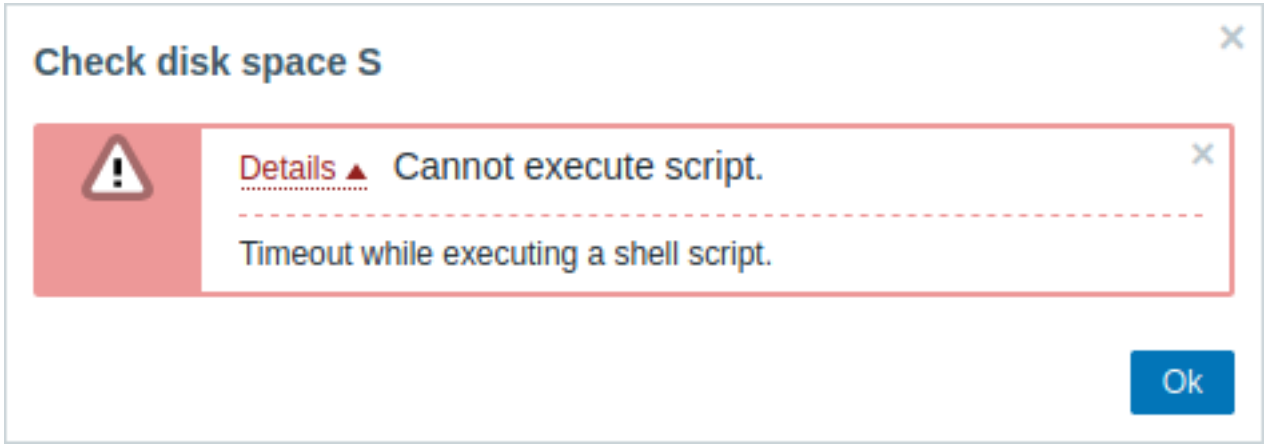
```
Get value from agent failed: ZBX_TCP_READ() timed out.
```

It means that modification was made in Zabbix agent configuration and it is required to modify Timeout setting also in Zabbix server configuration.

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).

9 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, connection problems or problems with agents, some items may get delayed and the information is displayed in this section. For more details, see the Queue section.

Note:
Queue is available only if Zabbix server is running.

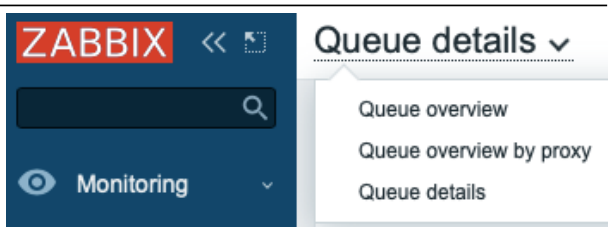
The *Administration* → *Queue* section contains the following pages:

- 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.

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

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.

Queue overview by proxy

Proxy	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Remote proxy	0	8	11	0	0	0
Server	0	0	0	0	0	0

Total: 2

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.

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

Displayed data:

Column	Description
Scheduled check	The time when the check was due is displayed.
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:

Details — Cannot display item queue.

Permission denied.

Error message in this case is the following:

```
Cannot display item queue. Permission denied
```

This happens when PHP configuration parameters $ZBX_SERVER_PORT or $ZBX_SERVER in zabbix.conf.php point to existing Zabbix server which uses different database.

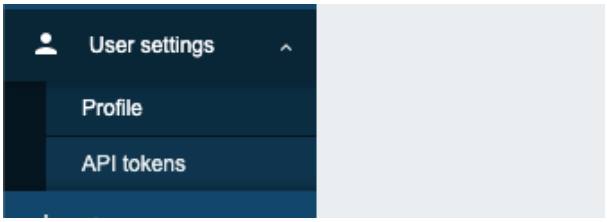
3 User settings

Overview

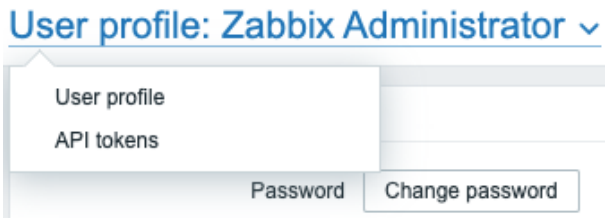
Depending on user role permissions, the *User settings* section may contain the following pages:

- *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.

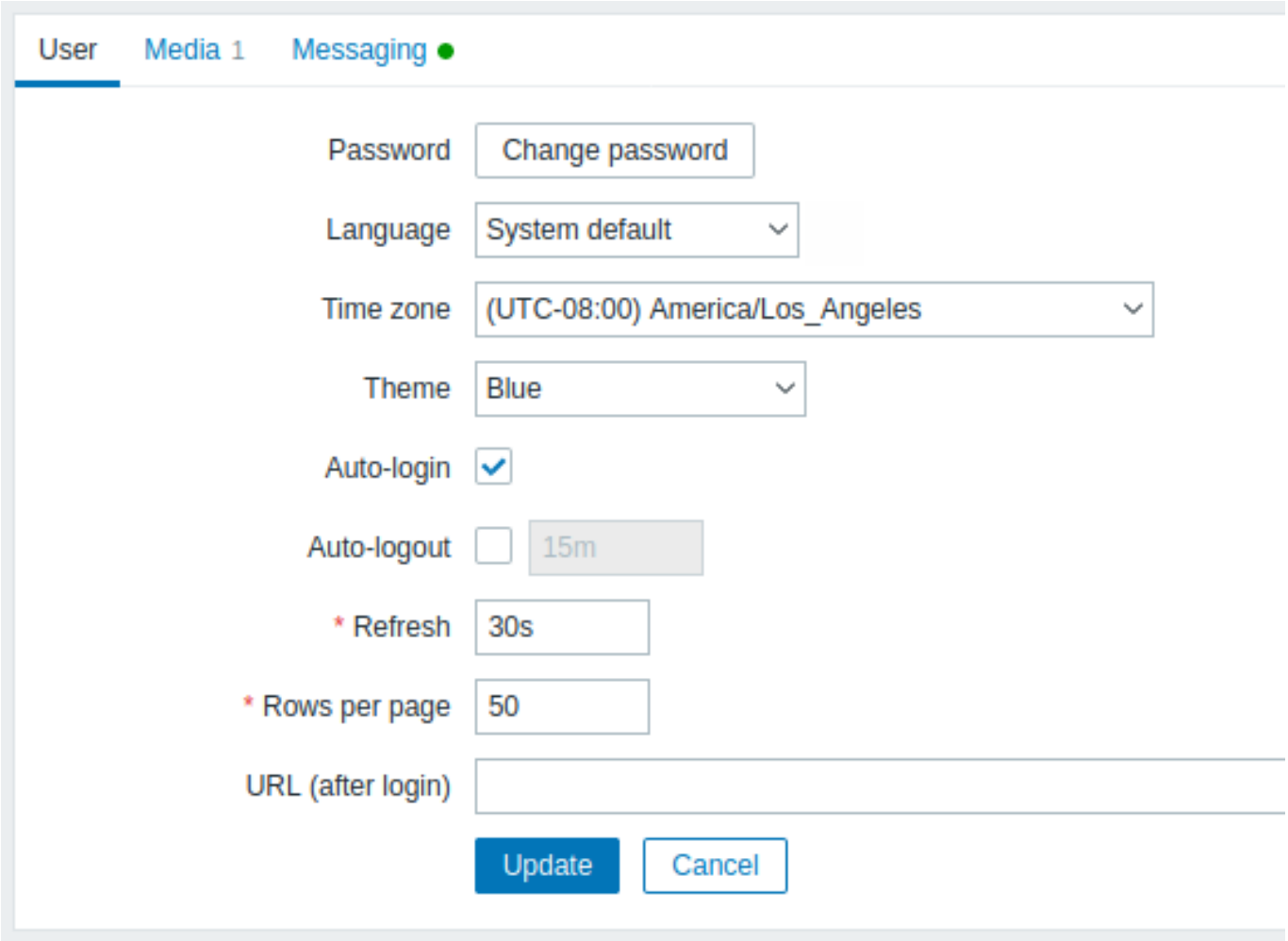


Title dropdown.

1 User profile

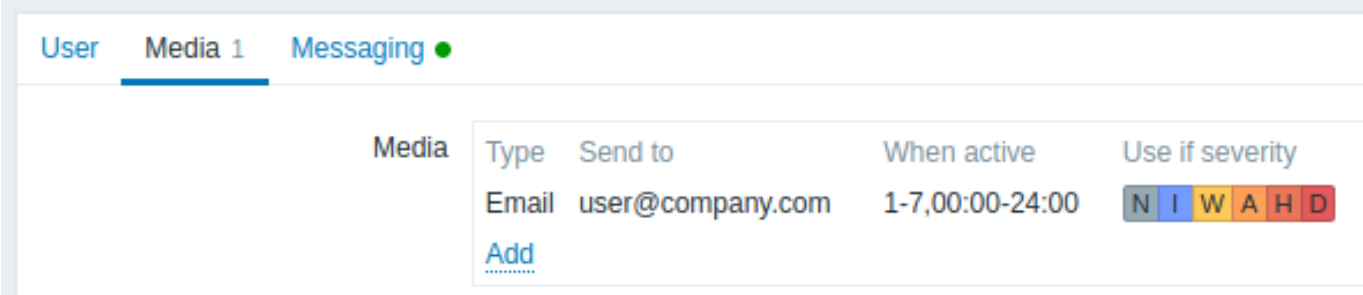
The **User profile** section provides options to set custom interface language, color theme, number of rows displayed in the lists, etc. The changes made here will be applied to the current user only.

The **User** tab allows you to set various user preferences.



Parameter	Description
Password	Click on the link to display two fields for entering a new password.
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, e.g. 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 accept 0, meaning that Auto-logout becomes disabled after profile settings update.
Refresh	You can set how often the information in the pages will be refreshed on the Monitoring menu, except for Dashboard, which uses its own refresh parameters for every widget. Time suffixes are supported, e.g. 30s, 5m, 2h, 1d.
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 *Monitoring* → *Dashboard* 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 the types, the addresses to use and when to use them to deliver notifications.

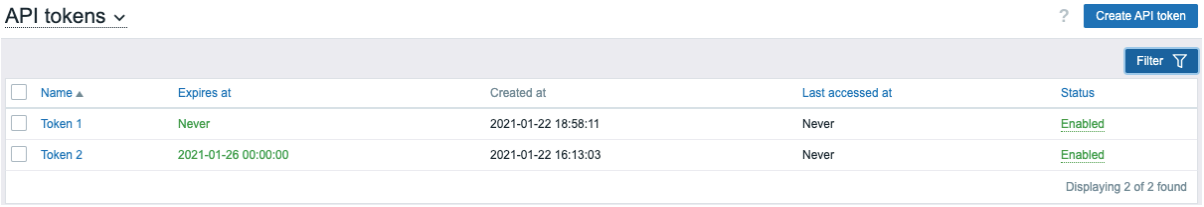


Note:
Only admin level users (Admin and Super admin) can change their own media details.

The **Messaging** tab allows you to set global notifications.

2 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.

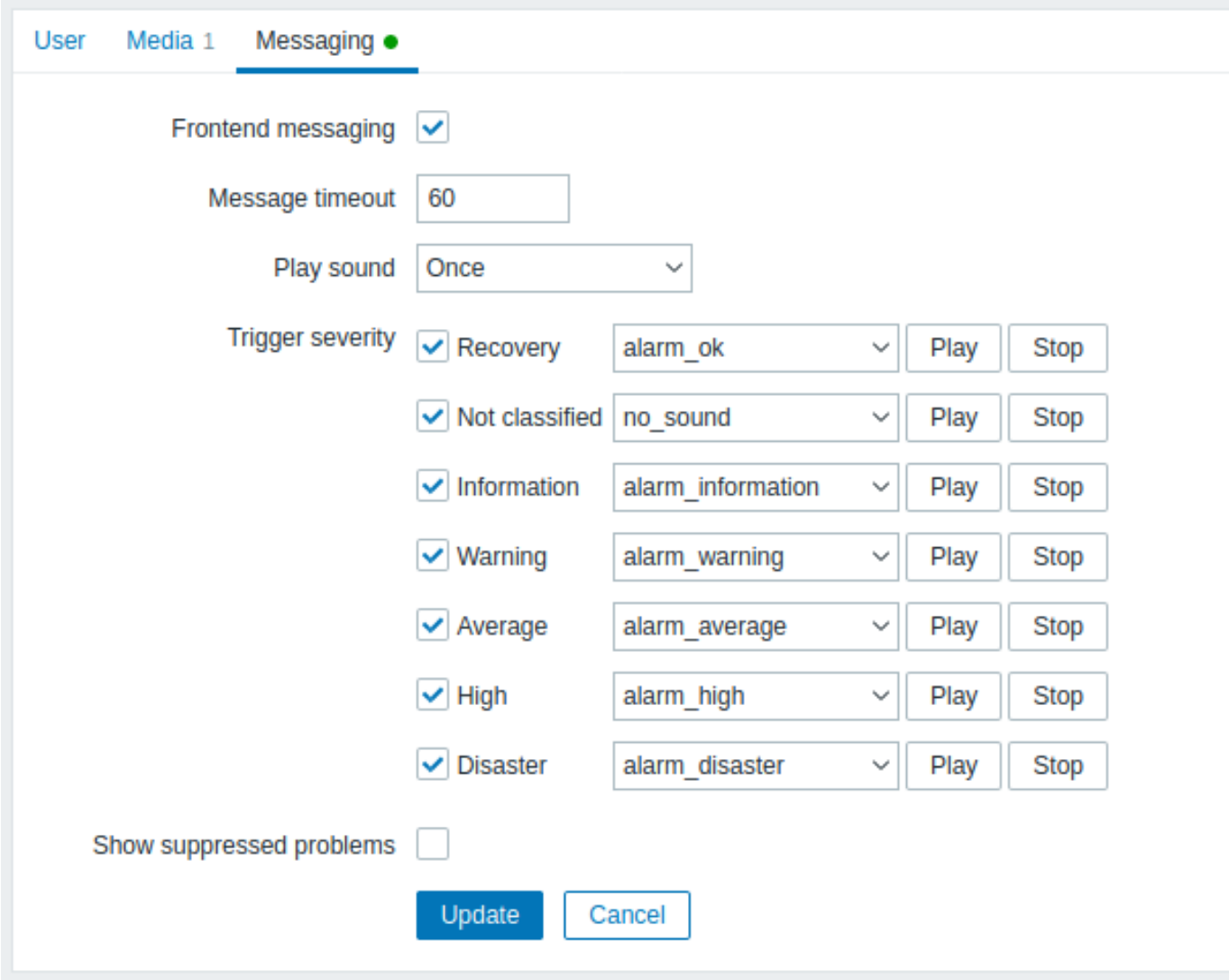
Without global notifications, working in some other location than *Problems* or the *Dashboard* would not show any information regarding issues that are currently happening. Global notifications will display this information regardless of where you are.

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

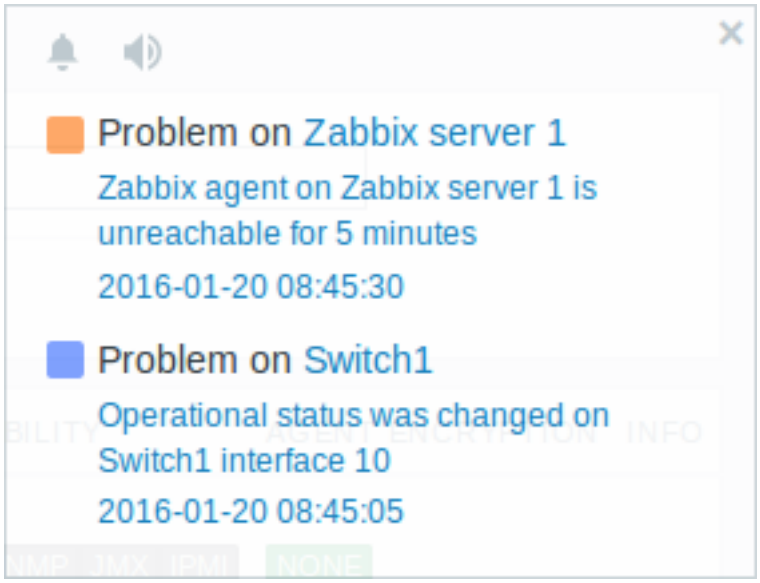
Global notifications can be enabled per user in the *Messaging* tab of profile configuration.



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 Chrome, Firefox, Microsoft Edge, Opera and Safari 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:

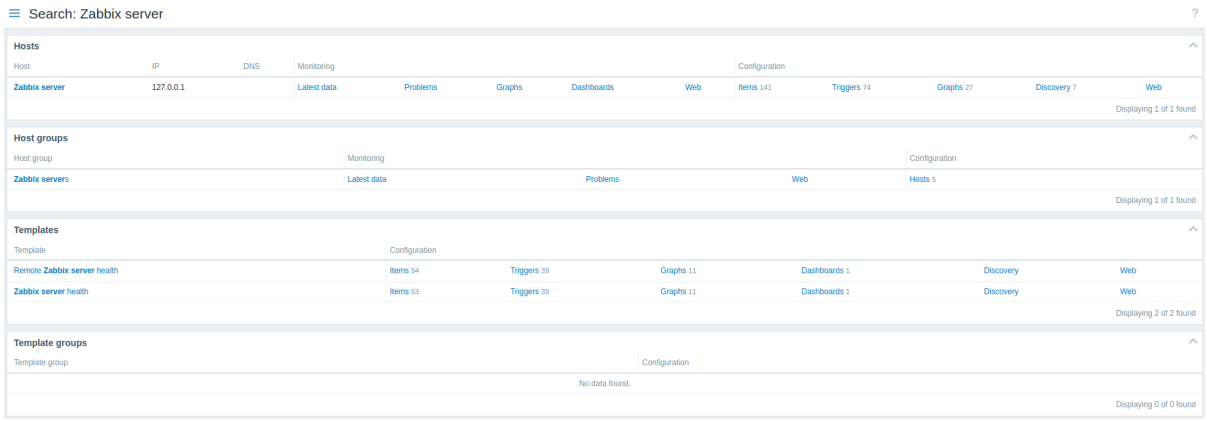
- Host name
- Visible name
- IP address
- DNS name

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:

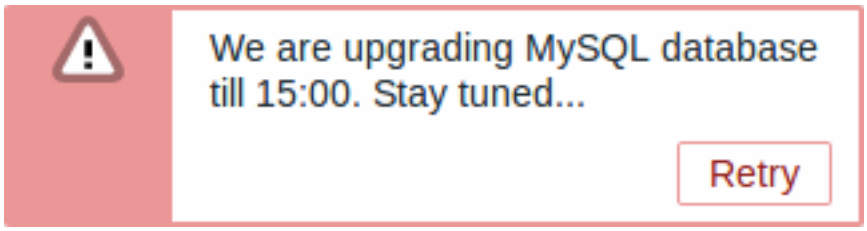
- SUSE and RedHat is `/etc/zabbix/web/maintenance.inc.php`.
- Debian-based systems is `/usr/share/zabbix/conf/`.

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	Message you can enter to inform users about the maintenance (optional).

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

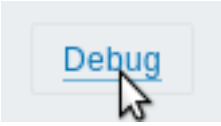
Debug mode may be used to diagnose performance problems with frontend pages.

Configuration

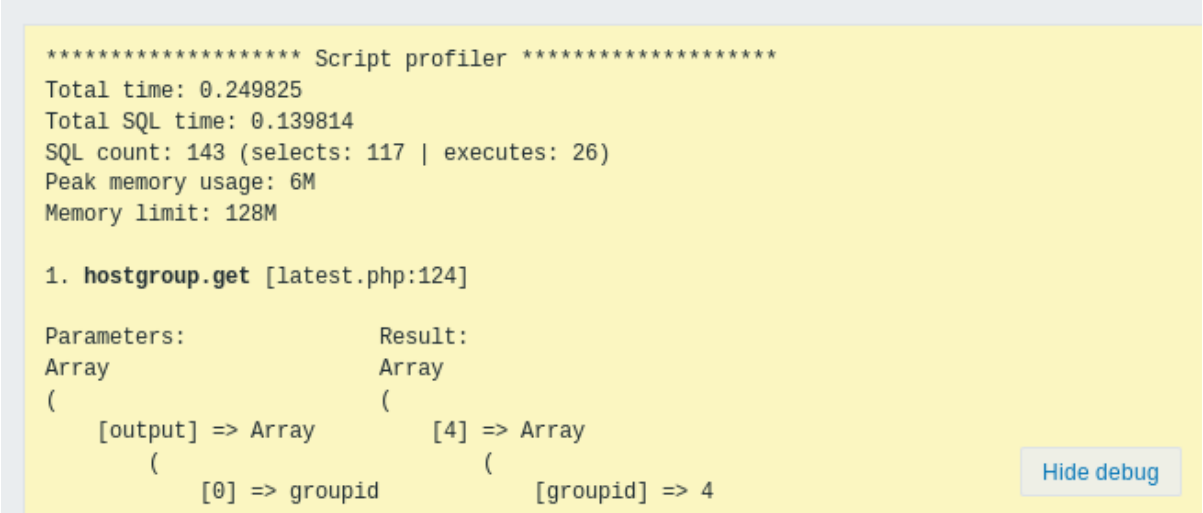
Debug mode can be activated for individual users who belong to a user group:

- 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[a]	Secure[a]
ZBX_SESSION_NAME	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)

[a] According to specification these are voltages on chip pins and generally speaking may need scaling.

[a] `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.

Name	Description	Values	Expires/Max-Age	HttpOnly[a]	Secure[a]
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	+	-

[a] According to specification these are voltages on chip pins and generally speaking may need scaling.

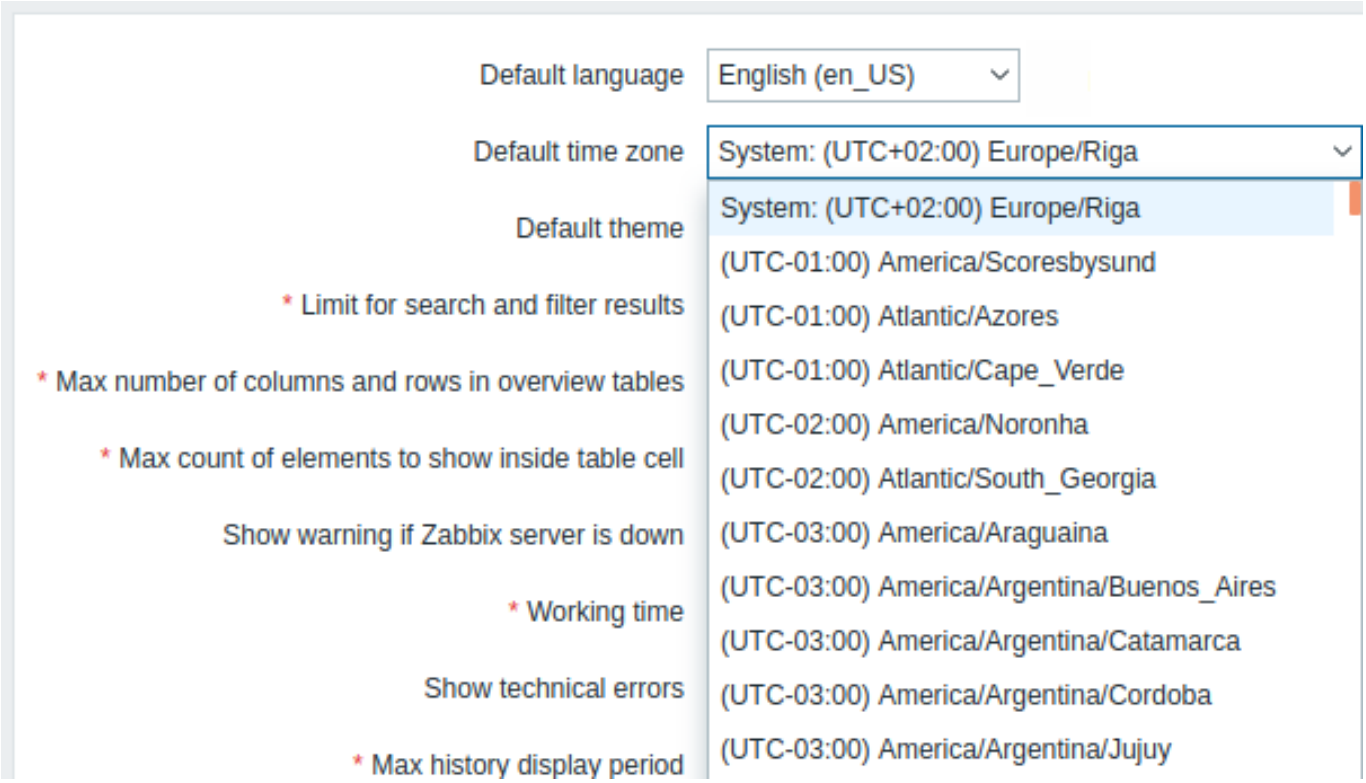
[a] `Secure` indicates that the cookie should only be transmitted over a secure HTTPS connection from the client. When set to 'true', the cookie will only be set if a secure connection exists.

Note:
Forcing 'HttpOnly' flag on Zabbix cookies by a webserver directive is not supported.

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 timezone 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 timezone:

- 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

19. API

Overview Zabbix API allows you to programmatically retrieve and modify the configuration of Zabbix and provides access to historical data. It is widely used to:

- Create new applications to work with Zabbix;
- Integrate Zabbix with third-party software;
- Automate routine tasks.

The Zabbix API is a web based API and is shipped as 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 info about the protocol and JSON can be found in the JSON-RPC 2.0 specification and the JSON format homepage.

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 of the APIs may provide a totally different set of methods.

Performing requests Once you've 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 *http://example.com/zabbix*, the HTTP request to call the `apiinfo.version` method may look like this:

```
POST http://example.com/zabbix/api_jsonrpc.php HTTP/1.1
Content-Type: application/json-rpc

{
    "jsonrpc": "2.0",
    "method": "apiinfo.version",
    "id": 1,
    "auth": null,
    "params": {}
}
```

The request must have the `Content-Type` header set to one of these values: `application/json-rpc`, `application/json` or `application/jsonrequest`.

Example workflow The following section will walk you through some usage examples in more 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:

```
{
    "jsonrpc": "2.0",
    "method": "user.login",
    "params": {
        "username": "Admin",
        "password": "zabbix"
    },
    "id": 1,
    "auth": null
}
```

Let's take a closer look at the request object. It has the following properties:

- `jsonrpc` - the version of the JSON-RPC protocol used by the API; the Zabbix API implements JSON-RPC version 2.0;
- `method` - the API method being called;
- `params` - parameters that will be passed to the API method;
- `id` - an arbitrary identifier of the request;
- `auth` - a user authentication token; since we don't have one yet, it's set to `null`.

If you provided the credentials correctly, the response returned by the API will contain the user authentication token:

```
{
    "jsonrpc": "2.0",
    "result": "0424bd59b807674191e7d77572075f33",
    "id": 1
}
```

The response object in turn contains the following properties:

- `jsonrpc` - again, the version of the JSON-RPC protocol;
- `result` - the data returned by the method;

- `id` - identifier of the corresponding request.

Retrieving hosts We now have a valid user authentication token that can be used to access the data in Zabbix. For example, let's use the host.get method to retrieve the IDs, host names and interfaces of all configured hosts:

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": [
            "hostid",
            "host"
        ],
        "selectInterfaces": [
            "interfaceid",
            "ip"
        ]
    },
    "id": 2,
    "auth": "0424bd59b807674191e7d77572075f33"
}
```

Attention:
Note that the `auth` property is now set to the authentication token we've obtained by calling `user.login`.

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 we recommend to always list the object properties you want to retrieve and avoid retrieving everything.

Creating a new item Let's create a new item on "Zabbix server" using the data we've obtained from the previous `host.get` request. This can be done by using the item.create method:

```
{
    "jsonrpc": "2.0",
    "method": "item.create",
    "params": {
        "name": "Free disk space on /home/joe/",
        "key_": "vfs.fs.size[/home/joe/,free]",
        "hostid": "10084",
        "type": 0,
        "value_type": 3,
        "interfaceid": "1",
        "delay": 30
    },
```

```
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 3
}
```

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 So if create methods accept arrays, we can add multiple triggers like so:

```
{
    "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",
        },
        {
            "description": "Too many processes on {HOST.NAME}",
            "expression": "avg(/Linux server/proc.num[],5m)>300",
        }
    ],
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 4
}
```

A 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, that is, set its status to "0":

```
{
    "jsonrpc": "2.0",
    "method": "item.update",
    "params": {
        "itemid": "10092",
        "status": 0
    },
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 5
```

```
}
```

A 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, that is, set their status to 0:

```
{
    "jsonrpc": "2.0",
    "method": "trigger.update",
    "params": [
        {
            "triggerid": "13938",
            "status": 0
        },
        {
            "triggerid": "13939",
            "status": 0
        }
    ],
    "auth": "0424bd59b807674191e7d77572075f33",
    "id": 6
}
```

A 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 like `host.massupdate` allow to write more simple code, but it's not recommended to use those methods, since they will be removed in the future releases.

Error handling Up to that point everything we've tried has worked fine. But what happens if we try to make an incorrect call to the API? Let's try to create another host by calling host.create but omitting the mandatory `groups` parameter.

```
{
    "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"
            }
        ]
    },
    "id": 7,
    "auth": "0424bd59b807674191e7d77572075f33"
}
```

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 occurred, instead of the `result` property, the response object will contain an `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 different cases, such as, using incorrect input values, a session timeout or trying to access unexisting 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're working with. This can be useful for adjusting your application to use version-specific features.

We guarantee feature backward compatibility inside of a major version. When making backward incompatible changes between major releases, we usually leave the old features as deprecated in the next release, and only remove them in the release after that. Occasionally, we 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 of the changes made to the API in the API changelog.

Further reading You now know enough to start working with the Zabbix API, but don't stop here. For further reading we suggest you 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.

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

Service monitoring

Create a hierarchy representation of monitored IT infrastructure/business services data.

Service API

Service Level Agreement

Define Service Level Objectives (SLO), retrieve detailed Service Level Indicators (SLI) information about service performance.

SLA API

Tasks

Interact with Zabbix server task manager, creating tasks and retrieving response.

Task API

Configuration 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

Manage templates and link them to hosts or other templates.

Template API | Value map API

Export and import

Export and import Zabbix configuration data for configuration backups, migration or large-scale configuration updates.

Configuration 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

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

Services

Manage services for service-level monitoring and retrieve detailed SLA information about any service.

Service API

Dashboards

Manage dashboards and make scheduled reports based on them.

Dashboard API | Template dashboard API | Report API

Maps

Configure maps to create detailed dynamic representations of your IT infrastructure.

Map API

Web monitoring

Configure web scenarios to monitor your web applications and services.

Web scenario 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

Administration With the Zabbix API you can change administration settings of your monitoring system.

Users

Add users that will have access to Zabbix, assign them to user groups and grant permissions. Make roles for granular management of user rights. Track configuration changes each user has done. Configure media types and multiple ways users will receive alerts.

User API | User group API | User role API | Media type API | Audit log API

General

Change certain global configuration options.

Autoregistration API | Icon map API | Image API | User macro API | Settings API | Housekeeping API

Regular expressions

Manage global regular expressions.

Regular expression API

Proxies

Manage the proxies used in your distributed monitoring setup.

Proxy API

Authentication

Change authentication configuration options.

Authentication API

API Tokens

Manage authorization tokens.

Token API

Scripts

Configure and execute scripts to help you with your daily tasks.

Script 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	*(readonly)* ID of the action.
esc_period (required)	string	Default operation step duration. Must be at least 60 seconds. Accepts seconds, time unit with suffix and user macro. Note that escalations are supported only for trigger, internal and service actions, and only in normal operations.
eventsource (required)	integer	*(constant)* Type of events that the action will handle. Refer to the event "source" property for a list of supported event types.
name (required)	string	Name of the action.
status	integer	Whether the action is enabled or disabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
pause_suppressed	integer	Whether to pause escalation during maintenance periods or not. Possible values: 0 - Don't pause escalation; 1 - *(default)* Pause escalation. Note that this parameter is valid for trigger actions only.

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. Note that this parameter is valid for trigger actions only.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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
operationid	string	*(readonly)* ID of the action operation.
operationtype (required)	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. Note that only types '0' and '1' are supported for trigger and service actions, only '0' is supported for internal actions. All types are supported for discovery and autoregistration actions.
actionid	string	*(readonly)* ID of the action that the operation belongs to.
esc_period	string	Duration of an escalation step in seconds. Must be greater than 60 seconds. Accepts seconds, time unit with suffix and user macro. If set to 0 or 0s, the default action escalation period will be used. Default: 0s. Note that escalations are supported only for trigger, internal and service actions, and only in normal operations.
esc_step_from	integer	Step to start escalation from. Default: 1. Note that escalations are supported only for trigger, internal and service actions, and only in normal operations.
esc_step_to	integer	Step to end escalation at. Default: 1. Note that escalations are supported only for trigger, internal and service actions, and only in normal operations.
evaltype	integer	Operation condition evaluation method. Possible values: 0 - *(default)* AND / OR; 1 - AND; 2 - OR.

Property	Type	Description
opcommand	object	Object containing data on global script run by the operation. Each object has one following property: `scriptid` - *(string)* ID of the script. Required for global script operations.
opcommand_grp	array	Host groups to run global scripts on. Each object has the following properties: `opcommand_grpid` - *(string, readonly)* ID of the object; `operationid` - *(string, readonly)* ID of the operation; `groupid` - *(string)* ID of the host group. Required for global script operations if `opcommand_hst` is not set.
opcommand_hst	array	Host to run global scripts on. Each object has the following properties: `opcommand_hstid` - *(string, readonly)* ID of the object; `operationid` - *(string, readonly)* ID of the operation; `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host. Required for global script operations if `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. Each object has the following properties: `operationid` - *(string, readonly)* ID of the operation; `groupid` - *(string)* ID of the host group. Required for ”add to host group” and ”remove from host group” operations.
opmessage	object	Object containing the data about the message sent by the operation. The operation message object is described in detail below. Required for message operations.
opmessage_grp	array	User groups to send messages to. Each object has the following properties: `operationid` - *(string, readonly)* ID of the operation; `usrgrpid` - *(string)* ID of the user group. Required for message operations if `opmessage_usr` is not set.
opmessage_usr	array	Users to send messages to. Each object has the following properties: `operationid` - *(string, readonly)* ID of the operation; `userid` - *(string)* ID of the user. Required for message operations if `opmessage_grp` is not set.
optemplate	array	Templates to link the hosts to. Each object has the following properties: `operationid` - *(string, readonly)* ID of the operation; `templateid` - *(string)* ID of the template. Required for ”link to template” and ”unlink from template” operations.

Property	Type	Description
opinventory	object	Inventory mode set host to. Object has the following properties: `operationid` - *(string, readonly)* ID of the operation; `inventory_mode` - *(string)* Inventory mode. Required for ”Set host inventory mode” operations.

Action operation message

The operation message object contains data about the message that will be sent by the operation.

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.
message	string	Operation message text.
subject	string	Operation message subject.

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
opconditionid	string	*(readonly)* ID of the action operation condition
conditiontype (required)	integer	Type of condition. Possible values: 14 - event acknowledged.
value (required)	string	Value to compare with.
operationid	string	*(readonly)* ID of the operation.
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 for trigger, internal and service actions. It has the following properties.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values for trigger and service actions: 0 - send message; 1 - global script; 11 - notify all involved. Possible values for internal actions: 0 - send message; 11 - notify all involved.
actionid	string	*(readonly)* ID of the action that the recovery operation belongs to.
opcommand	object	Object containnig data on global action type script run by the operation. Each object has one following property: `scriptid` - *(string)* ID of the action type script.
opcommand_grp	array	Required for global script operations. Host groups to run global scripts on. Each object has the following properties: `opcommand_grpid` - *(string, readonly)* ID of the object; `operationid` - *(string, readonly)* ID of the operation; `groupid` - *(string)* ID of the host group.
opcommand_hst	array	Required for global script operations if `opcommand_hst` is not set. Host to run global scripts on. Each object has the following properties: `opcommand_hstid` - *(string, readonly)* ID of the object; `operationid` - *(string, readonly)* ID of the operation; `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host.
opmessage	object	Required for global script operations if `opcommand_grp` is not set. Object containing the data about the message sent by the recovery operation. The operation message object is described in detail above.
opmessage_grp	array	Required for message operations. User groups to send messages to. Each object has the following properties: `operationid` - *(string, readonly)* ID of the operation; `usrgrpid` - *(string)* ID of the user group.
opmessage_usr	array	Required for message operations if `opmessage_usr` is not set. Users to send messages to. Each object has the following properties: `operationid` - *(string, readonly)* ID of the operation; `userid` - *(string)* ID of the user. Required for message operations if `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 for trigger and service actions. It has the

following properties.

Property	Type	Description
operationid	string	*(readonly)* ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values for trigger and service actions: 0 - send message; 1 - global script; 12 - notify all involved.
opcommand	object	Object containing data on global action type script run by the operation. Each object has one following property: `scriptid` - *(string)* ID of the action type script. Required for global script operations.
opcommand_grp	array	Host groups to run global scripts on. Each object has the following properties: `groupid` - *(string)* ID of the host group. Required for global script operations if `opcommand_hst` is not set.
opcommand_hst	array	Host to run global scripts on. Each object has the following properties: `hostid` - *(string)* ID of the host; if set to 0 the command will be run on the current host. Required for global script operations if `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. Each object has the following properties: `usrgrpid` - *(string)* ID of the user group. Required only for `send message` operations if `opmessage_usr` is not set. Is ignored for `send update message` operations.
opmessage_usr	array	Users to send messages to. Each object has the following properties: `userid` - *(string)* ID of the user. Required only for `send message` operations if `opmessage_grp` is not set. Is ignored for `send update message` operations.

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 (required)	array	Set of filter conditions to use for filtering results.

Property	Type	Description
evaltype (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
eval_formula	string	*(readonly)* 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.
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. Required for custom expression filters.

Action filter condition

The action filter condition object defines a specific condition that must be checked before running the action operations.

Property	Type	Description
conditionid	string	*(readonly)* ID of the action condition.

Property	Type	Description
conditiontype (required)	integer	Type of condition. Possible values for trigger actions: 0 - host group; 1 - host; 2 - trigger; 3 - trigger name; 4 - trigger severity; 6 - time period; 13 - host template; 16 - problem is suppressed; 25 - event tag; 26 - event tag value. Possible values for discovery actions: 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 for autoregistration actions: 20 - proxy; 22 - host name; 24 - host metadata. Possible values for internal actions: 0 - host group; 1 - host; 13 - host template; 23 - event type; 25 - event tag; 26 - event tag value. Possible values for service actions: 25 - event tag; 26 - event tag value; 27 - service; 28 - service name.
value (required)	string	Value to compare with.
value2	string	Secondary value to compare with. Required for trigger, internal and service actions when condition type is *26*.
actionid	string	*(readonly)* ID of the action that the condition belongs to.
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.

Property	Type	Description
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	Trigger name	contains, does not contain	Trigger 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.
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.

Condition	Condition name	Supported operators	Expected value
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.
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"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
                    }
                ]
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
                    }
                ]
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: `actionid`, `name` and `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 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
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.
update_operations	array	Action update operations to replace existing update operations.

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 the 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.

Property	Type	Description
mediatypeid	string	ID of the media type that was used to send the message.
message	text	Message text. Used for message alerts.
retries	integer	Number of times Zabbix tried to send the message.
sendto	string	Address, user name or other identifier of the recipient. Used for message alerts.
status	integer	Status indicating whether the action operation has been executed successfully. Possible values for message alerts: 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 for command alerts: 0 - command not run. 1 - command run. 2 - tried to run the command on the Zabbix agent but it was unavailable.
subject	string	Message subject. Used for message alerts.
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.

Parameter	Type	Description
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 are: `alertid`, `clock`, `eventid`, `mediatypeid`, `sendto` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": "4.0.0",
    "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	*(readonly)* 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 are: 0 - Add; 1 - Update; 2 - Delete; 4 - Logout; 7 - Execute; 8 - Login; 9 - Failed login; 10 - History clear; 11 - Config refresh.

Property	Type	Description
resourcetype	integer	Audit log entry resource type. Possible values are: 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 - Auth token; 46 - Scheduled report; 47 - High availability node; 48 - SLA; 49 - LDAP user directory. 50 - Template group.
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 JSON object where each property name is a path to property or nested object in which change occurred, and each value contain the data about the change of this property in array format. Possible value formats are: [”add”] - Nested object has been added; [”add”, ”<value>”] - The property of added object contain <value>; [”update”] - Nested object has been updated; [”update”, ”<new value>”, ”<old value>”] - The value of property of 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 are: `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. Additionally supports filtering by details property fields: `table_name`, `field_name`.
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	
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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	Enable HTTP authentication. Possible values: 0 - *(default)* Disable; 1 - Enable.
http_login_form	integer	Default login form. Possible values: 0 - *(default)* Zabbix login form; 1 - HTTP login form.
http_strip_domains	string	Remove domain name.
http_case_sensitive	integer	HTTP case sensitive login. Possible values: 0 - Off; 1 - *(default)* On.
ldap_configured	integer	Enable LDAP authentication. Possible values: 0 - *(default)* Disable; 1 - Enable.
ldap_case_sensitive	integer	LDAP case sensitive login. Possible values: 0 - Off; 1 - *(default)* On.
ldap_userdirectoryid	string	LDAP authentication default user directory for user groups with `gui_access` set to LDAP or System default. Required to be set when `ldap_configured` is set to 1.
saml_auth_enabled	integer	Enable SAML authentication. Possible values: 0 - *(default)* Disable; 1 - Enable.
saml_idp_entityid	string	SAML IdP entity ID.
saml_sso_url	string	SAML SSO service URL.
saml_slo_url	string	SAML SLO service URL.
saml_username_attribute	string	SAML username attribute.
saml_sp_entityid	string	SAML SP entity ID.
saml_nameid_format	string	SAML SP name ID format.
saml_sign_messages	integer	SAML sign messages. Possible values: 0 - *(default)* Do not sign messages; 1 - Sign messages.

Property	Type	Description
saml_sign_assertions	integer	SAML sign assertions. Possible values: 0 - *(default)* Do not sign assertations; 1 - Sign assertations.
saml_sign_authn_requests	integer	SAML sign AuthN requests. Possible values: 0 - *(default)* Do not sign AuthN requests; 1 - Sign AuthN requests.
saml_sign_logout_requests	integer	SAML sign logout requests. Possible values: 0 - *(default)* Do not sign logout requests; 1 - Sign logout requests.
saml_sign_logout_responses	integer	SAML sign logout responses. Possible values: 0 - *(default)* Do not sign logout responses; 1 - Sign logout responses.
saml_encrypt_nameid	integer	SAML encrypt name ID. Possible values: 0 - *(default)* Do not encrypt name ID; 1 - Encrypt name ID.
saml_encrypt_assertions	integer	SAML encrypt assertions. Possible values: 0 - *(default)* Do not encrypt assertions; 1 - Encrypt assertions.
saml_case_sensitive	integer	SAML case sensitive login. Possible values: 0 - Off; 1 - *(default)* On.
passwd_min_length	integer	Password minimal length requirement. Possible range of values: 1-70 8 - *default*
passwd_check_rules	integer	Password checking rules. Possible bitmap values are: 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, does not contain derivations of word ”Zabbix” or user’s name, last name or username.

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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_configured": "0",
        "ldap_case_sensitive": "1",
        "ldap_userdirectoryid": "0",
        "saml_auth_enabled": "0",
        "saml_idp_entityid": "",
        "saml_sso_url": "",
        "saml_slo_url": "",
        "saml_username_attribute": "",
        "saml_sp_entityid": "",
        "saml_nameid_format": "",
        "saml_sign_messages": "0",
        "saml_sign_assertions": "0",
        "saml_sign_authn_requests": "0",
        "saml_sign_logout_requests": "0",
        "saml_sign_logout_responses": "0",
        "saml_encrypt_nameid": "0",
        "saml_encrypt_assertions": "0",
        "saml_case_sensitive": "0",
        "passwd_min_length": "8",
        "passwd_check_rules": "8"
    },
    "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 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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(write-only)* PSK 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.
tls_psk	string	*(write-only)* PSK value string (an even number of hexadecimal characters).

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	string	Format in which the data must be exported. Possible values: `yaml` - YAML; `xml` - XML; `json` - JSON; `raw` - unprocessed PHP array.
prettyprint	boolean	Make the output more human readable by adding indentation. Possible values: `true` - add indentation; `false` - *(default)* do not add indentation.
options (required)	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.

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"
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n<zabbix_export><version>6.2</version><date>2023
    "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 (required)	string	Format of the serialized string. Possible values: `yaml` - YAML; `xml` - XML; `json` - JSON.
source (required)	string	Serialized string containing the configuration data.
rules (required)	object	Rules on how new and existing objects should be imported. The `rules` parameter is described in detail in the table below.

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`.

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`.
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`.
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`.
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`.
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>6.2</version><date>
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	string	Format of the serialized string. Possible values: `yaml` - YAML; `xml` - XML; `json` - JSON.
source (required)	string	Serialized string containing the configuration data.
rules (required)	object	Rules on how new and existing objects should be imported. The `rules` parameter is described in detail in the table below.

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`.
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`.

Parameter	Type	Description
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`.
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`.

Parameter	Type	Description
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`.
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>6.2</version><date>
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
```

```
    "result": {
        "templates": {
            "updated": [
                {
                    "before": {
                        "uuid": "5aef0444a82a4d8cb7a95dc4c0c85330",
                        "template": "New template",
                        "name": "New template"
                    },
                    "after": {
                        "uuid": "5aef0444a82a4d8cb7a95dc4c0c85330",
                        "template": "New template",
                        "name": "New template"
                    },
                    "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*.

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	*(readonly)* ID of the correlation.
name (required)	string	Name of the correlation.
description	string	Description of the correlation.
status	integer	Whether the correlation is enabled or disabled. Possible values are: 0 - *(default)* enabled; 1 - disabled.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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 (required)	integer	Type of operation. Possible values: 0 - close old events; 1 - close new event.

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 (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
conditions (required)	array	Set of filter conditions to use for filtering results.
eval_formula	string	*(readonly)* 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.
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. Required for custom expression filters.

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 (required)	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.
tag	string	Event tag (old or new). Required when type of condition is: 0, 1, 4, 5.
groupid	string	Host group ID. Required when type of condition is: 2.
oldtag	string	Old event tag. Required when type of condition is: 3.
newtag	string	New event tag. Required when type of condition is: 3.
value	string	Event tag (old or new) value. Required when type of condition is: 4, 5.
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.

Property	Type	Description
operator	integer	Condition operator. Required when type of condition is: 2, 4, 5.

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 (required)	array	Correlation operations to create for the correlation.
filter (required)	object	Correlation filter object for the correlation.

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
            }
        ]
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "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
            }
        ]
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
```

```
    "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"
    ],
    "auth": "343baad4f88b4106b9b5961e77437688",
    "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 are: `correlationid`, `name` and `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 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"
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "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"
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "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"
                }
            ]
        }
    },
    "auth": "343baad4f88b4106b9b5961e77437688",
    "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	*(readonly)* ID of the dashboard.
name (required)	string	Name of the dashboard.
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.

Property	Type	Description
auto_start	integer	Auto start slideshow. Possible values: 0 - do not auto start slideshow; 1 - *(default)* auto start slideshow.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Dashboard page

The dashboard page object has the following properties.

Property	Type	Description
dashboard_pageid	string	*(readonly)* ID of the dashboard page.
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	*(readonly)* ID of the dashboard widget.
type (required)	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; 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.
name	string	Custom widget name.
x	integer	A horizontal position from the left side of the dashboard. Valid values range from 0 to 23.

Property	Type	Description
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.

Dashboard widget field

The dashboard widget field object has the following properties.

Property	Type	Description
type (required)	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.
name (required)	string	Widget field name. Possible values: see Dashboard widget fields.
value (required)	mixed	Widget field value depending on the type. Possible values: see Dashboard widget fields.

Dashboard user group

List of dashboard permissions based on user groups. It has the following properties.

Property	Type	Description
usrgrpid (required)	string	User group ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

Dashboard user

List of dashboard permissions based on users. It has the following properties.

Property	Type	Description
userid (required)	string	User ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

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

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.
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).

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
                            }
                        ]
                    }
                ]
            }
```

```
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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 (required)	4	itemid	Item ID.

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*.
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”.

Parameter	type	name	value
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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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.

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.

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 *Tag name* 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 *Operator* required if configuring *Tags*.
Tag value	1	tags.value.0	Any string value. Parameter *Tag value* 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
            }
        ]
    },
```

```
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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.
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 *Tag name* 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 *Operator* required if configuring *Tags*.
Tag value	1	tags.value.0	Any string value. Parameter *Tag value* 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", or 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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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 Parameters if *Data set type* is set to "Item list"	0	ds.dataset_type.0	0 - Item list; 1 - *(default)* Item pattern.
Items (required)	4	ds.itemids.0.0	Item ID. Note: To configure multiple items, create a dashboard widget field object for each item.
Color (required)	1	ds.color.0.0	Hexadecimal color code (e.g. FF0000).
Parameters if *Data set type* is set to "Item pattern"			
Host pattern (required)	1	ds.hosts.0.0	Host name or pattern (e.g. Zabbix*).
Item pattern (required)	1	ds.items.0.0	Item name or pattern (e.g. *: Number of processed *values per second).
Color	1	ds.color.0	Hexadecimal color code (e.g. FF0000). Default: FF465C.

Parameter	type	name	value
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.
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.

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: 1, 2, 3, 4, 5 (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 *Tag name* 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 *Operator* required if configuring *Tags*.
Tag value	1	tags.value.0	Any string value. Parameter *Tag value* 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 (required)	1	or.hosts.0.0	Host name or pattern (e.g. `Zabbix*`).
Item pattern (required)	1	or.items.0.0	Item name or pattern (e.g. `*: Number of processed *values per second`).
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.
- 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": 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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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 *Graph* required if *Source* is set to "Graph".

Parameter	type	name	value
Item	4	itemid	Item ID. Parameter *Item* required if *Source* is set to ”Simple graph”.
Show legend	0	show_legend	0 - Disabled; 1 - *(default)* Enabled.
Dynamic item	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
```

```
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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 *Graph prototype* required if *Source* is set to ”Graph prototype”.
Item prototype	5	itemid	Item prototype ID. Parameter *Item prototype* required if *Source* is set to ”Simple graph prototype”.
Show legend	0	show_legend	0 - Disabled; 1 - *(default)* Enabled.
Dynamic item	0	dynamic	0 - *(default)* Disabled; 1 - Enabled.
Columns	0	columns	Valid values range from 1-24. Default: 2.

Parameter	type	name	value
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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

Parameter	type	name	value
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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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 (required)	4	itemid	Item ID.
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.
Dynamic item	0	dynamic	0 - *(default)* Disabled; 1 - Enabled.

Parameter	type	name	value
Advanced configuration	0	adv_conf	0 - *(default)* Disabled; 1 - Enabled.

Advanced configuration

The following parameters are supported if *Advanced configuration* is set to ”Enabled”.

Parameter	type	name	value
Background color	1	bg_color	Hexadecimal color code (e.g. FF0000). Default: "" (empty).

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			

Parameter	type	name	value
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			
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.

Parameter	type	name	value
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.

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": "bg_color",
            "value": "D1C4E9"
        },
        {
            "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"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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 *Map* required if *Source type* is set to ”Map”.
Linked widget reference	1	filter_widget_reference	Valid *Map navigation tree* widget parameter *Reference* value. Parameter *Linked widget reference* 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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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.
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.

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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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 (required)	4	itemids	Item ID. Note: To configure multiple items, create a dashboard widget field object for each item.
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.
Dynamic item	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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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.
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: 1, 2, 3, 4, 5 (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 *Tag name* 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 *Operator* required if configuring *Tags*.
Tag value	1	tags.value.0	Any string value. Parameter *Tag value* required if configuring *Tags*.
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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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.

Parameter	type	name	value
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: 1, 2, 3, 4, 5 (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 *Tag name* 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 *Operator* required if configuring *Tags*.

Parameter	type	name	value
Tag value	1	tags.value.0	Any string value. Parameter *Tag value* 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”.
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
        }
    ]
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "3"
        ]
    },
    "id": 1
}
```

See also

- Dashboard widget field
- `dashboard.create`
- `dashboard.update`

18 Problems by severity

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.

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).

Parameter	type	name	value
Severity	0	severities	0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster. Default: 1, 2, 3, 4, 5 (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 *Tag name* 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 *Operator* required if configuring *Tags*.
Tag value	1	tags.value.0	Any string value. Parameter *Tag value* 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”.

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.
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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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 (required)	10	slaid	SLA ID.
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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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.
Tag name	1	tags.tag.0	Any string value. Parameter *Tag name* 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 *Operator* required if configuring *Tags*.
Tag value	1	tags.value.0	Any string value. Parameter *Tag value* 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 (required)	0	columns.data.0	1 - Item value; 2 - Host name; 3 - Text.
Base color (required)	1	columns.base_color.0	Hexadecimal color code (e.g. FF0000).

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 (required)	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	type	name	value
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".
History data	0	columns.history.0	1 - *(default)* Auto; 2 - History; 3 - Trends.
Thresholds			
Color	1	columnsthresholds.color.0	Hexadecimal color code (e.g. FF0000). Default: "" (empty).
Threshold	1	columnsthresholds.threshold.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 *Text* 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", "Utilization", "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.item.0",
                                "value": "System name"
                            },
                            {
                                "type": 1,
                                "name": "columns.name.1",
                                "value": "Utilization"
                            },
                            {
                                "type": 0,
                                "name": "columns.data.1",
                                "value": 1
                            },
                            {
                                "type": 1,
                                "name": "columns.base_color.1",
                                "value": "4CAF50"
                            },
                            {
                                "type": 1,
                                "name": "columns.timeshift.1",
                                "value": ""
                            },
```

```
                {
                    "type": 1,
                    "name": "columns.item.1",
                    "value": "CPU utilization"
                },
                {
                    "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.timeshift.2",
                "value": ""
            },
            {
                "type": 1,
                "name": "columns.item.2",
                "value": "Load average (1m avg)"
            },
            {
                "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.timeshift.3",
                "value": ""
            },
            {
                "type": 1,
                "name": "columns.item.3",
                "value": "Load average (5m avg)"
            },
            {
                "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.timeshift.4",
                "value": ""
            },
            {
                "type": 1,
                "name": "columns.item.4",
                "value": "Load average (15m avg)"
```

```
                            },
                            {
                                "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.timeshift.5",
                                "value": ""
                            },
                            {
                                "type": 1,
                                "name": "columns.item.5",
                                "value": "Number of processes"
                            },
                            {
                                "type": 0,
                                "name": "column",
                                "value": 1
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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 *Tag name* 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 *Operator* required if configuring *Tags*.
Tag value	1	tags.value.0	Any string value. Parameter *Tag value* 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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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 (required)	1	url	Valid URL string.
Dynamic item	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/6.2/en"
                            }
                        ]
                    }
                ]
            }
        ],
        "userGroups": [
            {
                "usrgrpid": 7,
                "permission": 2
            }
        ],
        "users": [
            {
                "userid": 1,
                "permission": 3
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

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.
Tag name	1	tags.tag.0	Any string value. Parameter *Tag name* 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 *Operator* required if configuring *Tags*.
Tag value	1	tags.value.0	Any string value. Parameter *Tag value* required if configuring *Tags*.

Parameter	type	name	value
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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	array	Dashboard pages to be created for the dashboard. Dashboard pages will be ordered in the same order as specified. At least one dashboard page object is required for pages property.
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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 value is: `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"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 specified for each dashboard, all other properties are optional. Only the specified properties will be updated.

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. At least one dashboard page object is required for `pages` property.
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 2
}
```

Change dashboard owner

Available only for admins and super admins.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "dashboard.update",
    "params": {
        "dashboardid": "2",
        "userid": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 2
}
```

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.
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 are: `dhostid` and `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"
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.
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.

Parameter	Type	Description
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 are: `dserviceid`, `dhostid` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the discovery check.
druleid	string	*(readonly)* ID of the discovery rule that the check belongs to.
key_	string	The value of this property differs depending on the type of the check: - key to query for Zabbix agent checks, required; - SNMP OID for SNMPv1, SNMPv2 and SNMPv3 checks, required.
ports	string	One or several port ranges to check separated by commas. Used for all checks except for ICMP. Default: 0.
snmp_community	string	SNMP community. Required for SNMPv1 and SNMPv2 agent checks.
snmpv3_authpassphrase	string	Authentication passphrase used for SNMPv3 agent checks with security level set to *authNoPriv* or *authPriv*.

Property	Type	Description
snmpv3_authprotocol	integer	Authentication protocol used for SNMPv3 agent checks with security level set to *authNoPriv* or *authPriv*. Possible values: 0 - *(default)* MD5; 1 - SHA1; 2 - SHA224; 3 - SHA256; 4 - SHA384; 5 - SHA512.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 checks.
snmpv3_privpassphrase	string	Privacy passphrase used for SNMPv3 agent checks with security level set to *authPriv*.
snmpv3_privprotocol	integer	Privacy protocol used for SNMPv3 agent checks with security level set to *authPriv*. Possible values: 0 - *(default)* DES; 1 - AES128; 2 - AES192; 3 - AES256; 4 - AES192C; 5 - AES256C.
snmpv3_securitylevel	string	Security level used for SNMPv3 agent checks. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	Security name used for SNMPv3 agent checks.
type (required)	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.
uniq	integer	Whether to use this check as a device uniqueness criteria. Only a single unique check can be configured for a discovery rule. Used for Zabbix agent, SNMPv1, SNMPv2 and SNMPv3 agent checks. Possible values: 0 - *(default)* do not use this check as a uniqueness criteria; 1 - use this check as a uniqueness criteria.
host_source	integer	Source for host name. Possible values: 1 - *(default)* DNS; 2 - IP; 3 - discovery value of this check.

Property	Type	Description
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 are: `dcheckid` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the discovery rule.
iprange (required)	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.
name (required)	string	Name of the discovery rule.
delay	string	Execution interval of the discovery rule. Accepts seconds, time unit with suffix and user macro. Default: 1h.
nextcheck	timestamp	*(readonly)* Time when the discovery rule will be executed next.
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.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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 (required)	array	Discovery checks to create for the discovery rule.

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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: `druleid` and `name`.

Parameter	Type	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

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "druleid": "2",
            "proxy_hostid": "0",
            "name": "Local network",
            "iprange": "192.168.3.1-255",
            "delay": "5s",
            "nextcheck": "1348754327",
            "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",
            "nextcheck": "0",
            "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 for trigger events: 0 - trigger. Possible values for discovery events: 1 - discovered host; 2 - discovered service. Possible values for autoregistration events: 3 - auto-registered host. Possible values for internal events: 0 - trigger; 4 - item; 5 - LLD rule. Possible values for service events: 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 for trigger and service events: 0 - OK; 1 - problem. Possible values for discovery events: 0 - host or service up; 1 - host or service down; 2 - host or service discovered; 3 - host or service lost. Possible values for internal events: 0 - ”normal” state; 1 - ”unknown” or ”not supported” state. This parameter is not used for active agent autoregistration events.
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.
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 of Media type URLs	Active media types 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 URLs

Object with media type url have the following properties.

Property	Type	Description
name	string	Media type defined URL name.

Property	Type	Description
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 properties contain non expanded macro both properties will be excluded from results. Supported macros described on page.

event.acknowledge

Description

`object event.acknowledge(object/array parameters)`

This method allows to update events. 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 (required)	string/object	IDs of the events to acknowledge.
action (required)	integer	Event update action(s). This is bitmask field, any combination of values is acceptable. Possible values: 1 - close problem; 2 - acknowledge event; 4 - add message; 8 - change severity; 16 - unacknowledge event; 32 - suppress event; 64 - unsuppress event.
message	string	Text of the message. **Required**, if action contains 'add message' flag.

Parameter	Type	Description
severity	integer	New severity for events. **Required**, if action contains 'change severity' flag. Possible values: 0 - not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
suppress_until	integer	Unix timestamp until which event must be suppressed. **Required**, if action contains 'suppress event' flag. Set '0' to make indefinitely suppression.

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."
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "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.
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.
severities	integer/array	Return only events 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 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.
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; `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.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties. Possible values are: `eventid`, `objectid` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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",
            "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",
                    "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",
            "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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",
            "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",
            "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",
            "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	*(readonly)* ID of the graph.
height (required)	integer	Height of the graph in pixels.
name (required)	string	Name of the graph
width (required)	integer	Width of the graph in pixels.

Property	Type	Description
flags	integer	*(readonly)* Origin of the graph. Possible values are: 0 - *(default)* a plain graph; 4 - a discovered graph.
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.
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	*(readonly)* ID of the parent template graph.
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. Starting with Zabbix 6.2.1, if user have no access to specified item, the graph is rendered like ymax_type would be set to '0' (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. Starting with Zabbix 6.2.1, if user have no access to specified item, the graph is rendered like ymin_type would be set to '0' (calculated).

Property	Type	Description
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. Used only for graphs on templates. Auto-generated, if not given.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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 (required)	array	Graph items to be created for the graph.

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"
            }
```

```
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.

Parameter	Type	Description
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 are: `graphid`, `name` and `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	
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the graph item.
color (required)	string	Graph item's draw color as a hexadecimal color code.
itemid (required)	string	ID of the item.

Property	Type	Description
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.
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.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties. Possible values are: `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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",
            "key_": "system.cpu.util[,steal]",
            "hostid": "10001",
            "flags": "0",
            "host": "Linux"
        },
        {
            "gitemid": "1243",
            "graphid": "387",
            "itemid": "22668",
            "drawtype": "1",
            "sortorder": "2",
            "color": "55FF55",
            "yaxisside": "0",
```

```
            "calc_fnc": "2",
            "type": "0",
            "key_": "system.cpu.util[,softirq]",
            "hostid": "10001",
            "flags": "0",
            "host": "Linux"
        },
        {
            "gitemid": "1244",
            "graphid": "387",
            "itemid": "22671",
            "drawtype": "1",
            "sortorder": "3",
            "color": "009999",
            "yaxisside": "0",
            "calc_fnc": "2",
            "type": "0",
            "key_": "system.cpu.util[,interrupt]",
            "hostid": "10001",
            "flags": "0",
            "host": "Linux"
        }
    ],
    "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	*(readonly)* ID of the graph prototype.
height (required)	integer	Height of the graph prototype in pixels.
name (required)	string	Name of the graph prototype.

Property	Type	Description
width (required)	integer	Width of the graph prototype in pixels.
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.
show_work_period	integer	Whether to show the working time on the discovered graph. Possible values: 0 - hide; 1 - *(default)* show.
templateid	string	*(readonly)* ID of the parent template graph prototype.
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. Starting with Zabbix 6.2.1, if user have no access to specified item, the graph is rendered like ymax_type would be set to ’0’ (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. Starting with Zabbix 6.2.1, if user have no access to specified item, the graph is rendered like ymin_type would be set to ’0’ (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.

Property	Type	Description
uuid	string	Universal unique identifier, used for linking imported graph prototypes to already existing ones. Used only for graph prototypes on templates. Auto-generated, if not given.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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 (required)	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.

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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.
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 are: `graphid`, `name` and `graphtype`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	

Parameter	Type	Description
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 a LLD rule

Retrieve all graph prototypes from an LLD rule.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "graphprototype.get",
    "params": {
        "output": "extend",
        "discoveryids": "27426"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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, t.i. 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 are: `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	
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"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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"]
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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
        }
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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
        }
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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"]
        }
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.

Property	Type	Description
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"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

See also: known issues

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 - numeric unsigned; 4 - text. Default: 3.
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 are: `itemid` and `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	
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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the host.
host (required)	string	Technical name of the host.
description	text	Description of the host.
flags	integer	*(readonly)* Origin of the host. Possible values: 0 - a plain host; 4 - a discovered host.
inventory_mode	integer	Host inventory population mode. Possible values are: -1 - *(default)* disabled; 0 - manual; 1 - automatic.
ipmi_authtype	integer	IPMI authentication algorithm. Possible values are: -1 - *(default)* default; 0 - none; 1 - MD2; 2 - MD5 4 - straight; 5 - OEM; 6 - RMCP+.
ipmi_password	string	IPMI password.

Property	Type	Description
ipmi_privilege	integer	IPMI privilege level. Possible values are: 1 - callback; 2 - *(default)* user; 3 - operator; 4 - admin; 5 - OEM.
ipmi_username	string	IPMI username.
maintenance_from	timestamp	*(readonly)* Starting time of the effective maintenance.
maintenance_status	integer	*(readonly)* Effective maintenance status. Possible values are: 0 - *(default)* no maintenance; 1 - maintenance in effect.
maintenance_type	integer	*(readonly)* Effective maintenance type. Possible values are: 0 - *(default)* maintenance with data collection; 1 - maintenance without data collection.
maintenanceid	string	*(readonly)* ID of the maintenance that is currently in effect on the host.
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 are: 0 - *(default)* monitored host; 1 - unmonitored host.
tls_connect	integer	Connections to host. Possible values are: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate.
tls_accept	integer	Connections from host. Possible bitmap values are: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate.
tls_issuer	string	Certificate issuer.
tls_subject	string	Certificate subject.
tls_psk_identity	string	*(write-only)* PSK identity. Required if either `tls_connect` or `tls_accept` has PSK enabled. Do not put sensitive information in the PSK identity, it is transmitted unencrypted over the network to inform a receiver which PSK to use.
tls_psk	string	*(write-only)* The preshared key, at least 32 hex digits. Required if either `tls_connect` or `tls_accept` has PSK enabled.
active_available	integer	*(readonly)* Host active interface availability status. Possible values are: 0 - interface status is unknown; 1 - interface is available; 2 - interface is not available.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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
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

ID	Property	Type	Description	Maximum length
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 (required)	string	Host tag name.
value	string	Host tag value.
automatic	integer	Type of host tag. Possible values are: 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 (required)	object/array	Host groups to add the host to. The host groups must have the `groupid` property defined.
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"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10658"
        ]
    },
    "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"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.
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.

Parameter	Type	Description
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	array/object	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`.
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`.

Parameter	Type	Description
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.
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`.
search	object	Return results that match the given wildcard search. 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 with 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 are: `hostid`, `host`, `name`, `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	
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"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

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": 2
}
```

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,
    "auth": "70785d2b494a7302309b48afcdb3a401"
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "hostid": "10084",
            "parentTemplates": [
                {
                    "name": "Linux",
                    "templateid": "10001"
                },
                {
                    "name": "Zabbix Server",
                    "templateid": "10047"
                }
            ]
        }
    ],
    "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": 2,
    "auth": "7f9e00124c75e8f25facd5c093f3e9a0"
}
```

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
            }
        ]
    },
    "auth": "7f9e00124c75e8f25facd5c093f3e9a0",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	object/array	Hosts to be updated. The hosts must have the `hostid` property defined.

Parameter	Type	Description
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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	string/array	IDs of the hosts to be updated.
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	object/array	Hosts to be updated. The hosts must have the `hostid` property defined.
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, i.e., set their status to 0.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.massupdate",
    "params": {
        "hosts": [
            {
                "hostid": "69665"
            },
            {
                "hostid": "69666"
            }
        ],
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.
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, i.e. set its status to 0.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "host.update",
    "params": {
        "hostid": "10126",
        "status": 0
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": "CentOS 7"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "10387"
        ]
    },
    "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	*(readonly)* ID of the host group.
name (required)	string	Name of the host group.
flags	integer	*(readonly)* Origin of the host group. Possible values: 0 - a plain host group; 4 - a discovered host group.
uuid	string	Universal unique identifier, used for linking imported host groups to already existing ones. Auto-generated, if not given.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 can not 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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.
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.

Parameter	Type	Description
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 are: `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"
            ]
        }
    },
    "auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
    "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 (required)	object/array	Host groups to be updated. The host groups must have the `groupid` property defined.
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"
            }
        ]
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
    "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 (required)	string/array	IDs of the host groups to be updated.
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"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	object/array	Host groups to be updated. The host groups must have the `groupid` property defined.
hosts (required)	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.

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"
            }
        ]
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
    "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 hosts 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 (required)	object/array	Host groups to propagate. The host groups must have the `groupid` property defined.
permissions	boolean	Set `true` if need to propagate permissions.
tag_filters	boolean	Set `true` if need to propagate tag filters.

At least one parameter `permissions` or `tag_filters` is required.

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
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
    "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"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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 are required. If you do not want to use DNS, set it to an empty string.

Property	Type	Description
available	integer	*(readonly)* Availability of host interface. Possible values are: 0 - *(default)* unknown; 1 - available; 2 - unavailable.
details	array	Additional object for interface. **Required** if interface 'type' is SNMP.
disable_until	timestamp	*(readonly)* The next polling time of an unavailable host interface.
dns (required)	string	DNS name used by the interface. Can be empty if the connection is made via IP.
error	string	*(readonly)* Error text if host interface is unavailable.
errors_from	timestamp	*(readonly)* Time when host interface became unavailable.
hostid (required)	string	ID of the host the interface belongs to.
interfaceid	string	*(readonly)* ID of the interface.
ip (required)	string	IP address used by the interface. Can be empty if the connection is made via DNS.
main (required)	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 are: 0 - not default; 1 - default.
port (required)	string	Port number used by the interface. Can contain user macros.
type (required)	integer	Interface type. Possible values are: 1 - agent; 2 - SNMP; 3 - IPMI; 4 - JMX.

Property	Type	Description
useip (required)	integer	Whether the connection should be made via IP. Possible values are: 0 - connect using host DNS name; 1 - connect using host IP address for this host interface.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Details tag

The details object has the following properties.

Property	Type	Description
version (required)	integer	SNMP interface version. Possible values are: 1 - SNMPv1; 2 - SNMPv2c; 3 - SNMPv3
bulk	integer	Whether to use bulk SNMP requests. Possible values are: 0 - don't use bulk requests; 1 - (default) - use bulk requests.
community	string	SNMP community (required). Used only by SNMPv1 and SNMPv2 interfaces.
securityname	string	SNMPv3 security name. Used only by SNMPv3 interfaces.
securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 interfaces. Possible values are: 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 are: 0 - (default) - MD5; 1 - SHA1; 2 - SHA224; 3 - SHA256; 4 - SHA384; 5 - SHA512.
privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 interfaces. Possible values are: 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",
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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}"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `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	
nodeids	string/array	
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

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "interfaceid": "50039",
            "hostid": "30057",
            "main": "1",
            "type": "1",
            "useip": "1",
            "ip": "::1",
            "dns": "",
            "port": "10050",
            "available": "0",
            "error": "",
            "errors_from": "0",
            "disable_until": "0",
            "details": []
        },
        {
            "interfaceid": "55082",
            "hostid": "30057",
            "main": "0",
            "type": "1",
            "useip": "1",
            "ip": "127.0.0.1",
            "dns": "",
            "port": "10051",
            "available": "0",
            "error": "",
            "errors_from": "0",
            "disable_until": "0",
            "details": {
                "version": "2",
                "bulk": "0",
```

```
                "community": "{$SNMP_COMMUNITY}"
            }
        }
    ],
    "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
hosts (required)	object/array	Hosts to be updated. The hosts must have the `hostid` property defined.
interfaces (required)	object/array	Host interfaces to create on the given hosts.

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
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
hostids (required)	string/array	IDs of the hosts to be updated.
interfaces (required)	object/array	Host interfaces to remove from the given hosts. The host interface object must have the ip, dns and port properties defined

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"
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
hostid (required)	string	ID of the host to be updated.
interfaces (required)	object/array	Host interfaces to replace the current host interfaces with.

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
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the host prototype.

Property	Type	Description
host (required)	string	Technical name of the host prototype.
name	string	Visible name of the host prototype. Default: `host` property value.
status	integer	Status of the host prototype. Possible values are: 0 - *(default)* monitored host; 1 - unmonitored host.
inventory_mode	integer	Host inventory population mode. Possible values are: -1 - *(default)* disabled; 0 - manual; 1 - automatic.
templateid	string	*(readonly)* ID of the parent template host prototype.
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.
custom_interfaces	integer	Source of interfaces for hosts created by the host prototype. Possible values: 0 - *(default)* inherit interfaces from parent host; 1 - use host prototypes custom interfaces.
uuid	string	Universal unique identifier, used for linking imported host prototypes to already existing ones. Used only for host prototypes on templates. Auto-generated, if not given.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Group link

The group link object links a host prototype with a host group and has the following properties.

Property	Type	Description
groupid (required)	string	ID of the host group.

Group prototype

The group prototype object defines a group that will be created for a discovered host and has the following properties.

Property	Type	Description
name (required)	string	Name of the group prototype.

Host prototype tag

The host prototype tag object has the following properties.

Property	Type	Description
tag (required)	string	Host prototype tag name.
value	string	Host prototype tag value.

Custom interface

The custom interface object has the following properties.

Property	Type	Description
dns	string	DNS name used by the interface. **Required** if the connection is made via DNS. Can contain macros.
ip	string	IP address used by the interface. **Required** if the connection is made via IP. Can contain macros.
main (required)	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 are: 0 - not default; 1 - default.
port (required)	string	Port number used by the interface. Can contain user and LLD macros.
type (required)	integer	Interface type. Possible values are: 1 - agent; 2 - SNMP; 3 - IPMI; 4 - JMX.
useip (required)	integer	Whether the connection should be made via IP. Possible values are: 0 - connect using host DNS name; 1 - connect using host IP address for this host interface.
details	array	Additional object for interface. **Required** if interface 'type' is SNMP.

Custom interface details

The details object has the following properties.

Property	Type	Description
version (required)	integer	SNMP interface version. Possible values are: 1 - SNMPv1; 2 - SNMPv2c; 3 - SNMPv3
bulk	integer	Whether to use bulk SNMP requests. Possible values are: 0 - don't use bulk requests; 1 - *(default)* - use bulk requests.
community	string	SNMP community. Used only by SNMPv1 and SNMPv2 interfaces.
securityname	string	SNMPv3 security name. Used only by SNMPv3 interfaces.
securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 interfaces. Possible values are: 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.

Property	Type	Description
authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 interfaces. Possible values are: 0 - *(default)* - MD5; 1 - SHA1; 2 - SHA224; 3 - SHA256; 4 - SHA384; 5 - SHA512.
privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 interfaces. Possible values are: 0 - *(default)* - DES; 1 - AES128; 2 - AES192; 3 - AES256; 4 - AES192C; 5 - AES256C.
contextname	string	SNMPv3 context name. Used only by SNMPv3 interfaces.

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 (required)	array	Group links to be created for the host prototype.
ruleid (required)	string	ID of the LLD rule that the host prototype belongs to.
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}"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: `hostid`, `host`, `name` and `status`.
countOutput	boolean	These parameters being common for all `get` methods are described in detail on the Generic Zabbix API information 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 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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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}"
                    }
```

```
                }
            ]
        }
    ],
    "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.
groupPrototypes	array	Group prototypes to replace the existing group prototypes on the host prototype.
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.
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.
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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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}"
                }
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.
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.

Property	Type	Description
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	*(readonly)* Configuration flag DB extension. If this flag is set to "timescaledb" then the server changes its behavior for housekeeping and item deletion.
compression_status	integer	Enable TimescaleDB compression for history and trends. Possible values: 0 - *(default)* Off; 1 - On.
compress_older	string	Compress history and trends records older than specified period. Accepts seconds and time unit with suffix. Default: 7d.
compression_availability	integer	*(readonly)* Compression availability. Possible values: 0 - Unavailable; 1 - Available.

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the icon map.
default_iconid (reqiured)	string	ID of the default icon.

Property	Type	Description
name (required)	string	Name of the icon map.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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
iconmappingid	string	*(readonly)* ID of the icon map.
iconid (required)	string	ID of the icon used by the icon mapping.
expression (required)	string	Expression to match the inventory field against.
inventory_link (required)	integer	ID of the host inventory field. Refer to the host inventory object for a list of supported inventory fields.
iconmapid	string	*(readonly)* ID of the icon map that the icon mapping belongs to.
sortorder	integer	*(readonly)* Position of the icon mapping in the icon map.

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 (required)	array	Icon mappings to be created for the icon map.

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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: `iconmapid` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the image.
name (required)	string	Name of the image.
imagetype	integer	Type of image. Possible values: 1 - *(default)* icon; 2 - background image.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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.

Additionally to the standard image properties, the method accepts the following parameters.

Parameter	Type	Description
name (required)	string	Name of the image.
imagetype (required)	integer	Type of image. Possible values: 1 - *(default)* icon; 2 - background image.
image (required)	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 are: PNG, JPEG, GIF.

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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: `imageid` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

Additionally to the standard image properties, the method accepts the following parameters.

Parameter	Type	Description
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 are: PNG, JPEG, GIF.

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the item.
delay (required)	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}`). Optional for Zabbix trapper, dependent items and for Zabbix agent (active) with `mqtt.get` key.
hostid (required)	string	ID of the host or template that the item belongs to. For update operations this field is *readonly*.
interfaceid (required)	string	ID of the item's host interface. Used only for host items. Not required for Zabbix agent (active), Zabbix internal, Zabbix trapper, calculated, dependent, database monitor and script items. Optional for HTTP agent items.
key_ (required)	string	Item key.
name (required)	string	Name of the item.
type (required)	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
url (required)	string	URL string, required only for HTTP agent item type. Supports user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}.

Property	Type	Description
value_type (required)	integer	Type of information of the item. Possible values: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.
allow_traps	integer	HTTP agent item field. Allow to populate value as in trapper item type also. 0 - *(default)* Do not allow to accept incoming data. 1 - Allow to accept incoming data.
authtype	integer	Used only by SSH agent items or HTTP agent items. SSH agent authentication method possible values: 0 - *(default)* password; 1 - public key. HTTP agent authentication method possible values: 0 - *(default)* none 1 - basic 2 - NTLM 3 - Kerberos
description	string	Description of the item.
error	string	*(readonly)* Error text if there are problems updating the item.
flags	integer	*(readonly)* Origin of the item. Possible values: 0 - a plain item; 4 - a discovered item.
follow_redirects	integer	HTTP agent item field. Follow response redirects while pooling data. 0 - Do not follow redirects. 1 - *(default)* Follow redirects.
headers	object	HTTP agent item field. Object with HTTP(S) request headers, where header name is used as key and header value as value. Example: { "User-Agent": "Zabbix" }
history	string	A time unit of how long the history data should be stored. Also accepts user macro. Default: 90d.
http_proxy	string	HTTP agent item field. HTTP(S) proxy connection string.
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.
ipmi_sensor	string	IPMI sensor. Used only by IPMI items.
jmx_endpoint	string	JMX agent custom connection string. Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi
lastclock	timestamp	*(readonly)* Time when the item 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	Type	Description
lastns	integer	*(readonly)* Nanoseconds when the item 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.
lastvalue	string	*(readonly)* 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.
logtimefmt	string	Format of the time in log entries. Used only by log items.
master_itemid	integer	Master item ID. Recursion up to 3 dependent items and maximum count of dependent items equal to 29999 are allowed. Required by dependent items.
output_format	integer	HTTP agent item field. Should response be converted to JSON. 0 - *(default)* Store raw. 1 - Convert to JSON.
params	string	Additional parameters depending on the type of the item: - executed script for SSH and Telnet items; - SQL query for database monitor items; - formula for calculated items; - the script for script item.
parameters	array	Additional parameters for script items. Array of objects with 'name' and 'value' properties, where name must be unique.
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items. When used by JMX, username should also be specified together with password or both properties should be left blank.
post_type	integer	HTTP agent item field. Type of post data body stored in posts property. 0 - *(default)* Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent item field. HTTP(S) request body data. Used with post_type.
prevvalue	string	*(readonly)* 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.
privatekey	string	Name of the private key file.
publickey	string	Name of the public key file.
query_fields	array	HTTP agent item field. Query parameters. Array of objects with 'key':'value' pairs, where value can be empty string.
request_method	integer	HTTP agent item field. Type of request method. 0 - *(default)* GET 1 - POST 2 - PUT 3 - HEAD

Property	Type	Description
retrieve_mode	integer	HTTP agent item field. What part of response should be stored. 0 - *(default)* Body. 1 - Headers. 2 - Both body and headers will be stored. For request_method HEAD only 1 is allowed value.
snmp_oid	string	SNMP OID.
ssl_cert_file	string	HTTP agent item field. Public SSL Key file path.
ssl_key_file	string	HTTP agent item field. Private SSL Key file path.
ssl_key_password	string	HTTP agent item field. Password for SSL Key file.
state	integer	*(readonly)* State of the item. Possible values: 0 - *(default)* normal; 1 - not supported.
status	integer	Status of the item. Possible values: 0 - *(default)* enabled item; 1 - disabled item.
status_codes	string	HTTP agent item field. 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
templateid	string	(readonly) ID of the parent template item. *Hint*: Use the `hostid` property to specify the template that the item belongs to.
timeout	string	Item data polling request timeout. Used for HTTP agent and script items. Supports user macros. default: 3s maximum value: 60s
trapper_hosts	string	Allowed hosts. Used by trapper items or HTTP agent items.
trends	string	A time unit of how long the trends data should be stored. Also accepts user macro. Default: 365d.
units	string	Value units.
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, password should also be specified together with username or both properties should be left blank.
uuid	string	Universal unique identifier, used for linking imported item to already existing ones. Used only for items on templates. Auto-generated, if not given.
valuemapid	string	ID of the associated value map.
verify_host	integer	HTTP agent item field. Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. 0 - *(default)* Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent item field. Validate is host certificate authentic. 0 - *(default)* Do not validate. 1 - Validate.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Item tag

The item tag object has the following properties.

Property	Type	Description
tag (required)	string	Item tag name.
value	string	Item tag value.

Item preprocessing

The item preprocessing object has the following properties.

Property	Type	Description
type (required)	integer	The preprocessing option type. Possible values: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression matching; 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.
params (required)	string	Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character.
error_handler (required)	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.
error_handler_params (required)	string	Error handler parameters. Used with `error_handler`. Must be empty, if `error_handler` is 0 or 1. Can be empty if, `error_handler` is 2. Cannot be empty, if `error_handler` is 3.

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
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

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
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
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

[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.

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": "Disc usage"
            },
            {
                "tag": "Equipment",
                "value": "Workstation"
            }
        ],
        "delay": "30s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": ""
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": ""
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 3
}
```

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
```

```
        ]
    },
    "id": 3
}
```

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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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`.

Parameter	Type	Description
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.
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 matching; 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. `params` - (`string`) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\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.

Parameter	Type	Description
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 are: `itemid`, `name`, `key_`, `delay`, `history`, `trends`, `type` and `status`.
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
        }
```

```
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": 3,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

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": 3
}
```

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.
tags	array	Item tags.

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
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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": ""
            }
        ]
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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;"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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	*(readonly)* ID of the item prototype.
delay (required)	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}`). Optional for Zabbix trapper, dependent items and for Zabbix agent (active) with `mqtt.get` key.
hostid (required)	string	ID of the host that the item prototype belongs to. For update operations this field is *readonly*.
ruleid (required)	string	ID of the LLD rule that the item belongs to. For update operations this field is *readonly*.
interfaceid (required)	string	ID of the item prototype's host interface. Used only for host item prototypes. Not required for Zabbix agent (active), Zabbix internal, Zabbix trapper, calculated, dependent, database monitor and script item prototypes. Optional for HTTP agent item prototypes.
key_ (required)	string	Item prototype key.
name (required)	string	Name of the item prototype.

Property	Type	Description
type (required)	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.
url (required)	string	URL string required only for HTTP agent item prototypes. Supports LLD macros, user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}.
value_type (required)	integer	Type of information of the item prototype. Possible values: 0 - numeric float; 1 - character; 2 - log; 3 - numeric unsigned; 4 - text.
allow_traps	integer	HTTP agent item prototype field. Allow to populate value as in trapper item type also. 0 - *(default)* Do not allow to accept incoming data. 1 - Allow to accept incoming data.
authtype	integer	Used only by SSH agent item prototypes or HTTP agent item prototypes. SSH agent authentication method possible values: 0 - *(default)* password; 1 - public key. HTTP agent authentication method possible values: 0 - *(default)* none 1 - basic 2 - NTLM 3 - Kerberos
description	string	Description of the item prototype.
follow_redirects	integer	HTTP agent item prototype field. Follow response redirects while pooling data. 0 - Do not follow redirects. 1 - *(default)* Follow redirects.
headers	object	HTTP agent item prototype field. Object with HTTP(S) request headers, where header name is used as key and header value as value. Example: { ”User-Agent”: ”Zabbix” }

Property	Type	Description
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 agent item prototype field. HTTP(S) proxy connection string.
ipmi_sensor	string	IPMI sensor. Used only by IPMI item prototypes.
jmx_endpoint	string	JMX agent custom connection string. Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi
logtimefmt	string	Format of the time in log entries. Used only by log item prototypes.
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. Required by Dependent items.
output_format	integer	HTTP agent item prototype field. Should response be converted to JSON. 0 - *(default)* Store raw. 1 - Convert to JSON.
params	string	Additional parameters depending on the type of the item prototype: - executed script for SSH and Telnet item prototypes; - SQL query for database monitor item prototypes; - formula for calculated item prototypes.
parameters	array	Additional parameters for script item prototypes. Array of objects with ’name’ and ’value’ properties, where name must be unique.
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent item prototypes.
post_type	integer	HTTP agent item prototype field. Type of post data body stored in posts property. 0 - *(default)* Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent item prototype field. HTTP(S) request body data. Used with post_type.
privatekey	string	Name of the private key file.
publickey	string	Name of the public key file.
query_fields	array	HTTP agent item prototype field. Query parameters. Array of objects with ’key’:’value’ pairs, where value can be empty string.
request_method	integer	HTTP agent item prototype field. Type of request method. 0 - *(default)* GET 1 - POST 2 - PUT 3 - HEAD
retrieve_mode	integer	HTTP agent item prototype field. What part of response should be stored. 0 - *(default)* Body. 1 - Headers. 2 - Both body and headers will be stored. For request_method HEAD only 1 is allowed value.
snmp_oid	string	SNMP OID.
ssl_cert_file	string	HTTP agent item prototype field. Public SSL Key file path.
ssl_key_file	string	HTTP agent item prototype field. Private SSL Key file path.
ssl_key_password	string	HTTP agent item prototype field. Password for SSL Key file.

Property	Type	Description
status	integer	Status of the item prototype. Possible values: 0 - *(default)* enabled item prototype; 1 - disabled item prototype; 3 - unsupported item prototype.
status_codes	string	HTTP agent item prototype field. 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
templateid	string	(readonly) ID of the parent template item prototype.
timeout	string	Item data polling request timeout. Used for HTTP agent and script item prototypes. Supports user macros and LLD macros. default: 3s maximum value: 60s
trapper_hosts	string	Allowed hosts. Used by trapper item prototypes or HTTP item prototypes.
trends	string	A time unit of how long the trends data should be stored. Also accepts user macro and LLD macro. Default: 365d.
units	string	Value units.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent item prototypes. Required by SSH and Telnet item prototypes.
uuid	string	Universal unique identifier, used for linking imported item prototypes to already existing ones. Used only for item prototypes on templates. Auto-generated, if not given.
valuemapid	string	ID of the associated value map.
verify_host	integer	HTTP agent item prototype field. Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. 0 - *(default)* Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent item prototype field. Validate is host certificate authentic. 0 - *(default)* Do not validate. 1 - Validate.
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.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Item prototype tag

The item prototype tag object has the following properties.

Property	Type	Description
tag (required)	string	Item prototype tag name.
value	string	Item prototype tag value.

Item prototype preprocessing

The item prototype preprocessing object has the following properties.

Property	Type	Description
type (required)	integer	The preprocessing option type. Possible values: 1 - Custom multiplier; 2 - Right trim; 3 - Left trim; 4 - Trim; 5 - Regular expression matching; 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.
params (required)	string	Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character.
error_handler (required)	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.
error_handler_params (required)	string	Error handler parameters. Used with `error_handler`. Must be empty, if `error_handler` is 0 or 1. Can be empty if, `error_handler` is 2. Cannot be empty, if `error_handler` is 3.

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]			

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
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
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

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
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
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

[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.

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 (required)	string	ID of the LLD rule that the item belongs to.
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 disc 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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 3
}
```

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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 manual/api/reference/graphprototype/object#graph_prototype 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.

Parameter	Type	Description
selectTriggers	query	Return a triggers property with trigger prototypes that the item prototype is used in. Supports `count`.
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 matching; 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. `params` - (`string`) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\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 are: `itemid`, `name`, `key_`, `delay`, `type` and `status`.

Parameter	Type	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	
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": 17,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```


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": 17
}
```


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.
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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": []
    }
    "auth": "700ca65537074ec963db7efabda78259",
    "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"
            }
        ]
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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;"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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	*(readonly)* ID of the LLD rule.
delay (required)	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. Optional for Zabbix trapper, dependent items and for Zabbix agent (active) with `mqtt.get` key.
hostid (required)	string	ID of the host that the LLD rule belongs to.
interfaceid (required)	string	ID of the LLD rule's host interface. Used only for host LLD rules. Not required for Zabbix agent (active), Zabbix internal, Zabbix trapper, dependent, database monitor and script LLD rules. Optional for HTTP agent LLD rules.
key_ (required)	string	LLD rule key.
name (required)	string	Name of the LLD rule.
type (required)	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.
url (required)	string	URL string, required for HTTP agent LLD rule. Supports user macros, {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}.

Property	Type	Description
allow_traps	integer	HTTP agent LLD rule field. Allow to populate value as in trapper item type also. 0 - *(default)* Do not allow to accept incoming data. 1 - Allow to accept incoming data.
authtype	integer	Used only by SSH agent or HTTP agent LLD rules. SSH agent authentication method possible values: 0 - *(default)* password; 1 - public key. HTTP agent authentication method possible values: 0 - *(default)* none 1 - basic 2 - NTLM
description	string	Description of the LLD rule.
error	string	*(readonly)* Error text if there are problems updating the LLD rule.
follow_redirects	integer	HTTP agent LLD rule field. Follow response redirects while pooling data. 0 - Do not follow redirects. 1 - *(default)* Follow redirects.
headers	object	HTTP agent LLD rule field. Object with HTTP(S) request headers, where header name is used as key and header value as value. Example: { ”User-Agent”: ”Zabbix” }
http_proxy	string	HTTP agent LLD rule field. HTTP(S) proxy connection string.
ipmi_sensor	string	IPMI sensor. Used only by IPMI LLD rules.
jmx_endpoint	string	JMX agent custom connection string. Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi
lifetime	string	Time period after which items that are no longer discovered will be deleted. Accepts seconds, time unit with suffix and 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. Required for Dependent item.
output_format	integer	HTTP agent LLD rule field. Should response be converted to JSON. 0 - *(default)* Store raw. 1 - Convert to JSON.
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.
parameters	array	Additional parameters for script type LLD rule. Array of objects with ’name’ and ’value’ properties, where name must be unique.
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD rules.
post_type	integer	HTTP agent LLD rule field. Type of post data body stored in posts property. 0 - *(default)* Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent LLD rule field. HTTP(S) request body data. Used with post_type.

Property	Type	Description
privatekey	string	Name of the private key file.
publickey	string	Name of the public key file.
query_fields	array	HTTP agent LLD rule field. Query parameters. Array of objects with 'key':'value' pairs, where value can be empty string.
request_method	integer	HTTP agent LLD rule field. Type of request method. 0 - *(default)* GET 1 - POST 2 - PUT 3 - HEAD
retrieve_mode	integer	HTTP agent LLD rule field. What part of response should be stored. 0 - *(default)* Body. 1 - Headers. 2 - Both body and headers will be stored. For request_method HEAD only 1 is allowed value.
snmp_oid	string	SNMP OID.
ssl_cert_file	string	HTTP agent LLD rule field. Public SSL Key file path.
ssl_key_file	string	HTTP agent LLD rule field. Private SSL Key file path.
ssl_key_password	string	HTTP agent LLD rule field. Password for SSL Key file.
state	integer	*(readonly)* State of the LLD rule. Possible values: 0 - *(default)* normal; 1 - not supported.
status	integer	Status of the LLD rule. Possible values: 0 - *(default)* enabled LLD rule; 1 - disabled LLD rule.
status_codes	string	HTTP agent LLD rule field. 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
templateid	string	(readonly) ID of the parent template LLD rule.
timeout	string	Item data polling request timeout. Used for HTTP agent and script LLD rules. Supports user macros. default: 3s maximum value: 60s
trapper_hosts	string	Allowed hosts. Used by trapper LLD rules or HTTP agent LLD rules.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD rules. Required by SSH and Telnet LLD rules.
uuid	string	Universal unique identifier, used for linking imported LLD rules to already existing ones. Used only for LLD rules on templates. Auto-generated, if not given.
verify_host	integer	HTTP agent LLD rule field. Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate. 0 - *(default)* Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent LLD rule field. Validate is host certificate authentic. 0 - *(default)* Do not validate. 1 - Validate.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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 (required)	array	Set of filter conditions to use for filtering results.
evaltype (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
eval_formula	string	*(readonly)* 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.
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. Required for custom expression filters.

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 (required)	string	LLD macro to perform the check on.
value (required)	string	Value to compare with.
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 (required)	string	LLD macro.
path (required)	string	Selector for value which will be assigned to corresponding macro.

LLD rule preprocessing

The LLD rule preprocessing object has the following properties.

Property	Type	Description
type (required)	integer	The preprocessing option type. Possible values: 5 - Regular expression matching; 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; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace; 27 - XML to JSON.
params (required)	string	Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character.
error_handler (required)	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.
error_handler_params (required)	string	Error handler parameters. Used with `error_handler`. Must be empty, if `error_handler` is 0 or 1. Can be empty if, `error_handler` is 2. Cannot be empty, if `error_handler` is 3.

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

Preprocessing type	Name	Parameter 1	Parameter 2	Parameter 3	Supported error handlers
20	Discard unchanged with heartbeat	seconds[4, 5, 6]			
23	Prometheus to JSON	pattern[5, 7]			0, 1, 2, 3
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

[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] 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.
[8] Prometheus output following the syntax: `<label name>`.

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 (required)	string	Unique override name.
step (required)	integer	Unique order number of the override.
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 (required)	integer	Override filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
conditions (required)	array	Set of override filter conditions to use for matching the discovered objects.

Property	Type	Description
eval_formula	string	*(readonly)* 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.
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. Required for custom expression override filters.

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 (required)	string	LLD macro to perform the check on.
value (required)	string	Value to compare with.
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 (required)	integer	Type of discovered object to perform the action. Possible values: 0 - Item prototype; 1 - Trigger prototype; 2 - Graph prototype; 3 - Host prototype.
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.

Property	Type	Description
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 (required)	integer	Override the status for selected object. Possible values: 0 - Create enabled; 1 - Create disabled.

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 (required)	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.

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 (required)	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.

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 (required)	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.

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 (required)	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.

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 (required)	integer	Override the severity of trigger prototype. Possible values are: 0 - *(default)* not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.

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 (required)	string	New tag name.
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 (required)	string	Override the template of host prototype linked templates.

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 (required)	integer	Override the host prototype inventory mode. Possible values are: -1 - disabled; 0 - *(default)* manual; 1 - automatic.

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"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 object for the LLD rule.
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "27665"
        ]
    },
    "id": 1
}
```

Creating a 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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "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"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "d678e0b85688ce578ff061bd29a20d3b",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 35
}
```

Creating a 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": ""
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "44211"
        ]
    },
    "id": 1
}
```

Creating a 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"
                    }
                ]
            }
        ]
    }
```

```
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "23865"
        ]
    },
    "id": 3
}
```

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 `itemids` property.

Examples

Deleting multiple LLD rules

Delete two LLD rules.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "discoveryrule.delete",
    "params": [
        "27665",
        "27668"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.
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. It has the following properties: `type` - `(string)` The preprocessing option type: 5 - Regular expression matching; 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; 23 - Prometheus to JSON; 24 - CSV to JSON; 25 - Replace; 27 - XML to JSON. `params` - `(string)` Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\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.

Parameter	Type	Description
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: `selctItems`; `selectGraphs`; `selectTriggers`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `itemid`, `name`, `key_`, `delay`, `type` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": 39,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

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": 39
}
```


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": 39,
```

```
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

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": 39
}
```


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 object to replace the current filter.
preprocessing	array	LLD rule preprocessing options to replace the current 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 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"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": 36,
    "auth": "d678e0b85688ce578ff061bd29a20d3b"
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "itemids": [
            "28336"
        ]
    },
    "id": 36
}
```

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"
            }
        ]
```

```
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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;"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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	*(readonly)* ID of the maintenance.
name (required)	string	Name of the maintenance.
active_since (required)	timestamp	Time when the maintenance becomes active. The given value will be rounded down to minutes.
active_till (required)	timestamp	Time when the maintenance stops being active. The given value will be rounded down to minutes.
description	string	Description of the maintenance.
maintenance_type	integer	Type of maintenance. Possible values: 0 - *(default)* with data collection; 1 - without data collection.
tags_evaltype	integer	Problem tag evaluation method. Possible values: 0 - *(default)* And/Or; 2 - Or.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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. Used only for one time periods. The given value will be rounded down to minutes. Default: current date.
start_time	integer	Time of day when the maintenance starts in seconds. Used for daily, weekly and monthly periods. The given value will be rounded down to minutes. Default: 0.

Property	Type	Description
every	integer	Used for daily, weekly and monthly periods. For daily and weekly periods `every` defines day or week intervals at which the maintenance must come into effect. Default: 1. For monthly periods, if `dayofweek` property contains at least one selected day of week, the `every` property defines the week of the month when the maintenance must come into effect. Possible values: 1 - *(default)* first week; 2 - second week; 3 - third week; 4 - fourth week; 5 - last week.
dayofweek	integer	Days of the week when the maintenance must come into effect. Days are stored in binary form with each bit representing the corresponding day. For example, 4 equals 100 in binary and means, that maintenance will be enabled on Wednesday. Used for weekly and monthly time periods. Required only for weekly time periods. At least one `dayofweek` or `day` must be specified for monthly time periods.
day	integer	Day of the month when the maintenance must come into effect. Used only for monthly time periods. At least one `dayofweek` or `day` must be specified for monthly time periods.
month	integer	Months when the maintenance must come into effect. Months are stored in binary form with each bit representing the corresponding month. For example, 5 equals 101 in binary and means, that maintenance will be enabled in January and March. Required only for monthly time periods.

Problem tag

The problem tag object is used to define which problems must be suppressed when the maintenance comes into effect. It has the following properties.

Property	Type	Description
tag (required)	string	Problem tag name.
operator	integer	Condition operator. Possible values: 0 - Equals; 2 - *(default)* Contains.
value	string	Problem tag value.

Tags can only be specified for maintenance periods with data collection (`"maintenance_type":0`).

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. At least one object of `groups` or `hosts` must be specified.
hosts	object/array	Hosts that will undergo maintenance. The hosts must have the `hostid` property defined. At least one object of `groups` or `hosts` must be specified.
timeperiods (required)	object/array	Maintenance time periods.
tags	object/array	Problem tags. Define what problems must be suppressed. If no tags are given, all active maintenance host problems will be suppressed.
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": ""
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: `maintenanceid`, `name` and `maintenance_type`.
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	
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	Type	Description
hosts	object/array	Hosts to replace the current hosts. The hosts must have the `hostid` property defined.
timeperiods	object/array	Maintenance time periods to replace the current periods.
tags	object/array	Problem tags to replace the current tags.
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.

Attention:
At least one host or host group must be defined for each 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"}
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the map.
height (required)	integer	Height of the map in pixels.
name (required)	string	Name of the map.
width (required)	integer	Width of the map in pixels.
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.
grid_size	integer	Size of the map grid in pixels. Supported values: 20, 40, 50, 75 and 100. Default: 50.

Property	Type	Description
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. Required for maps with custom host label type.
label_string_hostgroup	string	Custom label for host group elements. Required for maps with custom host group label type.
label_string_image	string	Custom label for image elements. Required for maps with custom image label type.
label_string_map	string	Custom label for map elements. Required for maps with custom map label type.
label_string_trigger	string	Custom label for trigger elements. Required for maps with custom trigger label type.
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.
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.

Property	Type	Description
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.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Map element

The map element object defines an object displayed on a map. It has the following properties.

Property	Type	Description
selementid	string	*(readonly)* ID of the map element.

Property	Type	Description
elements (required)	array	Element data object. Required for host, host group, trigger and map type elements.
elementtype (required)	integer	Type of map element. Possible values: 0 - host; 1 - map; 2 - trigger; 3 - host group; 4 - image.
iconid_off (required)	string	ID of the image used to display the element in default state.
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. Unused for image elements.
iconid_maintenance	string	ID of the image used to display map elements in maintenance. Unused for image elements.
iconid_on	string	ID of the image used to display map elements with problems. Unused for image elements.
label	string	Label of the element.
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	*(readonly)* ID of the map that the element belongs to.
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.

Property	Type	Description
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 (required)	string	Map element tag name.

Property	Type	Description
operator	string	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	*(readonly)* ID of the map element URL.
name (required)	string	Link caption.
url (required)	string	Link URL.
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	*(readonly)* ID of the map link.
selementid1 (required)	string	ID of the first map element linked on one end.
selementid2 (required)	string	ID of the first map element linked on the other end.
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	*(readonly)* ID of the map link trigger.
triggerid (reqiuired)	string	ID of the trigger used as a link indicator.
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	*(readonly)* ID of the map URL.
name (required)	string	Link caption.
url (required)	string	Link URL.
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	*(readonly)* ID of the map user.
userid (required)	string	User ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

Map user group

List of map permissions based on user groups. It has the following properties:

Property	Type	Description
sysmapusrgrpid	string	*(readonly)* ID of the map user group.
usrgrpid (required)	string	User group ID.

Property	Type	Description
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

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	*(readonly)* ID of the map shape element.
type (required)	integer	Type of map shape element. Possible values: 0 - rectangle; 1 - ellipse. Property is required when new shapes are created.
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.
font_size	integer	Font size in pixels. Default: 11.
font_color	string	Font color. Default: ’000000’.

Property	Type	Description
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	*(readonly)* ID of the map shape element.
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.
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.

Property	Type	Description
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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: `name`, `width` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "sysmapids": [
            "9"
        ]
    },
    "id": 2
}
```

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	*(readonly)* ID of the media type.
name (required)	string	Name of the media type.
type (required)	integer	Transport used by the media type. Possible values: 0 - email; 1 - script; 2 - SMS; 4 - Webhook.
exec_path	string	For script media types `exec_path` contains the name of the executed script. Required for script media types.
gsm_modem	string	Serial device name of the GSM modem. Required for SMS media types.
passwd	string	Authentication password. Used for email media types.
smtp_email	string	Email address from which notifications will be sent. Required for email media types.
smtp_helo	string	SMTP HELO. Required for email media types.
smtp_server	string	SMTP server. Required for email media types.
smtp_port	integer	SMTP server port to connect to.
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.

Property	Type	Description
status	integer	Whether the media type is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
username	string	User name. Used for email media types.
exec_params	string	Script parameters. Each parameter ends with a new line feed.
maxsessions	integer	The maximum number of alerts that can be processed in parallel. Possible values for SMS: 1 - *(default)* Possible values for other media types: 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	string	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 value: 30s
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`.

Property	Type	Description
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 input parameters.
description	string	Media type description.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Webhook parameters

Parameters passed to webhook script when it is called, have the following properties.

Property	Type	Description
name (required)	string	Parameter name.
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 (required)	integer	Event source. Possible values: 0 - triggers; 1 - discovery; 2 - autoregistration; 3 - internal; 4 - services.
recovery (required)	integer	Operation mode. Possible values: 0 - operations; 1 - recovery operations; 2 - update operations.
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	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 e-mail media type

Create a new e-mail media type with a custom SMTP port and message templates.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.create",
    "params": {
        "type": "0",
        "name": "E-mail",
        "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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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",
        "exec_params": "{ALERT.SENDTO}\n{ALERT.SUBJECT}\n{ALERT.MESSAGE}\n",
        "maxattempts": "5",
        "attempt_interval": "11s"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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>"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 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
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.
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 are: `mediatypeid`.
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 media types

Retrieve all configured media types.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "mediatype.get",
    "params": {
        "output": "extend",
        "selectMessageTemplates": "extend"
    },
```

```
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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",
            "exec_params": "",
            "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": "",
            "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",
        "exec_params": "",
        "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": "",
        "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
}
```

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	Webhook parameters to replace the current webhook 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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "mediatypeids": [
            "6"
        ]
    },
    "id": 1
}
```

Source

CMediaType::update() in *ui/include/classes/api/services/CMediaType.php*.

Problem

This class is designed to work with problems.

Object references:

- Problem

Available methods:

- problem.get - retrieving problems

> Problem object

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.
object	integer	Type of object that is related to the problem event. Possible values for trigger events: 0 - trigger. Possible values for internal events: 0 - trigger; 4 - item; 5 - LLD rule. Possible values for service events: 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.

Property	Type	Description
r_clock	timestamp	Time when the recovery event was created.
r_ns	integer	Nanoseconds when the recovery event was created.
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 of Media type URLs	Active media types 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 URLs

Object with media type url have 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 properties contain non expanded macro both properties will be excluded from results. Supported macros described on page.

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.
severities	integer/array	Return only problems 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 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.

Parameter	Type	Description
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; 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 are: `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	
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"
    },
    "auth": "67f45d3eb1173338e1b1647c4bdc1916",
    "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",
            "opdata": "",
            "acknowledges": [
                {
                    "acknowledgeid": "14443",
                    "userid": "1",
                    "eventid": "1245463",
                    "clock": "1472457281",
                    "message": "problem solved",
                    "action": "6",
                    "old_severity": "0",
                    "new_severity": "0",
                    "suppress_until": "1472511600"
                }
            ],
            "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	*(readonly)* ID of the proxy.
host (required)	string	Name of the proxy.
status (required)	integer	Type of proxy. Possible values: 5 - active proxy; 6 - passive proxy.
description	text	Description of the proxy.
lastaccess	timestamp	*(readonly)* Time when the proxy last connected to the server.
tls_connect	integer	Connections to host. Possible values are: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate.
tls_accept	integer	Connections from host. Possible bitmap values are: 1 - *(default)* No encryption; 2 - PSK; 4 - certificate.
tls_issuer	string	Certificate issuer.
tls_subject	string	Certificate subject.
tls_psk_identity	string	*(write-only)* PSK identity. Required if either `tls_connect` or `tls_accept` has PSK enabled. Do not put sensitive information in the PSK identity, it is transmitted unencrypted over the network to inform a receiver which PSK to use.

Property	Type	Description
tls_psk	string	*(write-only)* The preshared key, at least 32 hex digits. Required if either `tls_connect` or `tls_accept` has PSK enabled.
proxy_address	string	Comma-delimited IP addresses or DNS names of active Zabbix proxy.
auto_compress	integer	*(readonly)* Indicates if communication between Zabbix server and proxy is compressed. Possible values are: 0 - No compression; 1 - Compression enabled;

Note that for some methods (update, delete) the required/optional parameter combination is different.

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 (required)	string	DNS name to connect to. Can be empty if connections are made via IP address.
ip (required)	string	IP address to connect to. Can be empty if connections are made via DNS names.
port (required)	string	Port number to connect to.
useip (required)	integer	Whether the connection should be made via IP address. Possible values are: 0 - connect using DNS name; 1 - connect using IP address.

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. Required for passive proxies.

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"
            }
        ]
    },
    "auth": "ab9638041ec6922cb14b07982b268f47",
    "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"
            }
        ]
    },
    "auth": "ab9638041ec6922cb14b07982b268f47",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.
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 are: `hostid`, `host` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "host": "Active proxy",
            "status": "5",
            "lastaccess": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "proxy_address": "",
            "auto_compress": "0",
            "proxyid": "30091",
            "interface": []
        },
        {
            "host": "Passive proxy",
            "status": "6",
            "lastaccess": "0",
            "description": "",
            "tls_connect": "1",
            "tls_accept": "1",
            "tls_issuer": "",
            "tls_subject": "",
            "proxy_address": "",
            "auto_compress": "0",
            "proxyid": "30092",
            "interface": {
                "interfaceid": "30109",
                "hostid": "30092",
                "useip": "1",
                "ip": "127.0.0.1",
                "dns": "",
                "port": "10051"
            }
        }
    ],
    "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.

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"
            },
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the regular expression.
name (required)	string	Name of the regular expression.
test_string	string	Test string.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Expressions object

The expressions object has the following properties.

Property	Type	Description
expression (required)	string	Regular expression.
expression_type (required)	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.
exp_delimiter	string	Expression delimiter. Only when `expression_type` *Any character string included*. Default value `,`. Possible values: `,`, `.`, `/`.
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.

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"
        }
      ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: `regexpid` and `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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
        }
      ]
    },
    "auth": "700ca65537074ec963db7efabda78259",
```

```
    "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	*(readonly)* ID of the report.
userid (required)	string	ID of the user who created the report.
name (required)	string	Unique name of the report.
dashboardid (required)	string	ID of the dashboard that the report is based on.
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.

Property	Type	Description
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. Required for weekly reports only. Days of the week are stored in binary form with each bit representing the corresponding week day. For example, 12 equals 1100 in binary and means that reports will be sent every Wednesday and Thursday. Default: 0.
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.
state	integer	*(readonly)* 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.
lastsent	timestamp	*(readonly)* Unix timestamp of the last successfully sent report.
info	string	*(readonly)* Error description or additional information.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Users

The users object has the following properties:

Property	Type	Description
userid (required)	string	ID of user to send the report to.

Property	Type	Description
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 (required)	string	ID of user group to send the report to.
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 of objects	Users to send the report to.
user_groups	object/array of objects	User groups to send the report to.

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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties. Possible values are: `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"]
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 of objects	Users to replace the current users assigned to the scheduled report.
user_groups	object/array of objects	User groups to replace the current user groups assigned to the scheduled report.

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the role.
name (required)	string	Name of the role.
type (required)	integer	User type. Possible values: 1 - *(default)* User; 2 - Admin; 3 - Super admin.
readonly	integer	*(readonly)* Whether the role is readonly. Possible values: 0 - *(default)* No; 1 - Yes.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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.

Property	Type	Description
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.
services.read.tag	object	Only used if `services.read.mode` is set to 0. 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.
services.write.mode	integer	Only used if `services.read.mode` is set to 0. 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.
services.write.tag	object	Only used if `services.write.mode` is set to 0. 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.
modules	array	Only used if `services.write.mode` is set to 0. 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 (required)	string	Name of the UI element. Possible values for users of any type: `monitoring.dashboard` - *Monitoring → Dashboard*; `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 only for users of *Admin* and *Super admin* user types: `monitoring.discovery` - *Monitoring → Discovery*; `services.actions` - *Services → Actions*; `services.sla` - *Services → SLA*; `reports.scheduled_reports` - *Reports → Scheduled reports*; `reports.notifications` - *Reports → Notifications*; `configuration.template_groups` - *Configuration → Template groups*; `configuration.host_groups` - *Configuration → Host groups*; `configuration.templates` - *Configuration → Templates*; `configuration.hosts` - *Configuration → Hosts*; `configuration.maintenance` - *Configuration → Maintenance*; `configuration.actions` - *Configuration → Actions*; `configuration.discovery` - *Configuration → Discovery*. Possible values only for users of *Super admin* user type: `reports.system_info` - *Reports → System information*; `reports.audit` - *Reports → Audit*; `reports.action_log` - *Reports → Action log*; `configuration.event_correlation` - *Configuration → Event correlation*; `administration.general` - *Administration → General*; `administration.proxies` - *Administration → Proxies*; `administration.authentication` - *Administration → Authentication*; `administration.user_groups` - *Administration → User groups*; `administration.user_roles` - *Administration → User roles*; `administration.users` - *Administration → Users*; `administration.media_types` - *Administration → Media types*; `administration.scripts` - *Administration → Scripts*; `administration.queue` - *Administration → Queue*.
status	integer	Whether access to the UI element is enabled. Possible values: 0 - Disabled; 1 - *(default)* Enabled.

Service

Property	Type	Description
serviceid (required)	string	ID of the Service.

Service tag

Property	Type	Description
tag (required)	string	Tag name. If empty string is specified, the service tag will not be used for service matching.
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 (required)	string	ID of the module.
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 (required)	string	Name of the action. Possible values for users of any type: `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 only for users of *Admin* and *Super admin* user types: `edit_maintenance` - Create and edit maintenances; `manage_scheduled_reports` - Manage scheduled reports; `manage_sla` - Manage SLA. Possible values only for users of *User* and *Admin* user types: `invoke_execute_now` - allows to execute item checks for users that have only read permissions on host.
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"
                }
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: `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	
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"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "roleid": "3",
```

```
            "name": "Super admin role",
            "type": "3",
            "readonly": "1",
            "rules": {
                "ui": [
                    {
                        "name": "inventory.hosts",
                        "status": "1"
                    },
                    {
                        "name": "inventory.overview",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.dashboard",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.hosts",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.latest_data",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.maps",
                        "status": "1"
                    },
                    {
                        "name": "monitoring.problems",
                        "status": "1"
                    },
                    {
                        "name": "reports.availability_report",
                        "status": "1"
                    },
                    {
                        "name": "reports.top_triggers",
                        "status": "1"
                    },
                    {
                        "name": "services.services",
                        "status": "1"
                    },
                    {
                        "name": "services.sla_report",
                        "status": "1"
                    },
                    {
                        "name": "configuration.actions",
                        "status": "1"
                    },
                    {
                        "name": "configuration.discovery",
                        "status": "1"
                    },
                    {
                        "name": "configuration.host_groups",
                        "status": "1"
                    },
                    {
```

```
            "name": "configuration.hosts",
            "status": "1"
        },
        {
            "name": "configuration.maintenance",
            "status": "1"
        },
        {
            "name": "configuration.templates",
            "status": "1"
        },
        {
            "name": "configuration.template_groups",
            "status": "1"
        },
        {
            "name": "monitoring.discovery",
            "status": "1"
        },
        {
            "name": "reports.notifications",
            "status": "1"
        },
        {
            "name": "reports.scheduled_reports",
            "status": "1"
        },
        {
            "name": "services.actions",
            "status": "1"
        },
        {
            "name": "services.sla",
            "status": "1"
        },
        {
            "name": "administration.authentication",
            "status": "1"
        },
        {
            "name": "administration.general",
            "status": "1"
        },
        {
            "name": "administration.media_types",
            "status": "1"
        },
        {
            "name": "administration.proxies",
            "status": "1"
        },
        {
            "name": "administration.queue",
            "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": "configuration.event_correlation",
                "status": "1"
            },
            {
                "name": "reports.action_log",
                "status": "1"
            },
            {
                "name": "reports.audit",
                "status": "1"
            },
            {
                "name": "reports.system_info",
                "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 replace the current access rules assigned to 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"
                    }
                ]
            }
        }
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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"]
            }
        }
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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	*(readonly)* ID of the script.
name (required)	string	Name of the script.

Property	Type	Description
type (required)	integer	Script type. Possible values: 0 - Script; 1 - IPMI; 2 - SSH; 3 - Telnet; 5 - *(default)* Webhook.
command (required)	string	Command to run.
scope	integer	Script scope. Possible values: 1 - *default* action operation; 2 - manual host action; 4 - manual event action.
execute_on	integer	Where to run the script. Used if `type` is 0 (script). Possible values: 0 - run on Zabbix agent; 1 - run on Zabbix server; 2 - *(default)* run on Zabbix server (proxy).
menu_path	string	Folders separated by slash that form a menu like navigation in frontend when clicked on host or event. Used if `scope` is 2 or 4.
authtype	integer	Authentication method used for SSH script type. Used if `type` is 2. Possible values: 0 - password; 1 - public key.
username	string	User name used for authentication. Required if `type` is 2 or 3.
password	string	Password used for SSH scripts with password authentication and Telnet scripts. Used if `type` is 2 and `authtype` is 0 or `type` is 3.
publickey	string	Name of the public key file used for SSH scripts with public key authentication. Required if type is 2 and `authtype` is 1.
privatekey	string	Name of the private key file used for SSH scripts with public key authentication. Required if type is 2 and `authtype` is 1.
port	string	Port number used for SSH and Telnet scripts. Used if type is 2 or 3.
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. Used if `scope` is 2 or 4. Default: 0.
host_access	integer	Host permissions needed to run the script. Used if `scope` is 2 or 4. Possible values: 2 - *(default)* read; 3 - write.

Property	Type	Description
confirmation	string	Confirmation pop up text. The pop up will appear when trying to run the script from the Zabbix frontend. Used if `scope` is 2 or 4.
timeout	string	Webhook script execution timeout in seconds. Time suffixes are supported, e.g. 30s, 1m. Required if `type` is 5. Possible values: 1-60s Default value: 30s
parameters	array	Array of webhook input parameters. Used if `type` is 5.
description	string	Description of the script.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Webhook parameters

Parameters passed to webhook script when it is called have the following properties.

Property	Type	Description
name (required)	string	Parameter name.
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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "scriptids": [
            "4"
        ]
    },
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.

> **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 (required)	string	ID of the script to run.
hostid	string	ID of the host to run the script on.
eventid	string	ID of the event to run the script on.

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 are: `scriptid` and `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	

Parameter	Type	Description
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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",
            "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",
```

```
            "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",
            "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",
            "parameters": [
                {
                    "name": "token",
                    "value": "{$WEBHOOK.TOKEN}"
                },
                {
                    "name": "host",
                    "value": "{HOST.HOST}"
                },
                {
                    "name": "v",
                    "value": "2.2"
                }
            ]
        }
    ],
    "id": 1
}
```


See also

- Host
- Host group

Source

CScript::get() 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.

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"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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",
                "hostid": "10001"
            },
            {
                "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",
                "hostid": "10001"
            },
```

```
            {
                "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",
                "hostid": "10001"
            }
        ],
        "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",
                "hostid": "10001"
            },
            {
                "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",
                "hostid": "10001"
            },
            {
                "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",
                "hostid": "10001"
            }
        ]
    },
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the service.
algorithm (required)	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.
name (required)	string	Name of the service.
sortorder (required)	integer	Position of the service used for sorting. Possible values: 0-999.
weight	integer	Service weight. Possible values: 0-1000000. Default: 0.
propagation_rule	integer	Status propagation rule. Must be set together with `propagation_value`. 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`.
propagation_value	integer	Status propagation value. Must be set together with `propagation_rule`. Possible values for `propagation_rule` with values 0 and 3: 0. Possible values for `propagation_rule` with values 1 and 2: 1-5. Possible values for `propagation_rule` with value 4: -1 - OK; 0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster.

Property	Type	Description
status	integer	*(readonly)* 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.
description	string	Description of the service.
uuid	string	Universal unique identifier. For update operations this field is *readonly*.
created_at	integer	Unix timestamp when service was created.
readonly	boolean	*(readonly)* Access to the service. Possible values: 0 - Read-write; 1 - Read-only.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Status rule

The status rule object has the following properties.

Property	Type	Description
type (required)	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`.
limit_value (required)	integer	Limit value. Possible values: - for N and W: 1-100000; - for N%: 1-100.
limit_status (required)	integer	Limit status. Possible values: -1 - OK; 0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster.

Property	Type	Description
new_status (required)	integer	New status value. Possible values: 0 - Not classified; 1 - Information; 2 - Warning; 3 - Average; 4 - High; 5 - Disaster.

Service tag

The service tag object has the following properties.

Property	Type	Description
tag (required)	string	Service tag name.
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 (required)	string	Problem tag name.
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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.

Parameter	Type	Description
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 of objects	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 of objects	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`.
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`.

Parameter	Type	Description
selectStatusTimeline	object/array of objects	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 are: `serviceid`, `name`, `status`, `sortorder` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

Parameter	Type	Description
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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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_GB`.
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.
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.

Property	Type	Description
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.
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.

Property	Type	Description
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.
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.

Property	Type	Description
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.
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.

Property	Type	Description
geomaps_tile_provider	string	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 if `geomaps_tile_provider` is set to empty string.
geomaps_max_zoom	integer	Geomap max zoom level if `geomaps_tile_provider` is set to empty string. Max zoom must be in the range between 0 and 30.
geomaps_attribution	string	Geomap attribution text 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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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_GB",
        "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 array with the names of updated parameters.

Examples

Request:


```
{
    "jsonrpc": "2.0",
    "method": "settings.update",
    "params": {
        "login_attempts": "1",
        "login_block": "1m"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 availability information as Service Level Indicator (SLI)
- 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	*(readonly)* ID of the SLA.
name (required)	string	Name of the SLA.
period (required)	integer	Reporting period of the SLA. Possible values: 0 - daily; 1 - weekly; 2 - monthly; 3 - quarterly; 4 - annually.
slo (required)	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).
effective_date	integer	Effective date of the SLA. Possible values: date timestamp in UTC.
timezone (required)	string	Reporting time zone, for example: `Europe/London`, `UTC`. For the full list of supported time zones please refer to PHP documentation.
status	integer	Status of the SLA. Possible values: 0 - *(default)* disabled SLA; 1 - enabled SLA.
description	string	Description of the SLA.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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 (required)	integer	Starting time of the recurrent weekly period of time (inclusive). Possible values: number of seconds (counting from Sunday).

Property	Type	Description
period_to (required)	integer	Ending time of the recurrent weekly period of time (exclusive). Possible values: number of seconds (counting from Sunday).

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.

Property	Type	Description
name (required)	string	Name of the excluded downtime.
period_from (required)	integer	Starting time of the excluded downtime (inclusive). Possible values: timestamp.
period_to (required)	integer	Ending time of the excluded downtime (exclusive). Possible values: timestamp.

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 (required)	string	SLA service tag name.
operator	integer	SLA service tag operator. Possible values: 0 - *(default)* equals; 2 - like
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 (required)	array	SLA service tags to be created for the SLA. At least one service tag must be specified.
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"
                }
            ]
        }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: `slaid`, `name`, `period`, `slo`, `effective_date`, `timezone`, `status` and `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
    },
```

```
    "auth": "85dd04b94cbfad794616eb923be13c71",
    "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.

> **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 (required)	string	IDs of services to return availability information for.
period_from	integer	Starting date (inclusive) to report the SLI for. Possible values: timestamp.
period_to	integer	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 demonstrates the arrangement of returned *period slices* based on combinations of parameters.

Parameters			Description
period_from	**period_to**	**periods**	
-	-	-	The last 20 periods (including the current one) but not past the first available period based on the effective date of the SLA.
-	-	specified	The last periods specified by the **periods** parameter.
-	specified	-	The last 20 periods before the **specified date**, but not past the first available period based on the effective date of the SLA.
-	specified	specified	The last periods specified by the **periods** parameter before the **specified date**.
specified	-	-	The first 20 periods (including the current one) but not past the current one.
specified	-	specified	The first periods specified by the **periods** parameter starting with the **specified date**.
specified	specified	-	Periods within the specified date range, but no more than 100 and not past the first available period based on the effective date of the SLA.
specified	specified	specified	Periods within the specified date range, but no more than the specified number of periods and not past the first available period based on the effective date of the SLA.

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.
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 on services with IDs ”50, 60 and 70” linked to an SLA with ID of ”5” 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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": []
                },
                {
                    "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. At least one service tag must be specified.
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"
                }
            ]
        }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": []
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the task.
type (required)	integer	Type of the task. Possible values: 1 - Diagnostic information; 2 - Refresh proxy configuration; 6 - Execute now.
status	integer	*(readonly)* Status of the task. Possible values: 1 - new task; 2 - task in progress; 3 - task is completed; 4 - task is expired.
clock	timestamp	*(readonly)* Time when the task was created.
ttl	integer	*(readonly)* The time in seconds after which task expires.
proxy_hostid	string	ID of the proxy about which diagnostic information statistic is collected. Ignored for 'Execute now' tasks.
request (required)	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	Type	Description
result	object	*(readonly)* Result object of the diagnostic information task. May contain NULL if result is not yet ready. Result object is described in detail below.

'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.

Diagnostic type	Available fields	Description
	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
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	*(readonly)* Status of the task result. Possible values: -1 - error occurred during performing task; 0 - task result is created.
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 the following parameters.

Parameter	Type	Description
type (required)	integer	Task type. Possible values: 1 - Diagnostic information; 2 - Refresh proxy configuration; 6 - Execute now.
request (required)	object	Task request object according to the task type. Correct format of request object is described in Task object section.
proxy_hostid	integer	Proxy about which Diagnostic information task will collect data. Ignored for 'Execute now' tasks.

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 ruls 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"
            }
        }
    ],
    "auth": "700ca65537074ec963db7efabda78259",
    "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"]
            }
        }
    ],
    "auth": "700ca65537074ec963db7efabda78259",
    "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
        }
    ],
    "auth": "700ca65537074ec963db7efabda78259",
    "id": 2
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "taskids": [
            "3"
        ]
    },
    "id": 2
}
```

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.

Parameter	Type	Description
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the template.
host (required)	string	Technical name of the template.
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.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Template tag

The template tag object has the following properties.

Property	Type	Description
tag (required)	string	Template tag name.
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 (required)	object/array	Template groups to add the template to. The template groups must have the `groupid` property defined.
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}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 are parent to the given templates.
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.

Parameter	Type	Description
tags	array/object	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 template is a child, in the `templates` property. Supports `count`.
selectParentTemplates	query	Return templates to which the template is a parent, 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`.
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`.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties. Possible values are: `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"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "templateid": "10001",
            "templategroups": [
                {
                    "groupid": "10",
                    "name": "Templates/Operating systems",
                    "uuid": "846977d1dfed4968bc5f8bdb363285bc"
                }
            ]
        }
    ],
    "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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	object/array	Templates to be updated. The templates must have the `templateid` property defined.
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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

Request:

```
{
    "jsonrpc": "2.0",
```

```
    "method": "template.massadd",
    "params": {
        "templates": [
            {
                "templateid": "10073"
            }
        ],
        "templates_link": [
            {
                "templateid": "10106"
            },
            {
                "templateid": "10104"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	string/array	IDs of the templates to be updated.
groupids	string/array	Template groups to remove the given templates from.
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).

Parameter	Type	Description
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}
```

Unlinking templates from a host

Unlink templates "10106", "10104" from template "10085".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": "10085",
        "templateids_link": [
            "10106",
            "10104"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	object/array	Templates to be updated. The templates must have the `templateid` property defined.
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 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

Replacing template groups

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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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}"
            }
```

```
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "hostids": [
            "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	*(readonly)* ID of the template dashboard.
name (required)	string	Name of the template dashboard.
templateid (required)	string	ID of the template the dashboard belongs to.
display_period	integer	Default page display period (in seconds). Possible values: 10, 30, 60, 120, 600, 1800, 3600. Default: 30.

Property	Type	Description
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.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Template dashboard page

The template dashboard page object has the following properties.

Property	Type	Description
dashboard_pageid	string	*(readonly)* ID of the dashboard page.
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	*(readonly)* ID of the dashboard widget.
type (required)	string	Type of the dashboard widget. Possible values: clock - Clock; graph - Graph (classic); graphprototype - Graph prototype; item - Item value; plaintext - Plain text; url - URL;
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;

Property	Type	Description
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 (required)	integer	Type of the widget field. Possible values: 0 - Integer; 1 - String; 4 - Item; 5 - Item prototype; 6 - Graph; 7 - Graph prototype.
name	string	Widget field name.
value (required)	mixed	Widget field value depending of type.

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 (required)	array	Template dashboard pages to be created for the dashboard. Dashboard pages will be ordered in the same order as specified. At least one dashboard page object is required for `pages` property.

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"
                            }
                        ]
                    }
                ]

            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 are: dashboardid and name.
countOutput	boolean	These parameters being common for all get methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	

Parameter	Type	Description
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
```

```
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": {
        "dashboardids": [
            "2"
        ]
    },
    "id": 2
}
```

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	*(readonly)* ID of the template group.
name (required)	string	Name of the template group.

Property	Type	Description
uuid	string	Universal unique identifier, used for linking imported template groups to already existing ones. Auto-generated, if not given. For update operations this field is *readonly*.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.
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.

Parameter	Type	Description
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.
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 are: `groupid`, `name`.

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	
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"
            ]
        }
    },
    "auth": "6f38cddc44cfbb6c1bd186f9a220b5a0",
    "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 (required)	object/array	Template groups to be updated. The template groups must have the `groupid` property defined.
templates	object/array	Templates to add to all template groups. The templates must have the `templateid` property defined.

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"
            }
        ]
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
    "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 (required)	string/array	IDs of the template groups to be updated.
templateids	string/array	Templates to remove from all template groups.

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": "hostgroup.massremove",
    "params": {
        "groupids": [
            "5",
            "6"
        ],
        "hostids": [
            "30050",
            "30001"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	object/array	Template groups to be updated. The template groups must have the `groupid` property defined.
templates (required)	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.

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"
            }
        ]
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
    "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 (required)	object/array	Template groups to propagate. The template groups must have the `groupid` property defined.
permissions (required)	boolean	Set `true` if need to propagate permissions.

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
    },
    "auth": "f223adf833b2bf2ff38574a67bba6372",
    "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"
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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	*(readonly)* ID of the token.
name (required)	string	Name of the token.
description	text	Description of the token.
userid	string	*(readonly for update)* A user the token has been assigned to. *Default: current user.*
lastaccess	timestamp	*(readonly)* Most recent date and time the token was authenticated. Zero if the token has never been authenticated.
status	integer	Token status. Possible values: 0 - *(default)* enabled token; 1 - disabled token.
expires_at	timestamp	Token expiration date and time. Zero for never-expiring tokens.
created_at	timestamp	*(readonly)* Token creation date and time.
creator_userid	string	*(readonly)* The creator user of the token.

Note that for some methods (update, delete) the required/optional parameter combination is different.

token.create

Description

`object token.create(object/array tokens)`

This method allows to create new tokens.

> **Note:**
> Only *Super admin* user type is allowed to manage tokens for other users.

> **Note:**
> A token created by this method 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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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:**
> Only *Super admin* user type is allowed 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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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:**
> Only *Super admin* user type is allowed to manage tokens for other users.

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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "tokenid": "1",
            "token": "bbcfce79a2d95037502f7e9a534906d3466c9a1484beb6ea0f4e7be28e8b8ce2"
        },
        {
            "tokenid": "2",
            "token": "fa1258a83d518eabd87698a96bd7f07e5a6ae8aeb8463cae33d50b91dd21bd6d"
        }
    ],
    "id": 0
}
```

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 are: `tokenid`, `name`, `lastaccess`, `status`, `expires_at` and `created_at`.
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

Retrieve a token

Retrieve all data for the token with ID "2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "token.get",
    "params": {
        "output": "extend",
        "tokenids": "2"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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:**
> Only *Super admin* user type is allowed 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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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. E. g. 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. E. g. 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.

Property	Type	Description
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 fields to output.

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the trigger.
description (required)	string	Name of the trigger.
expression (required)	string	Reduced trigger expression.
event_name	string	Event name generated by the trigger.
opdata	string	Operational data.
comments	string	Additional description of the trigger.
error	string	*(readonly)* Error text if there have been any problems when updating the state of the trigger.
flags	integer	*(readonly)* Origin of the trigger. Possible values are: 0 - *(default)* a plain trigger; 4 - a discovered trigger.
lastchange	timestamp	*(readonly)* Time when the trigger last changed its state.

Property	Type	Description
priority	integer	Severity of the trigger. Possible values are: 0 - *(default)* not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
state	integer	*(readonly)* State of the trigger. Possible values: 0 - *(default)* trigger state is up to date; 1 - current trigger state is unknown.
status	integer	Whether the trigger is enabled or disabled. Possible values are: 0 - *(default)* enabled; 1 - disabled.
templateid	string	*(readonly)* ID of the parent template trigger.
type	integer	Whether the trigger can generate multiple problem events. Possible values are: 0 - *(default)* do not generate multiple events; 1 - generate multiple events.
url	string	URL associated with the trigger.
value	integer	*(readonly)* Whether the trigger is in OK or problem state. Possible values are: 0 - *(default)* OK; 1 - problem.
recovery_mode	integer	OK event generation mode. Possible values are: 0 - *(default)* Expression; 1 - Recovery expression; 2 - None.
recovery_expression	string	Reduced trigger recovery expression.
correlation_mode	integer	OK event closes. Possible values are: 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 are: 0 - *(default)* No; 1 - Yes.
uuid	string	Universal unique identifier, used for linking imported triggers to already existing ones. Used only for triggers on templates. Auto-generated, if not given.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Trigger tag

The trigger tag object has the following properties.

Property	Type	Description
tag (required)	string	Trigger tag name.
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": ""
                }
            ]
        }
    ],
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 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.

Parameter	Type	Description
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.
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.

Parameter	Type	Description
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`.
sortfield	string/array	Sort the result by the given properties. Possible values are: `triggerid`, `description`, `status`, `priority`, `lastchange` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": "",
            "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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": ""
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the trigger prototype.
description (required)	string	Name of the trigger prototype.
expression (required)	string	Reduced trigger expression.
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	*(readonly)* ID of the parent template trigger prototype.

Property	Type	Description
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.
recovery_mode	integer	OK event generation mode. Possible values are: 0 - *(default)* Expression; 1 - Recovery expression; 2 - None.
recovery_expression	string	Reduced trigger recovery expression.
correlation_mode	integer	OK event closes. Possible values are: 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 are: 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. Used only for trigger prototypes on templates. Auto-generated, if not given.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Trigger prototype tag

The trigger prototype tag object has the following properties.

Property	Type	Description
tag (required)	string	Trigger prototype tag name.
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}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.
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 are: `triggerid`, `description`, `status` and `priority`.

Parameter	Type	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	
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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",
            "priority": "2",
            "comments": "",
            "templateid": "0",
            "type": "0",
            "flags": "2",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0",
```

```
            "opdata": "",
            "discover": "0",
            "functions": [
                {
                    "functionid": "12598",
                    "itemid": "22454",
                    "triggerid": "13272",
                    "parameter": "$",
                    "function": "last"
                }
            ]
        },
        {
            "triggerid": "13266",
            "expression": "{13500}<20",
            "description": "Free disk space is less than 20% on volume {#FSNAME}",
            "url": "",
            "status": "0",
            "priority": "2",
            "comments": "",
            "templateid": "0",
            "type": "0",
            "flags": "2",
            "recovery_mode": "0",
            "recovery_expression": "",
            "correlation_mode": "0",
            "correlation_tag": "",
            "manual_close": "0",
            "opdata": "",
            "discover": "0",
            "functions": [
                {
                    "functionid": "13500",
                    "itemid": "22686",
                    "triggerid": "13266",
                    "parameter": "$",
                    "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"
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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}"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the user.
username *(required)*	string	User name.
roleid *(required)*	string	Role ID of the user.
attempt_clock	timestamp	*(readonly)* Time of the last unsuccessful login attempt.
attempt_failed	integer	*(readonly)* Recent failed login attempt count.
attempt_ip	string	*(readonly)* IP address from where the last unsuccessful login attempt came from.
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_GB`. Default: `default` - system default.
name	string	Name of the user.
refresh	string	Automatic refresh period. Accepts seconds and time unit with suffix. Default: 30s.

Property	Type	Description
rows_per_page	integer	Amount of object rows to show per page. Default: 50.
surname	string	Surname of the user.
theme	string	User's theme. Possible values: `default` - *(default)* system default; `blue-theme` - Blue; `dark-theme` - Dark.
url	string	URL of the page to redirect the user to after logging in.
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.
alias (deprecated)	string	This property is deprecated, please use `username` instead. User alias.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Media

The media object has the following properties.

Property	Type	Description
mediatypeid (required)	string	ID of the media type used by the media.
sendto (required)	string/array	Address, user name or other identifier of the recipient. If type of Media type is e-mail, values are represented as array. For other types of Media types, value is represented as a string.
active	integer	Whether the media is enabled. Possible values: 0 - *(default)* enabled; 1 - disabled.
severity	integer	Trigger severities to send notifications about. Severities are stored in binary form with each bit representing the corresponding severity. For example, 12 equals 1100 in binary and means, that notifications will be sent from triggers with severities warning and average. Refer to the trigger object page for a list of supported trigger severities. 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 user session.

> **Attention:**
> Calling the **user.checkAuthentication** method with the parameter `sessionid` set prolongs 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	Used if `sessionid` is set. User session ID.
token	string	**Required** if `token` is not set. User API token. **Required** if `sessionid` is not set.

Return values

`(object)` Returns an object containing information about user.

Examples

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",
        "type": 3,
        "sessionid": "673b8ba11562a35da902c66cf5c23fa2",
```

```
        "debug_mode": 0,
        "userip": "127.0.0.1",
        "gui_access": 0,
        "userdirectoryid": 0
    },
    "id": 1
}
```

Note:
Response is similar to User.login call response with "userData" parameter set to true (the difference is that user data is retrieved by session id and not by username / password).

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
passwd *(required)*	string	User's password. Can be omitted if user is added only to groups that have LDAP access.
usrgrps (required)	array	User groups to add the user to. The user groups must have the `usrgrpid` property defined.
medias	array	User media to be created.
user_medias (deprecated)	array	This parameter is deprecated, please use `medias` instead. 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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.

Parameter	Type	Description
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 are: `userid` and `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	
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "userid": "1",
            "username": "Admin",
            "name": "Zabbix",
```

```
            "surname": "Administrator",
            "url": "",
            "autologin": "1",
            "autologout": "0",
            "lang": "en_GB",
            "refresh": "0s",
            "theme": "default",
            "attempt_failed": "0",
            "attempt_ip": "",
            "attempt_clock": "0",
            "rows_per_page": "50",
            "timezone": "default",
            "roleid": "3"
        },
        {
            "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"
        },
        {
            "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"
        }
    ],
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 (required)	string	User password.
username (required)	string	User name.

Parameter	Type	Description
userData	flag	Return information about the authenticated user.
user (deprecated)	string	This parameter is deprecated, please use `username` instead. User name.

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
debug_mode	boolean	Whether debug mode is enabled for the user.
gui_access	integer	User's authentication method to the frontend. Refer to the `gui_access` property of the user group object for a list of possible values.
sessionid	string	Authentication token, which must be used in the following API requests.
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",
        "type": 3,
        "debug_mode": 0,
        "userip": "127.0.0.1",
        "gui_access": "0",
        "userdirectoryid": 0,
        "sessionid": "5b56eee8be445e98f0bd42b435736e42"
    },
    "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,
    "auth": "16a46baf181ef9602e1687f3110abf8a"
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

See also

- user.login

Source

CUser::login() 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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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
passwd	string	User's password. Can be empty string if user belongs to or is moved only to groups that have LDAP access.
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.
user_medias (deprecated)	array	This parameter is deprecated, please use `medias` instead. 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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the user directory.
name (required)	string	Unique name of the user directory.
host (required)	string	LDAP server host name, IP or URI. URI should contain schema, host and port (optional).
port (required)	integer	LDAP server port.
base_dn (required)	string	LDAP base distinguished name string.
search_attribute (required)	string	LDAP attribute name to identify user by username in Zabbix database.
bind_dn	string	LDAP bind distinguished name string. Can be empty for anonymous binding.
bind_password	string	LDAP bind password. Can be empty for anonymous binding. *Available only for userdirectory.update and userdirectory.create requests.*
description	string	User directory description.
search_filter	string	LDAP custom filter string when authenticating user in LDAP. Default value: (%{attr}=%{user})
start_tls	integer	LDAP startTLS option. It cannot be used with `ldaps://` protocol hosts. Possible values: 0 - *(default)* disabled; 1 - enabled.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Filter `search_filter` supported placeholders:

Value	Description
%{attr}	Search attribute name *(uid, sAMAccountName)*.
%{user}	Username value.

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.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.create",
    "params": {
        "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 directory with the given IDs.
selectUsrgrps	query	Return a usrgrps property with user groups associated with user directory. Supports count.
sortfield	string/array	Sort the result by the given properties. Possible values are: name, host.

Parameter	Type	Description
filter	object	Return only those results that exactly match the given filter. Possible values are: `userdirectoryid`, `host`, `name`.
search	object	Return results that match the given wildcard search (case-insensitive). Possible values are: `base_dn`, `bind_dn`, `description`, `host`, `name`, `search_attribute`, `search_filter`.
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 property with count of user groups where user directory is used.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.get",
    "params": {
        "output": "extend",
        "selectUsrgrps": "count"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "userdirectoryid": "2",
            "name": "API user directory #1",
            "description": "",
            "host": "127.0.0.1",
            "port": "389",
            "base_dn": "ou=Users,dc=example,dc=org",
            "bind_dn": "cn=ldap_search,dc=example,dc=org",
            "search_attribute": "uid",
            "start_tls": "0",
            "search_filter": "",
            "usrgrps": "5"
        }
    ],
    "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 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

Test user directory for user ”user1”.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.test",
    "params": {
        "userdirectoryid": "2",
        "host": "127.0.0.1",
        "port": "3389",
        "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"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": true,
    "id": 1
}
```

Test user directory

Test non existing user "user2".

Request:

```
{
    "jsonrpc": "2.0",
    "method": "userdirectory.test",
    "params": {
        "userdirectoryid": "2",
        "host": "127.0.0.1",
        "port": "3389",
        "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"
    },
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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
}
```

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": {
        "userdirectory": "2",
        "bind_password": "newldappassword"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the user group.
name (required)	string	Name of the user group.
debug_mode	integer	Whether debug mode is enabled or disabled. Possible values are: 0 - *(default)* disabled; 1 - enabled.

Property	Type	Description
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. Possible values are: 0 - *(default)* enabled; 1 - disabled.
userdirectoryid	string	Authentication user directory when `gui_access` set to LDAP or System default.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Permission

The permission object has the following properties.

Property	Type	Description
id (required)	string	ID of the host group or template group to add permission to.
permission (required)	integer	Access level to the host group or template group. Possible values: 0 - access denied; 2 - read-only access; 3 - read-write access.

Tag-based permission

The tag-based permission object has the following properties.

Property	Type	Description
groupid (required)	string	ID of the host group to add permission to.
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"
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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.

> **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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 host group 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 template group 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.
sortfield	string/array	Sort the result by the given properties. Possible values are: `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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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 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 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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the global macro.
macro (required)	string	Macro string.
value (required)	string	Value of the macro.
type	integer	Type of macro. Possible values: 0 - *(default)* Text macro; 1 - Secret macro; 2 - Vault secret.
description	string	Description of the macro.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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	*(readonly)* ID of the host macro.
hostid (required)	string	ID of the host that the macro belongs to.
macro (required)	string	Macro string.
value (required)	string	Value of the macro.
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.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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 value: `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	

Parameter	Type	Description
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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
```

```
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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	*(readonly)* ID of the value map.
hostid (required)	id	Value map host or template ID.

Property	Type	Description
name (required)	string	Name of the value map.
mappings (required)	array	Value mappings for current value map. The mapping object is described in detail below.
uuid	string	Universal unique identifier, used for linking imported value maps to already existing ones. Used only for value maps on templates. Auto-generated, if not given.

Note that for some methods (update, delete) the required/optional parameter combination is different.

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. For type equal 0,1,2,3,4 `value` field cannot be empty, for type 5 `value` field should be empty. Possible values: 0 - (default) exact match ; 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), allow to define multiple ranges separated by comma character[1]; 4 - mapping will be applied if value match regular expression[2]; 5 - default value, mapping will be applied if no other match were found.
value (required)	string	Original value. Is not required for mapping of type "default".
newvalue (required)	string	Value to which the original value is mapped to.

[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"
            }
        ]
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "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"
    ],
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "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 are: `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	

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 value maps

Retrieve all configured value maps.

Request:

```
{
    "jsonrpc": "2.0",
    "method": "valuemap.get",
    "params": {
        "output": "extend"
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "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"]
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "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"
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "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"
            }
        ]
    },
    "auth": "57562fd409b3b3b9a4d916d45207bbcb",
    "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	*(readonly)* ID of the web scenario.
hostid (required)	string	ID of the host that the web scenario belongs to.
name (required)	string	Name of the web scenario.
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 and user macro. Default: 1m.
headers	array of HTTP fields	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.

Property	Type	Description
nextcheck	timestamp	*(readonly)* Time of the next web scenario execution.
retries	integer	Number of times a web scenario will try to execute each step before failing. Default: 1.
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 are: 0 - *(default)* enabled; 1 - disabled.
templateid	string	*(readonly)* ID of the parent template web scenario.
variables	array of HTTP fields	Web scenario variables.
verify_host	integer	Whether to verify that the host name specified in the SSL certificate matches the one used in the scenario. Possible values are: 0 - *(default)* skip host verification; 1 - verify host.
verify_peer	integer	Whether to verify the SSL certificate of the web server. Possible values are: 0 - *(default)* skip peer verification; 1 - verify peer.
uuid	string	Global unique identifier, used for linking imported web scenarios to already existing ones. Used only for web scenarios on templates.

Note that for some methods (update, delete) the required/optional parameter combination is different.

Web scenario tag

The web scenario tag object has the following properties.

Property	Type	Description
tag (required)	string	Web scenario tag name.
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
httpstepid	string	*(readonly)* ID of the scenario step.
name (required)	string	Name of the scenario step.
no (required)	integer	Sequence number of the step in a web scenario.
url (required)	string	URL to be checked.
follow_redirects	integer	Whether to follow HTTP redirects. Possible values are: 0 - don't follow redirects; 1 - *(default)* follow redirects.

Property	Type	Description
headers	array of HTTP fields	HTTP headers that will be sent when performing a request. Scenario step headers will overwrite headers specified for the web scenario.
httptestid	string	*(readonly)* ID of the web scenario that the step belongs to.
posts	string array of HTTP fields	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 are: 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 and user macro. Default: 15s. Maximum: 1h. Minimum: 1s.
variables	array of HTTP fields	Scenario step variables.
query_fields	array of HTTP fields	Query fields - array of HTTP fields that will be added to URL when performing a request

HTTP field

The HTTP field object defines a name and value that is used to specify variable, HTTP header, POST form field data of query field data. It has the following properties.

Property	Type	Description
name (required)	string	Name of header / variable / POST or GET field.
value (required)	string	Value of header / variable / POST or GET field.

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 (required)	array	Web scenario steps.
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
            }
        ]
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "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"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "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.

Parameter	Type	Description
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 the web scenario tags in tags property.
sortfield	string/array	Sort the result by the given properties. Possible values are: `httptestid` and `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"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
    "jsonrpc": "2.0",
    "result": [
        {
            "httptestid": "9",
            "name": "Homepage check",
            "nextcheck": "0",
            "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
    },
    "auth": "700ca65537074ec963db7efabda78259",
    "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` and `false` otherwise.
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 labels

Some of the objects properties are marked with short labels to describe their behavior. The following labels are used:

- *readonly* - the value of the property is set automatically and cannot be defined or changed by the client;
- *constant* - the value of the property can be set when creating an object, but cannot be changed after.

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 array, where the keys are property names, and the values are either a single value or an array of values to match against. Doesn't work for `text` fields.
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 wildcard search (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. Works only for `string` and `text` fields.
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`.
searchWildcardsEnabled	boolean	If set to `true` enables the use of "*" as a wildcard character in the `search` parameter. Default: `false`.

Parameter	Type	Description
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 are: `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
    },
    "auth": "766b71ee543230a1182ca5c44d353e36",
    "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
    },
    "auth": "766b71ee543230a1182ca5c44d353e36",
    "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
    },
    "auth": "766b71ee543230a1182ca5c44d353e36",
    "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
    },
    "auth": "766b71ee543230a1182ca5c44d353e36",
    "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 6.0 to 6.2

Backward incompatible changes authentication

Changes:
ZBXNEXT-2289 `authentication.get`, `authentication.create`, `authentication.update`: removed properties `ldap_host`, `ldap_port`, `ldap_base_dn`, `ldap_search_attribute`, `ldap_bind_dn`, `ldap_bind_password`.

configuration

Changes:
ZBXNEXT-2592 `configuration.export`: parameter `options` no longer supports `groups` as objects to be exported; instead, new options introduced: `host_groups` and `template_groups`.
ZBXNEXT-2592 `configuration.import`, `configuration.importcompare`: parameter `rules` no longer supports `groups`; instead, new rules introduced: `host_groups` and `template_groups`. Note that the parameter `source` still accepts `groups` as part of the configuration data to be imported/compared.

hostgroup

Changes:
ZBXNEXT-2592 `hostgroup` object does not have property `internal` anymore.
ZBXNEXT-2592 `hostgroup.get`: removed options `templated_hosts`, `with_hosts_and_templates`, `selectTemplates`.
ZBXNEXT-2592 `hostgroup.massadd`, `hostgroup.massupdate`: does not accept `templates` parameter anymore.
ZBXNEXT-2592 `hostgroup.massremove`: does not accept `templateids` parameter anymore.

template

Changes:
ZBXNEXT-2592 `template.create`, `template.massadd`, `template.massupdate`, `template.update`: parameter `groups` now accepts only template groups and must contain template group `groupid`.
ZBXNEXT-2592 `template.massremove`: parameter `groupids` is now template groups `groupids`.

trigger

Changes:
ZBX-20613 `trigger.adddependencies`, `trigger.deletedependencies`: support for the methods dropped.

Other changes and bug fixes auditlog

ZBXNEXT-2592 added new `resourcetype` *(50 - Template group)*.
ZBXNEXT-2289 added new `resourcetype` *(49 - LDAP user directory)*.
ZBXNEXT-1580 added new `action` *(11 - Config refresh)*.

authentication

ZBXNEXT-2289 `authentication.get`, `authentication.create`, `authentication.update`: added property `ldap_userdirectoryid`

event

ZBXNEXT-721 `event.acknowledge`: added support of new acknowledge actions 32 - suppress and 64 - unsuppress.
ZBXNEXT-721 `event.acknowledge`: added new property `suppress_until` to specify time for suppress action.
ZBXNEXT-721 `event.get`: added new parameter `userid` to `suppression_data` property.

graph

Changes:
ZBXNEXT-2592 `graph.get`: deprecated option `selectGroups`, new options `selectHostGroups` (returns property `hostgroups`) and `selectTemplateGroups` (returns property `templategroups`).

graphprototype

Changes:
ZBXNEXT-2592 `graphprototype.get`: deprecated option `selectGroups`, new options `selectHostGroups` (returns property `hostgroups`) and `selectTemplateGroups` (returns property `templategroups`).

host

Changes:
ZBXNEXT-2592 `host.get`: deprecated option `selectGroups`, new option `selectHostGroups` (returns property `hostgroups`).
ZBXNEXT-5517 `host.update`: added new field `automatic` to `macros` property with possible value "0" to change state of existing discovered host macro to manually user created macro.
ZBXNEXT-5517 `host.update`: added possibility to add user macros to discovered hosts.
ZBXNEXT-5517 `host.get`: added new field `automatic` to `macros` property that determines if macro on discovered host was created by discovery rule or manually created by user *(0 - user created macro, 1 - macro created by discovery rule)*.
ZBXNEXT-7591 `host.get`: added new field `automatic` to `tags` property that determines if tag on discovered host was created by discovery rule or manually created by user *(0 - user created tag, 1 - tag created by discovery rule)*.
ZBXNEXT-5088 `host.get`: added new property for host active interface availability status `active_available` *(0 - unknown, 1 - available, 2 - not available)*.
ZBXNEXT-7523 `host.get`: option `selectParentTemplates` returns a new property `link_type` *(0 - manually linked, 1 - linked by LLD)* and allows only to select template related fields "templateid", "host", "name", "description", "uuid" and "link_type". As well as "extend" or "count".
ZBXNEXT-7523 `host.massremove`: properties `templateids` and `templateids_clear` allow to remove only templates that are manually linked. Automatically linked templates by LLD are skipped.
ZBXNEXT-7523 `host.massupdate`, `host.update`: properties `templates` and `templates_clear` are now allowed for discovered hosts. Only manually linked templates can be removed.

hostgroup

Changes:
ZBXNEXT-2592 `hostgroup.get`: deprecated options `monitored_hosts`, `real_hosts`, new options `with_hosts`, `with_monitored_host`
ZBXNEXT-2592 `hostgroup.propagate`: new method.

maintenance

Changes:
ZBXNEXT-2592 `maintenance.get`: deprecated option `selectGroups`, new option `selectHostGroups` (returns property `hostgroups`).

problem

Changes:
ZBXNEXT-721 `problem.get`: added new parameter `userid` to `suppression_data` property.

role

Changes:
ZBXNEXT-4768 `rules` -> `actions` property `name` accepts new value "invoke_execute_now" and property `status` has possible values: 0 - *(default)* user cannot execute item check if user has only read permissions to host, 1 - user may execute item check even though user has only read permissions to host. This value is ignored for super admins.
ZBXNEXT-721 `rules` -> `actions` property `name` accepts new value "suppress_problems".

script

Changes:
ZBXNEXT-2592 `script.get`: deprecated option `selectGroups`, new option `selectHostGroups` (returns property `hostgroups`).

settings

Changes:
ZBXNEXT-7402 `settings.get`: new property `vault_provider` *(0 - HashiCorp Vault, 1 - CyberArk Vault)*.

task

Changes:
ZBXNEXT-1580 added new `type` *(2 - Refresh proxy configuration)* and new field `proxy_hostids` in `request` property.
ZBXNEXT-4768 `task.create`: tasks with type "6" *(Execute now)* now accepts dependent items, but only if top level master item is of allowed type: *(0 - Zabbix agent, 3 - Simple check, 5 - Zabbix internal, 10 - External check, 11 - Database monitor, 12 - IPMI*

agent, 13 - SSH agent, 14 - TELNET agent, 15 - Calculated, 16 - JMX agent, 19 - HTTP agent, 20 - SNMP agent, 21 - Script).

template

Changes:
ZBXNEXT-2592 `template.get`: deprecated option `selectGroups`, new option, `selectTemplateGroups` (returns property `templategroups`).
ZBXNEXT-7523 `template.get`: option `selectParentTemplates` returns a new property "link_type" (*0 - manually linked, 1 - linked by LLD*) and allows only to select template related fields "templateid", "host", "name", "description" and "uuid". As well as "extend" or "count".

templategroup

Changes:
ZBXNEXT-2592 added new templategroup API introducing new methods: `templategroup.create`, `templategroup.delete`, `templategroup.get`, `templategroup.massadd`, `templategroup.massremove`, `templategroup.massupdate`, `templategroup.pro` `templategroup.update`.

trigger

Changes:
ZBXNEXT-2592 `trigger.get`: deprecated option `selectGroups`, new options `selectHostGroups` (returns property `hostgroups`) and `selectTemplateGroups` (returns property `templategroups`).

triggerprototype

Changes:
ZBXNEXT-2592 `triggerprototype.get`: deprecated option `selectGroups`, new options `selectHostGroups` (returns property `hostgroups`) and `selectTemplateGroups` (returns property `templategroups`).

user

Changes:
ZBXNEXT-2289 `user.checkAuthentication`: returns a new property `userdirectoryid`.
ZBXNEXT-2289 `user.login`: option `userData` returns a new property `userdirectoryid`.

userdirectory

Changes:
ZBXNEXT-2289 added new `userdirectory` API introducing new methods `userdirectory.get`, `userdirectory.create`, `userdirectory.update`, `userdirectory.delete`, `userdirectory.test`.

usergroup

Changes:
ZBXNEXT-2592 `usergroup.get`: deprecated option `selectRights`, new options `selectHostGroupRights`, `selectTemplateGroupRig`
ZBXNEXT-2592 `usergroup.create`, `usergroup.update`: deprecated option `rights`, new options `hostgroup_rights`, `templategroup_rights`.
ZBXNEXT-2289 `usergroup.create`, `usergroup.update`, `usergroup.get`: added property `userdirectoryid`.

usermacro

Changes:
ZBXNEXT-2592 `usermacro.get`: deprecated option `selectGroups`, new options `selectHostGroups` (returns property `hostgroups`) and `selectTemplateGroups` (returns property `templategroups`).
ZBXNEXT-5517 `usermacro.update`: added new field `automatic` to `macros` property with possible value "0" to change state of existing discovered host macro to manually user created macro.
ZBXNEXT-5517 `usermacro.update`: added possibility to add user macros to discovered hosts.
ZBXNEXT-5517 `usermacro.get`: added new field `automatic` to `macros` property that determines if macro on discovered host was created by discovery rule or manually created by user *(0 - user created macro, 1 - macro created by discovery rule)*.

6.2.9 script

Changes:
ZBX-19466 changed validation of `script` object to be unique by combination of 2 properties: `name` and `menu_path`.

6.2.8 user

Changes:
ZBXNEXT-8012 `user.checkAuthentication`: added new parameter `token`.

6.2.7 configuration

Changes:
ZBXNEXT-7951 `configuration.import`, `configuration.importcompare`: option `deleteMissing: true` for `templateLinkage` will now unlink missing templates (instead of unlink and clear).

discoveryrule

Changes:
ZBXNEXT-7951 `discoveryrule.update`: the parameter `uuid` can now be updated.

graph

Changes:
ZBXNEXT-7951 `graph.update`: the parameter `uuid` can now be updated.

graphprototype

Changes:
ZBXNEXT-7951 `graphprototype.update`: the parameter `uuid` can now be updated.

hostgroup

Changes:
ZBXNEXT-7951 `hostgroup.update`: the parameter `uuid` can now be updated.

hostprototype

Changes:
ZBXNEXT-7951 `hostprototype.update`: the parameter `uuid` can now be updated.

httptest

Changes:
ZBXNEXT-7951 `httptest.update`: the parameter `uuid` can now be updated.

item

Changes:
ZBXNEXT-7951 `item.update`: the parameter `uuid` can now be updated.

itemprototype

Changes:
ZBXNEXT-7951 `itemprototype.update`: the parameter `uuid` can now be updated.

template

Changes:
ZBXNEXT-7951 `template.update`: the parameter `uuid` can now be updated.

templatedashboard

Changes:
ZBXNEXT-7951 `templatedashboard.update`: the parameter `uuid` can now be updated.

trigger

Changes:
ZBXNEXT-7951 `trigger.update`: the parameter `uuid` can now be updated.

triggerprototype

Changes:
ZBXNEXT-7951 `triggerprototype.update`: the parameter `uuid` can now be updated.

valuemap

Changes:
ZBXNEXT-7951 `valuemap.update`: the parameter `uuid` can now be updated.

6.2.3 user

Changes:
ZBXNEXT-7971 `user.create`, `user.update`: increased max length of the "url" field to 2048 characters.

6.2.1 graph

Changes:
ZBX-7706 `graph.get`: Graph availability doesn't depend on permissions to items specified in graph "ymin_itemid" and "ymax_itemid" fields.
Graph having MIN or MAX Y axis linked to inaccessible items will still be accessible but MIN/MAX Y axis works the same way as if specified calculation method is "Calculated".

graphprototype

Changes:
ZBX-7706 `graphprototype.get`: Graph prototype availability doesn't depend on permissions to items specified in graph prototype "ymin_itemid" and "ymax_itemid" fields.

20. 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 Modules are written in PHP language. Model-view-controller (MVC) software pattern design is preferred, as it is also used in Zabbix frontend and will ease the development. PHP strict typing is also welcome but not mandatory.

Please note that with modules you can easily add new menu items and respective views and actions to Zabbix frontend. Currently it is not possible to register new API or create new database tables through modules.

Module structure

Each module is a directory (placed within the `modules` directory) with sub-directories containing controllers, views and any other code:

```
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
        js/                                     JavaScript files used in views.
            example.something.view.js.php
    partials/                                   View partial files.
        example.something.reusable.php
        js/                                     JavaScript files used in partials.
            example.something.reusable.js.php
```

As you can see, the only mandatory file within the custom module directory is `manifest.json`. The module will not register without this file. `Module.php` is responsible for registering menu items and processing events such as 'onBeforeAction' and 'onTerminate'. The *actions*, *views* and *partials* directories contain PHP and JavaScript code needed for module actions.

Naming convention

Before you create a module, it is important to agree on the naming convention for different module items such as directories and files so that we could keep things well organized. You can also find examples above, in the Module structure section.

Item	Naming rules	Example
Module directory	Lowercase [a-z], underscore and decimal digits	example_v2
Action subdirectories	Lowercase [a-z] and underscore character	data_export
Action files	CamelCase, ending with action type	SomethingView.php
View and partial files	Lowercase [a-z] Words separated with dot Prefixed by `module.` followed by module name Ending with action type and .php file extension	module.example.something.view.php
Javascript files	The same rules apply as for view and partial files, except the .js.php file extension.	module.example.something.view.js.php

Note that the 'module' prefix and name inclusion is mandatory for view and partial file names, unless you need to override Zabbix core views or partials. This rule, however, does not apply to action file names.

Manifest preparation

Each module is expected to have a manifest.json file with the following fields in JSON format:

Parameter	Required	Type	Default	Description
manifest_version	Yes	Double	-	Manifest version of the module. Currently supported version is **1**.
id	Yes	String	-	Module ID. Only one module with given ID can be enabled at the same time.
name	Yes	String	-	Module name as displayed in the Administration section.
version	Yes	String	-	Module version as displayed in the Administration section.
namespace	Yes	String	-	PHP namespace for Module.php and action classes.
author	No	String	""	Module author as displayed in the Administration section.
url	No	String	""	Module URL as displayed in the Administration section.
description	No	String	""	Module description as displayed in the Administration section.
actions	No	Object	{}	Actions to register with this module. See Actions.
config	No	Object	{}	Module configuration.

For reference, please see an example of manifest.json in the Reference section.

Actions

The module will have control over frontend actions defined within the *actions* object in the manifest.json file. This way new actions are defined. In the same way you may redefine existing actions. Each key of actions should represent the action name and the corresponding value should contain `class` and optionally `layout` and `view` keys.

One action is defined by four counterparts: name, controller, view and layout. Data validation and preparation is typically done in the controller, output formatting is done in the view or partials, and the layout is responsible for decorating the page with elements such as menu, header, footer and others.

Module actions must be defined in the manifest.json file as *actions* object:

Parameter	Required	Type	Default	Description
key	Yes	String	-	Action name, in lowercase [a-z], separating words with dot.
class	Yes	String	-	Action class name, including subdirectory path (if used) within the `actions` directory.
layout	No	String	”layout.htmlpage”	Action layout.
view	No	String	null	Action view.

There are several predefined layouts, like `layout.json` or `layout.xml`. These are intended for actions which produce different result than an HTML. You may explore predefined layouts in the app/views/ directory or even create your own.

Sometimes it is necessary to only redefine the view part of some action leaving the controller intact. In such case just place the necessary view and/or partial files inside the `views` directory of the module.

For reference, please see an example action controller file in the Reference section. Please do not hesitate to explore current actions of Zabbix source code, located in the app/ directory.

Module.php

This optional PHP file is responsible for module initialization as well as event handling. Class ’Module’ is expected to be defined in this file, extending base class `\Core\CModule`. The Module class must be defined within the namespace specified in the manifest.json file.

```
<?php

namespace Modules\Example;
use Core\CModule as BaseModule;

class Module extends BaseModule {
    ...
}
```

For reference, please see an example of Module.php in the Reference section.

Reference This section contains basic versions of different module elements introduced in the previous sections.

manifest.json

```
{
    "manifest_version": 1.0,
    "id": "example_module",
    "name": "Example module",
    "version": "1.0",
    "namespace": "Example",
    "author": "John Smith",
    "url": "http://module.example.com",
    "description": "Short description of the module.",
    "actions": {
        "example.something.view": {
            "class": "SomethingView",
            "view": "module.example.something.view"
        },
        "example.something.create": {
            "class": "SomethingCreate",
            "layout": null
        },
        "example.something.delete": {
            "class": "SomethingDelete",
            "layout": null
        },
        "example.something.export.xml": {
            "class": "data_export/ExportAsXml",
            "layout": null
```

```
        },
        "example.something.export.excel": {
            "class": "data_export/ExportAsExcel",
            "layout": null
        }
    },
    "config": {
        "username": "john_smith"
    }
}
```

Module.php

```php
<?php declare(strict_types = 1);

namespace Modules\Example;

use APP;
use CController as CAction;

/**
 * Please see Core\CModule class for additional reference.
 */
class Module extends \Core\CModule {

    /**
     * Initialize module.
     */
    public function init(): void {
        // Initialize main menu (CMenu class instance).
        APP::Component()->get('menu.main')
            ->findOrAdd(_('Reports'))
                ->getSubmenu()
                    ->add((new \CMenuItem(_('Example wide report')))
                        ->setAction('example.report.wide.php')
                    )
                    ->add((new \CMenuItem(_('Example narrow report')))
                        ->setAction('example.report.narrow.php')
                    );
    }

    /**
     * Event handler, triggered before executing the action.
     *
     * @param CAction $action  Action instance responsible for current request.
     */
    public function onBeforeAction(CAction $action): void {
    }

    /**
     * Event handler, triggered on application exit.
     *
     * @param CAction $action  Action instance responsible for current request.
     */
    public function onTerminate(CAction $action): void {
    }
}
```

Action controller

```php
<?php declare(strict_types = 1);

namespace Modules\Example\Actions;
```

```php
use CControllerResponseData;
use CControllerResponseFatal;
use CController as CAction;

/**
 * Example module action.
 */
class SomethingView extends CAction {

    /**
     * Initialize action. Method called by Zabbix core.
     *
     * @return void
     */
    public function init(): void {
        /**
         * Disable SID (Sessoin ID) validation. Session ID validation should only be used for actions whic
         * modification, such as update or delete actions. In such case Session ID must be presented in th
         * the URL would expire as soon as the session expired.
         */
        $this→disableSIDvalidation();
    }

    /**
     * Check and sanitize user input parameters. Method called by Zabbix core. Execution stops if false is
     *
     * @return bool true on success, false on error.
     */
    protected function checkInput(): bool {
        $fields = [
            'name'  => 'required|string',
            'email' => 'required|string',
            'phone' => 'string'
        ];

        // Only validated data will further be available using $this→hasInput() and $this→getInput().
        $ret = $this→validateInput($fields);

        if (!$ret) {
            $this→setResponse(new CControllerResponseFatal());
        }

        return $ret;
    }

    /**
     * Check if the user has permission to execute this action. Method called by Zabbix core.
     * Execution stops if false is returned.
     *
     * @return bool
     */
    protected function checkPermissions(): bool {
        $permit_user_types = [USER_TYPE_ZABBIX_ADMIN, USER_TYPE_SUPER_ADMIN];

        return in_array($this→getUserType(), $permit_user_types);
    }

    /**
     * Prepare the response object for the view. Method called by Zabbix core.
     *
     * @return void
     */
```

```
    protected function doAction(): void {
        $contacts = $this→getInput('email');

        if ($this→hasInput('phone')) {
            $contacts .= ', '.$this→getInput('phone');
        }

        $data = [
            'name' => $this→getInput('name'),
            'contacts' => $contacts
        ];

        $response = new CControllerResponseData($data);

        $this→setResponse($response);
    }
}
```

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();
```

21. Appendixes

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

1 Frequently asked questions / Troubleshooting

Frequently asked questions or FAQ.

1. Q: Can I flush/clear the queue (as depicted in *Administration → Queue*)?
   A: No.
2. Q: How do I migrate from one database to another?
   A: Dump data only (for MySQL, use flag -t or --no-create-info), create the new database using schema files from Zabbix and import the data.
3. Q: I would like to replace all spaces with underscores in my item keys because they worked in older versions but space is not a valid symbol for an item key in 3.0 (or any other reason to mass-modify item keys). How should I do it and what should i beware of?
   A: You may use a database query to replace all occurrences of spaces in item keys with underscores:
   update items set key_=replace(key_,' ','_');
   Triggers will be able to use these items without any additional modifications, but you might have to change any item references in these locations:
   * Notifications (actions)
   * Map element and link labels
   * Calculated item formulas

4. Q: My graphs have dots instead of lines or empty areas. Why so?
   A: Data is missing. This can happen for a variety of reasons - performance problems on Zabbix database, Zabbix server, network, monitored devices...
5. Q: Zabbix daemons fail to start up with a message *Listener failed with error: socket() for [[-]:10050] failed with error 22: Invalid argument.*
   A: This error arises at attempt to run Zabbix agent compiled on version 2.6.27 or above on a platform with a kernel 2.6.26 and lower. Note that static linking will not help in this case because it is the socket() system call that does not support SOCK_CLOEXEC flag on earlier kernels. ZBX-3395
6. Q: I try to set up a flexible user parameter (one that accepts parameters) with a command that uses a positional parameter like $1, but it doesn't work (uses item parameter instead). How to solve this?
   A: Use a double dollar sign like **$$1**
7. Q: All dropdowns have a scrollbar and look ugly in Opera 11. Why so?
   A: It's a known bug in Opera 11.00 and 11.01; see Zabbix issue tracker for more information.
8. Q: How can I change graph background color in a custom theme?
   A: See graph_theme table in the database and theming guide.
9. Q: With DebugLevel 4 I'm seeing messages "Trapper got [] len 0" in server/proxy log - what's that?
   A: Most likely that is frontend, connecting and checking whether server is still running.
10. Q: My system had the time set in the future and now no data is coming in. How could this be solved?
    A: Clear values of database fields hosts.disable_until*, drules.nextcheck, httptest.nextcheck and restart the server/proxy.
11. Q: Text item values in frontend (when using *{ITEM.VALUE}* macro and in other cases) are cut/trimmed to 20 symbols. Is that normal?
    A: Yes, there is a hardcoded limit in include/items.inc.php currently.

If you haven't found answer to your question try Zabbix forum

2 Installation and setup

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.

> **Note:**
> If installing from Zabbix Git repository, you need to run the following command prior to proceeding to the next steps:
>

 `$ make dbschema`

MySQL

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. 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> 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

Since Zabbix 6.0, primary keys are used for all tables in new installations.

This section provides instructions for manually upgrading the history tables in existing installations to primary keys.

Instructions are available for:

- MySQL
- PostgreSQL
- TimescaleDB
- Oracle

Attention:
The instructions provided on this page are designed for advanced users. Note that these instructions might need to be adjusted for your specific configuration.

Important notes

- Make sure to back up the database before the upgrade.
- If the database uses partitions, contact the DB administrator or Zabbix support team for help.
- Stopping Zabbix server for the time of the upgrade is strongly recommended. However, if absolutely necessary, there is a way to perform an upgrade while the server is running (only for MySQL, MariaDB and PostgreSQL without TimescaleDB).
- CSV files can be removed after a successful upgrade to primary keys.
- Optionally, Zabbix frontend may be switched to maintenance mode.
- Upgrade to primary keys should be done after upgrading Zabbix server to 6.0.
- On proxy, history tables that are not used can be upgraded by executing history_pk_prepare.sql.

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
```

Attention:
The path to `history_pk_prepare.sql` has been changed since Zabbix 6.2.3. For versions older than 6.2.3, the following command should be used:


```
sudo -u zabbix psql zabbix < /usr/share/doc/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 `database/postgresql/tsdb_history_pk_upgrade_with`

```
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 `database/postgresql/tsdb_history_pk_upgrade_no_compression`:

```
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 /usr/share/zabbix/zabbix-sql-scripts/database/oracle
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):

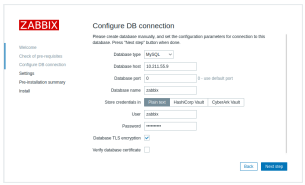
Parameter	Description
Database TLS CA file	Specify the full path to a valid TLS certificate authority (CA) file.
Database TLS key file	Specify the full path to a valid TLS key file.
Database TLS certificate file	Specify the full path to a valid TLS certificate file.
Database host verification	Mark this 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.

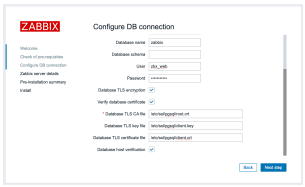
Use cases

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*
		6. 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* 6. 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

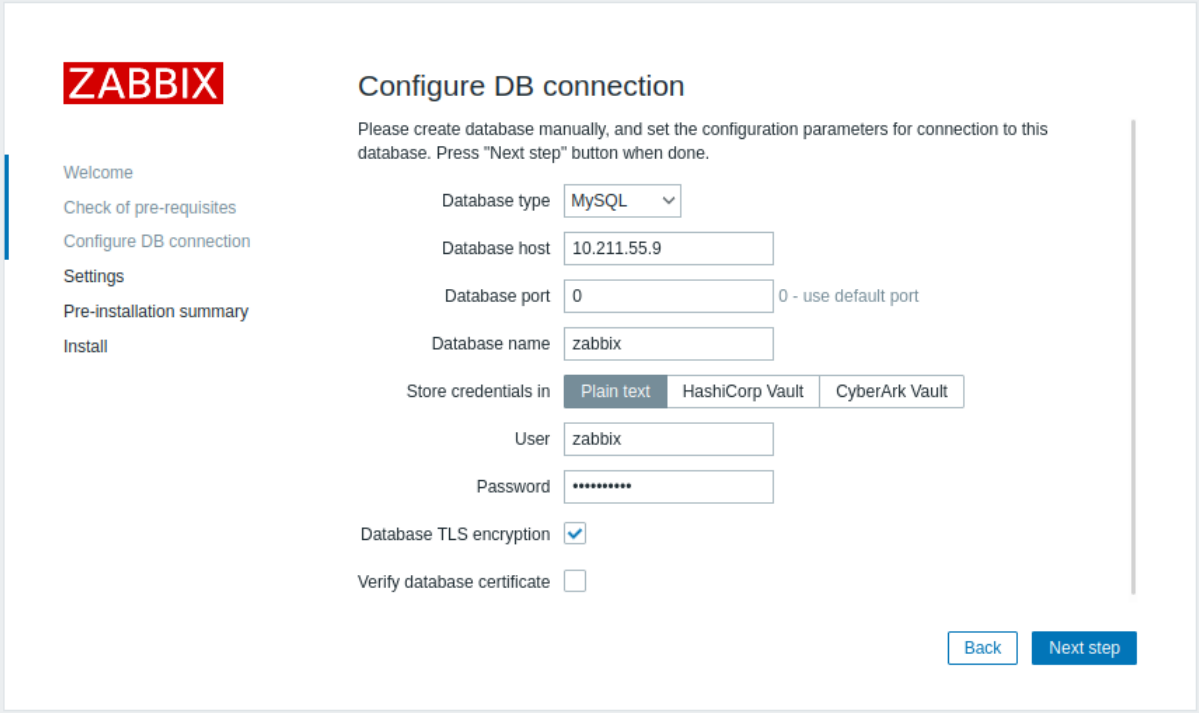

mysql> SHOW SESSION STATUS LIKE 'Ssl_cipher_list'\G;
*************************** 1. row ***************************
Variable_name: Ssl_cipher_list
Value: TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256:TLS_AES_128_CCM_SHA256:E
1 row in set (0.00 sec)

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

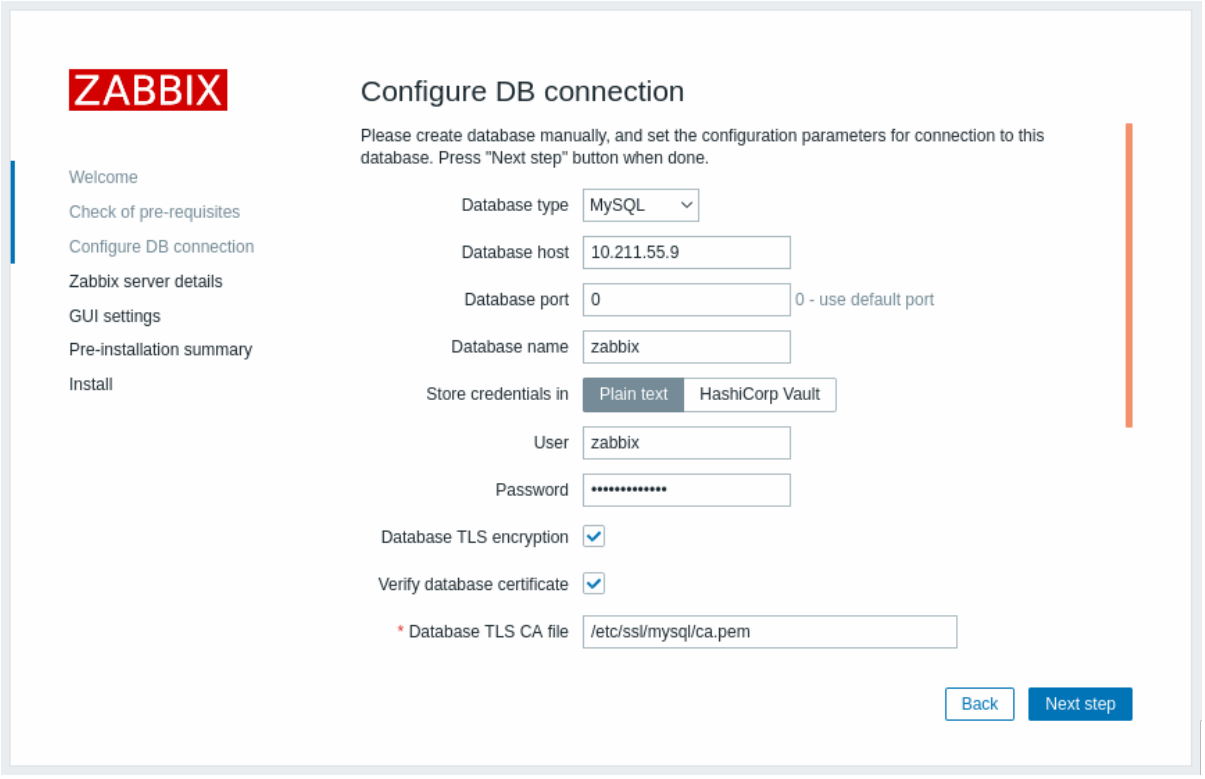
Copy required MySQL CA to the Zabbix frontend server, assign proper permissions to allow the webserver to read this file.

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=
```

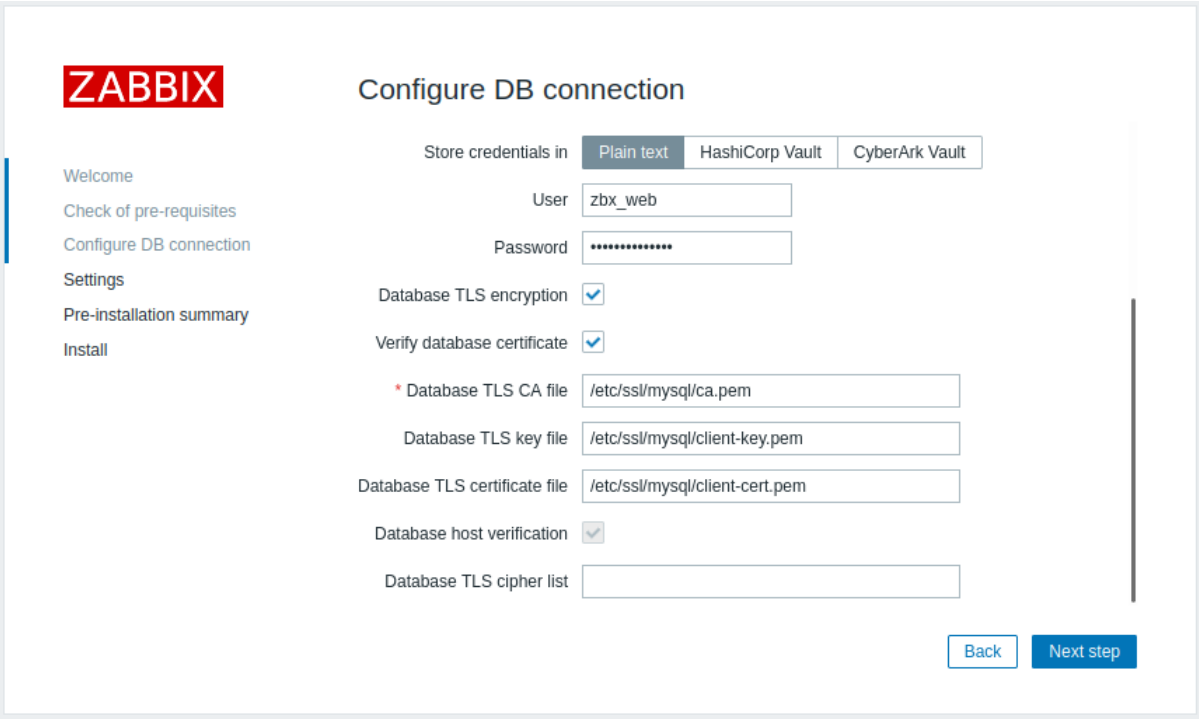
Frontend

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

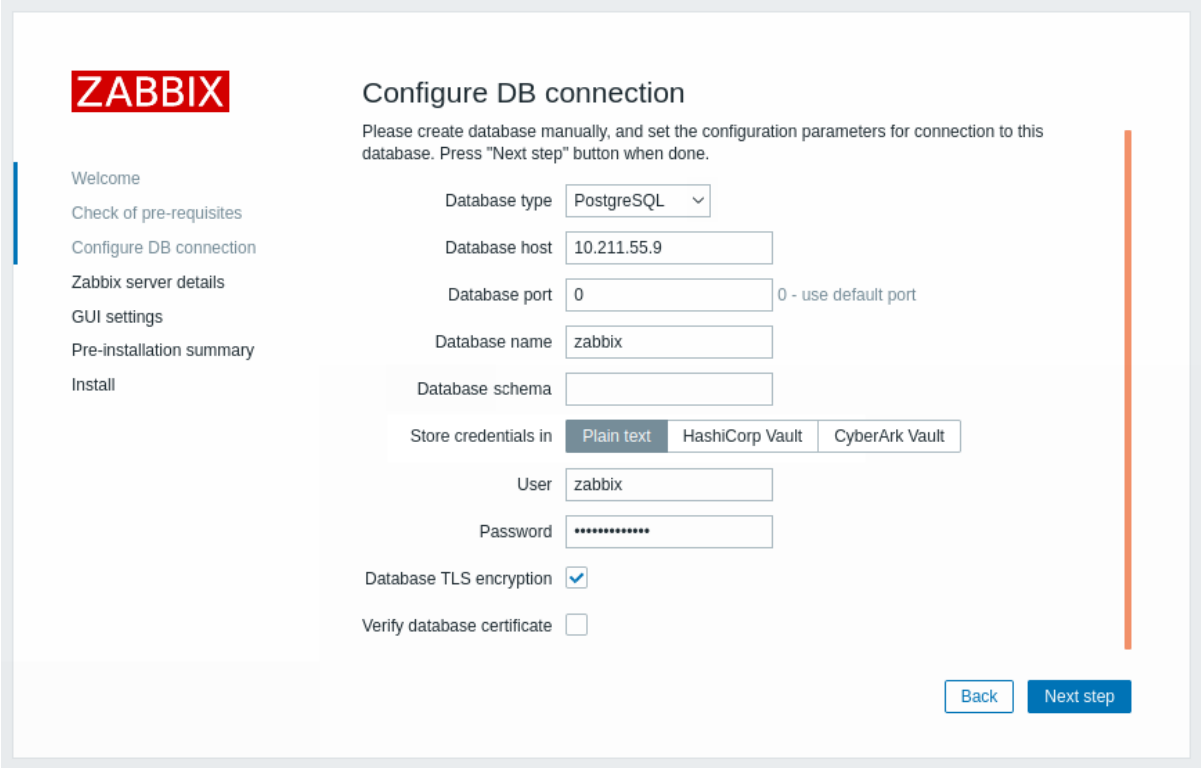
### verify full
hostssl all all 0.0.0.0/0 md5 clientcert=verify-full
...
```

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*:

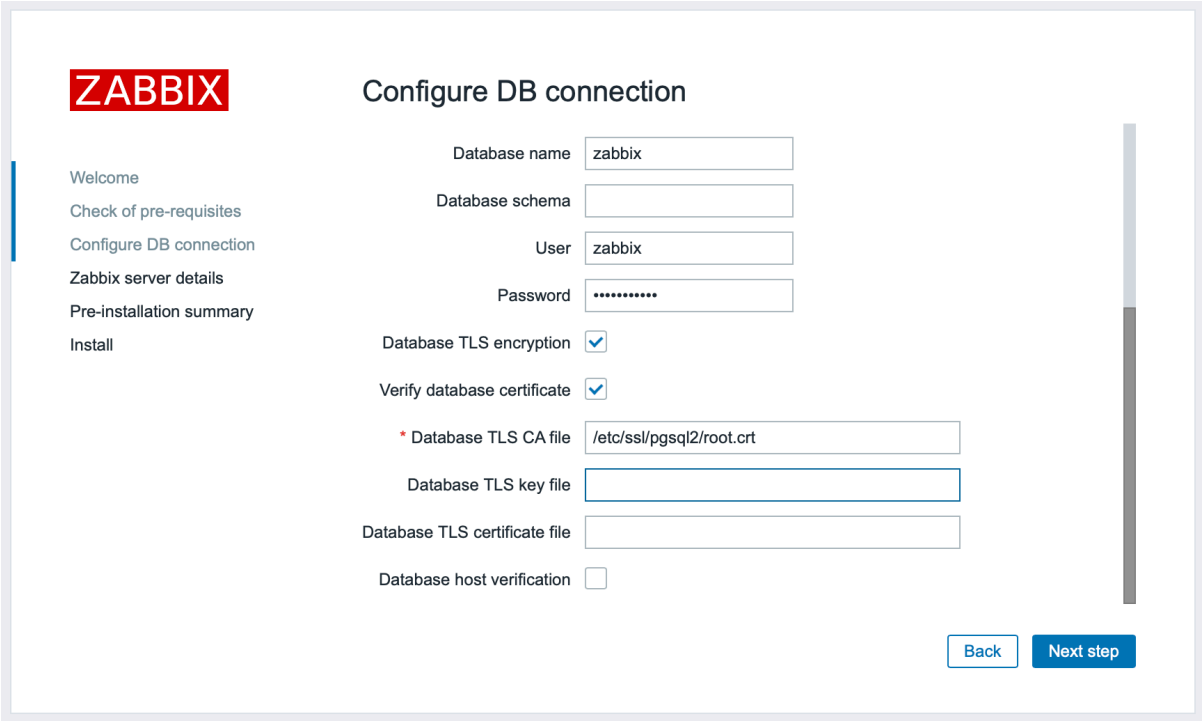
```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=required
...
```

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 key file*
- Specify path to *Database TLS CA file*
- Specify path to *Database TLS certificate 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:*

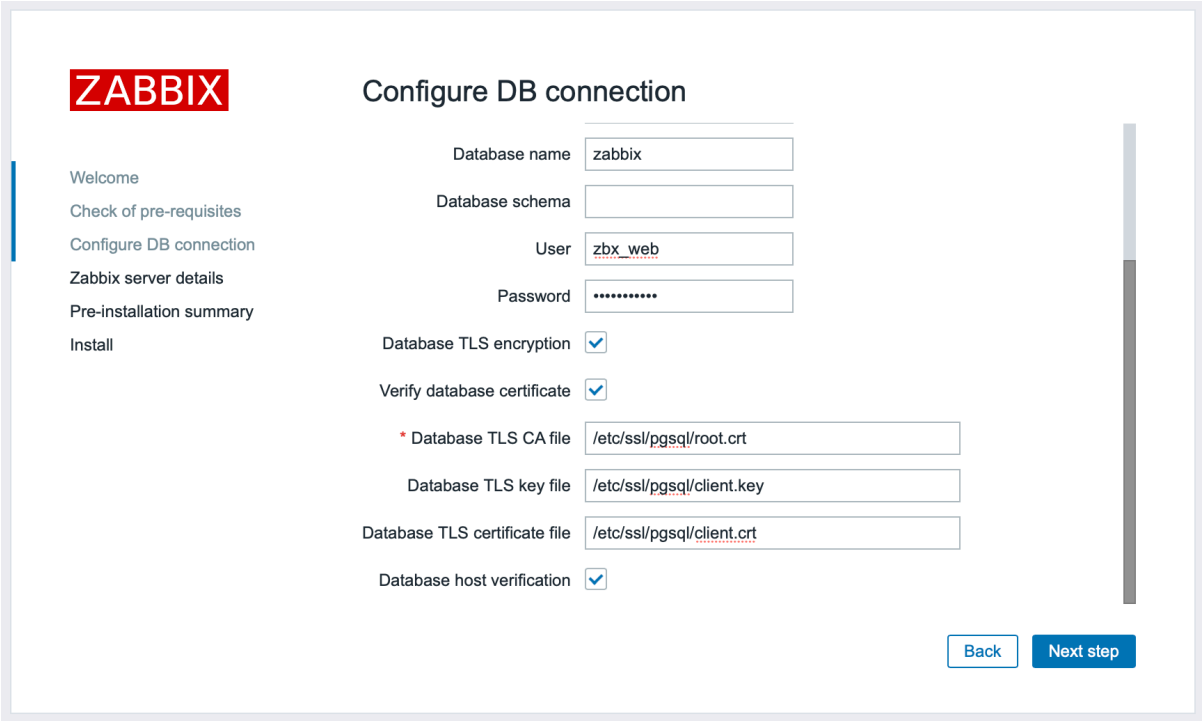
```
...
DBHost=10.211.55.9
DBName=zabbix
DBUser=zbx_srv
DBPassword=<strong_password>
DBTLSConnect=verify_ca
DBTLSCAFile=/etc/ssl/pgsql/root.crt
...
```

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).

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 `timescaledb.sql` script located in database/postgresql. 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

Compression can be used only if both *Override item history period* and *Override item trend period* options are 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 administration screen for *Housekeeping* and the *System information* section.

All of these parameters can be changed in *Administration → General → 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. Starting with Zabbix 6.2.1, Zabbix detects if compression is supported. If it 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 TimescaleDB compression 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 → General → 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 instructions in TimescaleDB 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 Real-time export of events, item values, trends

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.

8 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:

```
cp /etc/php5/fpm/php-fpm.conf{.default,}
sed -i 's/user = nobody/user = wwwrun/; s/group = nobody/group = www/' /etc/php5/fpm/php-fpm.conf
```

SLES 15

In SUSE Linux Enterprise Server 15 you need to configure php-fpm:

```
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
```

9 Running agent as root

Starting with version **5.0.0** the systemd service file for Zabbix agent in official packages was updated to explicitly include directives for `User` and `Group`. Both are set to `zabbix`.

This means that the old functionality of configuring which user Zabbix agent runs as via `zabbix_agentd.conf` file is bypassed and agent will always run as the user specified in the systemd service file.

To override this new behavior create a `/etc/systemd/system/zabbix-agent.service.d/override.conf` file with 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 2** this completely determines the user that it runs as.

For old **agent** this only re-enables the functionality of configuring user in the `zabbix_agentd.conf` file. Therefore in order to run zabbix agent as root you still have to edit the agent configuration file and specify `User=root` as well as `AllowRoot=1` options.

10 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 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

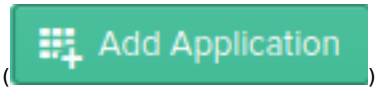
Zabbix agent for Windows does not support non-standard Windows configurations where CPUs are distributed non-uniformly across NUMA nodes. If logical CPUs are distributed non-uniformly, then CPU performance metrics may not be available for some CPUs. For example, if there are 72 logical CPUs with 2 NUMA nodes, both nodes must have 36 CPUs each.

11 SAML setup with Okta

This section describes how to configure Okta to enable SAML 2.0 authentication for Zabbix.

Okta configuration

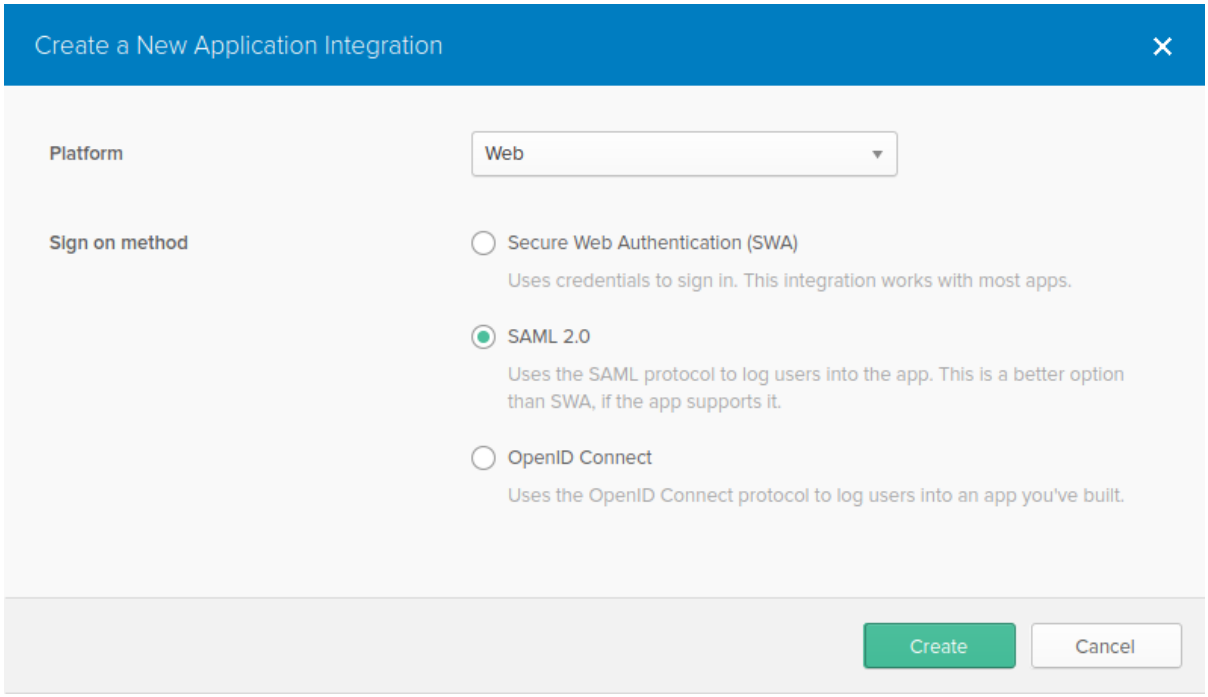
1. Go to https://okta.com and register or sign in to your account.

2. In the Okta web interface navigate to *Applications* → *Applications* and press "Add Application" button (

).

3. Press "Create New App" button (

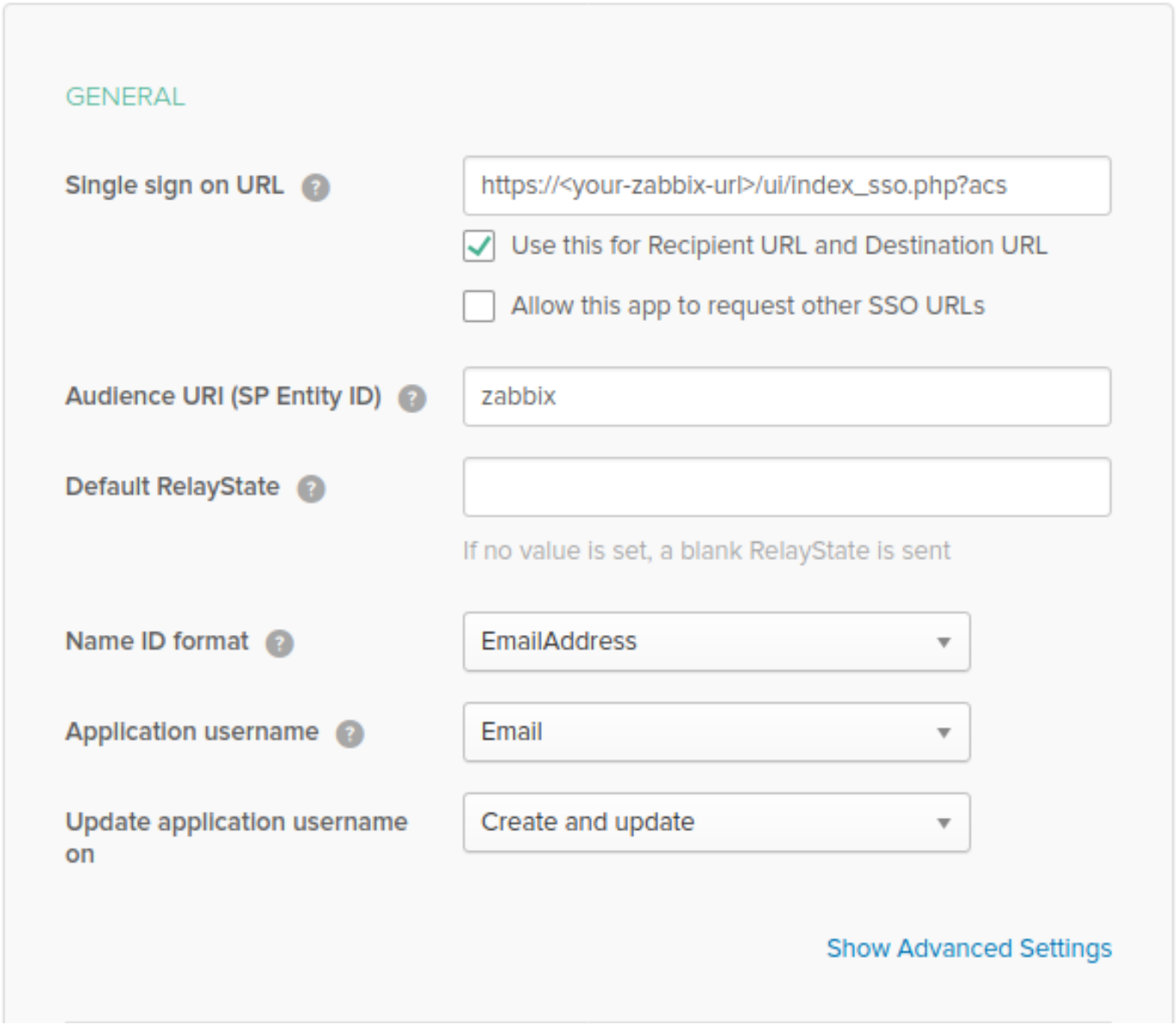
). In a popup window select *Platform:* Web, *Sign on method:* SAML 2.0 and press "Create" button.



4. Fill in the fields in the *General settings* tab (the first tab that appears) according to your preferences and press ”Next”.

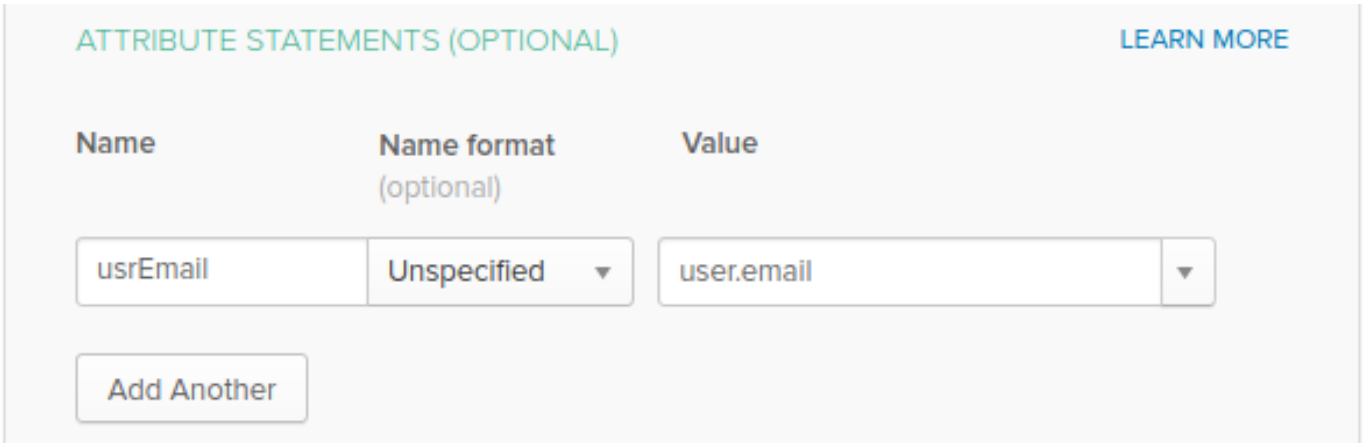
5. In the *Configure SAML* tab enter the values provided below, then press ”Next”.

- In the **GENERAL** section:
  - *Single sign on URL*: https://<your-zabbix-url>/ui/index_sso.php?acs
    The checkbox *Use this for Recipient URL and Destination URL* should 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 *Administration → Users* settings.
  - Fill in other fields according to your preferences.



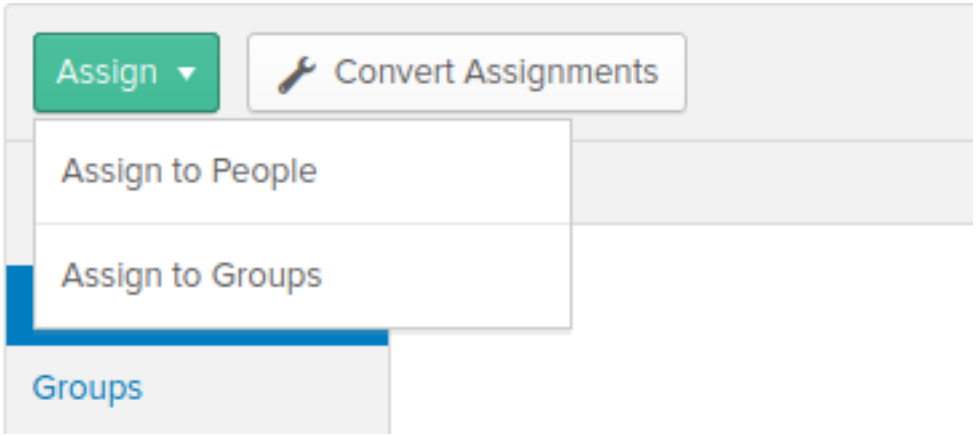
Note:
If planning to use encrypted connection, generate private and public encryption certificates, then upload public certificate to Okta. Certificate upload form appears when *Assertion Encryption* is set to Encrypted (click *Show Advanced Settings* to find this parameter).

- In the **ATTRIBUTE STATEMENTS (OPTIONAL)** section add an attribute statement with:
  - *Name:* usrEmail
  - *Name format:* Unspecified
  - *Value:* user.email



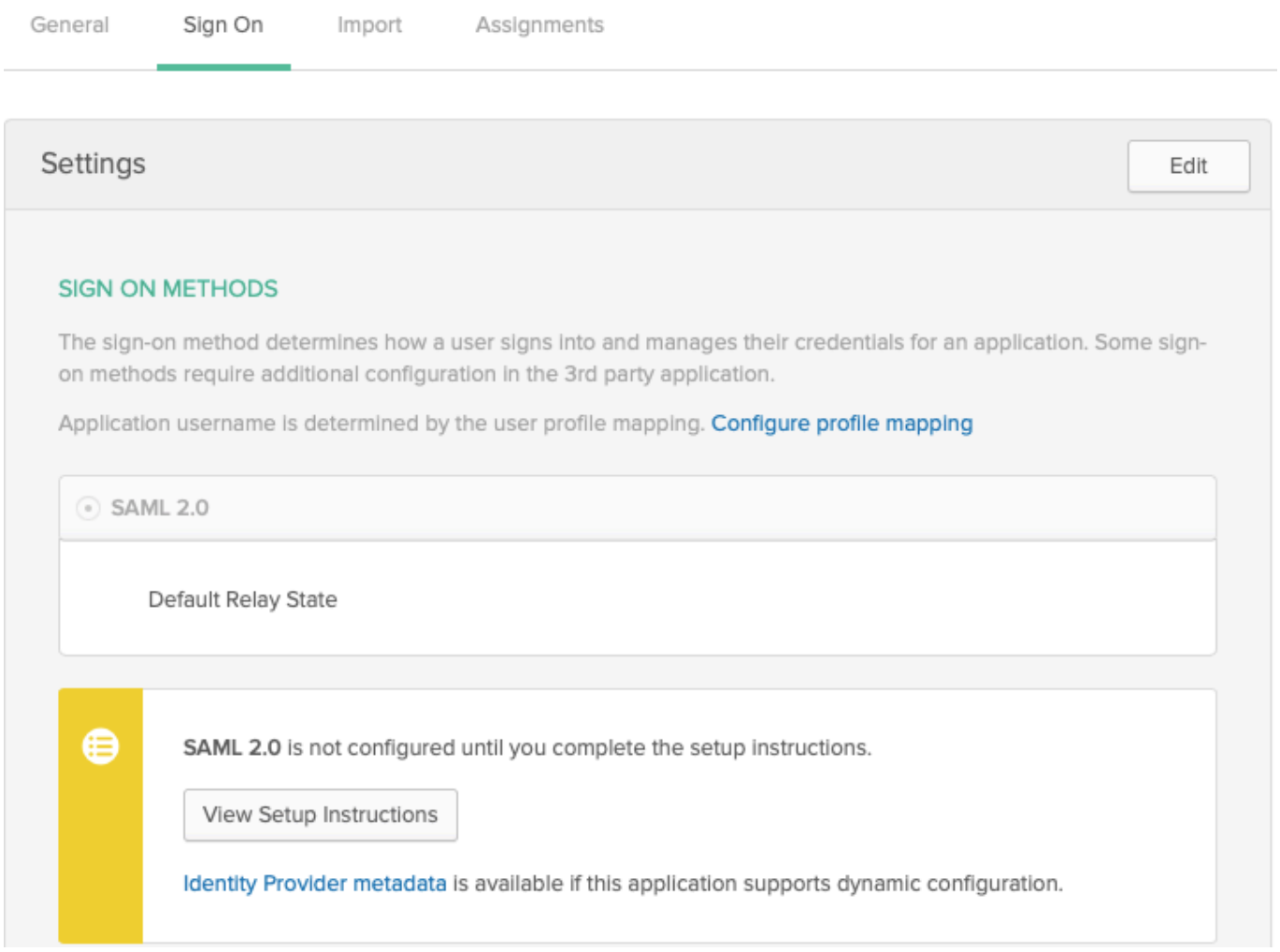
6. At the next tab, select ”I’m a software vendor. I’d like to integrate my app with Okta” and press ”Finish”.

7. Now, navigate to *Assignments* tab and press the ”Assign” button, then select *Assign to People* from the drop-down.



8. In a popup that appears, assign created app to people that will use SAML 2.0 to authenticate with Zabbix, then press ”Save and go back”.

9. Navigate to the *Sign On* tab and press the ”View Setup Instructions” button. Setup instructions will be displayed in a new tab; keep this tab open while configuring Zabbix.



Zabbix configuration

1. In Zabbix, go to SAML settings in the *Administration → Authentication* section and copy information from Okta setup instructions into corresponding fields:

- Identity Provider Single Sign-On URL → SSO service URL
- Identity Provider Issuer → IdP entity ID
- Username attribute → Attribute name (usrEmail)
- SP entity ID → Audience URI

2. Download the certificate provided in the Okta setup instructions page into *ui/conf/certs* folder as idp.crt, and set permission 644

by running:

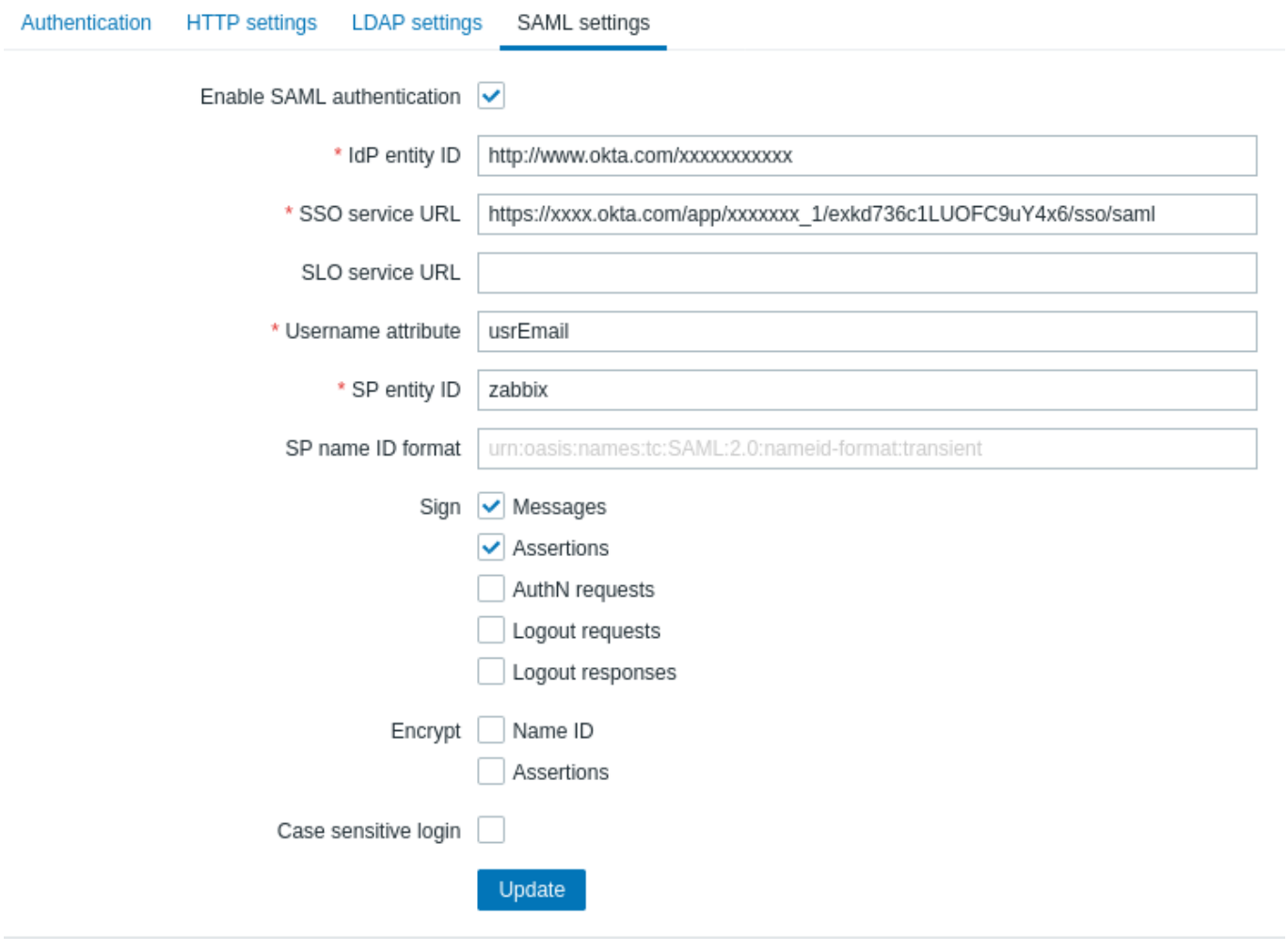
```
chmod 644 idp.crt
```

Note, that if you have upgraded to Zabbix 5.0 from an older version, you will also need to manually add these lines to zabbix.conf.php file (located in the //ui/conf/ // directory):

```
// Used for SAML authentication.
$SSO['SP_KEY'] = 'conf/certs/sp.key'; // Path to your private key.
$SSO['SP_CERT'] = 'conf/certs/sp.crt'; // Path to your public key.
$SSO['IDP_CERT'] = 'conf/certs/idp.crt'; // Path to IdP public key.
$SSO['SETTINGS'] = []; // Additional settings
```

See generic SAML Authentication instructions for more details.

3. If *Assertion Encryption* has been set to Encrypted in Okta, a checkbox "Assertions" of the *Encrypt* parameter should be marked in Zabbix as well.



4. Press the "Update" button to save these settings.

Note:
To sign in with SAML, the username in Zabbix should match the Okta e-mail. These settings can be changed in the *Administration → Users* section of Zabbix web interface.

12 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):

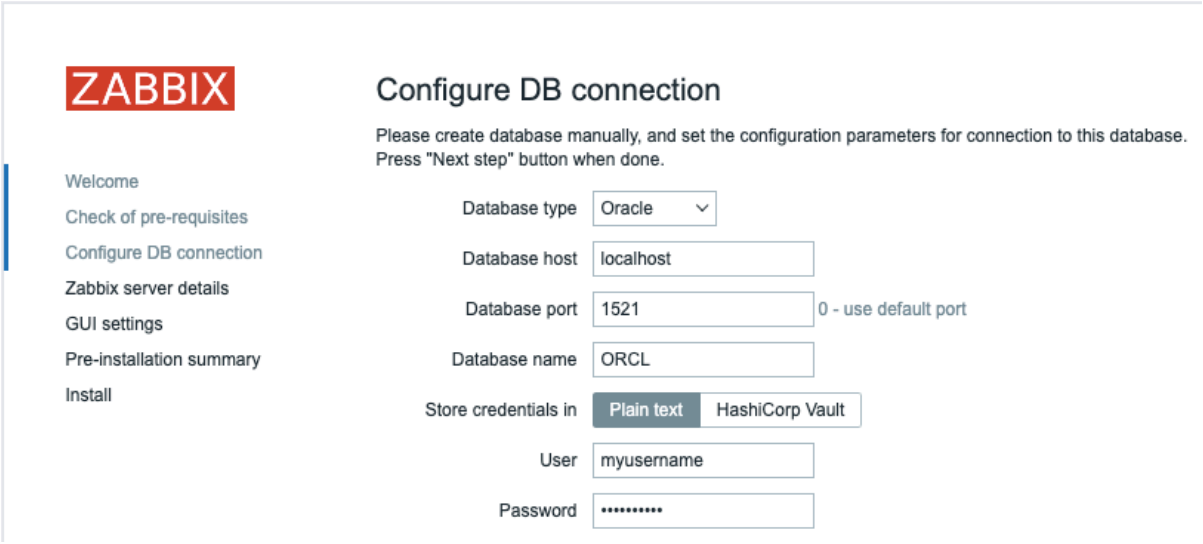
```
DBHost=localhost
DBPort=1521
DBUser=myusername
DBName=ORCL
DBPassword=mypassword
```

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_TAKS 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.

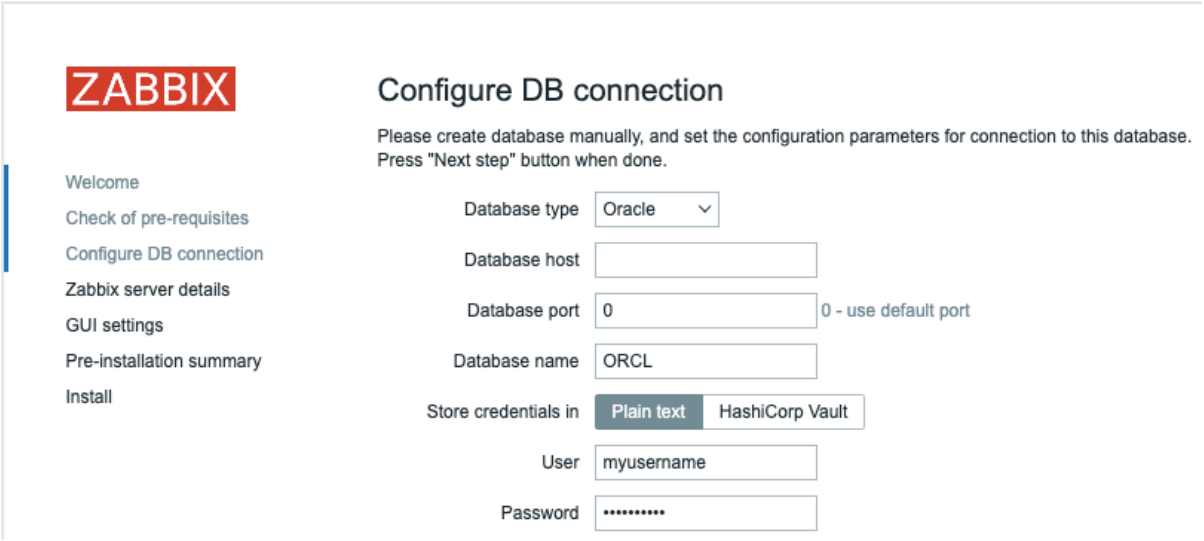
Example:

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
```

13 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
```

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.

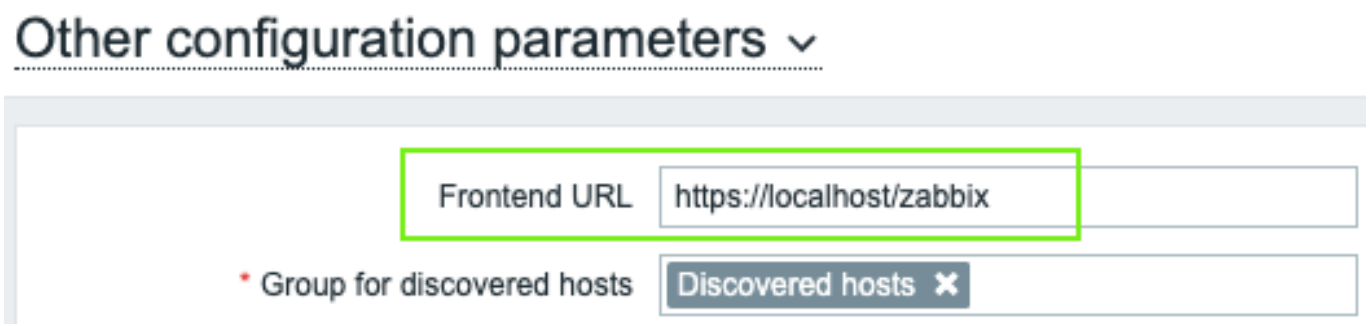
Example:

```
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.

14 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
```

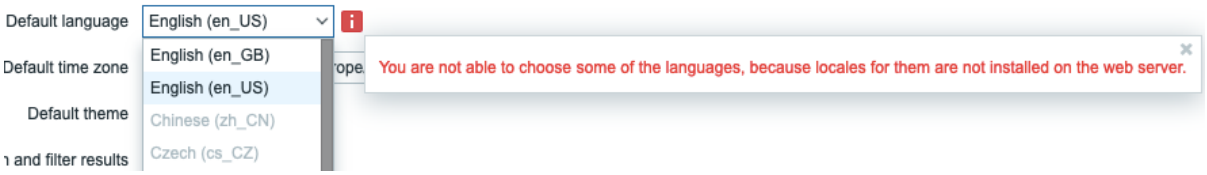
This step is not needed when installing Zabbix from packages or source tar.gz files.

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."



3 Process configuration

1 Zabbix server

Overview

This section lists parameters supported in a Zabbix server configuration file (zabbix_server.conf).

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.

Parameters

Parameter	Mandatory	Range	Default	Description
AlertScriptsPath	no		/usr/local/share/zabbix/alertscripts	Location of custom alert scripts (depends on compile-time installation variable *datadir*).
AllowRoot	no		0	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. 0 - do not allow 1 - allow
AllowUnsupportedDBVersions	no		0	Allow the server to work with unsupported database versions. 0 - do not allow 1 - allow
CacheSize	no	128K-64G	32M	Size of configuration cache, in bytes. Shared memory size for storing host, item and trigger data.
CacheUpdateFrequency	no	1-3600	60	Determines how often Zabbix will perform update of configuration cache in seconds. See also runtime control options.

Parameter	Mandatory	Range	Default	Description
DBHost	no		localhost	Database host name. In case of MySQL localhost or empty string results in using a socket. In case of PostgreSQL only empty string results in attempt to use socket. In case of 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.
DBName	yes			Database name. In case of 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.
DBPassword	no			Database password. Comment this line if no password is used.
DBPort	no	1024-65535		Database port when not using local socket. In case of 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.
DBSchema	no			Schema name. Used for PostgreSQL.
DBSocket	no			Path to MySQL socket file.
DBUser	no			Database user.
DBTLSConnect	no			Setting this option enforces to use TLS connection to 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_ca”, ”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. This parameter is supported since Zabbix 5.0.0.
DBTLSCAFile	no (yes, if DBTLSConnect set to one of: verify_ca, verify_full)			Full pathname of a file containing the top-level CA(s) certificates for database certificate verification. This parameter is supported since Zabbix 5.0.0.
DBTLSCertFile	no			Full pathname of file containing Zabbix server certificate for authenticating to database. This parameter is supported since Zabbix 5.0.0.
DBTLSKeyFile	no			Full pathname of file containing the private key for authenticating to database. This parameter is supported since Zabbix 5.0.0.
DBTLSCipher	no			The list of encryption ciphers that Zabbix server permits for TLS protocols up through TLSv1.2. Supported only for MySQL. This parameter is supported since Zabbix 5.0.0.
DBTLSCipher13	no			The list of encryption ciphersuites that Zabbix server permits for TLSv1.3 protocol. Supported only for MySQL, starting from version 8.0.16. This parameter is supported since Zabbix 5.0.0.

Parameter	Mandatory	Range	Default	Description
DebugLevel	no	0-5	3	Specifies 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.
ExportDir	no			Directory for real-time export of events, history and trends in newline-delimited JSON format. If set, enables real-time export. This parameter is supported since Zabbix 4.0.0.
ExportFileSize	no	1M-1G	1G	Maximum size per export file in bytes. Only used for rotation if ExportDir is set. This parameter is supported since Zabbix 4.0.0.
ExportType	no			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. e.g.: ExportType=history,trends - export history and trends only ExportType=events - export events only
ExternalScripts	no		/usr/local/share/zabbix/externalscripts	Location of external scripts (depends on compile-time installation variable *datadir*).
Fping6Location	no		/usr/sbin/fping6	Location of fping6. Make sure that fping6 binary has root ownership and SUID flag set. Make empty (”Fping6Location=”) if your fping utility is capable to process IPv6 addresses.
FpingLocation	no		/usr/sbin/fping	Location of fping. Make sure that fping binary has root ownership and SUID flag set!
HANodeName	no			The high availability cluster node name. When empty the server is working in standalone mode and a node with empty name is created.
HistoryCacheSize	no	128K-2G	16M	Size of history cache, in bytes. Shared memory size for storing history data.
HistoryIndexCacheSize	no	128K-2G	4M	Size of history index cache, in bytes. Shared memory size for indexing history data stored in history cache. The index cache size needs roughly 100 bytes to cache one item. This parameter is supported since Zabbix 3.0.0.
HistoryStorageDateIndex			0	Enable preprocessing of history values in history storage to store values in different indices based on date: 0 - disable 1 - enable
HistoryStorageURL				History storage HTTP[S] URL. This parameter is used for Elasticsearch setup.
HistoryStorageTypes	no		uint,dbl,str,log,text	Comma separated list of value types to be sent to the history storage. This parameter is used for Elasticsearch setup.

Parameter	Mandatory	Range	Default	Description
HousekeepingFrequency	no	0-24	1	Determines how often Zabbix will perform 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. 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 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.
Include	no			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. For example: `/absolute/path/to/config/files/*.conf`. See special notes about limitations.
JavaGateway	no			IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started.
JavaGatewayPort	no	1024-32767	10052	Port that Zabbix Java gateway listens on.
ListenBacklog	no	0 - INT_MAX	SOMAXCONN	The maximum number of pending connections in the TCP queue. Default value is a hard-coded constant, which depends on the system. Maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.
ListenIP	no		0.0.0.0	List of comma delimited IP addresses that the trapper should listen on. Trapper will listen on all network interfaces if this parameter is missing.
ListenPort	no	1024-32767	10051	Listen port for trapper.
LoadModule	no			Module to load at server startup. Modules are used to extend functionality of the server. Formats: LoadModule=<module.so> LoadModule=<path/module.so> LoadModule=</abs_path/module.so> Either the module must be located in 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.
LoadModulePath	no			Full path to location of server modules. Default depends on compilation options.

Parameter	Mandatory	Range	Default	Description
LogFile	yes, if LogType is set to *file*, otherwise no			Name of log file.
LogFileSize	no	0-1024	1	Maximum size of 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.
LogType	no		file	Log output type: *file* - write log to file specified by LogFile parameter, *system* - write log to syslog, *console* - write log to standard output. This parameter is supported since Zabbix 3.0.0.
LogSlowQueries	no	0-3600000	0	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.
MaxHousekeeperDelete	no	0-1000000	5000	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.
NodeAddress	no		10051	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.
PidFile	no		/tmp/zabbix_server.pid	Name of PID file.
ProblemHousekeepingFrequency	no	1-3600	60	Determines how often Zabbix will delete problems for deleted triggers in seconds.
ProxyConfigFrequency	no	1-604800	3600	Determines how often Zabbix server sends configuration data to a Zabbix proxy in seconds. Used only for proxies in a passive mode.
ProxyDataFrequency	no	1-3600	1	Determines how often Zabbix server requests history data from a Zabbix proxy in seconds. Used only for proxies in a passive mode.
ServiceManagerSyncFrequency	no	1-3600	60	Determines how often Zabbix will synchronize configuration of a service manager in seconds.
SNMPTrapperFile	no		/tmp/zabbix_traps.tmp	Temporary file used for passing data from SNMP trap daemon to the server. Must be the same as in zabbix_trap_receiver.pl or SNMPTT configuration file.
SocketDir	no		/tmp	Directory to store IPC sockets used by internal Zabbix services. This parameter is supported since Zabbix 3.4.0.

Parameter	Mandatory	Range	Default	Description
SourceIP	no			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	no			Location of public and private keys for SSH checks and actions
SSLCertLocation	no			Location of SSL client certificate files for client authentication. This parameter is used in web monitoring only.
SSLKeyLocation	no			Location of SSL private key files for client authentication. This parameter is used in web monitoring only.
SSLCALocation	no			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 since Zabbix 2.4.0 and in SMTP authentication since Zabbix 3.0.0.
StartAlerters	no	1-100	3	Number of pre-forked instances of alerters. This parameter is supported since Zabbix 3.4.0.
StartDBSyncers	no	1-100	4	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.
StartDiscoverers	no	0-250	1	Number of pre-forked instances of discoverers.
StartEscalators	no	1-100	1	Number of pre-forked instances of escalators. This parameter is supported since Zabbix 3.0.0.
StartHistoryPollers	no	0-1000	5	Number of pre-forked instances of history pollers. Only required for calculated checks. This parameter is supported since Zabbix 5.4.0.
StartHTTPPollers	no	0-1000	1	Number of pre-forked instances of HTTP pollers[1].
StartIPMIPollers	no	0-1000	0	Number of pre-forked instances of IPMI pollers.
StartJavaPollers	no	0-1000	0	Number of pre-forked instances of Java pollers[1].
StartLLDProcessors	no	1-100	2	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. This parameter is supported since Zabbix 4.2.0.
StartODBCPollers	no	0-1000	1	Number of pre-forked instances of ODBC pollers[1].
StartPingers	no	0-1000	1	Number of pre-forked instances of ICMP pingers[1].
StartPollersUnreachable	no	0-1000	1	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.
StartPollers	no	0-1000	5	Number of pre-forked instances of pollers[1].
StartPreprocessors	no	1-1000	3	Number of pre-forked instances of preprocessing workers[1]. The preprocessing manager process is automatically started when a preprocessor worker is started. This parameter is supported since Zabbix 3.4.0.
StartProxyPollers	no	0-250	1	Number of pre-forked instances of pollers for passive proxies[1].
StartReportWriters	no	0-100	0	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. This parameter is supported since Zabbix 5.4.0.

Parameter	Mandatory	Range	Default	Description
StartSNMPTrapper	no	0-1	0	If set to 1, SNMP trapper process will be started.
StartTimers	no	1-1000	1	Number of pre-forked instances of timers. Timers process maintenance periods.
StartTrappers	no	0-1000	5	Number of pre-forked instances of trappers[1]. Trappers accept incoming connections from Zabbix sender, active agents and active proxies.
StartVMwareCollectors	no	0-250	0	Number of pre-forked VMware collector instances.
StatsAllowedIP	no			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 This parameter is supported since Zabbix 4.2.0.
Timeout	no	1-30	3	Specifies how long we wait for agent, SNMP device or external check in seconds.
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSCertFile	no			Full pathname of a file containing the server certificate or certificate chain, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSCipherAll	no			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_1 This parameter is supported since Zabbix 4.4.7.
TLSCipherAll13	no			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:+COMP-NULL::+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+A This parameter is supported since Zabbix 4.4.7.
TLSCipherCert	no			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:+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128 This parameter is supported since Zabbix 4.4.7.
TLSCipherCert13	no			Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption. This parameter is supported since Zabbix 4.4.7.

Parameter	Mandatory	Range	Default	Description
TLSCipherPSK	no			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:+SIGN-ALL Example for OpenSSL: kECDHEPSK+AES128:kPSK+AES128 This parameter is supported since Zabbix 4.4.7.
TLSCipherPSK13	no			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 This parameter is supported since Zabbix 4.4.7.
TLSCRLFile	no			Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSKeyFile	no			Full pathname of a file containing the server private key, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TmpDir	no		/tmp	Temporary directory.
TrapperTimeout	no	1-300	300	Specifies how many seconds trapper may spend processing new data.
TrendCacheSize	no	128K-2G	4M	Size of trend cache, in bytes. Shared memory size for storing trends data.
TrendFunctionCacheSize	no	128K-2G	4M	Size of trend function cache, in bytes. Shared memory size for caching calculated trend function data.
UnavailableDelay	no	1-3600	60	Determines how often host is checked for availability during the unavailability period in seconds.
UnreachableDelay	no	1-3600	15	Determines how often host is checked for availability during the unreachability period in seconds.
UnreachablePeriod	no	1-3600	45	Determines after how many seconds of unreachability treats a host as unavailable.
User	no		zabbix	Drop privileges to a specific, existing user on the system. Only has effect if run as 'root' and AllowRoot is disabled.
ValueCacheSize	no	0,128K-64G	8M	Size of history value cache, in bytes. Shared memory size for caching item history data requests. Setting to 0 disables value cache (not recommended). When value cache runs out of the shared memory a warning message is written to the server log every 5 minutes.
Vault	no		HashiCorp	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.
VaultDBPath	no			Specifies a location, from where database credentials should be retrieved by keys. Depending on the Vault, can be vault path or query. Keys used for HashiCorp are 'password' and 'username'. Example: secret/zabbix/database Keys used for CyberArk are 'Content' and 'UserName'. Example: AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_proxy_data This option can only be used if DBUser and DBPassword are not specified.

Parameter	Mandatory	Range	Default	Description
VaultTLSCertFile	no			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	no			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	yes, if Vault is set to *HashiCorp*, otherwise no			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.
VaultURL	no		https://127.0.0.1:8200	Vault server HTTP[S] URL. System-wide CA certificates directory will be used if SSLCALocation is not specified.
VMwareCacheSize	no	256K-2G	8M	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.
VMwareFrequency	no	10-86400	60	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.
VMwarePerfFrequency	no	10-86400	60	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.
VMwareTimeout	no	1-300	10	The maximum number of seconds vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor).
WebServiceURL	no			HTTP[S] URL to Zabbix web service in the format `<host:port>/report`. For example: `http://localhost:10053/report` This parameter is supported since Zabbix 5.4.0.

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

This section lists parameters supported in a Zabbix proxy configuration file (zabbix_proxy.conf).

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.

Parameters

Parameter	Mandatory	Range	Default	Description
AllowRoot	no		0	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. 0 - do not allow 1 - allow
AllowUnsupportedDBVersions	no		0	Allow the proxy to work with unsupported database versions. 0 - do not allow 1 - allow
CacheSize	no	128K-64G	32M	Size of configuration cache, in bytes. Shared memory size, for storing host and item data.
ConfigFrequency	no	1-604800	3600	How often proxy retrieves configuration data from Zabbix server in seconds. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).
DataSenderFrequency	no	1-3600	1	Proxy will send collected data to the server every N seconds. Note that active proxy will still poll Zabbix server every second for remote command tasks. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).
DBHost	no		localhost	Database host name. In case of MySQL localhost or empty string results in using a socket. In case of PostgreSQL only empty string results in attempt to use socket. In case of 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.
DBName	yes			Database name or path to database file for SQLite3 (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 Zabbix server is using. In case of 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.
DBPassword	no			Database password. Ignored for SQLite. Comment this line if no password is used.
DBSchema	no			Schema name. Used for PostgreSQL.
DBSocket	no		3306	Path to MySQL socket.
				Database port when not using local socket. Ignored for SQLite.
DBUser				Database user. Ignored for SQLite.

Parameter	Mandatory	Range	Default	Description
DBTLSConnect	no			Setting this option enforces to use TLS connection to 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. This parameter is supported since Zabbix 5.0.0.
DBTLSCAFile	no (yes, if DBTLSConnect set to one of: verify_ca, verify_full)			Full pathname of a file containing the top-level CA(s) certificates for database certificate verification. This parameter is supported since Zabbix 5.0.0.
DBTLSCertFile	no			Full pathname of file containing Zabbix server certificate for authenticating to database. This parameter is supported since Zabbix 5.0.0.
DBTLSKeyFile	no			Full pathname of file containing the private key for authenticating to database. This parameter is supported since Zabbix 5.0.0.
DBTLSCipher	no			The list of encryption ciphers that Zabbix server permits for TLS protocols up through TLSv1.2. Supported only for MySQL. This parameter is supported since Zabbix 5.0.0.
DBTLSCipher13	no			The list of encryption ciphersuites that Zabbix server permits for TLSv1.3 protocol. Supported only for MySQL, starting from version 8.0.16. This parameter is supported since Zabbix 5.0.0.
DebugLevel	no	0-5	3	Specifies 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)
EnableRemoteCommands	no		0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed This parameter is supported since Zabbix 3.4.0.
ExternalScripts	no		/usr/local/share/zabbix/externalscripts	Location of external scripts (depends on compile-time installation variable *datadir*).
Fping6Location	no		/usr/sbin/fping6	Location of fping6. Make sure that fping6 binary has root ownership and SUID flag set. Make empty (”Fping6Location=”) if your fping utility is capable to process IPv6 addresses.
FpingLocation	no		/usr/sbin/fping	Location of fping. Make sure that fping binary has root ownership and SUID flag set!

Parameter	Mandatory	Range	Default	Description
HeartbeatFrequency	no	0-3600	60	Frequency of heartbeat messages in seconds. Used for monitoring availability of proxy on server side. 0 - heartbeat messages disabled. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter).
HistoryCacheSize	no	128K-2G	16M	Size of history cache, in bytes. Shared memory size for storing history data.
HistoryIndexCacheSize	no	128K-2G	4M	Size of history index cache, in bytes. Shared memory size for indexing history data stored in history cache. The index cache size needs roughly 100 bytes to cache one item. This parameter is supported since Zabbix 3.0.0.
Hostname	no		Set by HostnameItem	Unique, case sensitive Proxy name. Make sure the proxy name is known to the server! Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128
HostnameItem	no		system.hostname	Item used for setting Hostname if it is undefined (this will be run on the proxy similarly as on an agent). Does not support UserParameters, performance counters or aliases, but does support system.run[]. Ignored if Hostname is set.
HousekeepingFrequency	no	0-24	1	How often Zabbix will perform housekeeping procedure (in hours). Housekeeping is removing outdated information from the database. *Note*: To prevent housekeeper from being overloaded (for example, when configuration parameters ProxyLocalBuffer or ProxyOfflineBuffer are greatly reduced), no more than 4 times HousekeepingFrequency hours of outdated information are deleted in one housekeeping cycle. 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 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 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.
Include	no			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. For example: `/absolute/path/to/config/files/*.conf`. See special notes about limitations.
JavaGateway	no			IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started.
JavaGatewayPort	no	1024-32767	10052	Port that Zabbix Java gateway listens on.

Parameter	Mandatory	Range	Default	Description
ListenBacklog	no	0 - INT_MAX	SOMAXCONN	The maximum number of pending connections in the TCP queue. Default value is a hard-coded constant, which depends on the system. Maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.
ListenIP	no		0.0.0.0	List of comma delimited IP addresses that the trapper should listen on. Trapper will listen on all network interfaces if this parameter is missing.
ListenPort	no	1024-32767	10051	Listen port for trapper.
LoadModule	no			Module to load at proxy startup. Modules are used to extend functionality of the proxy. Formats: LoadModule=<module.so> LoadModule=<path/module.so> LoadModule=</abs_path/module.so> Either the module must be located in 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.
LoadModulePath	no			Full path to location of proxy modules. Default depends on compilation options.
LogFile	yes, if LogType is set to *file*, otherwise no			Name of log file.
LogFileSize	no	0-1024	1	Maximum size of 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.
LogRemoteCommands	no		0	Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled This parameter is supported since Zabbix 3.4.0.
LogType	no		file	Log output type: *file* - write log to file specified by LogFile parameter, *system* - write log to syslog, *console* - write log to standard output. This parameter is supported since Zabbix 3.0.0.
LogSlowQueries	no	0-3600000	0	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.
PidFile	no		/tmp/zabbix_proxy.pid	Name of PID file.
ProxyLocalBuffer	no	0-720	0	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.
ProxyMode	no	0-1	0	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.

Parameter	Mandatory	Range	Default	Description
ProxyOfflineBuffer	no	1-720	1	Proxy will keep data for N hours in case of no connectivity with Zabbix server. Older data will be lost.
Server	yes			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
SNMPTrapperFile	no		/tmp/zabbix_traps.tmp	Temporary file used for passing data from SNMP trap daemon to the proxy. Must be the same as in zabbix_trap_receiver.pl or SNMPTT configuration file.
SocketDir	no		/tmp	Directory to store IPC sockets used by internal Zabbix services. This parameter is supported since Zabbix 3.4.0.
SourceIP	no			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	no			Location of public and private keys for SSH checks and actions
SSLCertLocation	no			Location of SSL client certificate files for client authentication. This parameter is used in web monitoring only.
SSLKeyLocation	no			Location of SSL private key files for client authentication. This parameter is used in web monitoring only.
SSLCALocation	no			Location of certificate authority (CA) files for SSL server certificate verification. 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 since Zabbix 2.4.0 and in SMTP authentication since Zabbix 3.0.0.
StartDBSyncers	no	1-100	4	Number of pre-forked instances of history syncers. *Note*: Be careful when changing this value, increasing it may do more harm than good.
StartDiscoverers	no	0-250	1	Number of pre-forked instances of discoverers.
StartHTTPPollers	no	0-1000	1	Number of pre-forked instances of HTTP pollers.
StartIPMIPollers	no	0-1000	0	Number of pre-forked instances of IPMI pollers.
StartJavaPollers	no	0-1000	0	Number of pre-forked instances of Java pollers.
StartODBCPollers	no	0-1000	1	Number of pre-forked instances of ODBC pollers.
StartPingers	no	0-1000	1	Number of pre-forked instances of ICMP pingers.
StartPollersUnreachable	no	0-1000	1	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.

Parameter	Mandatory	Range	Default	Description
StartPollers	no	0-1000	5	Number of pre-forked instances of pollers.
StartPreprocessors	no	1-1000	3	Number of pre-forked instances of preprocessing workers[1]. The preprocessing manager process is automatically started when a preprocessor worker is started. This parameter is supported since Zabbix 4.2.0.
StartSNMPTrapper	no	0-1	0	If set to 1, SNMP trapper process will be started.
StartTrappers	no	0-1000	5	Number of pre-forked instances of trappers. Trappers accept incoming connections from Zabbix sender and active agents.
StartVMwareCollectors	no	0-250	0	Number of pre-forked VMware collector instances.
StatsAllowedIP	no			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 This parameter is supported since Zabbix 4.2.0.
Timeout	no	1-30	3	Specifies how long we wait for agent, SNMP device or external check (in seconds).
TLSAccept	yes for passive proxy, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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 This parameter is supported since Zabbix 3.0.0.
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSCertFile	no			Full pathname of a file containing the proxy certificate or certificate chain, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSCipherAll	no			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_1 This parameter is supported since Zabbix 4.4.7.
TLSCipherAll13	no			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:+COMP-NULL::+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+A This parameter is supported since Zabbix 4.4.7.

Parameter	Mandatory	Range	Default	Description
TLSCipherCert	no			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:+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128 This parameter is supported since Zabbix 4.4.7.
TLSCipherCert13	no			Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption. This parameter is supported since Zabbix 4.4.7.
TLSCipherPSK	no			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:+SIGN-ALL Example for OpenSSL: kECDHEPSK+AES128:kPSK+AES128 This parameter is supported since Zabbix 4.4.7.
TLSCipherPSK13	no			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 This parameter is supported since Zabbix 4.4.7.
TLSConnect	yes for active proxy, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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 This parameter is supported since Zabbix 3.0.0.
TLSCRLFile	no			Full pathname of a file containing revoked certificates.This parameter is used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSKeyFile	no			Full pathname of a file containing the proxy private key, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSPSKFile	no			Full pathname of a file containing the proxy pre-shared key. used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.
TLSServerCertIssuer	no			Allowed server certificate issuer. This parameter is supported since Zabbix 3.0.0.
TLSServerCertSubject	no			Allowed server certificate subject. This parameter is supported since Zabbix 3.0.0.
TmpDir	no		/tmp	Temporary directory.
TrapperTimeout	no	1-300	300	Specifies how many seconds trapper may spend processing new data.
User	no		zabbix	Drop privileges to a specific, existing user on the system. Only has effect if run as 'root' and AllowRoot is disabled.

Parameter	Mandatory	Range	Default	Description
UnavailableDelay	no	1-3600	60	How often host is checked for availability during the unavailability period, in seconds.
UnreachableDelay	no	1-3600	15	How often host is checked for availability during the unreachability period, in seconds.
UnreachablePeriod	no	1-3600	45	After how many seconds of unreachability treat a host as unavailable.
Vault	no		HashiCorp	Specifies secret management tool: *HashiCorp* - HashiCorp KV Secrets Engine version 2 *CyberArk* - CyberArk Central Credential Provider Must match the vault provider set in the frontend.
VaultDBPath	no			Specifies a location, from where database credentials should be retrieved by keys. Depending on the vault, can be vault path or query. Keys used for HashiCorp are 'password' and 'username'. Example: secret/zabbix/database Keys used for CyberArk are 'Content' and 'UserName'. Example: AppID=zabbix_server&Query=Safe=passwordSafe;Object=zabbix_proxy_data This option can only be used if DBUser and DBPassword are not specified.
VaultTLSCertFile	no			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	no			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	yes, if Vault is set to *HashiCorp*, otherwise no			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 VAULT_TOKEN environment variable are defined at the same time.
VaultURL	no		https://127.0.0.1:8200	Vault server HTTP[S] URL. System-wide CA certificates directory will be used if SSLCALocation is not specified.
VMwareCacheSize	no	256K-2G	8M	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.
VMwareFrequency	no	10-86400	60	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.
VMwarePerfFrequency	no	10-86400	60	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.
VMwareTimeout	no	1-300	10	The maximum number of seconds vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor).

3 Zabbix agent (UNIX)

Overview

This section lists parameters supported in a Zabbix agent configuration file (zabbix_agentd.conf).

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.

Parameters

Parameter	Mandatory	Range	Default	Description
Alias	no			Sets an alias for an item key. It can be used to substitute long and complex item key with a smaller and simpler one. Multiple *Alias* parameters may be present. Multiple parameters with the same *Alias* key are allowed. Different *Alias* keys may reference the same item key. Aliases can be used in *HostMetadataItem* but not in *HostnameItem* parameters. Examples: 1. Retrieving the ID of user 'zabbix'. Alias=zabbix.userid:vfs.file.regexp[/etc/passwd,"^zabbix:.:([0-9]+)"„„\1] Now shorthand key **zabbix.userid** may be used to retrieve data. 2. Getting CPU utilization with default and custom parameters. Alias=cpu.util:system.cpu.util Alias=cpu.util[*]:system.cpu.util[*] This allows use **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. 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	no			Allow execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports "*" 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. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.
AllowRoot	no		0	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. 0 - do not allow 1 - allow
BufferSend	no	1-3600	5	Do not keep data longer than N seconds in buffer.
BufferSize	no	2-65535	100	Maximum number of values in a memory buffer. The agent will send all collected data to Zabbix server or proxy if the buffer is full.

Parameter	Mandatory	Range	Default	Description
DebugLevel	no	0-5	3	Specifies 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)
DenyKey	no			Deny execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports "*" 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. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.
EnableRemoteCommands	no		0	Whether remote commands from Zabbix server are allowed. This parameter is **deprecated**, use AllowKey=system.run[*] or DenyKey=system.run[*] instead It is internal alias for AllowKey/DenyKey parameters depending on value: 0 - DenyKey=system.run[*] 1 - AllowKey=system.run[*]
HeartbeatFrequency	no	0-3600	60	Frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks. 0 - heartbeat messages disabled.
HostInterface	no	0-255 characters		Optional parameter that defines host interface. Host interface is used at host autoregistration process. An agent will issue an error and not start if the value is over the limit of 255 characters. If not defined, value will be acquired from HostInterfaceItem. Supported since Zabbix 4.4.0.
HostInterfaceItem	no			Optional parameter that defines an item used for getting host interface. Host interface is used at host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by specified item is over 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. Supported since Zabbix 4.4.0.
HostMetadata	no	0-255 characters		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. An agent will issue an error and not start if the specified value is over the limit or a non-UTF-8 string.
HostMetadataItem	no			Optional parameter that defines a *Zabbix agent* item used for getting host metadata. This option is only used when HostMetadata is not defined. Supports UserParameters and aliases. Supports *system.run[]* regardless of AllowKey/DenyKey values. HostMetadataItem value is retrieved on each autoregistration attempt and is used only at host autoregistration process (active agent). During an autoregistration request an agent will log a warning message if the value returned by the specified item is over the limit of 255 characters. The value returned by the item must be a UTF-8 string otherwise it will be ignored.

Parameter	Mandatory	Range	Default	Description
Hostname	no		Set by HostnameItem	List of comma-delimited unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. Value is acquired from HostnameItem if undefined. Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128 characters per hostname, 2048 characters for the entire line.
HostnameItem	no		system.hostname	Optional parameter that defines a *Zabbix agent* item used for getting host name. This option is only used when Hostname is not defined. Does not support UserParameters or aliases, but does support *system.run[]* regardless of AllowKey/DenyKey values. The output length is limited to 512KB.
Include	no			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. For example: `/absolute/path/to/config/files/*.conf`. See special notes about limitations.
ListenBacklog	no	0 - INT_MAX	SOMAXCONN	The maximum number of pending connections in the TCP queue. Default value is a hard-coded constant, which depends on the system. Maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.
ListenIP	no		0.0.0.0	List of comma delimited IP addresses that the agent should listen on. Multiple IP addresses are supported in version 1.8.3 and higher.
ListenPort	no	1024-32767	10050	Agent will listen on this port for connections from the server.
LoadModule	no			Module to load at agent startup. Modules are used to extend functionality of the agent. Formats: LoadModule=<module.so> LoadModule=<path/module.so> LoadModule=</abs_path/module.so> Either the module must be located in 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.
LoadModulePath	no			Full path to location of agent modules. Default depends on compilation options.
LogFile	yes, if LogType is set to *file*, otherwise no			Name of log file.
LogFileSize	no	0-1024	1	Maximum size of 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.
LogType	no		file	Log output type: *file* - write log to file specified by LogFile parameter, *system* - write log to syslog, *console* - write log to standard output. This parameter is supported since Zabbix 3.0.0.

Parameter	Mandatory	Range	Default	Description
LogRemoteCommands	no		0	Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled 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.
MaxLinesPerSecond	no	1-1000	20	Maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log' 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.
PidFile	no		/tmp/zabbix_agentd.pid	Name of PID file.
RefreshActiveChecks	no	60-3600	120	How often list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted after 60 seconds.
Server	yes, if StartAgents is not explicitly set to 0			List of comma delimited IP addresses, optionally in CIDR notation, or hostnames 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. Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.domain Spaces are allowed.
ServerActive	no			Zabbix server/proxy address or cluster configuration to get active checks from. Server/proxy address is 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 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.clus Example for high availability with two clusters and one server: ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.clus

Parameter	Mandatory	Range	Default	Description
SourceIP	no			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	no	0-100	3	Number of pre-forked instances of zabbix_agentd that process passive checks. If set to 0, disables passive checks and the agent will not listen on any TCP port.
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing.
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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 This parameter is supported since Zabbix 3.0.0.
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSCertFile	no			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSCipherAll	no			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_1 This parameter is supported since Zabbix 4.4.7.
TLSCipherAll13	no			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:+COMP-NULL::+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128:kECDHEPSK+AES128:kPSK+A This parameter is supported since Zabbix 4.4.7.
TLSCipherCert	no			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:+SIGN-ALL:+CTYPE-X.509 Example for OpenSSL: EECDH+aRSA+AES128:RSA+aRSA+AES128 This parameter is supported since Zabbix 4.4.7.
TLSCipherCert13	no			Cipher string for OpenSSL 1.1.1 or newer in TLS 1.3. Override the default ciphersuite selection criteria for certificate-based encryption. This parameter is supported since Zabbix 4.4.7.

Parameter	Mandatory	Range	Default	Description
TLSCipherPSK	no			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:+SIGN-ALL Example for OpenSSL: kECDHEPSK+AES128:kPSK+AES128 This parameter is supported since Zabbix 4.4.7.
TLSCipherPSK13	no			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 This parameter is supported since Zabbix 4.4.7.
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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 This parameter is supported since Zabbix 3.0.0.
TLSCRLFile	no			Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSKeyFile	no			Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSPSKFile	no			Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.
TLSServerCertIssuer	no			Allowed server (proxy) certificate issuer. This parameter is supported since Zabbix 3.0.0.
TLSServerCertSubject	no			Allowed server (proxy) certificate subject. This parameter is supported since Zabbix 3.0.0.
UnsafeUserParameters	no	0,1	0	Allow all characters to be passed in arguments to user-defined parameters. 0 - do not allow 1 - allow The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) > | # @ Additionally, newline characters are not allowed.
User	no		zabbix	Drop privileges to a specific, existing user on the system. Only has effect if run as 'root' and AllowRoot is disabled.
UserParameter	no			User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=<key>,<shell command> Note that shell command must not return empty string or EOL only. Shell commands may have relative paths, if UserParameterDir parameter is specified. Examples: UserParameter=system.test,who|wc -l UserParameter=check_cpu,./custom_script.sh

Parameter	Mandatory	Range	Default	Description
UserParameterDir	no			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.

This section lists parameters supported in a Zabbix agent 2 configuration file (zabbix_agent2.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 in the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Alias	no			Sets an alias for an item key. It can be used to substitute long and complex item key with a smaller and simpler one. Multiple *Alias* parameters may be present. Multiple parameters with the same *Alias* key are allowed. Different *Alias* keys may reference the same item key. Aliases can be used in *HostMetadataItem* but not in *HostnameItem* parameters. Examples: 1. Retrieving the ID of user ‘zabbix’. Alias=zabbix.userid:vfs.file.regexp[/etc/passwd,”^zabbix:.:([0-9]+)”,,,\1] Now shorthand key **zabbix.userid** may be used to retrieve data. 2. Getting CPU utilization with default and custom parameters. Alias=cpu.util:system.cpu.util Alias=cpu.util[*]:system.cpu.util[*] This allows use **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. 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.

Parameter	Mandatory	Range	Default	Description
AllowKey	no			Allow execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports "*" 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. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.
BufferSend	no	1-3600	5	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.
BufferSize	no	2-65535	100	Maximum number of values in a memory buffer. The agent will send all collected data to Zabbix server or proxy if the buffer is full. This parameter should only be used if persistent buffer is disabled (*EnablePersistentBuffer=0*).
ControlSocket	no		/tmp/agent.sock	The control socket, used to send runtime commands with '-R' option.
DebugLevel	no	0-5	3	Specifies 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)
DenyKey	no			Deny execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports "*" 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. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.
EnablePersistentBuffer	no	0-1	0	Enable usage of local persistent storage for active items. 0 - disabled 1 - enabled If persistent storage is disabled, the memory buffer will be used.
ForceActiveChecksOnStart	no	0-1	0	Perform active checks immediately after restart for the first received configuration. 0 - disabled 1 - enabled Also available as per plugin configuration parameter, for example: `Plugins.Uptime.System.ForceActiveChecksOnStart=1`
HeartbeatFrequency	no	0-3600	60	Frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks. 0 - heartbeat messages disabled.
HostInterface	no	0-255 characters		Optional parameter that defines host interface. Host interface is used at host autoregistration process. An agent will issue an error and not start if the value is over the limit of 255 characters. If not defined, value will be acquired from HostInterfaceItem. Supported since Zabbix 4.4.0.

Parameter	Mandatory	Range	Default	Description
HostInterfaceItem	no			Optional parameter that defines an item used for getting host interface. Host interface is used at host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by specified item is over 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. Supported since Zabbix 4.4.0.
HostMetadata	no	0-255 characters		Optional parameter that defines host metadata. Host metadata is used at host autoregistration process. An agent will issue an error and not start if the specified value is over the limit or a non-UTF-8 string. If not defined, the value will be acquired from HostMetadataItem.
HostMetadataItem	no			Optional parameter that defines an item used for getting host metadata. Host metadata item value is retrieved on each autoregistration attempt for host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by the specified item is over the limit of 255 characters. This option is only used when HostMetadata is not defined. Supports UserParameters and aliases. Supports *system.run[]* regardless of AllowKey/DenyKey values. The value returned by the item must be a UTF-8 string otherwise it will be ignored.
Hostname	no		Set by HostnameItem	List of comma-delimited unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. Value is acquired from HostnameItem if undefined. Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128 characters per hostname, 2048 characters for the entire line.
HostnameItem	no		system.hostname	Item used for generating Hostname if it is not defined. Ignored if Hostname is defined. Does not support UserParameters or aliases, but does support *system.run[]* regardless of AllowKey/DenyKey values. The output length is limited to 512KB.
Include	no			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. To only include relevant files in the specified directory, the asterisk wildcard character is supported for pattern matching. For example: `/absolute/path/to/config/files/*.conf`. Since Zabbix 6.0.0 a path can be relative to *zabbix_agent2.conf* file location. See special notes about limitations.
ListenIP	no		0.0.0.0	List of comma-delimited IP addresses that the agent should listen on. The first IP address is sent to Zabbix server, if connecting to it, to retrieve the list of active checks.
ListenPort	no	1024-32767	10050	Agent will listen on this port for connections from the server.

Parameter	Mandatory	Range	Default	Description
LogFile	yes, if LogType is set to *file*, otherwise no		/tmp/zabbix_agent2.log	Log file name if LogType is 'file'.
LogFileSize	no	0-1024	1	Maximum size of 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.
LogType	no		file	Specifies where log messages are written to: *system* - syslog, *file* - file specified by LogFile parameter, *console* - standard output.
PersistentBufferFile	no			The file, where Zabbix Agent2 should keep SQLite database. Must be a full filename. This parameter is only used if persistent buffer is enabled (*EnablePersistentBuffer=1*).
PersistentBufferPeriod	no	1m-365d	1h	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*).
PidFile	no		/tmp/zabbix_agent2.pid	Name of PID file.
Plugin	no			Since Zabbix 6.0.0 most of the plugins have their own configuration files. The agent configuration file contains plugin parameters listed below.
Plugins.Log.MaxLinesPerSecond	no	1-1000	20	Maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log' 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. This parameter is supported since 4.4.2 and replaces MaxLinesPerSecond.
Plugins.System.Run.LogRemoteCommands	no		0	Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled 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. This parameter is supported since 4.4.2 and replaces LogRemoteCommands.
PluginSocket	no		/tmp/agent.plugin.sock	Path to unix socket for loadable plugin communications.
PluginTimeout	no	1-30	Global timeout	Timeout for connections with loadable plugins.
RefreshActiveChecks	no	60-3600	120	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 after 60 seconds.

Parameter	Mandatory	Range	Default	Description
Server	yes			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', '::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: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com Spaces are allowed.
ServerActive	no			Zabbix server/proxy address or cluster configuration to get active checks from. Server/proxy address is 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 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.clu Example for high availability with two clusters and one server: ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.clu
SourceIP	no			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	no	1024-32767		If set, agent will listen on this port for HTTP status requests (http://localhost:<port>/status).
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing.
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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

Parameter	Mandatory	Range	Default	Description
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	no			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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
TLSCRLFile	no			Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components.
TLSKeyFile	no			Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components.
TLSPSKFile	no			Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components.
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server.
TLSServerCertIssuer	no			Allowed server (proxy) certificate issuer.
TLSServerCertSubject	no			Allowed server (proxy) certificate subject.
UnsafeUserParameters	no	0,1	0	Allow all characters to be passed in arguments to user-defined parameters. The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) > | # @ Additionally, newline characters are not allowed.
UserParameter	no			User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=<key>,<shell command> Note that shell command must not return empty string or EOL only. Shell commands may have relative paths, if UserParameterDir parameter is specified. Examples: UserParameter=system.test,who|wc -l UserParameter=check_cpu,./custom_script.sh
UserParameterDir	no			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

This section lists parameters supported in a Zabbix agent (Windows) configuration file (zabbix_agent.conf).

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.

Parameters

Parameter	Mandatory	Range	Default	Description
Alias	no			Sets an alias for an item key. It can be used to substitute long and complex item key with a smaller and simpler one. Multiple *Alias* parameters may be present. Multiple parameters with the same *Alias* key are allowed. Different *Alias* keys may reference the same item key. Aliases can be used in *HostMetadataItem* but not in *HostnameItem* or *PerfCounter* parameters. Examples: 1. Retrieving paging file usage in percents from the server. Alias=pg_usage:perf_counter[\Paging File(_Total)\% Usage] Now shorthand key **pg_usage** may be used to retrieve data. 2. Getting 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 CPU utilization percentage with default parameters as well as use **cpu.load[percpu,avg15]** to get specific data about CPU load. 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	no			Allow execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports ”*” 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. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.
BufferSend	no	1-3600	5	Do not keep data longer than N seconds in buffer.
BufferSize	no	2-65535	100	Maximum number of values in a memory buffer. The agent will send all collected data to Zabbix server or proxy if the buffer is full.
DebugLevel	no	0-5	3	Specifies 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)
DenyKey	no			Deny execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports ”*” 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. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.

Parameter	Mandatory	Range	Default	Description
EnableRemoteCommands	no		0	Whether remote commands from Zabbix server are allowed. This parameter is **deprecated**, use AllowKey=system.run[*] or DenyKey=system.run[*] instead It is internal alias for AllowKey/DenyKey parameters depending on value: 0 - DenyKey=system.run[*] 1 - AllowKey=system.run[*].
HeartbeatFrequency	no	0-3600	60	Frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks. 0 - heartbeat messages disabled.
HostInterface	no	0-255 characters		Optional parameter that defines host interface. Host interface is used at host autoregistration process. An agent will issue an error and not start if the value is over the limit of 255 characters. If not defined, value will be acquired from HostInterfaceItem. Supported since Zabbix 4.4.0.
HostInterfaceItem	no			Optional parameter that defines an item used for getting host interface. Host interface is used at host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by specified item is over 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. Supported since Zabbix 4.4.0.
HostMetadata	no	0-255 characters		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. An agent will issue an error and not start if the specified value is over the limit or a non-UTF-8 string.
HostMetadataItem	no			Optional parameter that defines a *Zabbix agent* item used for getting host metadata. This option is only used when HostMetadata is not defined. Supports UserParameters, performance counters and aliases. Supports *system.run[]* regardless of *EnableRemoteCommands* value. HostMetadataItem value is retrieved on each autoregistration attempt and is used only at host autoregistration process (active agent). During an autoregistration request an agent will log a warning message if the value returned by the specified item is over the limit of 255 characters. The value returned by the item must be a UTF-8 string otherwise it will be ignored.
Hostname	no		Set by HostnameItem	List of comma-delimited unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. Value is acquired from HostnameItem if undefined. Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128 characters per hostname, 2048 characters for the entire line.
HostnameItem	no		system.hostname	Optional parameter that defines a *Zabbix agent* item used for getting host name. This option is only used when Hostname is not defined. Does not support UserParameters, performance counters or aliases, but does support *system.run[]* regardless of *EnableRemoteCommands* value. The output length is limited to 512KB. See also a more detailed description.

Parameter	Mandatory	Range	Default	Description
Include	no			You may include individual files or all files in a directory in the configuration file. 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. For example: `C:\Program Files\Zabbix Agent\zabbix_agentd.d\*.conf`. See special notes about limitations.
ListenBacklog	no	0 - INT_MAX	SOMAXCONN	The maximum number of pending connections in the TCP queue. Default value is a hard-coded constant, which depends on the system. Maximum supported value depends on the system, too high values may be silently truncated to the 'implementation-specified maximum'.
ListenIP	no		0.0.0.0	List of comma-delimited IP addresses that the agent should listen on.
ListenPort	no	1024-32767	10050	Agent will listen on this port for connections from the server.
LogFile	yes, if LogType is set to *file*, otherwise no		C:\zabbix_agentd.log	Name of the agent log file.
LogFileSize	no	0-1024	1	Maximum size of 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.
LogType	no		file	Log output type: *file* - write log to file specified by LogFile parameter, *system* - write log Windows Event Log, *console* - write log to standard output. This parameter is supported since Zabbix 3.0.0.
LogRemoteCommands	no		0	Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled
MaxLinesPerSecond	no	1-1000	20	Maximum number of new lines the agent will send per second to Zabbix server or proxy processing 'log', 'logrt' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log', 'logrt' or 'eventlog' item keys. *Note*: Zabbix will process 10 times more new lines than set in *MaxLinesPerSecond* to seek the required string in log items.
PerfCounter	no			Defines a new parameter <parameter_name> which is an 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 average number of processor interrupts per second for last minute, you can define a new parameter "interrupts" as the following: PerfCounter = interrupts,"\Processor(0)\Interrupts/sec",60 Please note double quotes around performance counter path. The parameter name (interrupts) is to be used as the item key when creating an item. Samples for calculating average value will be taken every second. You may run "typeperf -qx" to get list of all performance counters available in Windows.

Parameter	Mandatory	Range	Default	Description
PerfCounterEn	no			Defines a new parameter <parameter_name> which is an average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds). Syntax: <parameter_name>,”<perf_counter_path>”,<period> Compared to PerfCounter, perfcounter paths must be in English. Supported only on **Windows Server 2008/Vista** and above. For example, if you wish to receive average number of processor interrupts per second for last minute, you can define a new parameter ”interrupts” as the following: PerfCounterEn = interrupts,”\Processor(0)\Interrupts/sec”,60 Please note double quotes around performance counter path. The parameter name (interrupts) is to be used as the item key when creating an item. Samples for calculating 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`. This parameter is supported since Zabbix 4.0.13 and 4.2.7.
RefreshActiveChecks	no	60-3600	120	How often list of active checks is refreshed, in seconds. Note that after failing to refresh active checks the next refresh will be attempted after 60 seconds.
Server	yes, if StartAgents is not explicitly set to 0			List of comma delimited IP addresses, optionally in CIDR notation, or hostnames of Zabbix servers. 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: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.domain Spaces are allowed.

Parameter	Mandatory	Range	Default	Description
ServerActive	no	(*)		Zabbix server/proxy address or cluster configuration to get active checks from. Server/proxy address is 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 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.clus Example for high availability with two clusters and one server: ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.clus
SourceIP	no			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	no	0-63 (*)	3	Number of pre-forked instances of zabbix_agentd that process passive checks. If set to 0, disables passive checks and the agent will not listen on any TCP port.
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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 This parameter is supported since Zabbix 3.0.0.
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSCertFile	no			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.

Parameter	Mandatory	Range	Default	Description
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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 This parameter is supported since Zabbix 3.0.0.
TLSCRLFile	no			Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSKeyFile	no			Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSPSKFile	no			Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components. This parameter is supported since Zabbix 3.0.0.
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server. This parameter is supported since Zabbix 3.0.0.
TLSServerCertIssuer	no			Allowed server (proxy) certificate issuer. This parameter is supported since Zabbix 3.0.0.
TLSServerCertSubject	no			Allowed server (proxy) certificate subject. This parameter is supported since Zabbix 3.0.0.
UnsafeUserParameters	no	0-1	0	Allow all characters to be passed in arguments to user-defined parameters. 0 - do not allow 1 - allow The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) > | # @ Additionally, newline characters are not allowed.
UserParameter	no			User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=<key>,<shell command> Note that shell command must not return empty string or EOL only. Shell commands may have relative paths, if UserParameterDir parameter is specified. Examples: UserParameter=system.test,who|wc -l UserParameter=check_cpu,./custom_script.sh
UserParameterDir	no			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.

This section lists parameters supported in a Zabbix agent 2 configuration file (zabbix_agent2.win.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 in the beginning of the line.

Parameters

Parameter	Mandatory	Range	Default	Description
Alias	no			Sets an alias for an item key. It can be used to substitute long and complex item key with a smaller and simpler one. Multiple *Alias* parameters may be present. Multiple parameters with the same *Alias* key are allowed. Different *Alias* keys may reference the same item key. Aliases can be used in *HostMetadataItem* but not in *HostnameItem* parameters. Examples: 1. Retrieving the ID of user 'zabbix'. Alias=zabbix.userid:vfs.file.regexp[/etc/passwd,"^zabbix:.:([0-9]+)"„„\1] Now shorthand key **zabbix.userid** may be used to retrieve data. 2. Getting CPU utilization with default and custom parameters. Alias=cpu.util:system.cpu.util Alias=cpu.util[*]:system.cpu.util[*] This allows use **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. 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	no			Allow execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports "*" 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. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.
BufferSend	no	1-3600	5	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.
BufferSize	no	2-65535	100	Maximum number of values in a memory buffer. The agent will send all collected data to Zabbix server or proxy if the buffer is full. This parameter should only be used if persistent buffer is disabled (*EnablePersistentBuffer=0*).

Parameter	Mandatory	Range	Default	Description
ControlSocket	no		\\.\pipe\agent.sock	The control socket, used to send runtime commands with '-R' option.
DebugLevel	no	0-5	3	Specifies 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)
DenyKey	no			Deny execution of those item keys that match a pattern. Key pattern is a wildcard expression that supports "*" 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. This parameter is supported since Zabbix 5.0.0. See also: Restricting agent checks.
EnablePersistentBuffer	no	0-1	0	Enable usage of local persistent storage for active items. 0 - disabled 1 - enabled If persistent storage is disabled, the memory buffer will be used.
ForceActiveChecksOnStart	no	0-1	0	Perform active checks immediately after restart for the first received configuration. 0 - disabled 1 - enabled Also available as per plugin configuration parameter, for example: `Plugins.Uptime.System.ForceActiveChecksOnStart=1`
HeartbeatFrequency	no	0-3600	60	Frequency of heartbeat messages in seconds. Used for monitoring the availability of active checks. 0 - heartbeat messages disabled.
HostInterface	no	0-255 characters		Optional parameter that defines host interface. Host interface is used at host autoregistration process. An agent will issue an error and not start if the value is over the limit of 255 characters. If not defined, value will be acquired from HostInterfaceItem. Supported since Zabbix 4.4.0.
HostInterfaceItem	no			Optional parameter that defines an item used for getting host interface. Host interface is used at host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by specified item is over 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. Supported since Zabbix 4.4.0.
HostMetadata	no	0-255 characters		Optional parameter that defines host metadata. Host metadata is used at host autoregistration process. An agent will issue an error and not start if the specified value is over the limit or a non-UTF-8 string. If not defined, the value will be acquired from HostMetadataItem.

Parameter	Mandatory	Range	Default	Description
HostMetadataItem	no			Optional parameter that defines an item used for getting host metadata. Host metadata item value is retrieved on each autoregistration attempt for host autoregistration process. During an autoregistration request an agent will log a warning message if the value returned by the specified item is over the limit of 255 characters. This option is only used when HostMetadata is not defined. Supports UserParameters and aliases. Supports *system.run[]* regardless of *EnableRemoteCommands* value. The value returned by the item must be a UTF-8 string otherwise it will be ignored.
Hostname	no		Set by HostnameItem	List of comma-delimited unique, case-sensitive hostnames. Required for active checks and must match hostnames as configured on the server. Value is acquired from HostnameItem if undefined. Allowed characters: alphanumeric, '.', ' ', '_' and '-'. Maximum length: 128 characters per hostname, 2048 characters for the entire line.
HostnameItem	no		system.hostname	Item used for generating Hostname if it is not defined. Ignored if Hostname is defined. Does not support UserParameters or aliases, but does support *system.run[]* regardless of *EnableRemoteCommands* value. The output length is limited to 512KB.
Include	no			You may include individual files or all files in a directory in the configuration file. 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. For example: `C:\Program Files\Zabbix Agent2\zabbix_agent2.d\*.conf`. See special notes about limitations.
ListenIP	no		0.0.0.0	List of comma-delimited IP addresses that the agent should listen on. The first IP address is sent to Zabbix server, if connecting to it, to retrieve the list of active checks.
ListenPort	no	1024-32767	10050	Agent will listen on this port for connections from the server.
LogFile	yes, if LogType is set to *file*, otherwise no		c:\zabbix_agent2.log	Log file name if LogType is 'file'.
LogFileSize	no	0-1024	1	Maximum size of 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.
LogType	no		file	Specifies where log messages are written to: *file* - file specified by LogFile parameter, *console* - standard output.
PersistentBufferFile	no			The file, where Zabbix Agent2 should keep SQLite database. Must be a full filename. This parameter is only used if persistent buffer is enabled (*EnablePersistentBuffer=1*).

Parameter	Mandatory	Range	Default	Description
PersistentBufferPeriod	no	1m-365d	1h	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*).
Plugins	no			Since Zabbix 6.0.0 most of the plugins have their own configuration files. The agent configuration file contains plugin parameters listed below.
Plugins.Log.MaxLinesPerSecond	no	1-1000	20	Maximum number of new lines the agent will send per second to Zabbix server or proxy when processing 'log' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log' 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. This parameter is supported since 4.4.2 and replaces MaxLinesPerSecond.
Plugins.SystemRun.LogRemoteCommands	no		0	Enable logging of executed shell commands as warnings. 0 - disabled 1 - enabled 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. This parameter is supported since 4.4.2 and replaces LogRemoteCommands.
PluginSocket	no		\\.\pipe\agent.plugin.sock	Path to Unix socket for loadable plugin communications.
PluginTimeout	no	1-30	Global timeout	Timeout for connections with loadable plugins.
RefreshActiveChecks	no	60-3600	120	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 after 60 seconds.
Server	yes			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', '::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: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com Spaces are allowed.

Parameter	Mandatory	Range	Default	Description
ServerActive	no			Zabbix server/proxy address or cluster configuration to get active checks from. Server/proxy address is 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 addresses can be provided to use several independent Zabbix servers in parallel. Spaces are allowed. If 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.clu Example for high availability with two clusters and one server: ServerActive=zabbix.cluster.node1;zabbix.cluster.node2:20051,zabbix.clu
SourceIP	no			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	no	1024-32767		If set, agent will listen on this port for HTTP status requests (http://localhost:<port>/status).
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing.
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	no			Full pathname of a file containing the agent certificate or certificate chain, used for encrypted communications with Zabbix components.

Parameter	Mandatory	Range	Default	Description
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for *unencrypted* connection), otherwise no			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
TLSCRLFile	no			Full pathname of a file containing revoked certificates. This parameter is used for encrypted communications with Zabbix components.
TLSKeyFile	no			Full pathname of a file containing the agent private key used for encrypted communications with Zabbix components.
TLSPSKFile	no			Full pathname of a file containing the agent pre-shared key used for encrypted communications with Zabbix components.
TLSPSKIdentity	no			Pre-shared key identity string, used for encrypted communications with Zabbix server.
TLSServerCertIssuer	no			Allowed server (proxy) certificate issuer.
TLSServerCertSubject	no			Allowed server (proxy) certificate subject.
UnsafeUserParameters	no	0,1	0	Allow all characters to be passed in arguments to user-defined parameters. The following characters are not allowed: \ ' " ` * ? [ ] { } ~ $! & ; ( ) > | # @ Additionally, newline characters are not allowed.
UserParameter	no			User-defined parameter to monitor. There can be several user-defined parameters. Format: UserParameter=<key>,<shell command> Note that shell command must not return empty string or EOL only. Shell commands may have relative paths, if UserParameterDir parameter is specified. Examples: UserParameter=system.test,who|wc -l UserParameter=check_cpu,./custom_script.sh
UserParameterDir	no			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.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			https://localhost:8003	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 (since version 5.2.3). 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.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.

Plugins.Memcached.Sessions.<SessionName>.Uri			tcp://localhost:11211	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 (since version 5.2.3).
				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 seral 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).

Since Zabbix 6.2.7, MongoDB is a loadable plugin, which is available and fully described in the MongoDB plugin repository

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.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. Supported in plugin versions 1.2.1, 6.2.7 and newer[1].

Parameter	Mandatory	Range	Default	Description
Plugins.MongoDB.Sessions.<SessionName>.TLSCertFile	No (yes, 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. Supported in plugin versions 1.2.1, 6.2.7 and newer[1].
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. Accepted values: *required* - require TLS connection; *verify_ca* - verify certificates; *verify_full* - verify certificates and IP address. Supported in plugin versions 1.2.1, 6.2.7 and newer[1].
Plugins.MongoDB.Sessions.<SessionName>.TLSKeyFile	No (yes, 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. Supported in plugin versions 1.2.1, 6.2.7 and newer[1].
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

Footnotes

[1] - Since Zabbix 6.2.7, loadable plugins started using the same versioning system as Zabbix itself. As a result, MongoDB plugin version has changed from 1.2.1 to 6.2.7.

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.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

7 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	no	1-30	global timeout	The maximum amount of time in seconds to wait for a request to be done.
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				Named session password. **<SessionName>** - define name of a session for using in item keys.
Plugins.Mysql.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 monitored databases. **<SessionName>** - define name of a session for using in item keys.
Plugins.Mysql.Sessions.<SessionName>.TLSCertFile				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. Accepted values: *required* - require TLS connection; *verify_ca* - verify certificates; *verify_full* - verify certificates and IP address.
Plugins.Mysql.Sessions.<SessionName>.TLSKeyFile				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.

Parameter	Mandatory	Range	Default	Description

Plugins.Mysql.Sessions.<SessionName>.Uri			tcp://localhost:3306	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 (since version 5.2.3).
				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

8 Oracle plugin

Overview

This section lists parameters supported in the Oracle 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.Oracle.CallTimeout		1-30	global timeout	The maximum wait time in seconds for a request to be completed.
Plugins.Oracle.ConnectTimeout		1-30	global timeout	The maximum wait time in seconds for a connection to be established.
Plugins.Oracle.CustomQueriesPath				Full pathname of a directory containing .sql files with custom queries. Disabled by default. Example: `/etc/zabbix/oracle/sql`
Plugins.Oracle.KeepAlive		60-900	300	The maximum time of waiting (in seconds) before unused plugin connections are closed.
Plugins.Oracle.Sessions.<SessionName>.Password				Named session password. **<SessionName>** - define name of a session for using in item keys.
Plugins.Oracle.Sessions.<SessionName>.Service				Named session service name to be used for connection (SID is not supported). Supported for: *Oracle*. **<PluginName>** - name of the plugin. **<SessionName>** - define name of a session for using in item keys.

Parameter	Mandatory	Range	Default	Description
Plugins.Oracle.Sessions.<SessionName>.Uri			tcp://localhost:1521	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 (since version 5.2.3). A port can be omitted (default=1521). Examples: `tcp://127.0.0.1:1521` `localhost`
Plugins.Oracle.Sessions.<SessionName>.User				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 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.KeepAlive		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			tcp://localhost:6379	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 (since version 5.2.3).
				A port can be omitted (default=6379).
				Examples: `tcp://localhost:6379`
				`localhost`
				`unix:/var/run/redis.sock`

Plugins.Redis.Sessions.<SessionName>.User				Named session username. **<SessionName>** - define name of a session for using in item keys.
Plugins.Redis.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 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

Zabbix web service is a process that is used for communication with external web services.

This section lists parameters supported in Zabbix web service configuration file (zabbix_web_service.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
AllowedIP	yes			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.exampl`
DebugLevel	no	0-5	3	Specifies 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)
ListenPort	no	1024-32767	10053	The port service listens on for connections from the server.
LogFile	yes, if LogType is set to *file*, otherwise no			Log file name for LogType 'file' parameter. Example: `/tmp/zabbix_web_service.log`
LogFileSize	no	0-1024	1	Maximum size of log file in MB. 0 - disable automatic log rotation.
LogType	no	system / file / console	file	Specifies where log messages are written to: *system* - syslog *file* - file specified with LogFile parameter *console* - standard output
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing.
TLSAccept	no	unencrypted / cert	unencrypted	Specifies what type of connection to use: *unencrypted* - accept connections without encryption (default) *cert* - accept connections with TLS and a certificate
TLSCAFile	no			Full pathname of a file containing the top-level CA(s) certificates for peer certificate verification, used for encrypted communications between Zabbix components.
TLSCertFile	no			Full pathname of a file containing the service certificate or certificate chain, used for encrypted communications with Zabbix components.
TLSKeyFile	no			Full pathname of a file containing the service private key used for encrypted communications with 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).

4 Protocols

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

Proxy config request

The `proxy config` request is sent by server to provide proxy configuration data. This request is sent every `ProxyConfigFrequency` (server configuration parameter) seconds.

name	value type	description
server→proxy:		
request	*string*	'proxy config'
<table>	*object*	One or more objects with <table> data.
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.
proxy→server:		
response	*string*	Request success information ('success' or 'failed').
version	*string*	Proxy version (<major>.<minor>.<build>).

Example:

server→proxy:

```
{
    "request": "proxy config",
    "globalmacro":{
        "fields":[
            "globalmacroid",
            "macro",
            "value"
        ],
```

```
        "data":[
            [
                2,
                "{$SNMP_COMMUNITY}",
                "public"
            ]
        ]
    },
    "hosts":{
        "fields":[
            "hostid",
            "host",
            "status",
            "ipmi_authtype",
            "ipmi_privilege",
            "ipmi_username",
            "ipmi_password",
            "name",
            "tls_connect",
            "tls_accept",
            "tls_issuer",
            "tls_subject",
            "tls_psk_identity",
            "tls_psk"
        ],
        "data":[
            [
                10001,
                "Linux",
                3,
                -1,
                2,
                "",
                "",
                "Linux",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10050,
                "Zabbix Agent",
                3,
                -1,
                2,
                "",
                "",
                "Zabbix Agent",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10105,
                "Logger",
                0,
```

```
                -1,
                2,
                "",
                "",
                "Logger",
                1,
                1,
                "",
                "",
                "",
                ""
            ]
        ]
    },
    "interface":{
        "fields":[
            "interfaceid",
            "hostid",
            "main",
            "type",
            "useip",
            "ip",
            "dns",
            "port",
            "bulk"
        ],
        "data":[
            [
                2,
                10105,
                1,
                1,
                1,
                "127.0.0.1",
                "",
                "10050",
                1
            ]
        ]
    },
    ...
}
```

proxy→server:

```
{
  "response": "success",
  "version": "5.4.0"
}
```

Proxy 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.

name	value type	description
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	*array*	*(optional)* Array of history data objects.
data		
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*
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.

name	value type	description
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').
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":"5.4.0"
}
```


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

Proxy heartbeat request

The `proxy heartbeat` request is sent by proxy to report that proxy is running. This request is sent every `HeartbeatFrequency` (proxy configuration parameter) seconds.

name	value type	description
proxy→server:		
request	*string*	'proxy heartbeat'
host	*string*	Proxy name.
version	*string*	Proxy version (<major>.<minor>.<build>).
server→proxy:		
response	*string*	Request success information ('success' or 'failed').

proxy→server:

```
{
   "request": "proxy heartbeat",
   "host": "Proxy #12",
   "version": "5.4.0"
}
```

server→proxy:

```
{
  "response": "success"
}
```

Proxy config request

The `proxy config` request is sent by proxy to obtain proxy configuration data. This request is sent every `ConfigFrequency` (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>).
server→proxy:		
request	*string*	'proxy config'
<table>	*object*	One or more objects with <table> data.
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 the column type in database schema.
proxy→server:		
response	*string*	Request success information ('success' or 'failed').

Example:

proxy→server:

```
{
  "request": "proxy config",
  "host": "Proxy #12",
  "version":"5.4.0"
}
```

server→proxy:

```
{
    "globalmacro":{
        "fields":[
            "globalmacroid",
            "macro",
            "value"
        ],
        "data":[
            [
                2,
                "{$SNMP_COMMUNITY}",
                "public"
            ]
        ]
    },
    "hosts":{
        "fields":[
            "hostid",
            "host",
            "status",
            "ipmi_authtype",
            "ipmi_privilege",
            "ipmi_username",
            "ipmi_password",
            "name",
            "tls_connect",
            "tls_accept",
            "tls_issuer",
            "tls_subject",
            "tls_psk_identity",
            "tls_psk"
        ],
        "data":[
            [
                10001,
                "Linux",
                3,
                -1,
                2,
                "",
                "",
                "Linux",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10050,
                "Zabbix Agent",
                3,
                -1,
                2,
```

```
                "",
                "",
                "Zabbix Agent",
                1,
                1,
                "",
                "",
                "",
                ""
            ],
            [
                10105,
                "Logger",
                0,
                -1,
                2,
                "",
                "",
                "Logger",
                1,
                1,
                "",
                "",
                "",
                ""
            ]
        ]
    },
    "interface":{
        "fields":[
            "interfaceid",
            "hostid",
            "main",
            "type",
            "useip",
            "ip",
            "dns",
            "port",
            "bulk"
        ],
        "data":[
            [
                2,
                10105,
                1,
                1,
                1,
                "127.0.0.1",
                "",
                "10050",
                1
            ]
        ]
    },
    ...
}
```

proxy→server:

```
{
  "response": "success"
}
```

Proxy 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.

name	value type	description
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 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.

name	value type	description
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:

proxy→server:


```
{
    "request": "proxy data",
    "host": "Proxy #12",
    "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": 2,
            "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": 2,
            "clock":1478608381,
            "ttl": 600,
            "commandtype": 1,
            "command": "restart_service2.sh",
            "execute_on": 0,
            "authtype": 0,
            "username": "",
            "password": "",
            "publickey": "",
            "privatekey": "",
            "parent_taskid": 20,
            "hostid": 10084
        }
    ],
```

```
    "tasks":[
        {
            "type": 0,
            "status": 0,
            "parent_taskid": 10
        },
        {
            "type": 0,
            "status": 1,
            "error": "No permissions to execute task.",
            "parent_taskid": 20
        }
    ],
    "version":"5.4.0"
}
```

server→proxy:

```
{
  "response": "success",
  "upload": "enabled",
  "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
      }
  ]
}
```

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.
port	*number*	no	The configuration parameter ListenPort value if set and not default agent listening port.

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
}
```

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.
	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 sensitiviness setting.

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
    }
  ]
}
```


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.
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 Real-time export protocol

This section presents details of the real-time export protocol in a newline-delimited JSON format for:

- trigger events
- item values
- trends

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.
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.
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"],"itemid":3,"nam
```


Numeric (float) value:


```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"itemid":4,"nam
```


Character, text value:


```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"itemid":2,"nam
```


Log value:


```
{"host":{"host":"Host A","name":"Host A visible"},"groups":["Group X","Group Y","Group Z"],"itemid":1,"nam
```


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.
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"],"itemid":3,"nam
```


Numeric (float) value:


```
{"host":{"host":"Host B","name":"Host B visible"},"groups":["Group X","Group Y","Group Z"],"itemid":4,"nam
```


5 Items

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.

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
}
```

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.

Field	Type	Mandatory	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.

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
    }
  ]
}
```

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.
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.
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 sensitiviness setting.

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.
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.

::: noteimportant 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
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

Item key	User group	
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*.[5]

```
$ 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	

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.

[5] According to specification these are voltages on chip pins and generally speaking may need scaling.

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
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
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, since 6.2.2)	jname: name of jail (FreeBSD only, since 6.2.2)	pmem: percentage of real memory (FreeBSD only, since 6.2.2)
name: process name	name: process name	pmem: percentage of real memory (FreeBSD only)
cmdline: command with arguments	user: user (real) the process runs under	rss: resident set size
user: user (real) the process runs under	group: group (real) the process runs under	size: size of process (code + data + stack)
group: group (real) the process runs under	uid: user ID	tsize: text (code) size
uid: user ID	gid: ID of the group the process runs under	dsize: data size
gid: ID of the group the process runs under	tid: thread ID	ssize: stack size
vsize: virtual memory size	tname: thread name	cputime_user: total CPU seconds (user)
pmem: percentage of real memory (FreeBSD only)	cputime_user: total CPU seconds (user)	cputime_system: total CPU seconds (system)
rss: resident set size	cputime_system: total CPU seconds (system)	ctx_switches: number of context switches
size: size of process (code + data + stack)	state: thread state	threads: number of threads (not supported for NetBSD)
tsize: text (code) size	ctx_switches: number of context switches	stk: size of stack segment
dsize: data size	io_read_op: number of times the system had to perform input	page_faults: number of page faults
ssize: stack size	io_write_op: number of times the system had to perform output	fds: number of file descriptors (OpenBSD only)
cputime_user: total CPU seconds (user)		swap: size of swap space used
cputime_system: total CPU seconds (system)		io_read_op: number of times the system had to perform input
state: process state (disk sleep/running/sleeping/tracing stop/zombie/other)		io_write_op: number of times the system had to perform output
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

mode=process	mode=thread	mode=summary
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
handles: number of handles		io_read_op: IO read operations
io_read_b: IO bytes read		io_write_op: IO write operations
io_write_b: IO bytes written		io_other_b: IO bytes transferred, other than read and write operations
io_read_op: IO read operations		io_other_op: IO operations, other than read and write 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.

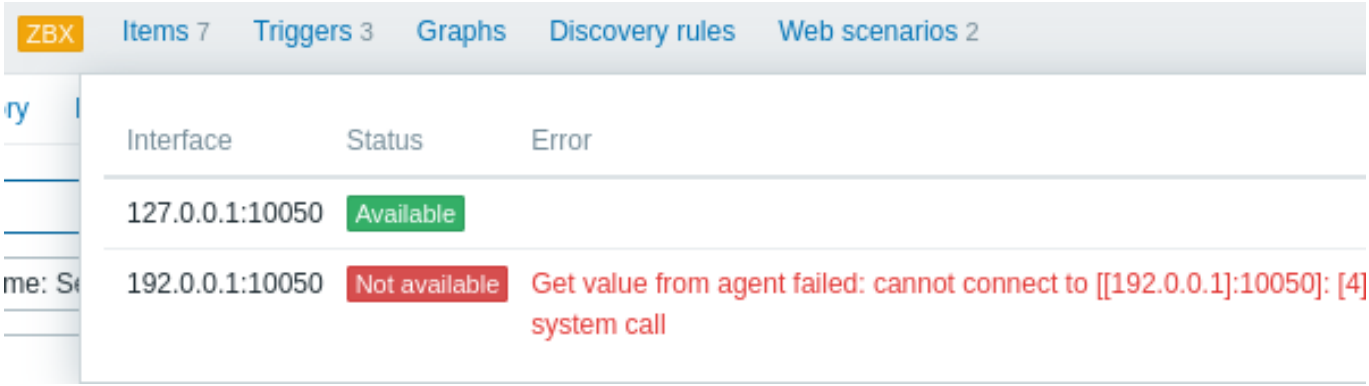
Unavailable interface

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.

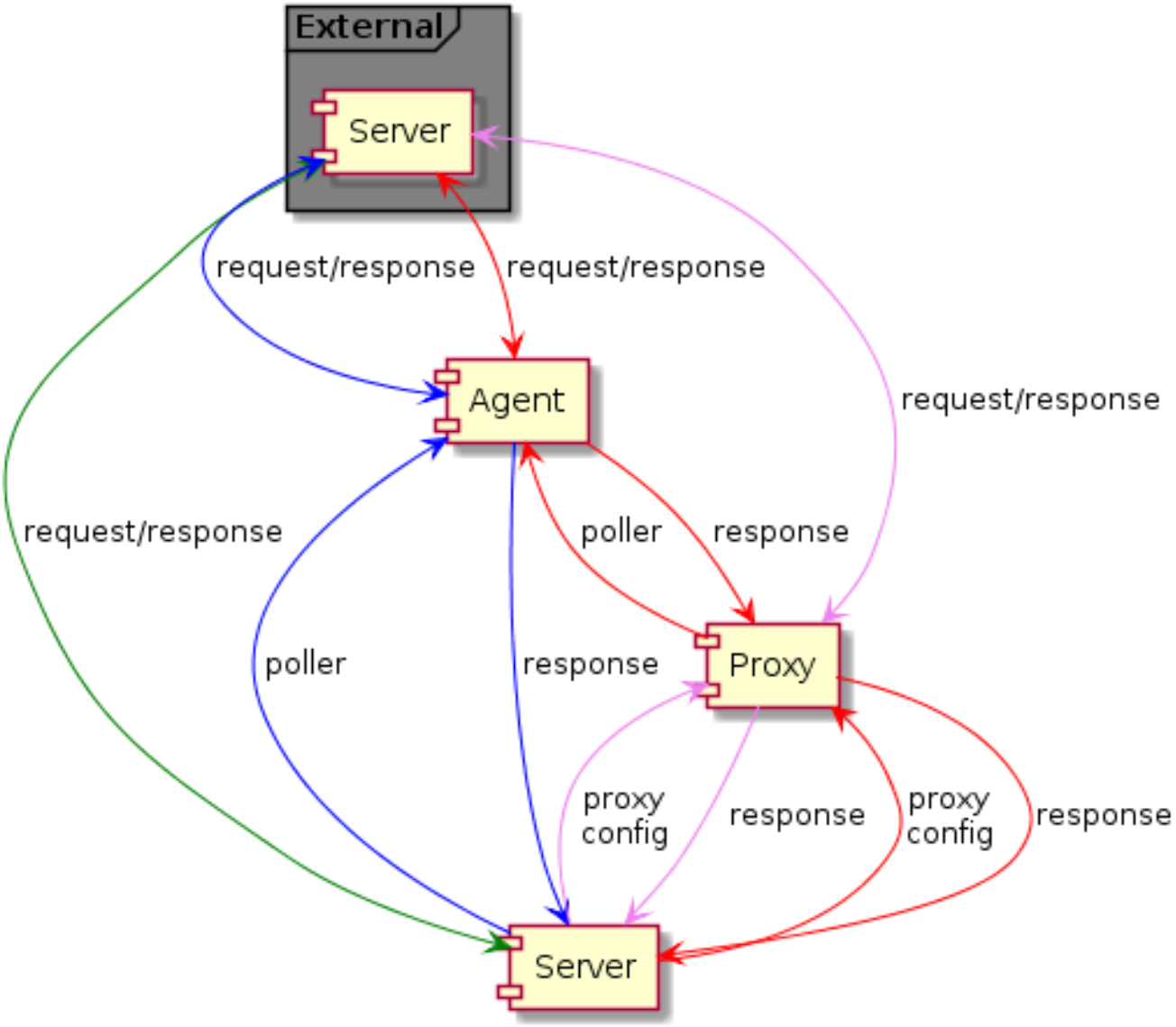
Items

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
- Remote Zabbix proxy

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/CentOS:

```
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*

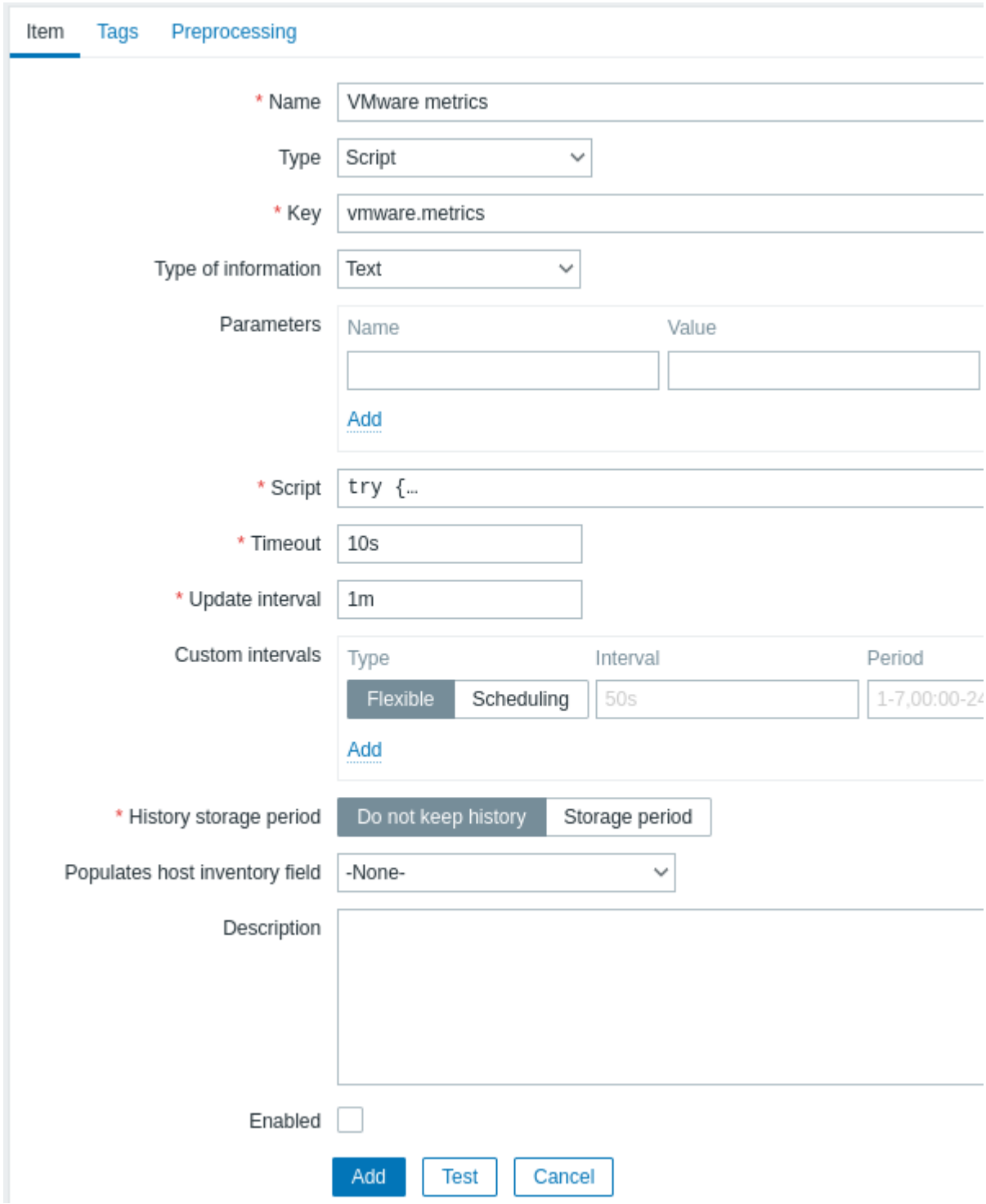
So the above example would give the following counter path: `cpu/usagemhz[average]`

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]`

6 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
Bitwise functions	bitand, bitlshift, bitnot, bitor, bitrshift, bitxor

Function group	Functions
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
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 calculatations.

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))`

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

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.

Aggregate functions

FUNCTION		
Description	**Function-specific parameters**	**Comments**
avg (/host/key,(sec|#num)<:time shift>)		

FUNCTION		
Average value of an item within the defined evaluation period.	See common parameters.	Supported value types: float, int Examples: => **avg**(/host/key,**1h**) → average value for the last hour until now => **avg**(/host/key,**1h:now-1d**) → average value for an hour from 25 hours ago to 24 hours ago from now => **avg**(/host/key,**#5**) → average value of the five latest values => **avg**(/host/key,**#5:now-1d**) → average value of the five latest values excluding the values received in the last 24 hours Time shift is useful when there is a need to compare the current average value with the average value some time ago.
bucket_percentile (item filter,time period,percentage) Calculates the percentile from the buckets of a histogram.	**item filter** - see item filter **time period** - see time period **percentage** - percentage (0-100)	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>)) Count of values in an array returned by a foreach function.	**func_foreach** - foreach function for which the number of returned values should be counted (with supported arguments). See foreach functions for details.	Supported value type: int Example: => **count**(max_foreach(/*/net.if.in[*],1h)) → number of net.if.in items that received data in the last hour until now Note that 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.
histogram_quantile (quantile,bucket1,value1,bucket2,value2,...) Calculates the φ-quantile from the buckets of a histogram.	**quantile** - 0 ≤ φ ≤ 1 **bucketN, valueN** - manually entered pairs (>=2) of parameters or response of bucket_rate_foreach	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. Example: => **histogram_quantile**(0.75,1.0,last(/host/rate_bucket[1.0 => **histogram_quantile**(0.5,bucket_rate_foreach(//item_ke
item_count (item filter)		

FUNCTION		
Count of existing items in configuration that match filter criteria.	**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.	Supported only in calculated items. Supported value type: int Works as an alias for the *count(exists_foreach(item_filter))* function. Example: => **item_count**(/*/agent.ping?[group="Host group 1"]) → number of hosts with the *agent.ping* item in the "Host group 1"
kurtosis (/host/key,(sec|#num)<:time shift>) "Tailedness" of the probability distribution in collected values within the defined evaluation period. See also: Kurtosis	See common parameters.	Supported value types: float, int Example: => **kurtosis**(/host/key,**1h**) → kurtosis for the last hour until now
mad (/host/key,(sec|#num)<:time shift>) Median absolute deviation in collected values within the defined evaluation period. See also: Median absolute deviation	See common-parameters.	Supported value types: float, int Example: => **mad**(/host/key,**1h**) → median absolute deviation for the last hour until now
max (/host/key,(sec|#num)<:time shift>) Highest value of an item within the defined evaluation period.	See common parameters.	Supported value types: float, int Example: => **max**(/host/key,**1h**) - **min**(/host/key,**1h**) → calculate the difference between the maximum and minimum values within the last hour until now (delta of values)
min (/host/key,(sec|#num)<:time shift>) Lowest value of an item within the defined evaluation period.	See common parameters.	Supported value types: float, int Example: => **max**(/host/key,**1h**) - **min**(/host/key,**1h**) → calculate the difference between the maximum and minimum values within the last hour until now (delta of values)
skewness (/host/key,(sec|#num)<:time shift>) Asymmetry of the probability distribution in collected values within the defined evaluation period. See also: Skewness	See common parameters.	Supported value types: float, int Example: => **skewness**(/host/key,**1h**) → skewness for the last hour until now
stddevpop (/host/key,(sec|#num)<:time shift>)		

FUNCTION		
Population standard deviation in collected values within the defined evaluation period. See also: Standard deviation	See common parameters.	Supported value types: float, int Example: => **stddevpop**(/host/key,**1h**) → population standard deviation for the last hour until now
stddevsamp (/host/key,(sec|#num)<:time shift>) Sample standard deviation in collected values within the defined evaluation period. See also: Standard deviation	See common parameters.	Supported value types: float, int At least two data values are required for this function to work. Example: => **stddevsamp**(/host/key,**1h**) → sample standard deviation for the last hour until now
sum (/host/key,(sec|#num)<:time shift>) Sum of collected values within the defined evaluation period.	See common parameters.	Supported value types: float, int Example: => **sum**(/host/key,**1h**) → 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.	See common parameters.	Supported value types: float, int Example: => **sumofsquares**(/host/key,**1h**) → sum of squares for the last hour until now
varpop (/host/key,(sec|#num)<:time shift>) Population variance of collected values within the defined evaluation period. See also: Variance	See common parameters.	Supported value types: float, int Example: => **varpop**(/host/key,**1h**) → population variance for the last hour until now
varsamp (/host/key,(sec|#num)<:time shift>) Sample variance of collected values within the defined evaluation period. See also: Variance	See common parameters.	Supported value types: float, int At least two data values are required for this function to work. Example: => **varsamp**(/host/key,**1h**) → sample variance for the last hour until now

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**.

For example, the *avg_foreach* function will return the *average* value from the history of each 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 the number of currently enabled items.
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.

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.

2 Bitwise functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

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 < >

FUNCTION		
Description	**Function-specific parameters**	**Comments**
bitand (value,mask)		

FUNCTION		
Value of ”bitwise AND” of an item value and mask.	**value** - value to check **mask** (mandatory) - 64-bit unsigned integer (0 - 18446744073709551615)	Supported value types: int 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 time => **bitand**(last(/host/key),**20**)=16 → 3rd bit not set and 5th bit set.
bitlshift (value,bits to shift) Bitwise shift left of an item value.	**value** - value to check **bits to shift** (mandatory) - number of bits to shift	Supported value types: int 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) Value of ”bitwise NOT” of an item value.	**value** - value to check	Supported value types: int 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) Value of ”bitwise OR” of an item value and mask.	**value** - value to check **mask** (mandatory) - 64-bit unsigned integer (0 - 18446744073709551615)	Supported value types: int 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) Bitwise shift right of an item value.	**value** - value to check **bits to shift** (mandatory) - number of bits to shift	Supported value types: int 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) Value of ”bitwise exclusive OR” of an item value and mask.	**value** - value to check **mask** (mandatory) - 64-bit unsigned integer (0 - 18446744073709551615)	Supported value types: int 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.

manual/appendix/functions/time

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.

FUNCTION		
Description	**Function-specific parameters**	**Comments**
date Current date in YYYYMMDD format.		Example: => **date**()<20220101
dayofmonth Day of month in range of 1 to 31.		Example: => **dayofmonth**()=1
dayofweek Day of week in range of 1 to 7 (Mon - 1, Sun - 7).		Example: => **dayofweek**()<6
now Number of seconds since the Epoch (00:00:00 UTC, January 1, 1970).		Example: => **now**()<1640998800
time Current time in HHMMSS format.		Example: => **time**()>000000 and **time**()<060000

4 History functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

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

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.

History functions

FUNCTION		
Description	**Function-specific parameters**	**Comments**
change (/host/key)		

FUNCTION		
The amount of difference between the previous and latest value.		Supported value types: float, int, str, text, log For strings returns: 0 - values are equal 1 - values differ Example: => **change**(/host/key)>10 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
changecount (/host/key,(sec|#num)<:time shift>,<mode>) Number of changes between adjacent values within the defined evaluation period.	See common parameters. **mode** (optional; must be double-quoted) Supported `modes`: *all* - count all changes (default) *dec* - count decreases *inc* - count increases	Supported value types: float, int, str, text, log For non-numeric value types, *mode* parameter is ignored. Examples: => **changecount**(/host/key, 1w) → number of value changes for the last week until now => **changecount**(/host/key,#10,"inc") → number of value increases (relative to the adjacent value) among the last 10 values => **changecount**(/host/key,24h,"dec") → number of value decreases (relative to the adjacent value) for the last 24 hours until now
count (/host/key,(sec|#num)<:time shift>,<operator>,<pattern>)		

FUNCTION		
Number of values within the defined evaluation period.	See common parameters. **operator** (optional; must be double-quoted) Supported `operators`: *eq* - equal (default) *ne* - not equal *gt* - greater *ge* - greater or equal *lt* - less *le* - less or equal *like* - 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** (optional) - required pattern (string arguments must be double-quoted)	Supported value types: float, integer, string, text, log Float items match with the precision of 2.22e-16; if database is not upgraded the precision is 0.000001. With *bitand* as the third parameter, 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 the third parameter, 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**) → number of values for the last 10 minutes until now => **count**(/host/key,**10m**,"like","error") → number of values for the last 10 minutes until now that contain 'error' => **count**(/host/key,**10m**,,12) → number of values for the last 10 minutes until now that equal '12' => **count**(/host/key,**10m**,"gt",12) → number of values for the last 10 minutes until now that are over '12' => **count**(/host/key,**#10**,"gt",12) → number of values within the last 10 values until now that are over '12' => **count**(/host/key,**10m:now-1d**,"gt",12) → number of values between 24 hours and 10 minutes and 24 hours ago from now that were over '12' => **count**(/host/key,**10m**,"bitand","6/7") → number of values for the last 10 minutes until now having '110' (in

FUNCTION
countunique (/host/key,(sec|#num)<:time shift>,<operator>,<pattern>)

FUNCTION		
Number of unique values within the defined evaluation period.	See common parameters. **operator** (optional; must be double-quoted) Supported `operators`: *eq* - equal (default) *ne* - not equal *gt* - greater *ge* - greater or equal *lt* - less *le* - less or equal *like* - 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** (optional) - required pattern (string arguments must be double-quoted)	Supported value types: float, integer, string, text, log Float items match with the precision of 2.22e-16; if database is not upgraded the precision is 0.000001. With *bitand* as the third parameter, 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 the third parameter, 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**) → number of unique values for the last 10 minutes until now => **countunique**(/host/key,**10m**,"like","error") → number of unique values for the last 10 minutes until now that contain 'error' => **countunique**(/host/key,**10m**,"gt",12) → number of unique values for the last 10 minutes until now that are over '12' => **countunique**(/host/key,**#10**,"gt",12) → number of unique values within the last 10 values until now that are over '12' => **countunique**(/host/key,**10m:now-1d**,"gt",12) → number of unique values between 24 hours and 10 minutes and 24 hours ago from now that were over '12' => **countunique**(/host/key,**10m**,"bitand","6/7")

FUNCTION		
find (/host/key,<(sec|#num)<:time shift»,<operator>,<pattern>) Find a value match.	See common parameters. **sec** or **#num** (optional) - defaults to the latest value if not specified **operator** (optional; must be double-quoted) Supported `operators`: *eq* - equal (default) *ne* - not equal *gt* - greater *ge* - greater or equal *lt* - less *le* - less or equal *like* - value 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** - required pattern (string arguments must be double-quoted); Perl Compatible Regular Expression (PCRE) regular expression if `operator` is *regexp*, *iregexp*.	Supported value types: float, int, str, text, log Returns: 1 - found 0 - otherwise If more than one value is processed, '1' is returned if there is at least one matching value. With *regexp* or *iregexp* as the third parameter, 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. Example: => **find**(/host/key,**10m**,"like","error") → find a value that contains 'error' within the last 10 minutes until now
first (/host/key,sec<:time shift>) The first (the oldest) value within the defined evaluation period.	See common parameters.	Supported value types: float, int, str, text, log Example: => **first**(/host/key,**1h**) → retrieve the oldest value within the last hour until now See also last().
fuzzytime (/host/key,sec)		

FUNCTION		
Checking how much the passive agent time differs from the Zabbix server/proxy time.	See common-parameters.	Supported value types: float, int Returns: 1 - difference between the passive item value (as timestamp) and Zabbix server/proxy timestamp (clock of value collection) is less than or equal to T seconds 0 - otherwise 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 vfs.file.time[/path/file,modify] key to check that file didn't get updates for long time. Example: => **fuzzytime**(/host/key,**60s**)=0 → detect a problem if the time difference is over 60 seconds 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,60s)=0 or last(/Host/trap)<>0`
last (/host/key,<#num<:time shift>) The most recent value.	See common parameters. **#num** (optional) - the Nth most recent value	Supported value types: float, int, str, text, log 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) - 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. 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 See also first().

FUNCTION		
logeventid (/host/key,<#num<:time shift»,<pattern>) Checking if event ID of the last log entry matches a regular expression.	See common parameters. **#num** (optional) - the Nth most recent value **pattern** (optional) - regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style (string arguments must be double-quoted).	Supported value types: log Returns: 0 - does not match 1 - matches
logseverity (/host/key,<#num<:time shift») Log severity of the last log entry.	See common parameters. **#num** (optional) - the Nth most recent value	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). Zabbix takes log severity from **Information** field of Windows event log.
logsource (/host/key,<#num<:time shift»,<pattern>) Checking if log source of the last log entry matches a regular expression.	See common parameters. **#num** (optional) - the Nth most recent value **pattern** (optional) - regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style (string arguments must be double-quoted).	Supported value types: log Returns: 0 - does not match 1 - matches Normally used for Windows event logs. For example, logsource(”VMware Server”).
monodec (/host/key,(sec|#num)<:time shift>,<mode>) Check if there has been a monotonous decrease in values.	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)	Supported value types: int Returns 1 if all elements in the time period continuously decrease, 0 otherwise. Example: => **monodec**(/Host1/system.swap.size[all,free],**60s**) + **monodec**(/Host2/system.swap.size[all,free],**60s**) + **monodec**(/Host3/system.swap.size[all,free],**60s**) - calculate in how many hosts there has been a decrease in free swap size
monoinc (/host/key,(sec|#num)<:time shift>,<mode>)		

FUNCTION		
Check if there has been a monotonous increase in values.	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)	Supported value types: int Returns 1 if all elements in the time period continuously increase, 0 otherwise. Example: => **monoinc**(/Host1/system.localtime,**#3**,"strict")=0 - check if system local time has been increasing consistently
nodata (/host/key,sec,<mode>) Checking for no data received.	See common parameters. **sec** 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).	All value types are supported. Returns: 1 - if no data received during the defined period of time 0 - otherwise Since Zabbix 5.0, 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 & ProxyUpdateFrequency 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 work the same as before 5.0.0 and fire as soon as the evaluation period (five minutes) without data has past. Note that 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)		

FUNCTION		
P-th percentile of a period, where P (percentage) is specified by the third parameter.	See common parameters. **percentage** - a floating-point number between 0 and 100 (inclusive) with up to 4 digits after the decimal point	Supported value types: float, int
rate (/host/key,sec<:time shift>) Per-second average rate of the increase in a monotonically increasing counter within the defined time period.	See common parameters.	Supported value types: float, int 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.

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 TrendCacheSize 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

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

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)

Trend functions

FUNCTION		
Description	**Function-specific parameters**	**Comments**
baselinedev (/host/key,data period:time shift,season_unit,num_seasons)		

FUNCTION		
Returns the number of deviations (by stddevpop algorithm) between the last data period and the same data periods in preceding seasons. **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. **trendavg** (/host/key,time period:time shift)	**data period** - the data gathering period within a season, defined as <N><time unit> where `N` - 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 (see examples) **season_unit** - duration of one season (h, d, w, M, y), cannot be smaller than data period **num_seasons** - number of seasons to evaluate	Examples: => **baselinedev**(/host/key,1d:now/d,”M”,6) → calculating the number of standard deviations (population) between the previous day and the same day in the previous 6 months. If the date doesn’t exist in a previous month, the last day of the month will be used (Jul,31 will be analysed against Jan,31, Feb, 28,... June, 30). => **baselinedev**(/host/key,1h:now/h,”d”,10) → calculating the number of standard deviations (population) between the previous hour and the same hours over the period of ten days before yesterday. Examples: => **baselinewma**(/host/key,1h:now/h,”d”,3) → calculating baseline based on the last full hour within a 3-day period that ended yesterday. If ”now” is Monday 13:30, the data for 12:00-12:59 on Friday, Saturday, and Sunday will be analyzed. => **baselinemwa**(/host/key,2h:now/h,”d”,3) → calculating baseline based on the last two hours within a 3-day period that ended yesterday. If ”now” is Monday 13:30, the data for 10:00-11:59 on Friday, Saturday, and Sunday will be analyzed. => **baselinewma**(/host/key,1d:now/d,”M”,4) → calculating baseline based on the same day of month as ‘yesterday’ in the 4 months preceding the last full month. If required date doesn’t exist, the last day of month is taken. If today is September 1st, the data for July 31st, June 30th, May 31st, April 30th will be analyzed.

FUNCTION		
Average of trend values within the defined time period. **trendcount** (/host/key,time period:time shift) Number of successfully retrieved history values used to calculate the trend value within the defined time period. **trendmax** (/host/key,time period:time shift) The maximum in trend values within the defined time period. **trendmin** (/host/key,time period:time shift)	See common parameters. See common parameters. See common parameters.	Examples: => **trendavg**(/host/key,**1h:now/h**) → average for the previous hour (e.g. 12:00-13:00) => **trendavg**(/host/key,**1h:now/h-1h**) → average for two hours ago (11:00-12:00) => **trendavg**(/host/key,**1h:now/h-2h**) → average for three hours ago (10:00-11:00) => **trendavg**(/host/key,**1M:now/M-1y**) → average for the previous month a year ago Examples: => **trendcount**(/host/key,**1h:now/h**) → count for the previous hour (e.g. 12:00-13:00) => **trendcount**(/host/key,**1h:now/h-1h**) → count for two hours ago (11:00-12:00) => **trendcount**(/host/key,**1h:now/h-2h**) → count for three hours ago (10:00-11:00) => **trendcount**(/host/key,**1M:now/M-1y**) → count for the previous month a year ago Examples: => **trendmax**(/host/key,**1h:now/h**) → 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 and minimum values (trend delta) for the previous hour (12:00-13:00) => **trendmax**(/host/key,**1h:now/h-1h**) → maximum for two hours ago (11:00-12:00) => **trendmax**(/host/key,**1h:now/h-2h**) → maximum for three hours ago (10:00-11:00) => **trendmax**(/host/key,**1M:now/M-1y**) → maximum for the previous month a year ago

FUNCTION		
The minimum in trend values within the defined time period.	See common parameters.	Examples: => **trendmin**(/host/key,**1h:now/h**) → minimum for the previous hour (e.g. 12:00-13:00) => **trendmin**(/host/key,**1h:now/h**) - **trendmin**(/host/key,**1h:now/h**) → calculate the difference between the maximum and minimum values (trend delta) for the previous hour (12:00-13:00) => **trendmin**(/host/key,**1h:now/h-1h**) → minimum for two hours ago (11:00-12:00) => **trendmin**(/host/key,**1h:now/h-2h**) → minimum for three hours ago (10:00-11:00) => **trendmin**(/host/key,**1M:now/M-1y**) → minimum for the previous month a year ago
trendstl (/host/key,eval period:time shift,detection period,season,<deviations>,<devalg>,<s_window>)		

FUNCTION		
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))`. **trendsum** (/host/key,time period:time shift)	**eval period** - the time period that must be decomposed (minimum '1h'), defined as <N><time unit> where `N` - number of time units `time unit` - h (hour), d (day), w (week), M (month) or y (year). Time shift - the time period offset (see examples) **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` - 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, 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` - 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) - 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: => **trend-stl**(/host/key,**100h:now/h**,10h,2h) → analyse the last 100 hours of trend data, find the anomaly rate for the last 10 hours of that period, expecting the periodicity to be 2h, the remainder series values of the evaluation period are considered anomalies if they reach the value of 3 deviations of the MAD of that remainder series => **trendstl**(/host/key,**100h:now/h-10h**,100h,2h,2.1,"mad") → analyse the period of 100 hours of trend data, up to 10 hours ago, find the anomaly rate for that entire period expecting the periodicity to be 2h, the remainder series values of the evaluation period are considered anomalies if they reach the value of 2,1 deviations of the MAD of that remainder series => **trendstl**(/host/key,**100d:now/d-1d**,10d,1d,4„10) → analyse 100 days of trend data up to a day ago, find the anomaly rate for the period of last 10d of that period, expecting the periodicity to be 1d, the remainder series values of the evaluation period are considered anomalies if they reach the value of 4 deviations of the MAD of that remainder series, overriding the default span of the loess window for seasonal extraction of "10 * number of entries in eval period + 1" with the span of 10 lags => **trendstl**(/host/key,**1M:now/M-1y**,1d,2h„"stddevsamp") → analyse the previous month a year ago, find the anomaly rate of the last day of that period expecting the periodicity to be 2h, the remainder series values of the evaluation period are considered anomalies if they reach the value of 3 deviation of the sample standard deviation of that remainder series

FUNCTION		
Sum of trend values within the defined time period.	See common parameters.	Examples: => **trendsum**(/host/key,**1h:now/h**) → sum for the previous hour (e.g. 12:00-13:00) => **trendsum**(/host/key,**1h:now/h-1h**) → sum for two hours ago (11:00-12:00) => **trendsum**(/host/key,**1h:now/h-2h**) → sum for three hours ago (10:00-11:00) => **trendsum**(/host/key,**1M:now/M-1y**) → sum for the previous month a year ago

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.

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 < >

FUNCTION		
Description	**Function-specific parameters**	**Comments**
abs (value) The absolute value of a value.	**value** - value to check	Supported value types: float, int, str, text, log For strings returns: 0 - values are equal 1 - values differ Example: => **abs**(last(/host/key))>10 Absolute numeric difference will be calculated, as seen with these incoming example values ('previous' and 'latest' value = absolute difference): '1' and '5' = 4 '3' and '1' = 2 '0' and '-2.5' = 2.5
acos (value) The arccosine of a value as an angle, expressed in radians.	**value** - 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)		

FUNCTION		
The arcsine of a value as an angle, expressed in radians.	**value** - 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.	**value** - value to check	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 (exprue) and abscissa coordinates specified as an angle, expressed in radians.	**value** - value to check **abscissa** - abscissa value	For example, the arctangent of the ordinate and abscissa coordinates of a value ’1’ will be ’2.21429744’. Example: => **atan2**(last(/host/key),2)
avg (<value1>,<value2>,...) Average value of the referenced item values.	**valueX** - value returned by one of history functions	Example: => **avg**(avg(/host/key),avg(/host2/key2))
cbrt (value) Cube root of a value.	**value** - 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.	**value** - value to check	For example, ’2.4’ will be rounded up to ’3’. Example: => **ceil**(last(/host/key)) See also floor().
cos (value) The cosine of a value, where the value is an angle expressed in radians.	**value** - 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.	**value** - value to check	For example, the hyperbolic cosine of a value ’1’ will be ’1.54308063482’. Returns value as a real number, not as scientific notation. Example: => **cosh**(last(/host/key))
cot (value) The cotangent of a value, where the value is an angle, expressed in radians.	**value** - value to check	For example, the cotangent of a value ’1’ will be ’0.54030230586’. Example: => **cot**(last(/host/key))
degrees (value)		

FUNCTION		
Converts a value from radians to degrees.	**value** - value to check	For example, a value ’1’ converted to degrees will be ’57.2957795’. Example: => **degrees**(last(/host/key))
e Euler’s number (2.718281828459045).		Example: => **e**()
exp (value) Euler’s number at a power of a value.	**value** - 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) Euler’s number at a power of a value minus 1.	**value** - 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.	**value** - value to check	For example, ’2.6’ will be rounded down to ’2’. Example: => **floor**(last(/host/key)) See also ceil().
log (value) Natural logarithm.	**value** - value to check	For example, the natural logarithm of a value ’2’ will be ’0.69314718055994529’. Example: => **log**(last(/host/key))
log10 (value) Decimal logarithm.	**value** - value to check	For example, the decimal logarithm of a value ’5’ will be ’0.69897000433’. Example: => **log10**(last(/host/key))
max (<value1>,<value2>,...) Highest value of the referenced item values.	**valueX** - value returned by one of history functions	Example: => **max**(avg(/host/key),avg(/host2/key2))
min (<value1>,<value2>,...) Lowest value of the referenced item values.	**valueX** - value returned by one of history functions	Example: => **min**(avg(/host/key),avg(/host2/key2))
mod (value,denominator) Division remainder.	**value** - value to check **denominator** - division denominator	For example, division remainder of a value ’5’ with division denominator ’2’ will be ’1’. Example: => **mod**(last(/host/key),2)
pi Pi constant (3.14159265358979).		Example: => **pi**()

FUNCTION		
power (value,power value) The power of a value.	**value** - 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) Convert a value from degrees to radians.	**value** - 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.	**value** - 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.	**value** - value to check	Example: => **signum**(last(/host/key))
sin (value) The sine of a value, where the value is an angle expressed in radians.	**value** - 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.	**value** - value to check	For example, the hyperbolical sine of a value '1' will be '1.17520119364'. Example: => **sinh**(last(/host/key))
sqrt (value) Square root of a value.	**value** - value to check	This function will fail with a negative value. For example, the square root of a value '3.5' will be '1.87082869339'. Example: => **sqrt**(last(/host/key))
sum (<value1>,<value2>,...) Sum of the referenced item values.	**valueX** - value returned by one of history functions	Example: => **sum**(avg(/host/key),avg(/host2/key2))
tan (value) The tangent of a value.	**value** - value to check	For example, the tangent of a value '1' will be '1.55740772465'. Example: => **tan**(last(/host/key))
truncate (value,decimal places)		

FUNCTION		
Truncate the value to decimal places.	**value** - value to check **decimal places** - specify decimal places for truncating (0 is also possible)	Example: => **truncate**(last(/host/key),2)

7 Operator functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

Some general notes on function parameters:

- Function parameters are separated by a comma
- Expressions are accepted as parameters

FUNCTION		
Description	**Function-specific parameters**	**Comments**
between (value,min,max) Check if a value belongs to the given range.	**value** - value to check **min** - minimum value **max** - maximum value	Supported value types: integer, float Returns: 1 - in range 0 - otherwise Example: => **between**(last(/host/key),**1**,**10**)=1 - trigger if the value is between 1 and 10.
in (value,value1,value2,...valueN) Check if a value is equal to at least one of the listed values.	**value** - value to check **value1,value2,...valueN** - listed values (string values must be double-quoted)	Supported value types: all Returns: 1 - if equal 0 - otherwise 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.

8 Prediction functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

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

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.

Prediction functions

FUNCTION		
Description	**Function-specific parameters**	**Comments**
forecast (/host/key,(sec|#num)<:time shift>,time,<fit>,<mode>) Future value, max, min, delta or avg of the item.	See common parameters. **time** - forecasting horizon in seconds (time suffixes can be used); negative values are supported **fit** (optional; must be double-quoted) - function used to fit historical data Supported `fits`: *linear* - linear function *polynomialN* - polynomial of degree N (1 <= N <= 6) *exponential* - exponential function *logarithmic* - logarithmic function *power* - power function Note that: *linear* is default, *polynomial1* is equivalent to *linear* **mode** (optional; must be double-quoted) - demanded output Supported `modes`: *value* - value (default) *max* - maximum *min* - minimum *delta* - *max-min* *avg* - average Note that: *value* estimates item value at the moment `now + time` *max*, *min*, *delta* and *avg* investigate item value estimate on the interval between `now` and `now + time`	Supported value types: float, int If value to return is larger than 1.7976931348623157E+308 or less than -1.7976931348623157E+308, return value is cropped to 1.7976931348623157E+308 or -1.7976931348623157E+308 correspondingly. Becomes unsupported only if misused in expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors. Examples: => **forecast**(/host/key,**#10**,**1h**) → forecast item value in one hour based on the last 10 values => **forecast**(/host/key,**1h**,**30m**) → forecast item value in 30 minutes based on the last hour data => **forecast**(/host/key,**1h:now-1d**,**12h**) → forecast item value in 12 hours based on one hour one day ago => **forecast**(/host/key,**1h**,**10m**,"exponential") → forecast item value in 10 minutes based on the last hour data and exponential function => **forecast**(/host/key,**1h**,**2h**,"polynomial3","max") → forecast the maximum value the item can reach in the next two hours based on last hour data and cubic (third degree) polynomial => **forecast**(/host/key,**#2**,**-20m**) → estimate the item value 20 minutes ago based on the last two values (this can be more precise than using last(), especially if item is updated rarely, say, once an hour) See also additional information on predictive trigger functions.

FUNCTION		
timeleft (/host/key,(sec|#num)<:time shift>,threshold,<fit>) Time in seconds needed for an item to reach a specified threshold.	See common parameters. **threshold** - value to reach (unit suffixes can be used) **fit** (optional; must be double-quoted) - see forecast()	Supported value types: float, int If value to return is larger than 1.7976931348623157E+308, return value is cropped to 1.7976931348623157E+308. Returns 1.7976931348623157E+308 if threshold cannot be reached. Becomes unsupported only if misused in the expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors. Examples: => **timeleft**(/host/key,**#10**,**0**) → time until the item value reaches zero based on the last 10 values => **timeleft**(/host/key,**1h**,**100**) → time until the item value reaches 100 based on the last hour data => **timeleft**(/host/key,**1h:now-1d**,**100**) → time until the item value reaches 100 based on one hour one day ago => **timeleft**(/host/key,**1h**,**200**,”polynomial2”) → time until the item value reaches 200 based on the last hour data and assumption that the item behaves like quadratic (second degree) polynomial See also additional information on predictive trigger functions.

9 String functions

All functions listed here are supported in:

- Trigger expressions
- Calculated items

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 < >

FUNCTION		
Description	**Function-specific parameters**	**Comments**
ascii (value)		

FUNCTION		
The ASCII code of the leftmost character of the value.	**value** - value to check	Supported value types: string, text, log 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.	**value** - value to check	Supported value types: string, text, log, integer Example: => **bitlength**(last(/host/key))
bytelength (value) The length of value in bytes.	**value** - value to check	Supported value types: string, text, log, integer Example: => **bytelength**(last(/host/key))
char (value) Return the character by interpreting the value as ASCII code.	**value** - value to check	Supported value types: integer 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 referenced item values or constant values.	**value** - a value returned by one of the history functions or a constant value (string, integer, or float number)	Supported value types: string, text, log, float, integer For example, a value like ’Zab’ concatenated to ’bix’ (the constant string) will return ’Zabbix’. Must contain at least two parameters. 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.	**value** - value to check **start** - start position **length** - positions to replace **replacement** - replacement string	Supported value types: string, text, log 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)		

FUNCTION		
The leftmost characters of the value.	**value** - value to check **count** - number of characters to return	Supported value types: string, text, log For example, you may return 'Zab' from 'Zabbix' by specifying 3 leftmost characters to return. Example: => **left**(last(/host/key),**3**) - return three leftmost characters See also right().
length (value) The length of value in characters.	**value** - value to check	Supported value types: str, text, log Example: => **length**(last(/host/key)) → length of the latest value => **length**(last(/host/key,#3)) → length of the third most recent value => **length**(last(/host/key,#1:now-1d)) → length of the most recent value one day ago
ltrim (value,<chars>) Remove specified characters from the beginning of string.	**value** - value to check **chars** - (optional) specify characters to remove Whitespace is left-trimmed by default (if no optional characters are specified).	Supported value types: string, text, log Example: => **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 See also: rtrim(), trim()
mid (value,start,length) Return a substring of N characters beginning at the character position specified by 'start'.	**value** - value to check **start** - start position of substring **length** - positions to return in substring	Supported value types: string, text, log 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.	**value** - value to check **count** - number of times to repeat	Supported value types: string, text, log Example: => **repeat**(last(/host/key),**2**) - repeat the value two times
replace (value,pattern,replacement)		

FUNCTION		
Find pattern in the value and replace with replacement. All occurrences of the pattern will be replaced.	**value** - value to check **pattern** - pattern to find **replacement** - string to replace the pattern with	Supported value types: string, text, log Example: => **replace**(last(/host/key),**"ibb"**,**"abb"**) - replace all 'ibb' with 'abb'
right (value,count) The rightmost characters of the value.	**value** - value to check **count** - number of characters to return	Supported value types: string, text, log For example, you may return 'bix' from 'Zabbix' by specifying 3 rightmost characters to return. Example: => **right**(last(/host/key),**3**) - return three rightmost characters See also left().
rtrim (value,<chars>) Remove specified characters from the end of string.	**value** - value to check **chars** - (optional) specify characters to remove Whitespace is right-trimmed by default (if no optional characters are specified).	Supported value types: string, text, log Example: => **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' or space from the end of string See also: ltrim(), trim()
trim (value,<chars>) Remove specified characters from the beginning and end of string.	**value** - value to check **chars** - (optional) specify characters to remove Whitespace is trimmed from both sides by default (if no optional characters are specified).	Supported value types: string, text, log Example: => **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 also: ltrim(), rtrim()

7 Macros

1 Supported macros

Overview

The table contains a complete list of macros supported by Zabbix out-of-the-box.

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*.

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	*'Default message' value from action configuration.* Supported since 3.0.0.
{ALERT.SENDTO}	→ Alert script parameters	*'Send to' value from user media configuration.* Supported since 3.0.0.
{ALERT.SUBJECT}	→ Alert script parameters	*'Default subject' value from action configuration.* Supported since 3.0.0.
{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.*
{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.

Macro	Supported in	Description
{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.* ||*{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*).*
{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.
{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.

Macro	Supported in	Description
{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.
{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.*

Macro	Supported in	Description
{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.
{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.*

Macro	Supported in	Description
{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).*
{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>}	→ Trigger 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.

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, Proxy, 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	*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	*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, Proxy, 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	*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, Proxy, 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	*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 → Description of item value 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, Proxy, 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	*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, Proxy, 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	*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	*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.
{HOSTGROUP.ID}	→ Map element labels, map URL names and values	*Host group ID.*

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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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	*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.
{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	*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.

Macro	Supported in	Description
{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	*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	*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	*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	*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.
{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.

Macro	Supported in	Description
{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). Note that since 4.0, when used in the problem name, it will not resolve to the latest item value when viewing problem events, instead it will keep the item value from the time of problem happening. 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; - 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.
{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.

Macro	Supported in	Description
{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; - 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.
{LLDRULE.DESCRIPTION}	→ LLD-rule based internal notifications	Supported since 5.4.0. *Description of the low-level discovery rule which caused a notification.*
{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.*

Macro	Supported in	Description
{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.
{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.
{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.
{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.
{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.

Macro	Supported in	Description
{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.
{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.*
{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.
{TRIGGER.EXPRESSION.RECOVERY.EXPLAIN}	→ 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.*

Macro	Supported in	Description
{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*.
{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.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.

Macro	Supported in	Description
{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.
{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.
{USER.USERNAME}	→ 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.
{$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 → Map element labels[3] → Map shape labels[3] → Link labels in maps[3] → Graph names[5]	See expression macros. Supported since 5.2.0.

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

Location		Multiple macros/mix with text[1]
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
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
TELNET agent		
	Username	yes
	Password	yes
	Script	yes
Zabbix trapper		
	Allowed hosts	yes
Tags[2]		
	Tag names	yes
	Tag values	yes
Preprocessing		

Location		Multiple macros/mix with text[1]
	Step parameters (including custom scripts)	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		
	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		

Location		Multiple macros/mix with text[1]
	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
Expression (only in constants and function parameters; secret macros are not supported)	yes
Description	yes
URL[3]	yes
Tag for matching	yes
Tags[2]	
Tag names	yes

Location	Multiple macros/mix with text[1]
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
	SSL key password	yes
Tags[2]		
	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 (script, SSH, Telnet, IPMI), including confirmation text	yes
Webhooks	
JavaScript script	no
JavaScript script parameter name	no
JavaScript script parameter value	yes

Location	Multiple macros/mix with text[1]
Monitoring → *Dashboards*	
Description field of *Item value* dashboard widget	yes
URL[3] field of *dynamic URL* dashboard widget	yes
Administration → *Users* → *Media*	
When active	no
Administration → *General* → *GUI*	
Working time	no
Administration → *Media types* → *Message templates*	
Subject	yes
Message	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 "******".

8 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 *Monitoring* → *Dashboard* ('Time shift' field)
- *Administration* → *General* → *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
```

9 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 01-07,00:00-24:00, 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`

10 Command execution

Zabbix uses common functionality for external checks, user parameters, system.run items, custom alert scripts, remote commands and user scripts.

Execution steps

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 is created for remote commands on Zabbix server to save script execution output and can be enabled using LogRemoteCommands agent parameter.

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).

Read more about:

- External checks
- User parameters
- system.run items
- Custom alert scripts
- Remote commands
- Global scripts

11 Version compatibility

Supported agents

To be compatible with Zabbix 6.2, Zabbix agent must not be older than version 1.4 and must not be newer than 6.2.

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 6.2; Zabbix agent 2 must not be newer than 6.2.

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 compatible with Zabbix 6.2, the proxy must be of the same major version; thus only Zabbix 6.2.x proxies can work with Zabbix 6.2.x server.

See also: Known issues in Zabbix 6.2.5

> **Attention:**
> It is no longer possible to start the upgraded server and have older and unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled, as the server will ignore data from unupgraded proxies. See also the upgrade procedure.

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 6.2.

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.

12 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

13 Zabbix sender dynamic link library for Windows

In a Windows environment applications can send data to Zabbix server/proxy directly by using the Zabbix sender dynamic link library (zabbix_sender.dll) instead of having to launch an external process (zabbix_sender.exe).

The dynamic link library with the development files is located in bin\winXX\dev folders. To use it, include the zabbix_sender.h header file and link with the zabbix_sender.lib library. An example file with Zabbix sender API usage can be found in build\win32\examples\zabbix_sender folder.

The following functionality is provided by the Zabbix sender dynamic link library:

```
int zabbix_sender_send_values(const char *address, unsigned short port,const char *source, const zabbix_
char **result);‘{.c}
```

The following data structures are used by the Zabbix sender dynamic link library:

```
typedef struct
{
    /* host name, must match the name of target host in Zabbix */
    char    *host;
    /* the item key */
    char    *key;
    /* the item value */
    char    *value;
}
zabbix_sender_value_t;

typedef struct
{
    /* number of total values processed */
    int total;
    /* number of failed values */
    int failed;
    /* time in seconds the server spent processing the sent values */
    double  time_spent;
```

```
}
zabbix_sender_info_t;
```

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		

Parameter	Zabbix agent	Zabbix agent 2
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
Timeout settings	Defined on an agent level only.	Plugin timeout can override the timeout defined on an agent level.
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

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

loglevel increase
Increase log level

loglevel 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 by man2html, using the manual pages.
Time: 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[=*target*]
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 by man2html, using the manual pages.
Time: 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 by man2html, using the manual pages.
Time: 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, heartbeat sender, 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 by man2html, using the manual pages.
Time: 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 by man2html, using the manual pages.
Time: 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*.

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.

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*, *alerting*, *lld*, *valuecache*. 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 (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”

-h, **--help**
Display this help and exit.

-V, **--version**
Output version information and exit.

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 by man2html, using the manual pages.
Time: 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 by man2html, using the manual pages.
Time: 12:58:30 GMT, June 11, 2021