

Documentation 4.2

ZABBIX

02.01.2026

Contents

Zabbix Manual	4
Copyright notice	4
1. Introduction	4
1 Manual structure	4
2 What is Zabbix	5
3 Zabbix features	5
4 Zabbix overview	6
5 What's new in Zabbix 4.2.0	7
6 What's new in Zabbix 4.2.1	18
7 What's new in Zabbix 4.2.2	19
8 What's new in Zabbix 4.2.3	20
9 What's new in Zabbix 4.2.4	20
10 What's new in Zabbix 4.2.5	21
11 What's new in Zabbix 4.2.6	22
12 What's new in Zabbix 4.2.7	22
13 What's new in Zabbix 4.2.8	23
2. Definitions	23
3. Zabbix processes	25
1 Server	25
2 Agent	28
3 Proxy	31
4 Java gateway	33
5 Sender	37
6 Get	38
4. Installation	38
1 Getting Zabbix	38
2 Requirements	39
3 Installation from sources	47
4 Installation from packages	60
5 Installation from containers	75
6 Upgrade procedure	82
7 Known issues	91
8 Template changes	92
9 Upgrade notes for 4.2.0	93
10 Upgrade notes for 4.2.1	95
11 Upgrade notes for 4.2.2	95
12 Upgrade notes for 4.2.3	95
13 Upgrade notes for 4.2.4	95
14 Upgrade notes for 4.2.5	95
15 Upgrade notes for 4.2.6	95
16 Upgrade notes for 4.2.7	95
17 Upgrade notes for 4.2.8	96
18 Upgrade notes for 4.2.9	96
5. Quickstart	96
1 Login and configuring user	96
2 New host	100
3 New item	101
4 New trigger	104
5 Receiving problem notification	106
6 New template	110

6. Zabbix appliance	112
7. Configuration	117
1 Hosts and host groups	123
2 Items	131
3 Triggers	314
4 Events	329
5 Event correlation	332
6 Visualisation	340
7 Templates	386
8 Templates out of the box	387
9 Notifications upon events	391
10 Macros	435
11 Users and user groups	441
8. Service monitoring	446
9. Web monitoring	450
1 Web monitoring items	458
2 Real life scenario	460
10. Virtual machine monitoring	467
1 Virtual machine discovery key fields	471
11. Maintenance	474
12. Regular expressions	478
13. Problem acknowledgement	483
14. Configuration export/import	486
1 Host groups	487
2 Templates	488
3 Hosts	507
4 Network maps	526
5 Screens	534
15. Discovery	540
1 Network discovery	540
2 Active agent auto-registration	549
3 Low-level discovery	552
16. Distributed monitoring	586
1 Proxies	587
17. Encryption	590
1 Using certificates	593
2 Using pre-shared keys	599
3 Troubleshooting	601
18. Web interface	604
1 Frontend sections	605
2 User profile	699
3 Global search	703
4 Frontend maintenance mode	705
5 Page parameters	706
6 Definitions	707
7 Creating your own theme	708
8 Debug mode	709
19. API	710
Method reference	715
Appendix 1. Reference commentary	1103
Appendix 2. Changes from 4.0 to 4.2	1108
Zabbix API changes in 4.2	1110
20. Appendixes	1110
1 Frequently asked questions / Troubleshooting	1110
2 Installation	1111
3 Daemon configuration	1120
4 Protocols	1159
5 Items	1178
6 Triggers	1212
7 Macros	1233
8 Unit symbols	1251
9 Setting time periods	1252
10 Command execution	1253

11 Recipes for monitoring	1253
12 Performance tuning	1255
13 Version compatibility	1257
14 Database error handling	1258
15 Zabbix sender dynamic link library for Windows	1258
16 Issues with SELinux	1259
17 Other issues	1259
Zabbix manpages	1259
zabbix_agentd	1260
NAME	1260
SYNOPSIS	1260
DESCRIPTION	1260
OPTIONS	1260
FILES	1261
SEE ALSO	1261
AUTHOR	1261
Index	1261
zabbix_get	1261
NAME	1262
SYNOPSIS	1262
DESCRIPTION	1262
OPTIONS	1262
EXAMPLES	1263
SEE ALSO	1263
AUTHOR	1263
Index	1263
zabbix_proxy	1263
NAME	1263
SYNOPSIS	1264
DESCRIPTION	1264
OPTIONS	1264
FILES	1264
SEE ALSO	1265
AUTHOR	1265
Index	1265
zabbix_sender	1265
NAME	1265
SYNOPSIS	1265
DESCRIPTION	1266
OPTIONS	1266
EXIT STATUS	1268
EXAMPLES	1268
SEE ALSO	1268
AUTHOR	1268
Index	1269
zabbix_server	1269
NAME	1269
SYNOPSIS	1269
DESCRIPTION	1269
OPTIONS	1269
FILES	1270
SEE ALSO	1270
AUTHOR	1270
Index	1270

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.

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 Zabbix 4.2 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.

IT services 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 acknowledgement** 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 software that monitors numerous parameters of a network and the health and integrity of servers. 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 visualisation 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 organisations 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.

Learn more about [Zabbix features](#).

Users of Zabbix

Many organisations 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 visualisation options

- ability to create custom graphs that can combine multiple items into a single view
- network maps
- custom screens and slide shows for 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 auto registration
- 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, 3rd 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, the responsibilities of which 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.

Data flow

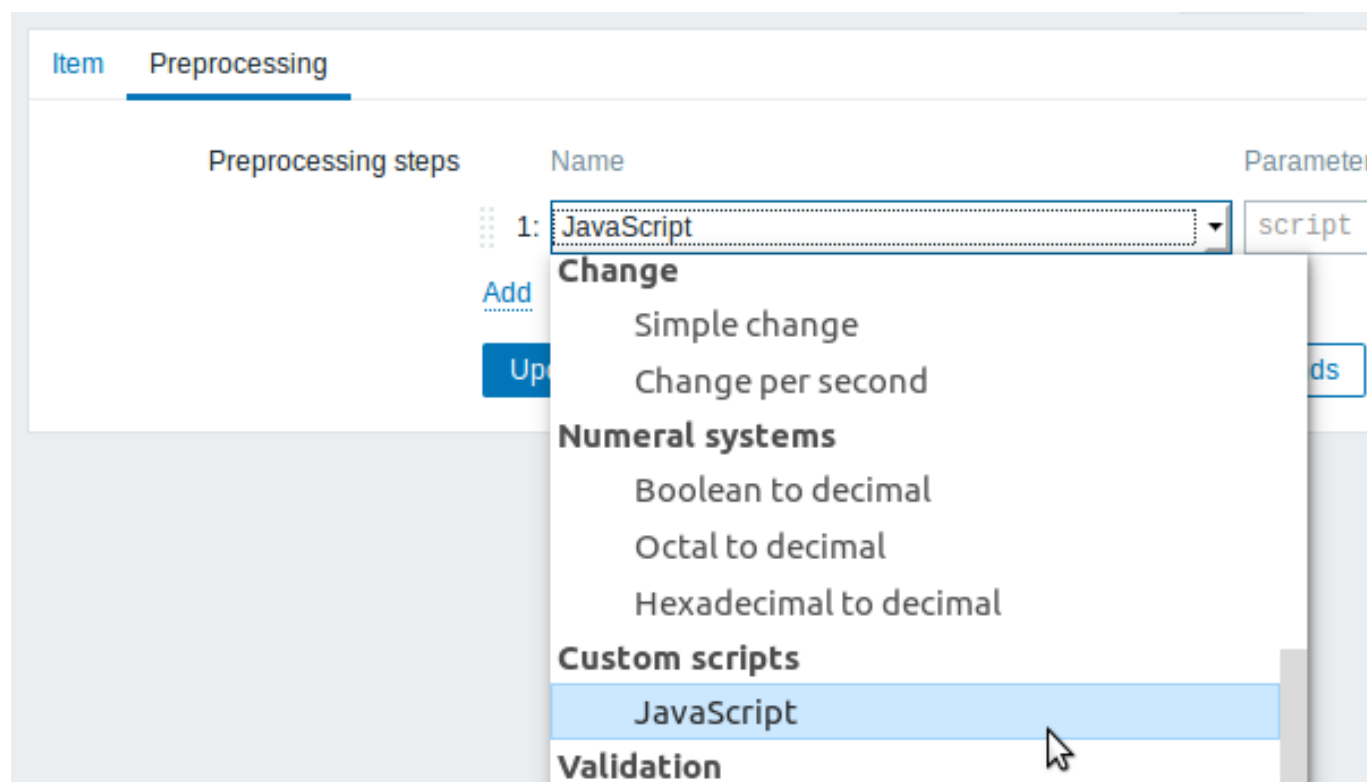
In addition it is important to take a step back and have a look at the overall data flow within Zabbix. In order to create an item that gathers data you must first create a host. Moving to the other end of the Zabbix spectrum you must first have an item to create a trigger. You must have a trigger to 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 4.2.0

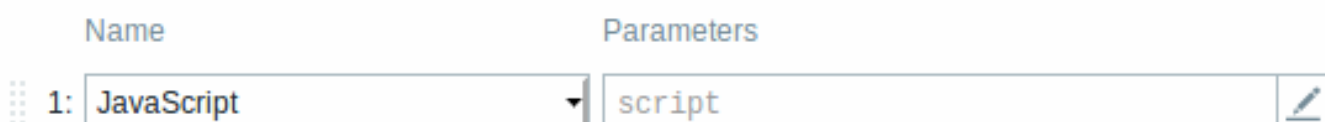
This section introduces new and updated features of Zabbix 4.2.

Extended item value preprocessing JavaScript

It is now possible to apply preprocessing to item value or low-level discovery values using JavaScript.



The JavaScript step accepts one parameter - a non-empty JavaScript code block.



Clicking in the parameter field or on the view/edit button next to the parameter field will open a window for entering the code block.

JavaScript

```
function (value) {  
  1  var stats = {};  
  2  var lines = value.split("\n");  
  3  var i = 0;  
  4  
  5  var reVersion = new RegExp("Server Version: ([^$]+)");  
  6  var reUptime = new RegExp("Server uptime: ([0-9]+) days? ([0-9]+) hours? ([0-9]  
  7  var reLoad = new RegExp("Server load: ([^ ]+) ([^ ]+) ([^ ]+)");  
  8  var reHits = new RegExp("Total accesses: ([^ ]+) \- Total Traffic: ([^ ]+) ([^  
  9  var reCpu = new RegExp("CPU Usage: u([^ ]+) s([^ ]+)");  
10  var lines_num = lines.length;  
11  
12  for (i = 0; i < lines_num; i++)  
13  {  
14      var match = lines[i].match(reVersion);  
15      if (match)  
16      {  
17          stats.version = match[1];  
18          break;  
19      }  
20  }
```

Validation and throttling rules

New options have been added to item value preprocessing focusing on validation and throttling.

New validation options such as defining the acceptable range or some regular expression match allow to remove clearly incorrect or noise values which may happen when, for example, a counter is not initialized yet (empty or 'N/A') or a temperature sensor returns +999°C when it is off, the normal range being -100°C up to +100°C. Validation and custom error handling allows to discard these noise values or replace them with some default value or simply replace them with a human-readable error message.

Extracting an error message provided in incoming data is another new feature. For example, a JSON object may contain a field "error". If the field exists preprocessing is able to extract its value and set it as an error message for the item.

Item
Preprocessing

Preprocessing steps	Name	Parameter
1:	Discard unchanged with heartbeat	seconds
2:	In range	0

Add
Up

Change

- Simple change
- Change per second

Numeral systems

- Boolean to decimal
- Octal to decimal
- Hexadecimal to decimal

Custom scripts

- JavaScript

Validation

- In range
- Matches regular expression
- Does not match regular expression
- Check for error in JSON
- Check for error in XML
- Check for error using regular expression

Throttling

- Discard unchanged
- Discard unchanged with heartbeat

Throttling rules allow to process value only if it is changed, thus reducing the amount of incoming data when it's mostly static. For example, dependent items may be updated less frequently as the master item, provided there's no change to the values. If the master is updated every 10 seconds, a dependent item may be updated every minute.

For more details, see: [Item value preprocessing](#).

Custom error handling

Previously an item would become unsupported if any of the preprocessing steps failed. Now a custom error handling option has been added to the following preprocessing steps:

- Regular expression
- XML XPath
- JSON Path
- Custom multiplier
- Simple change
- Change per second

- Boolean to decimal
- Octal to decimal
- Hexadecimal to decimal
- In range (new validation option)
- Matches regular expression (new validation option)
- Does not match regular expression (new validation option)
- Prometheus pattern
- Prometheus to JSON

If you mark the *Custom on fail* checkbox, the item will not become unsupported in case of a failed preprocessing step and it is possible to specify a custom error handling option, e.g.:

- discard the value
- set to a specified value
- set a specified error message

Testing preprocessing steps

As new item preprocessing steps are being added the preprocessing pipeline has become more complex and with increased chance of mistakes. To help with getting it right Zabbix provides a way for the user to test the configured preprocessing steps.

Each step can be tested individually as well as all steps can be tested together using the new *Test* or *Test all steps* buttons respectively in the Actions block.

See also: [Testing preprocessing steps](#).

Preprocessing support on Zabbix proxy

For hosts monitored by proxies, all item (including low-level discovery rules, dependent items) value preprocessing will now be done on the proxy.

Given new preprocessing options such as Javascript, extensive validation and throttling, preprocessing may become a server bottleneck thus preprocessing by proxies offers the necessary scalability. As this is not a configurable setting it is important to be aware of the performance implications - while preprocessing by proxies is beneficial for Zabbix server performance, as data from proxies will no longer go through the preprocessing manager, the affected proxies will require more resources than before.

Warning:

The change affects the upgrade process to Zabbix 4.2. Please see the [upgrade notes](#).

Extended error messages

Previously, in case of a preprocessing error, the item would become unsupported and the error message would report the number of the failed preprocessing step along with the error description.

Item preprocessing step #<N> failed: <error description>

This was not always enough as the real problem could lie in the step before the one that failed. Therefore, now the error message is extended with a log of the executed steps, reporting on the result or failure of each preprocessing step.

Preprocessing failed for: <value>

1. Result (<custom on fail action>): <value>
2. Result (<custom on fail action>): <value>
3. Failed: <error description>

Note that if the summary error message is larger than 2048 bytes, some of the initial steps may be replaced with '...'. If the header plus the last step error description exceeds 2048 character limit the error message will be truncated.

Low-level discovery Separate processing for low-level discovery

Processing low-level discovery rules has been split from data gathering processes into its own processing.

See: [Daemon improvements](#).

Expected format change in low-level discovery

In order to support preprocessing of low-level discovery result and custom paths to low-level discovery (LLD) macro values in a JSON document, the format of JSON returned by low-level discovery rules has been changed. It is no longer expected that the JSON will contain the "data" object:

```
{
  "data": [
    {<discovery row>},
    {<discovery row>}
  ]
}
```

Low-level discovery will now accept a normal JSON containing an array.

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 as dependent item

Low-level discovery rule now can also be a dependent item, depending on a regular item. Low-level discovery rule cannot be dependent on another low-level discovery rule.

To configure it as a dependent item, just select *Dependent item* for 'Type' and specify the master item in the 'Master item' field in discovery rule [configuration](#).

Preprocessing and custom macros in low-level discovery Low-level discovery configuration has gained additional options of preprocessing the discovery result and extracting custom macros from the preprocessed result. To better understand the changes in the discovery data flow, let's compare it with the previous versions:

Version	LLD data flow
Before Zabbix 4.2	<i>Discovery rule → Discovery rule filter</i>
In Zabbix 4.2	<i>Discovery rule → Preprocessing → Custom macros → Discovery rule filter</i>

It's important to understand that this logic works in the order from left to right, the same as when defining the rule: first the discovery happens, then optionally preprocessing is applied to it, then custom macros may be extracted and then the filter conditions are applied to the result.

Preprocessing

Low-level discovery rules now have a preprocessing step, containing the following options:

- Regular expression match
- JSONPath (structured data)
- Javascript
- Does not match regular expression (validation)
- Check for error in JSON (validation)
- Discard unchanged with heartbeat (throttling)
- Prometheus to JSON

Discovery rule			Preprocessing	LLD macros	Filters
Preprocessing steps			Name	Parameters	
1:	JSONPath	\$pools			
2:	Regular expression	pattern			
			Add		

For more details, see: [Preprocessing](#) in low-level discovery rules.

LLD macro extraction with JSONPath

Low-level discovery rules now also accept extracting user-defined low-level discovery (LLD) macro values using a custom path specified in JSONPath syntax.

LLD macro = JSONPath mappings can now be defined where each macro is defined with a custom JSON path to the value location. The values pointed to by the defined JSON path are used to replace the macros in item, trigger, etc prototype fields.

Discovery rule			Preprocessing	LLD macros	Filters
LLD macros			LLD macro	JSONPath	
			{#NAME}	\$.name	
			{#FSNAME}	\$.fs.name	
			{#FSMOUNT}	\$.fs.mountpoint	

For more details, see: [Custom macros](#) in low-level discovery rules.

Prometheus integration Prometheus integration offers the ability to monitor any object that exposes metrics in Prometheus line format, including Kubernetes, Docker, GitLab, Ceph, Collectd, etcd, InfluxDB, InfluxDB Telegraf and others.

Zabbix uses HTTP items to access Prometheus metrics and then uses preprocessing to query the metrics for specific values.

See [Prometheus checks](#).

TimescaleDB support In the new version Zabbix adds support of a time-series database in the form of **experimental** support of TimescaleDB, a PostgreSQL-based database solution of automatically partitioning data into time-based chunks to support faster performance at scale.

A migration script is available to migrate from existing PostgreSQL tables to a TimescaleDB layout. For more details see: [Migration to TimescaleDB](#).

Warning:

Currently TimescaleDB is not supported by Zabbix proxy.

Using item value to name discovered hosts Previously names of new hosts added as a result of network discovery were using either the DNS name or, in the absence of it, the IP address. In the new version it is possible to give much more descriptive naming to discovered hosts using discovery item values collected either from Zabbix agent or SNMP agent items:

* Name

Discovery by proxy

* IP range

* Update interval

* Checks [Edit](#) [Remove](#)
[New](#)

Device uniqueness criteria ☐ IP address ☒ SNMPv2 agent "iso.3.6.1.2.1.1.1.0"

Host name ☐ DNS name ☐ IP address ☒ SNMPv2 agent "iso.3.6.1.2.1.1.1.0"

Visible name ☒ Host name ☐ DNS name ☐ IP address ☐ SNMPv2 agent "iso.3.6.1.2.1.1.1.0"

Enabled ☒

Zabbix agent or SNMP agent checks are listed as naming options in the new *Host name* and *Visible name* fields, in addition to the ability to specify DNS name or IP address as the name.

Template and host-level tags The possibility to tag events has been a feature of Zabbix since 3.2.0, however, previously, tagging was confined to individual triggers. In the new version tagging can be applied on template and host levels:

- On template level that means all problems of host triggers inherited from these templates will be tagged accordingly
- On host level that means that all host problems will be tagged accordingly

Tags can be defined in a new tab in template and host configuration forms, for example, for a template:

The screenshot shows the 'Templates' configuration page with the 'Tags' tab selected. At the top, there are tabs for 'Template', 'Linked templates', and 'Tags'. Below the tabs, there is a 'Tags' checkbox which is checked, followed by 'Add', 'Replace', and 'Remove' buttons. A table with two columns, 'Name' and 'Value', contains two rows: 'App' with 'MySQL' and 'tag' with 'value'. Below the table is an 'Add' link with a dotted underline. At the bottom are 'Update' and 'Cancel' buttons.

In trigger configuration, tags are now also separated into their own tab. In addition the *Inherited and trigger tags* option allows to view tags defined on template level (but not on host level), if the trigger comes from that template:

The screenshot shows the 'Triggers' configuration page with the 'Tags' tab selected. At the top, there is a breadcrumb trail: 'All hosts / Remote proxy: My host' followed by status indicators 'Enabled', 'ZBX', 'SNMP', 'JMX', 'IPMI', and counts for 'Applications 10', 'Items 43', 'Triggers 22', 'Graphs 10', and 'Discoverers 10'. Below the breadcrumb, there are tabs for 'Trigger', 'Tags', and 'Dependencies'. The 'Tags' tab is active. It shows two sub-tabs: 'Trigger tags' and 'Inherited and trigger tags', with the latter selected. A table with columns 'Name', 'Value', 'Action', and 'Parent templates' contains two rows: 'App' with 'MySQL', 'Remove' action, and 'Template OS Linux' parent; and 'tag' with 'value', 'Remove' action, and no parent. Below the table is an 'Add' link with a dotted underline. At the bottom are 'Update', 'Clone', 'Delete', and 'Cancel' buttons.

An event when it happens inherits all tags from the whole chain of templates, hosts, triggers. Completely identical tag:value combinations are merged into one rather than being duplicated, when marking the event.

See more on [tags](#).

More intuitive dashboard editing Several improvements have been made to the way widgets behave in dashboard editing mode:

- When a widget is made larger, it no longer "throws down" other widgets. Instead the other widgets in any direction try to shrink accordingly using all the available space;
- When a widget is moved down, the widgets below it try to take its place if possible, instead of being dragged along downwards;
- When a widget is moved up, directly adjoining widgets below it are dragged up as well, instead of being left behind;
- If previously it was possible to add a widget only at the end and only in its default size, now it is now possible to add a widget in any open slot and size. See [Adding widgets](#) for details on the changes.

Animated GIF support in maps It is now supported to upload animated GIF images to use in Zabbix maps. To support animated GIF upload, the required minimum GD library version is now 2.0.28.

HTML notifications Email notifications can now be sent in HTML format. HTML format allows for more feature-rich emails containing font colouring, clickable links, interactive elements, company style, etc.

Media types

Media type

Options

* Name

Email

Type

Email

* SMTP server

mail.example.com

SMTP server port

25

* SMTP helo

example.com

* SMTP email

zabbix@example.com

Connection security

None

STARTTLS

SSL/TLS

Authentication

None

Username and password

Message format

HTML

Plain text

Enabled

☒

HTML format can be selected in the *Message format* option when configuring email as the media type in *Administration → Media types*.

Testing media types It is now possible to test if a configured media type works. To do that in the media type list, click on *Test* in the last column.

Create media type

Filter 

	Action
ifo@zabbix.com"	Test
	Test
	Test

Displaying 3 of 3 found

A testing window will open where you can enter a *Send to* recipient address and send a test message subject and body by clicking on *Test*.

Test media type

*

Send to

myaddress@mydomain.com

Subject

Test subject

*

Message

This is the test message from Zabbix

Test

Cancel

Test success or failure message will be displayed in the same window.

Regular expression-based matching in auto-registration conditions Regular expression match is now supported when configuring conditions for active agent **auto-registration** actions. It is supported if using *matches* and *does not match* operators for host name and host metadata conditions.

Action

Operations

*

Name

Active agent auto-registration

Conditions

Label	Name
New condition	Host metadata matches Add

Enabled

☒

*

At least one operation

Note that *contains* and *does not contain* operators that were supported before perform only a substring match.

Web monitoring Content retrieval based on regular expressions also in headers

While previously it was possible to retrieve headers only, it was not possible to seek for matching content in those. Now it is possible to search headers as well for a regular expression match either in variables or the required string.

Additionally, a *Retrieve mode* option has been added to web monitoring **steps**, allowing to:

- Retrieve body only
- Retrieve headers only
- Retrieve both body and headers

Retrieve mode

Body

Headers

Body and headers

Retrieve mode replaces the *Retrieve only headers* option. Thus it is now possible to seek matching content in body only, headers only or in both.

Frontend This section introduces miscellaneous frontend improvements:

Mass update

Mass update for item prototypes

It is now possible to also mass update item prototypes used in **low-level discovery** rules. To update properties of several item prototypes at once, mark the checkboxes before the items and click on *Mass update*:

The screenshot shows the 'Item prototypes' management interface. At the top, there are navigation links: 'All hosts / Remote proxy: My host', status 'Enabled', and discovery methods 'ZBX', 'SNMP', 'JMX', 'IPMI'. On the right, it says 'Discovery list / Network interface'. Below this is a table with columns 'Wizard', 'Name', and 'Key'. Two items are listed, both with checkboxes checked: 'Template OS Linux: Outgoing network traffic on {#IFNAME}' and 'Template OS Linux: Incoming network traffic on {#IFNAME}'. At the bottom, there are buttons: '2 selected', 'Create enabled', 'Create disabled', 'Mass update' (highlighted with a green box), and 'Delete'.

Then update the properties you need in the mass **update form** that is identical to that of regular items.

Mass update for templates

Mass update is now available in the **list** of templates.

Filtering

Trigger filtering

Several new options have been added to the filter in the trigger list. Importantly:

- it is now possible to specify multiple hosts and host groups in the filter allowing to view triggers of more than one host, greatly increasing the usability of such functions as mass update. Also, the dropdowns of host and host group selection have been removed as these selections now can be made in the filter;
- it is also possible to filter by several severities:

The screenshot shows the 'Triggers' filter interface. It includes sections for 'Host groups' (with a search box and 'Select' button), 'Hosts' (with 'Zabbix server' and 'My host' selected and a 'Select' button), 'Name' (with a search box), 'Severity' (with checkboxes for 'Not classified', 'Information', 'Warning', 'Average', 'High', 'Disaster'), 'State' (with buttons 'all', 'Normal', 'Unknown'), 'Status' (with buttons 'all', 'Enabled', 'Disabled'), and 'Value' (with buttons 'all', 'Ok', 'Problem'). On the right, there are 'Tags' (with 'And/Or' and 'Or' buttons and a search box), 'Inherited' (with buttons 'all', 'Yes', 'No'), 'Discovered' (with buttons 'all', 'Yes', 'No'), and 'With dependencies' (with buttons 'all', 'Yes', 'No').

See trigger **filter details** for more information.

Item filtering

The possibility to specify multiple hosts and host groups in the filter, introduced in the trigger filter (see above), is also available in the item filter.

If no host or host group is specified in the item filter, now all items will be displayed instead of none and a message to select something.

See item [filter details](#) for more information.

Macros New macros supported in maps

More macros are now supported in map element labels, URL names and URL values:

New supported location in maps	Macros	
	Host element	Host TriggerMap groupel- el- el- e- e- e- ment ment ment
Element label	{HOST.ID}, {INVENTORY.*}	{HOST.TRIGGER.ID}, {HOST.GROUP.ID}, {INVENTORY.MACNAME}
URL name	macros	macros
URL name and value	{HOST.ID}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.IP}, {INVENTORY.*}	{HOST.TRIGGER.ID}, {HOST.GROUP.ID}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.IP}, {INVENTORY.*}
	macros	macros

Note that this table lists only the new locations. So, even though {HOST.ID}, {HOSTGROUP.ID}, {TRIGGER.ID} and {MAP.ID} are listed as now supported in map URL names, they were supported in the URL values already. Similarly, the many {HOST.*} macros were already supported in element labels.

See also: [All supported macros](#).

Daemons Separate processing for low-level discovery

Processing low-level discovery rules has been split from data gathering processes into its own processing, consisting of a configurable number of low-level discovery workers and a low-level discovery manager. This change greatly reduces low-level discovery impact on data gathering (poller, trapper processes) and proxy communications (trapper, proxypoller processes). Before data gathering could be delayed because the corresponding data gathering processes were busy with low-level discovery.

See also: [StartLLDProcessors](#) parameter

Sender uses all ServerActive addresses

Zabbix sender now uses all addresses defined in the agent ServerActive configuration parameter for sending data, instead of the first entry only. 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.

See also

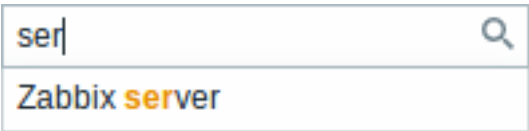
- [Template changes](#)

6 What's new in Zabbix 4.2.1

Frontend

Global search

The [global search](#) field in the top-right corner has been improved. Starting to type in it will show all hosts with matching strings in any part of the name, not only in the beginning of the name as before.



The global search field can now also match the technical name to the visible name of the host. In this case the matching visible name is listed as an option, but not highlighted:

0017| 

Zabbix server

Discovery rule filtering

A filter has been added for discovery rule filtering in *Monitoring* → *Discovery*. The filter replaces the dropdown from previous Zabbix versions. If you start typing the name in the filter, all matching enabled discovery rules will be listed for selection.

Status of discovery 

Filter 

Discovery rule

Local network

Local network_2

More than one discovery rule can be selected now.

Filter 

Discovery rule

type here to search

Copying items, triggers, graphs

The form for item/trigger/graph copying to other targets has been changed:

- The *Target type* field is now a radio button type, instead of a dropdown;
- The *Target* field allows to multiselect, rather than displaying all possible targets in what could be an endless list. Starting to type in the *Target* field will show all available corresponding targets.

|<|<|<|<|

Separated preprocessing tab in item mass update

In item **mass update** the preprocessing options are now separated into their own tab.

7 What's new in Zabbix 4.2.2

Item value preprocessing testing

The preprocessing testing form now contains a switch between CRLF/LF end of line sequences for multiline input:

Test item preprocessing

Value 

Previous value 

End of line sequence ☐ ☐

Preprocessing steps

Name

1: Regular expression

Previously multiline input was sent with CRLF (\r\n) by default, which was incompatible with, for example, Prometheus line format.

Custom headers in web scenarios/HTTP checks

The length of custom header value fields has been increased from 255 to 1000 characters.

Frontend

Human readable time in Plain text widget

When displaying time in the *Plain text* dashboard **widget**, it is now displayed in a human readable form.

|<| |<| |-| |<|

Daemons

Warnings about incompatible versions

Warnings are now being logged if incompatible Zabbix daemon versions are used.

Windows agent compilation revision

Generating a Windows file properties revision number has been added for agent compilation on MS Windows. It follows a {b}{t}{nn} format where:

- {b} - source (1 - feature or release, 2 - tag)
- {t} - type (1 - alpha, 2 - beta, 3 - release candidate, 4 - release)
- {nn} - sequence number for the 'type'

For example:

Tag	Branch	Version	Result
4.2.2		Zabbix 4.2.2	4.2.2.2400
	release/4.2	Zabbix 4.2.2rc3	4.2.2.1303
	feature/ZBX-16074	Zabbix 4.2.2rc1	4.2.2.1301

8 What's new in Zabbix 4.2.3

This minor version contains no functional changes.

9 What's new in Zabbix 4.2.4

VMware datastore discovery

VMware datastore discovery has been moved from hypervisor level to vCenter level. As a result datastores are discovered and datastore info is collected only once for each datastore instead of one time per each hypervisor.

Within this development the following new items have been added:

- vmware.datastore.discovery[<url>]
- vmware.datastore.hv.list[<url>,<datastore>]
- vmware.datastore.read[<url>,<datastore>,<mode>]
- vmware.datastore.write[<url>,<datastore>,<mode>]
- vmware.datastore.size[<url>,<datastore>,<mode>]
- vmware.hv.datastore.list[<url>,<uuid>]

For more information on the added items, see: **VMware monitoring item keys**

Item changes

web.page.get, **web.page.perf**, **web.page.regexp** agent **items** now along with hostname also support URL as the first parameter, allowing to specify both HTTP and HTTPS protocols.

As a prerequisite of SSL (HTTPS) functionality, the agent should be compiled with cURL support. Previously, only domain name or IP address was allowed in the first parameter and there was no support for HTTPS protocol.

Additionally, Unicode is now supported in the hostnames of these items (Punycode support was added).

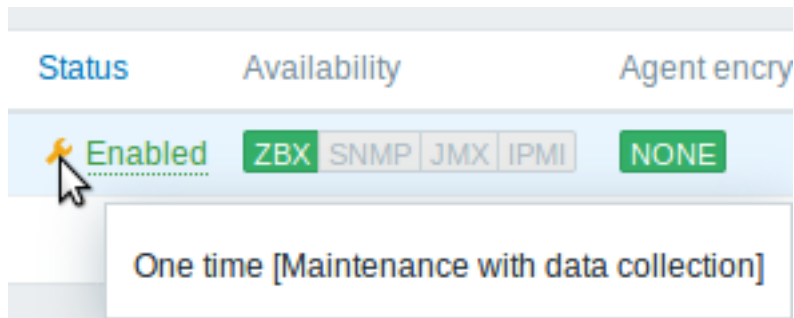
Configuration parameters

The **LoadModule** server/proxy/agent parameter now supports specifying relative or absolute path to the module.

Frontend

Maintenance icon in host list

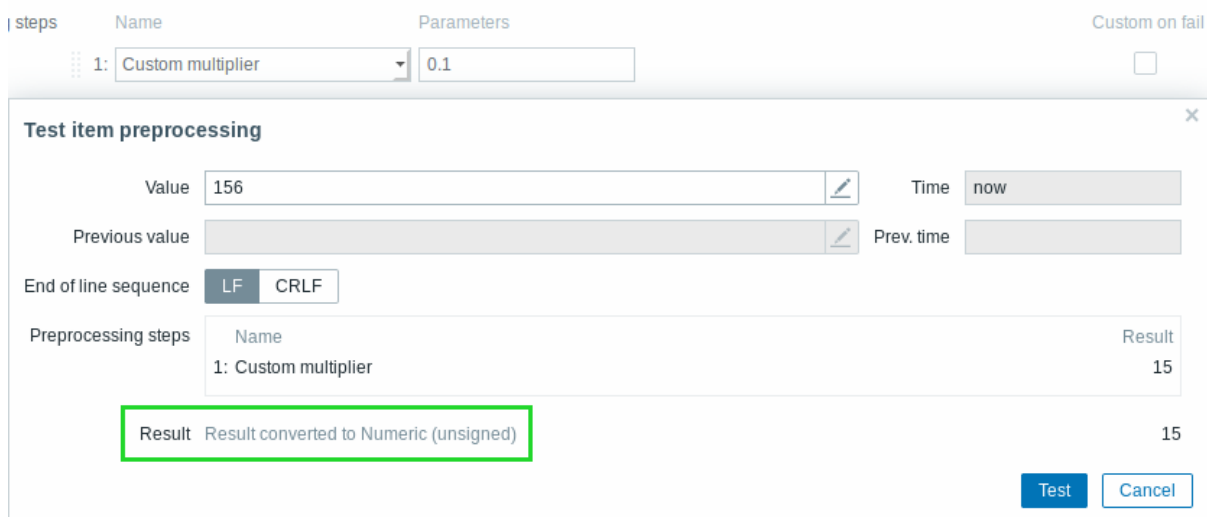
An orange wrench icon  next to the host status now indicates that this host is in maintenance in *Configuration → Hosts*. Maintenance details are displayed when the mouse pointer is positioned over the icon:



Previously, an orange *In maintenance* string was displayed as status instead of the icon.

Result of preprocessing testing displayed

A new *Result* line has been added in the preprocessing testing form. In it, 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 image shows a 'Test item preprocessing' dialog box. At the top, there's a table with columns 'steps', 'Name', 'Parameters', and 'Custom on fail'. The first row shows '1: Custom multiplier' with a parameter value of '0.1'. Below this, the dialog has fields for 'Value' (156), 'Time' (now), 'Previous value', and 'Prev. time'. There are also buttons for 'End of line sequence' (LF, CRLF). A section for 'Preprocessing steps' shows a table with one step: '1: Custom multiplier' with a 'Result' of '15'. Below this, a green-bordered box highlights the text 'Result Result converted to Numeric (unsigned)' with a '15' to its right. At the bottom right are 'Test' and 'Cancel' buttons.

The type of conversion to the value type of the item is also displayed, for example *Result converted to Numeric (unsigned)*. An error icon is displayed in case of errors.

Regular expressions

The *File systems for discovery* global **regular expression** has been updated to include "apfs".

10 What's new in Zabbix 4.2.5

Full JSONPath functionality supported

Full support of **JSONPath functionality** has been added in item value preprocessing and low-level discovery macros.

Item history

Global housekeeper settings no longer override item-level history/trends settings in the case when keeping history/trends is entirely disabled on the item level.

In addition, if previously disabling item history/trends had to be specified by entering '0' as the storage period value, now the same can be done using a dedicated radio button.

* History storage period

Do not keep history

Storage period

* Trend storage period

Do not keep trends

Storage period

365d

11 What's new in Zabbix 4.2.6

Manual closing of problems

When **updating a problem**, the checkbox for closing a problem is now 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*. Already closed problems will not be closed repeatedly. If no problem is manually closeable, the checkbox is disabled.

Previously, closing a problem in this way was only possible if the *Allow manual close* option was checked in **trigger configuration** for each selected problem. If at least one problem was not manually closeable, the checkbox was disabled.

Similarly, when changing problem severity the relevant checkbox is now available if at least one selected problem is read-writable. Previously each selected problem had to be read-writable, otherwise the checkbox was disabled.

Global notifications

Several improvements have been made to **global notification** messages and sounds:

- Notifications are synchronized across tabs and windows. Any changes to the list of notifications are immediately rendered into other tabs.
- Notification timeout is calculated since the first rendering time of the notification.
- The *Display OK triggers for setting* is now applied to recovery type notifications. That becomes relevant if it is shorter than the regular problem notification message timeout thus the notification disappears sooner.
- Default audio clips are now in MP3 (it used to be wave files). An embedded media player is not used any more (it was used for Internet Explorer only), so now, in case of custom audio files, you must be sure that your browser can play them (or just use the MP3 format).
- The sound tends to follow the focused tab/window, except when *Play sound* is set to 'Once'.
- When *Play sound* is set to 'Once' it will not replay the sound on page reload.
- In case of several notifications, the sound of the most severe or the most recent notification is played.
- There is now an icon next to each snoozed notification.
- It is possible to snooze a whole list of notifications of *High* severity and be assured to receive an alarm when an *Average* severity notification arrives, meanwhile keeping the notification list with all event links open.
- If you snooze a notification in problem state, it will not play audio when the particular problem becomes resolved.
- Regardless of how many tabs are opened the server is polled by only one of them, reducing server load. Additionally the server response traffic is minimized, by not containing the message body and title in response, if the client receives the same notification ID.
- Recovery of a crashed tab is implemented. Any tab that is still able to execute JavaScript will detect a crashed tab and would continue to be the one who polls server. In case of a slow connection and heavy page load during the request, any other tab would continue to poll for notifications. Thus with a notification timeout (30 seconds) and a performed request (40 seconds) one can be assured that no notifications will miss the eye if there is another tab..
- Slide animations are improved. Fade out will happen when all remaining notifications reach the message timeout at once, else the consecutive slide-up transition is applied.
- Updated user settings, such as the timeout and audiofile, are applied to the list of notifications dynamically.

12 What's new in Zabbix 4.2.7

Increased proxy-server limit per connection

Limit per connection has been increased from 128MB to 1GB packet size (ZBX_MAX_RECV_DATA_SIZE constant).

Working with English perfcounters on Windows

It is now possible to work with exclusively English names of Windows performance counters, regardless of the locale, using the new items/parameters:

- perf_counter_en Windows agent **item**
- PerfCounterEn Windows agent **parameter**

Note that these items will only work on **Windows Server 2008/Vista** and above.

13 What's new in Zabbix 4.2.8

Cisco UCS server template

A Cisco UCS server monitoring template - *Template Server Cisco UCS SNMPv2* has been added.

It will be available in *Configuration → Templates* in new installations. If you are upgrading from previous versions, you can find this template in the `templates` directory of the downloaded latest Zabbix version. Then, while in *Configuration → Templates* you can import it manually into Zabbix.

Macros

A new {EVENT.RECOVERY.NAME} macro is supported, returning the recovery event name.

{TRIGGER.ID} macro is now 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.

JSONPath preprocessing

Quoted numeric values are now accepted by the JSONPath **aggregate functions** (sum, min, max, etc.) in item value preprocessing. It means that the values are converted from string type to numeric if aggregation is required.

Acknowledged problems in trigger overview

The icon that indicates an acknowledged problem in *Monitoring → Overview* now is displayed only if all problems or resolved problems of the trigger are acknowledged. Previously it was enough for the last problem to be acknowledged.

2. Definitions

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

Definitions **host**

- a networked device that you want to monitor, with IP/DNS.

host group

- a logical grouping of hosts; it may contain hosts and templates. Hosts and templates within a host group are not in any way linked to each other. 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 auto-registration 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, screens, applications, 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.

application

- a grouping of items in a logical group.

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 visualisations 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 agent

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

Zabbix proxy

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

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, application prototypes.

agent auto-registration

- 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.conf)
-R --runtime-control <option>	perform administrative functions
-h --help	give this help
-V --version	display version number

Note:

Runtime control is not supported on OpenBSD and NetBSD.

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.	
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.	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.	

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
```

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"
```

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** - manager of alerter tasks
- **alerter** - process for sending notifications
- **configuration_syncer** - process for managing in-memory cache of configuration data
- **discoverer** - process for discovery of devices
- **escalator** - process for escalation of actions
- **history_syncer** - history DB writer
- **housekeeper** - process for removal of old historical data
- **http_poller** - web monitoring poller
- **icmp_ping** - poller for icmping 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
- **poller** - normal poller for passive checks
- **preprocessing_manager** - manager of preprocessing tasks
- **preprocessing_worker** - process for data preprocessing
- **proxy_poller** - poller for passive proxies
- **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.

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 for:

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris: 9, 10, 11
- Windows: all desktop and server versions since XP

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\win64\zabbix_agentd.exe and conf\zabbix_agentd.win.conf files to c:\zabbix.

Edit the c:\zabbix\zabbix_agentd.win.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.win.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	

Parameter	Description
-m --multiple-agents	Use multiple agent instances (with -i,-d,-s,-x functions). To distinguish service names of instances, each service name will include the Hostname value from the specified configuration file.
Windows agent only (functions)	
-i --install	Install Zabbix Windows agent as service
-d --uninstall	Uninstall Zabbix Windows agent service
-s --start	Start Zabbix Windows agent service
-x --stop	Stop Zabbix Windows agent service

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

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

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

Runtime control

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

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

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 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 or IBM DB2 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
-R --runtime-control <option>	perform administrative functions
-h --help	give this help
-V --version	display version number

Note:

Runtime control is not supported on OpenBSD and NetBSD.

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.	
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.	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.	

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
```

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

- `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 icmping checks
- `ipmi_manager` - IPMI poller manager
- `ipmi_poller` - poller for IPMI checks
- `java_poller` - poller for Java checks
- `poller` - normal poller for passive checks
- `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 [server#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.

4 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, since Zabbix 2.0.15, Zabbix 2.2.10 and Zabbix 2.4.5 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
<i>Sources</i>	Installation	Setup
<i>RHEL/CentOS packages</i>	Installation	Setup
<i>Debian/Ubuntu packages</i>	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.

5 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.

6 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.regex[www.zabbix.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:

<code>-s --host <host name or IP></code>	Specify host name or IP address of a host.
<code>-p --port <port number></code>	Specify port number of agent running on the host. Default is 10050.
<code>-I --source-address <IP address></code>	Specify source IP address.
<code>-k --key <item key></code>	Specify key of item to retrieve value of.
<code>-h --help</code>	Give this help.
<code>-V --version</code>	Display version number.

See also [Zabbix get manpage](#) for more information.

Zabbix get on Windows can be run similarly:

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

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. 128 MB of physical memory and 256 MB of free disk space could be a good starting point. However, 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. Amount of memory allocated for the connection depends on 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 several examples of hardware configurations:

Name	Platform	CPU/Memory	Database	Monitored hosts
<i>Small</i>	CentOS	Virtual Appliance	MySQL InnoDB	100
<i>Medium</i>	CentOS	2 CPU cores/2GB	MySQL InnoDB	500
<i>Large</i>	RedHat Enterprise Linux	4 CPU cores/8GB	RAID10 MySQL InnoDB or PostgreSQL	>1000
<i>Very large</i>	RedHat Enterprise Linux	8 CPU cores/16GB	Fast RAID10 MySQL InnoDB or PostgreSQL	>10000

Note:

Actual configuration depends on the number of active items and refresh rates very much. It is highly recommended to run the database on a separate box for large installations.

Supported platforms

Due to security requirements and mission-critical nature of 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 is tested on the following platforms:

- Linux
- IBM AIX
- FreeBSD
- NetBSD
- OpenBSD
- HP-UX
- Mac OS X
- Solaris
- Windows: all desktop and server versions since XP (Zabbix agent only)

Note:

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

Attention:

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

Software

Zabbix is built around a modern Apache web server, leading database engines, and PHP scripting language.

Database management system

Software	Version	Comments
<i>MySQL</i>	5.0.3 - 8.0.x	Required if MySQL is used as Zabbix backend database. InnoDB engine is required. MariaDB also works with Zabbix.
<i>Oracle</i>	10g or later	Required if Oracle is used as Zabbix backend database.
<i>PostgreSQL</i>	8.1 or later	Required if PostgreSQL is used as Zabbix backend database. It is suggested to use at least PostgreSQL 8.3, which introduced much better VACUUM performance .
<i>TimescaleDB</i>	1.0 or later, OSS (free) version	Required if TimescaleDB is used as Zabbix backend database.
<i>IBM DB2</i>	9.7 or later	Required if IBM DB2 is used as Zabbix backend database.
<i>SQLite</i>	3.3.5 or later	SQLite is only supported with Zabbix proxies. Required if SQLite is used as Zabbix proxy database.

Attention:

IBM DB2 and TimescaleDB support is experimental!

Frontend

The following software is required to run Zabbix frontend:

Software	Version	Comments
<i>Apache</i>	1.3.12 or later	
<i>PHP</i>	5.4.0 or later	
PHP extensions: <i>gd</i>	2.0.28 or later	PHP GD extension must support PNG images (<code>--with-png-dir</code>), JPEG (<code>--with-jpeg-dir</code>) images and FreeType 2 (<code>--with-freetype-dir</code>).
<i>bcmath</i>		php-bcmath (<code>--enable-bcmath</code>)
<i>ctype</i>		php-ctype (<code>--enable-ctype</code>)
<i>libXML</i>	2.6.15 or later	php-xml or php5-dom, if provided as a separate package by the distributor.

Software	Version	Comments
<i>xmlreader</i>		php-xmlreader, if provided as a separate package by the distributor.
<i>xmlwriter</i>		php-xmlwriter, if provided as a separate package by the distributor.
<i>session</i>		php-session, if provided as a separate package by the distributor.
<i>sockets</i>		php-net-socket (<i>--enable-sockets</i>). Required for user script support.
<i>mbstring</i>		php-mbstring (<i>--enable-mbstring</i>)
<i>gettext</i>		php-gettext (<i>--with-gettext</i>). Required for translations to work.
<i>ldap</i>		php-ldap. Required only if LDAP authentication is used in the frontend.
<i>ibm_db2</i>		Required if IBM DB2 is used as Zabbix backend database.
<i>mysqli</i>		Required if MySQL is used as Zabbix backend database.
<i>oci8</i>		Required if Oracle is used as Zabbix backend database.
<i>pgsql</i>		Required if PostgreSQL is used as Zabbix backend database.

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.

Web browser on client side

Cookies and Java Script must be enabled.

Latest versions of Google Chrome, Mozilla Firefox, Microsoft Internet Explorer and Opera are supported. Other browsers (Apple Safari, Konqueror) may work with Zabbix as well.

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 screens or dashboards on <http://secure-zabbix.com/zabbix/>, will have full JS access to Zabbix.

Server

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

Requirement	Status	Description
<i>libpcre</i>	Mandatory	PCRE 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'. Note that you need exactly PCRE (v8.x); PCRE2 (v10.x) library is not used.

Requirement	Status	Description
<i>libevent</i>	Optional	Required for bulk metric support and IPMI monitoring. Version 1.4 or higher. Note that for Zabbix proxy this requirement is optional; it is needed for IPMI monitoring support.
<i>libpthread</i>		Required for mutex and read-write lock support.
<i>zlib</i>		Required for compression support.
<i>OpenIPMI</i>		Required for IPMI support.
<i>libssh2</i>		Required for SSH support. Version 1.0 or higher.
<i>fping</i>		Required for ICMP ping items .
<i>libcurl</i>		Required for web monitoring, VMware monitoring, SMTP authentication, <code>web.page.*</code> 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
<i>libiksemel</i>		Required for Jabber support.
<i>libxml2</i>		Required for VMware monitoring and XML XPath preprocessing.
<i>net-snmp</i>		Required for SNMP support.

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	License	Website	Comments
<i>logback-core-0.9.27.jar</i>	EPL 1.0, LGPL 2.1	http://logback.qos.ch/	Tested with 0.9.27, 1.0.13, and 1.1.1.
<i>logback-classic-0.9.27.jar</i>	EPL 1.0, LGPL 2.1	http://logback.qos.ch/	Tested with 0.9.27, 1.0.13, and 1.1.1.
<i>slf4j-api-1.6.1.jar</i>	MIT License	http://www.slf4j.org/	Tested with 1.6.1, 1.6.6, and 1.7.6.
<i>android-json-4.3_r3.1.jar</i>	Apache License 2.0	https://android.googlesource.com/platform/libcore/+/_master/json	Tested with 2.3.3_r1.1 and 4.3_r3.1. See <code>src/zabbix_java/lib/README</code> for instructions on creating a JAR file.

Java gateway compiles and runs with Java 1.6 and above. It is recommended that those who provide a precompiled version of the gateway for others use Java 1.6 for compilation, so that it runs on all versions of Java up to the latest one.

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 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, 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 items². In our case, it means that 130M of values will require $130M * 90 \text{ bytes} = 10.9GB$ 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 customised.

Zabbix database, depending on 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 space¹. 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 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 space¹. 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 $3*365*24*3600*(250+80+\#tags*100) = \sim 30GB + \#tags*100B$ disk space².

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)
<i>Zabbix configuration</i>	Fixed size. Normally 10MB or less.
<i>History</i>	$\text{days} * (\text{items} / \text{refresh rate}) * 24 * 3600 * \text{bytes}$ 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.
<i>Trends</i>	$\text{days} * (\text{items} / 3600) * 24 * 3600 * \text{bytes}$ items : number of items days : number of days to keep history bytes : number of bytes required to keep single trend, depends on database engine, normally ~90 bytes.
<i>Events</i>	$\text{days} * \text{events} * 24 * 3600 * \text{bytes}$ 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 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 synchronisation

It is very important to have precise system date on 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 synchronised system date on all systems Zabbix components are running on.

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.

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 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 "Zabbix Admin" user type has less privileges than "Zabbix 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.

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.

Setting up SSL for Zabbix frontend

On RHEL/CentOS, install mod_ssl package:

```
yum 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
```

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 / http://example.com
</VirtualHost>
```

Restart the Apache service to apply the changes:

```
systemctl restart httpd.service
```

Enabling HTTP Strict Transport Security (HSTS) on web server

To protect Zabbix frontend against protocol downgrade attacks, we recommend to enable [HSTS](#) policy on webserver.

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:

▼ **Response Headers** [view source](#)

Cache-Control: no-store, no-cache, must-revalidate
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 1160
Content-Type: text/html; charset=UTF-8
Keep-Alive: timeout=5, max=100
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)

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:

Not Found

The requested URL /custom-text was not found on this server.

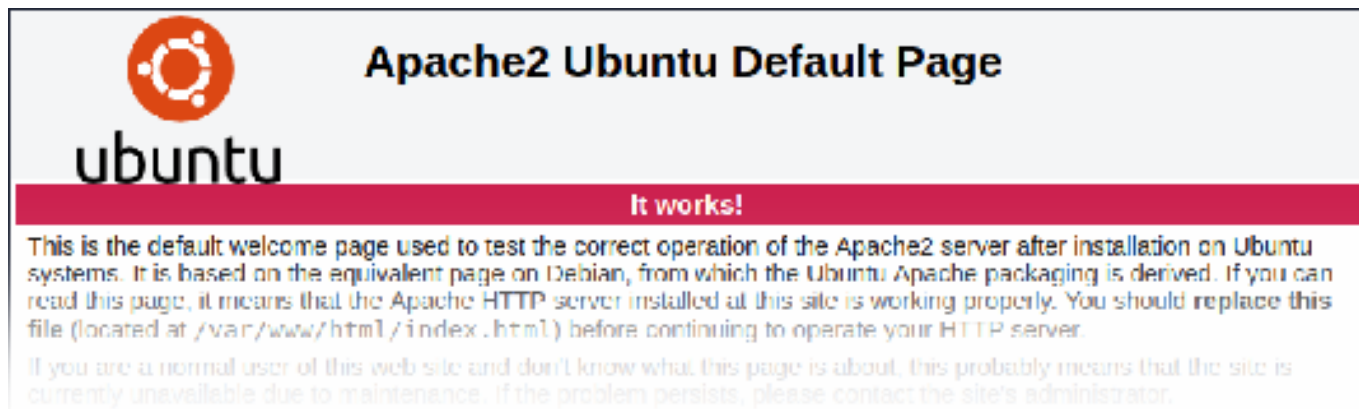
Apache/2.4.18 (Ubuntu) Server at localhost Port 80

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.

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-4.2.0.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

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
```

Note:

For virtual machine monitoring **--with-libcurl** and **--with-libxml2** configuration options are required; **--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.

Attention:

Since version 3.4.0, Zabbix will always compile with the PCRE library; 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.

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
```

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**.

Attention:

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.

Note:

Command-line utilities **zabbix_get** and **zabbix_sender** are compiled if **--enable-agent** option is used.

Note:

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.

Note:

Use `--with-ibm-db2` flag to specify location of the CLI API.
Use `--with-oracle` flag to specify location of the OCI API.

For encryption support see [Compiling Zabbix with encryption support](#).

5 Make and install everything

Note:

If installing from SVN, 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. See the [performance tuning](#) section for more details.

- 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.
See the ["See also"](#) section at the bottom of this page to find out how to configure shared memory.

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 `frontends/php` 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 frontends/php
cp -a . <htdocs>/zabbix
```

If installing from SVN and planning to use any other language than English, you must generate translation files. To do so, run:

```
locale/make_mo.sh
```

`msgfmt` utility from `gettext` package is required.

Note:

Additionally, to use any other language than English, its locale should be installed on the web server. See the ["See also"](#) section in the "User profile" page to find out how to install it if required.

Note:

In Zabbix 4.2.2 (only) the web server user requires *write* permissions to the `assets` directory.

Installing frontend

Step 1

In your browser, open Zabbix URL: `http://<server_ip_or_name>/zabbix`

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



Step 2

Make sure that all software prerequisites are met.



Check of pre-requisites

Welcome

Check of pre-requisites

Configure DB connection

Zabbix server details

Pre-installation summary

Install

	CURRENT VALUE	REQUIRED	
PHP version	5.4.20	5.4.0	OK
PHP option memory_limit	128M	128M	OK
PHP option post_max_size	32M	16M	OK
PHP option upload_max_filesize	16M	2M	OK
PHP option max_execution_time	600	300	OK
PHP option max_input_time	600	300	OK
PHP time zone	Europe/Riga		OK
PHP databases support	MySQL		OK
PHP bcmath	on		OK

Back

Next step

Pre-requisite	Minimum value	Description
PHP version	5.4.0	
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, IBM DB2	One of the following modules must be installed: mysql, oci8, pgsql, ibm_db2
bcmath		php-bcmath
mbstring		php-mbstring
PHP mbstring.func_overload option	must be disabled	In php.ini: mbstring.func_overload = 0
PHP always_populate_raw_post_data option	must be disabled	Required only for PHP versions 5.6.0 or newer. In php.ini: always_populate_raw_post_data = -1
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 or php5-dom
xmlwriter		php-xmlwriter
xmlreader		php-xmlreader
ctype		php-ctype
session		php-session

Pre-requisite	Minimum value	Description
<i>gettext</i>		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.

Step 3

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

Step 4

Enter Zabbix server details.



- Welcome
- Check of pre-requisites
- Configure DB connection
- Zabbix server details
- Pre-installation summary
- Install

Zabbix server details

Please enter the host name or host IP address and port number of the Zabbix server, as well as the name of the installation (optional).

Host

Port

Name

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

Step 5

Review a summary of settings.



- Welcome
- Check of pre-requisites
- Configure DB connection
- Zabbix server details
- Pre-installation summary
- Install

Pre-installation summary

Please check configuration parameters. If all is correct, press "Next step" button, or "Back" button to change configuration parameters.

Database type	MySQL
Database server	localhost
Database port	default
Database name	zabbix
Database user	zabbix
Database password	*****
Zabbix server	localhost
Zabbix server port	10051
Zabbix server name	

Step 6

Download the configuration file and place it under conf/ in the webserver HTML documents subdirectory where you copied Zabbix PHP files to.

ZABBIX

Welcome

Check of pre-requisites

Configure DB connection

Zabbix server details

Pre-installation summary

Install

Install

⚠

Details ▲ Cannot create the configuration file.

Unable to create the configuration file.

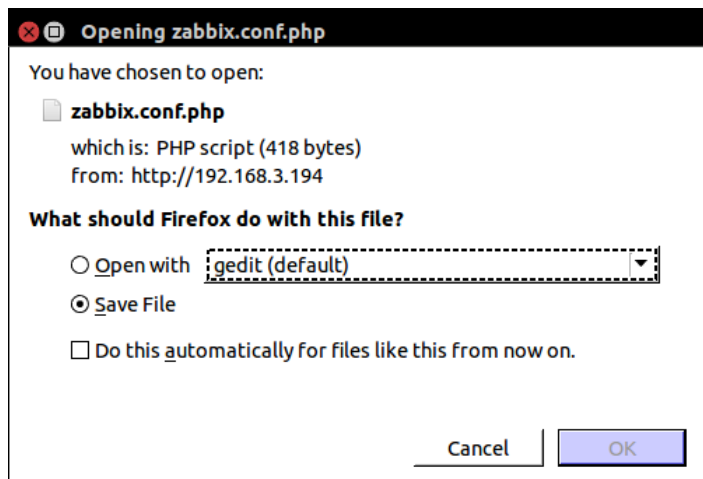
Alternatively, you can install it manually:

1. [Download the configuration file](#)

2. Save it as "var/www/html/zabbix/conf/zabbix.conf.php"

Back

Finish

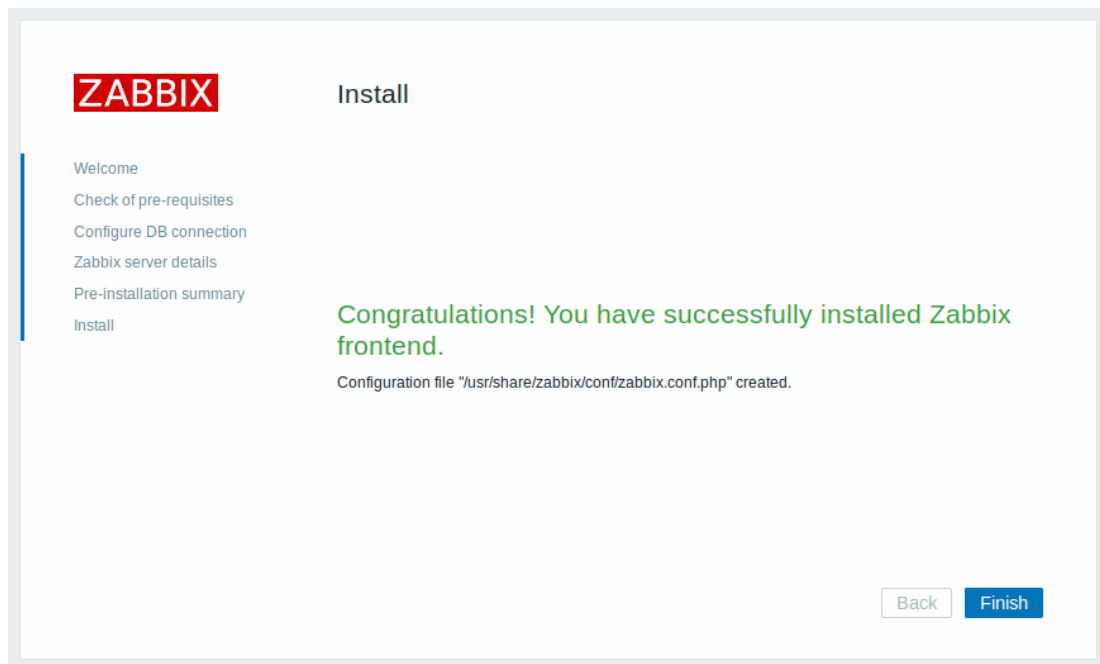


Note:

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.

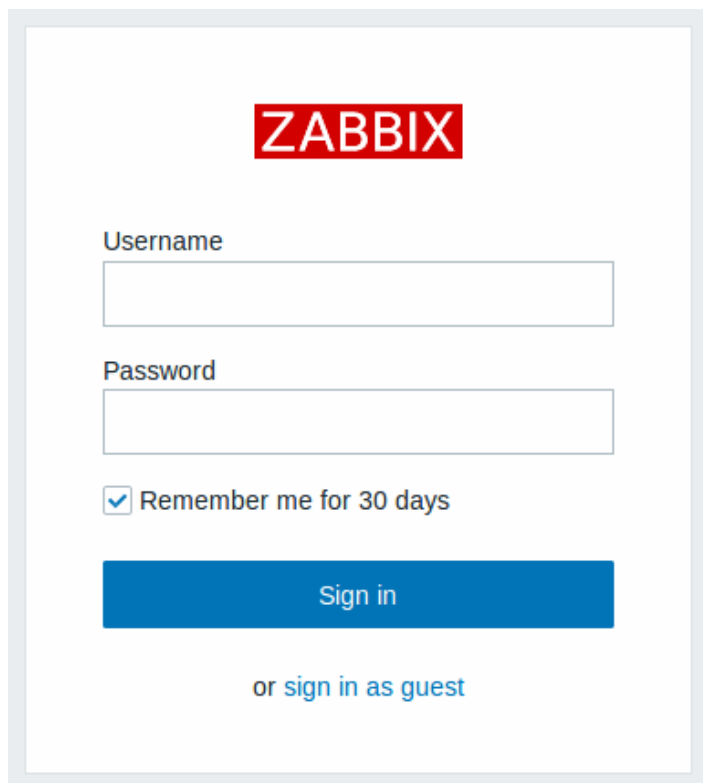
Step 7

Finish the installation.



Step 8

Zabbix frontend is ready! The default user name is **Admin**, password **zabbix**.



Proceed to [getting started with Zabbix](#).

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.

See also

1. [How to configure shared memory for Zabbix daemons](#)

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). If you want to build agent binaries with TLS, you will also need OpenSSL or GnuTLS.

To install command line developer tools, open terminal and run `svn` command. Instead of printing short help message, it will display informative message about missing command line developer tools and open new GUI window for installing them. Press "Install" in the dialog window.

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 PCRE:

```
$ brew install pcre
```

When building with TLS, install OpenSSL and/or GnuTLS:

```
$ brew install openssl
$ brew install gnutls
```

Download Zabbix source:

```
$ svn export svn://svn.zabbix.com/trunk zabbix-source
```

Build agent without TLS:

```
$ cd zabbix-source/
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6
$ make
$ make install
```

Build agent with OpenSSL:

```
$ cd zabbix-source/
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-openssl=/usr/local/opt
$ 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 PCRE 8.42.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre-8.42"
```

Download and build PCRE with Unicode properties support:

```
$ mkdir static-libs-source
$ cd static-libs-source
$ curl --remote-name https://ftp.pcre.org/pub/pcre/pcre-8.42.tar.gz
$ tar xf pcre-8.42.tar.gz
$ cd pcre-8.42
$ ./configure --prefix="$PCRE_PREFIX" --disable-shared --enable-static --enable-unicode-properties
$ make
$ make check
$ make install
```

Download Zabbix source and build agent:

```
$ svn export svn://svn.zabbix.com/trunk zabbix-source
$ cd zabbix-source/
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre="$PCRE_PREFIX
$ 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 PCRE 8.42 and OpenSSL 1.1.1a.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre-8.42"
$ 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://ftp.pcre.org/pub/pcre/pcre-8.42.tar.gz
$ tar xf pcre-8.42.tar.gz
$ cd pcre-8.42
$ ./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:

```
$ svn export svn://svn.zabbix.com/trunk zabbix-source
$ cd zabbix-source/
$ ./bootstrap.sh
$ ./configure --sysconfdir=/usr/local/etc/zabbix --enable-agent --enable-ipv6 --with-libpcre="$PCRE_PREFIX
$ 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 PCRE 8.42, Nettle 3.4.1 and GnuTLS 3.6.5.

```
$ PCRE_PREFIX="$HOME/static-libs/pcre-8.42"
$ 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:

```
$ svn export svn://svn.zabbix.com/trunk zabbix-source
$ cd zabbix-source/
$ ./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-libpcre="$PCRE_PREFIX
$ 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`
 - 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 be 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 library (mandatory library since Zabbix 4.0) from pcre.org, version 8.XX; not pcre2 (<ftp://ftp.csx.cam.ac.uk/pub/software/8.41.zip>)
2. Extract to directory E:\pcre-8.41
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:\pcre-8.41\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:\pcre-8.41 and E:\pcre-8.41\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:\pcre-8.41-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:\pcre-8.41-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:\pcre-8.41\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 SVN: `$ svn co svn://svn.zabbix.com/tags/4.2.0` `$ cd 4.2.0/` `$ ./bootstrap.sh` `$ ./configure --enable-agent --enable-ipv6 --prefix=`pwd`` `$ make dbschema` `$ make dist`
2. Copy and unpack the archive, e.g. `zabbix-4.2.0.tar.gz`, on a Windows machine.
3. Let's assume that sources are in `e:\zabbix-4.2.0`. Open a commandline window e.g. the x64 Native Tools Command Prompt for VS 2017 RC. Go to `E:\zabbix-4.2.0\build\win32\project`.
4. Compile `zabbix_get`, `zabbix_sender` and `zabbix_agent`.
 - without TLS: `E:\zabbix-4.2.0\build\win32\project> nmake /K PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-install\lib`
 - with TLS: `E:\zabbix-4.2.0\build\win32\project> nmake /K -f Makefile_get TLS=openssl TLSINCDIR=C:\OpenSSL-Win64-111-static\include TLSLIBDIR=C:\OpenSSL-Win64-111-static\lib PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-install\lib`
`E:\zabbix-4.2.0\build\win32\project> nmake /K -f Makefile_sender TLS=openssl TLSINCDIR="C:\OpenSSL-Win64-111-static\include" PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-install\lib`
`E:\zabbix-4.2.0\build\win32\project> nmake /K -f Makefile_agent TLS=openssl TLSINCDIR=C:\OpenSSL-Win64-111-static\include TLSLIBDIR=C:\OpenSSL-Win64-111-static\lib PCREINCDIR=E:\pcre-8.41-install\include PCRELIBDIR=E:\pcre-8.41-install\lib`
5. New binaries are located in `e:\zabbix-4.2.0\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).pdb /NODEFAULTLIB:LIBCMT
```

4 Installation from packages

From distribution packages

Several popular OS distributions have Zabbix packages provided. You can use these packages to install Zabbix.

Note:

OS distributions may lack the latest version of Zabbix in their repositories.

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 are available at repo.zabbix.com. Yum and apt repositories are also available on the server.

A step-by-step tutorial for installing Zabbix from packages is provided in sub-pages here.

1 Red Hat Enterprise Linux/CentOS

Overview

Official Zabbix packages are available for RHEL 7, CentOS 7 and Oracle Linux 7. In this documentation we will refer to all 3 using the term RHEL.

Some agent and proxy packages are available for [RHEL 6](#) and [RHEL 5](#) as well.

Adding Zabbix repository

Install the repository configuration package. This package contains yum (software package manager) configuration files.

RHEL 7:

```
# rpm -Uvh https://repo.zabbix.com/zabbix/4.2/rhel/7/x86_64/zabbix-release-4.2-2.el7.noarch.rpm
```

RHEL 6:

```
# rpm -Uvh https://repo.zabbix.com/zabbix/4.2/rhel/6/x86_64/zabbix-release-4.2-2.el6.noarch.rpm
```

RHEL 5:

```
# rpm -Uvh https://repo.zabbix.com/zabbix/4.2/rhel/5/x86_64/zabbix-release-4.2-2.el5.noarch.rpm
```

Frontend installation prerequisites

Zabbix frontend requires additional packages not available in basic installation. You need to enable repository of optional rpms in the system you will run Zabbix frontend on:

RHEL 7:

```
# yum-config-manager --enable rhel-7-server-optional-rpms
```

Server/proxy/frontend installation

To install Zabbix server (available for RHEL 7, **deprecated on RHEL 6**) with MySQL support:

```
# yum install zabbix-server-mysql
```

To install Zabbix proxy with MySQL support:

```
# yum install zabbix-proxy-mysql
```

To install Zabbix frontend (available for RHEL 7, **deprecated on RHEL 6**) with MySQL support:

```
# yum install zabbix-web-mysql
```

Substitute 'mysql' in the commands with 'pgsql' to use PostgreSQL, or with 'sqlite3' to use SQLite3 (proxy only).

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/doc/zabbix-server-mysql*/create.sql.gz | mysql -uzabbix -p zabbix
```

You will be prompted to enter your newly created database password.

With PostgreSQL:

```
# zcat /usr/share/doc/zabbix-server-pgsql*/create.sql.gz | sudo -u <username> psql zabbix
```

With TimescaleDB, in addition to the previous command, also run:

```
# zcat /usr/share/doc/zabbix-server-pgsql*/timescaledb.sql.gz | sudo -u <username> psql zabbix
```

Warning:

TimescaleDB is supported with Zabbix server only.

For **proxy**, import initial schema:

```
# zcat /usr/share/doc/zabbix-proxy-mysql*/schema.sql.gz | mysql -uzabbix -p zabbix
```

For proxy with PostgreSQL (or SQLite):

```
# zcat /usr/share/doc/zabbix-proxy-pgsql*/schema.sql.gz | sudo -u <username> psql zabbix
```

```
# zcat /usr/share/doc/zabbix-proxy-sqlite3*/schema.sql.gz | sqlite3 zabbix.db
```

Configure database for Zabbix server/proxy

Edit `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. See **SELinux configuration** below for instructions.

Starting Zabbix server process

It's time to start Zabbix server process:

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

and make it start at system boot:

RHEL 7 and later:

```
# systemctl enable zabbix-server
```

RHEL prior to 7:

```
# chkconfig --level 12345 zabbix-server on
```

Substitute 'zabbix-server' with 'zabbix-proxy' if you are installing Zabbix proxy.

Zabbix frontend configuration

For RHEL 7 and later the Apache configuration file for Zabbix frontend is located in `/etc/httpd/conf.d/zabbix.conf`.

If you use RHEL 6 please read the section about [using Zabbix frontend on RHEL 6](#) on how to configure the frontend.

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
```

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.

Note:

Zabbix official repository provides `fping`, `iksemel`, `libssh2` packages as well. These packages are located in the [non-supported](#) directory.

SELinux configuration

Having SELinux status enabled in enforcing mode, you need to execute the following commands to enable communication between Zabbix frontend and server:

RHEL 7 and later:

```
# setsebool -P httpd_can_connect_zabbix on
```

If the database is accessible over network (including 'localhost' in case of PostgreSQL), you need to allow

```
# 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
```

As frontend and SELinux configuration is done, you need to restart Apache web server:

```
# service httpd restart
```

Zabbix frontend and server on RHEL 6

Zabbix frontend on RHEL 6 is not supported because of PHP version. Since Zabbix 3.0 the requirements are to have PHP 5.4.0 or later while RHEL 6 latest version is 5.3.3 .

In most cases Zabbix server and frontend are installed on the same machine. When upgrading 2.2 to 3.0 Zabbix server will perform database upgrade and frontend will stop working. There is no way to roll back the database changes so users will be forced to upgrade PHP using 3rd party packages. This is why Zabbix server is also deprecated on RHEL 6.

If you still want to use Zabbix frontend on RHEL 6 and upgraded your PHP using 3rd party packages you would need to enable zabbix-deprecated repository first:

- open file /etc/yum.repos.d/zabbix.repo
- find section [zabbix-deprecated]
- set enabled=1
- save the file

You will have to do some more manual configuration. This is because we cannot identify the Apache version required for your PHP which makes it impossible for us to provide proper Apache configuration for Zabbix frontend. We have included 2 Apache configuration files to our zabbix-web package, one for Apache 2.2 and another for 2.4, which you would need to integrate with the Apache configuration yourself:

- httpd22-example.conf
- httpd24-example.conf

To get the full path to the files execute:

```
$ rpm -ql zabbix-web | grep example.conf
```

Agent installation

To install the agent, run

```
# yum install zabbix-agent
```

To start the agent, run:

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

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:

```
# yum install zabbix-java-gateway
```

Proceed to **setup** for more details on configuring and running Java gateway.

Installing debuginfo packages

Debuginfo packages are currently available for RHEL/CentOS versions 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 - $basearch
baseurl=http://repo.zabbix.com/zabbix/4.2/rhel/7/$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.

```
# yum install zabbix-debuginfo
```

This single packages contains debug information for all binary Zabbix componets.

2 Debian/Ubuntu/Raspbian

Overview

Official Zabbix packages are available for:

- Debian 10 (Buster)
- Debian 9 (Stretch)
- Debian 8 (Jessie)
- Ubuntu 18.04 (Bionic Beaver) LTS
- Ubuntu 16.04 (Xenial Xerus) LTS
- Ubuntu 14.04 (Trusty Tahr) LTS
- Raspbian (Buster)
- Raspbian (Stretch)

Adding Zabbix repository

Install the repository configuration package. This package contains apt (software package manager) configuration files.

For **Debian 10**, run the following commands:

Note! For Debian 9, substitute 'buster' with 'stretch' in the commands. For Debian 8, substitute 'buster' with 'jessie' in the commands.

```
# wget https://repo.zabbix.com/zabbix/4.2/debian/pool/main/z/zabbix-release/zabbix-release_4.2-2+buster_all.deb
# dpkg -i zabbix-release_4.2-2+buster_all.deb
# apt update
```

For **Ubuntu 18.04 (bionic)**, run the following commands:

```
# wget https://repo.zabbix.com/zabbix/4.2/ubuntu/pool/main/z/zabbix-release/zabbix-release_4.2-2+bionic_all.deb
# dpkg -i zabbix-release_4.2-2+bionic_all.deb
# apt update
```

- For Ubuntu 16.04, substitute 'bionic' with 'xenial' in the commands.
- For Ubuntu 14.04, substitute 'bionic' with 'trusty' in the commands.

For **Raspbian**, run the following commands:

```
# wget https://repo.zabbix.com/zabbix/4.2/raspbian/pool/main/z/zabbix-release/zabbix-release_4.2-2+buster_all.deb
# dpkg -i zabbix-release_4.2-2+buster_all.deb
# apt update
```

Server/proxy/frontend installation

To install Zabbix server with MySQL support:

```
# apt install zabbix-server-mysql
```

To install Zabbix proxy with MySQL support:

```
# apt install zabbix-proxy-mysql
```

To install Zabbix frontend:

```
# apt install zabbix-frontend-php
```

Substitute 'mysql' in the commands with 'pgsql' to use PostgreSQL, or with 'sqlite3' to use SQLite3 (proxy only).

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/doc/zabbix-server-mysql/create.sql.gz | mysql -uzabbix -p zabbix
```

You will be prompted to enter your newly created database password.

With PostgreSQL:

```
# zcat /usr/share/doc/zabbix-server-pgsql/create.sql.gz | sudo -u <username> psql zabbix
```

With TimescaleDB, in addition to the previous command, also run:

```
# zcat /usr/share/doc/zabbix-server-pgsql*/timescaledb.sql.gz | sudo -u <username> psql zabbix
```

Warning:

TimescaleDB is supported with Zabbix server only.

For **proxy**, import initial schema:

```
# zcat /usr/share/doc/zabbix-proxy-mysql/schema.sql.gz | mysql -uzabbix -p zabbix
```

For proxy with PostgreSQL (or SQLite):

```
# zcat /usr/share/doc/zabbix-proxy-pgsql/schema.sql.gz | sudo -u <username> psql zabbix
# zcat /usr/share/doc/zabbix-proxy-sqlite3/schema.sql.gz | sqlite3 zabbix.db
```

Configure database for Zabbix server/proxy

Edit `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. Refer to the [respective section](#) for RHEL/CentOS for instructions.

Starting Zabbix server process

It's time to start Zabbix server process and make it start at system boot:

```
# service zabbix-server start
# update-rc.d zabbix-server enable
```

Substitute 'zabbix-server' with 'zabbix-proxy' to start Zabbix proxy process.

SELinux configuration

Refer to the [respective section](#) for RHEL/CentOS.

As frontend and SELinux configuration is done, you need to restart Apache web server:

```
# service apache2 restart
```

Frontend configuration

Apache configuration file for Zabbix frontend is located in `/etc/apache2/conf-enabled/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
```

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.

Agent installation

To install the agent, run

```
# apt install zabbix-agent
```

To start the agent, run:

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

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:

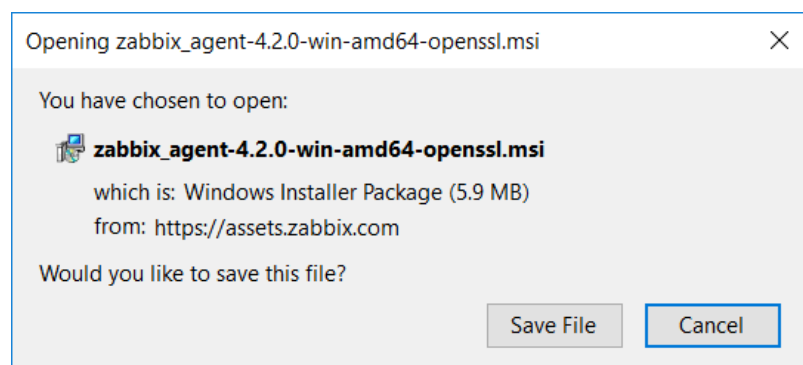
```
# apt install zabbix-java-gateway
```

Proceed to **setup** for more details on configuring and running Java gateway.

3 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](#):



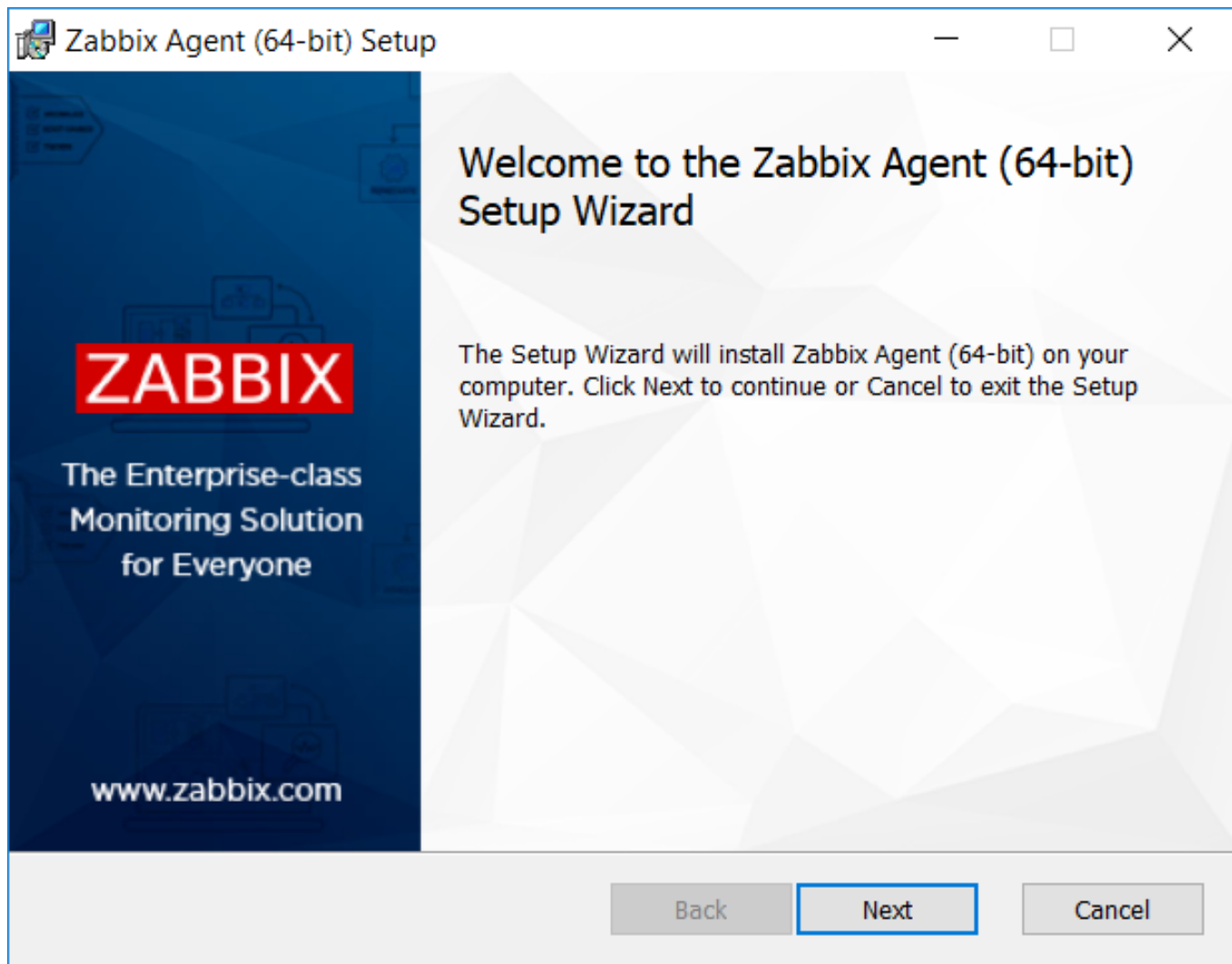
32-bit package cannot be installed on a 64-bit Windows.

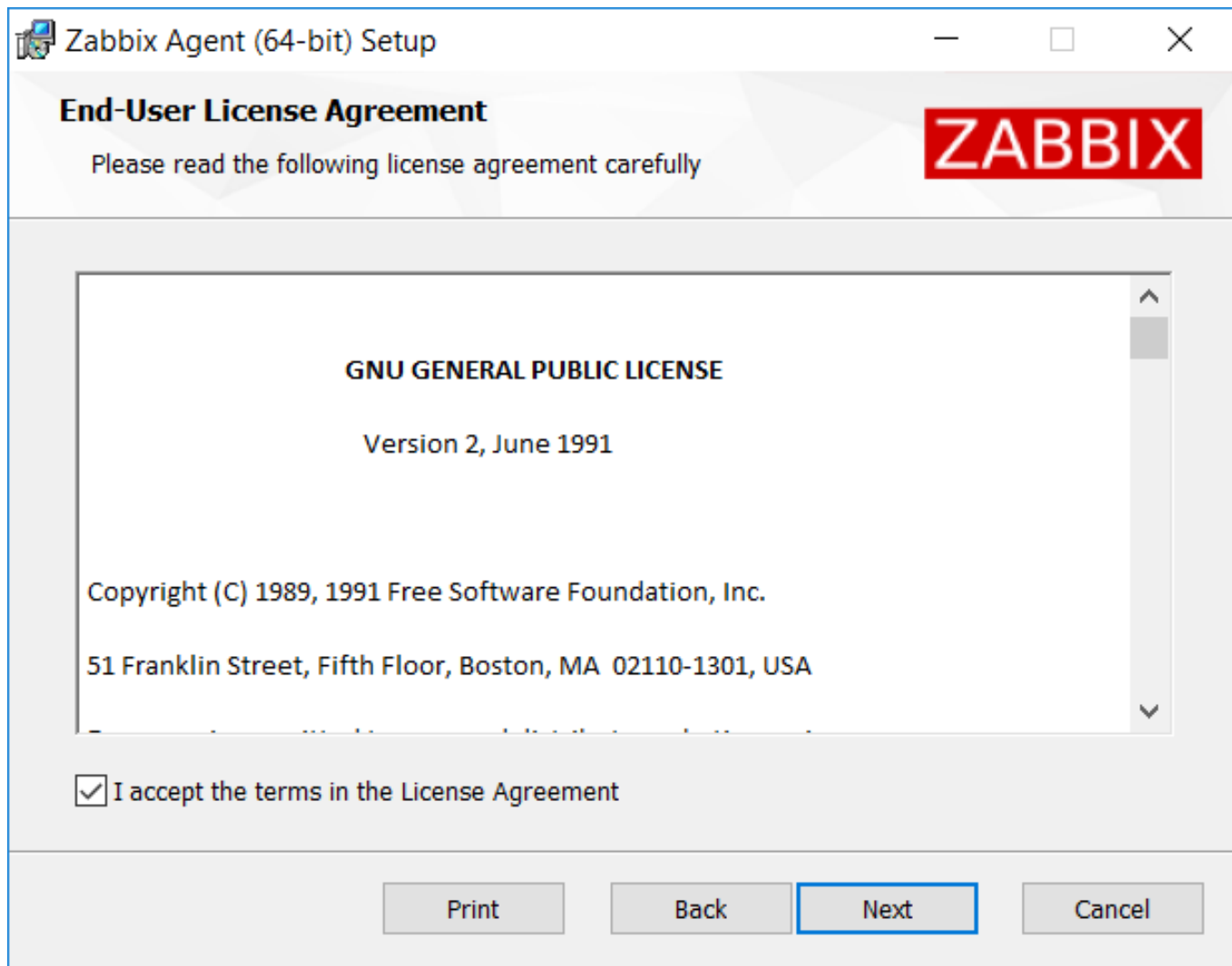
All packages come with TLS support, however, configuring TLS is optional.

Both UI and command-line based installation is supported.

Installation steps

To install, double-click the downloaded MSI file.





Accept the licence to proceed to the next step.


Zabbix Agent (64-bit) Setup
✕

Zabbix Agent service configuration

Please enter the information for configure Zabbix Agent



Host name:

LAPTOP-IKP7S50S

Zabbix server IP/DNS:

127.0.0.1

Agent listen port:

10050

Server or Proxy for active checks:

127.0.0.1

Remote command:

☒

Enable PSK:

☒

Add agent location to the PATH:

☒

Back

Next

Cancel

Specify the following parameters.

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

 Zabbix Agent (64-bit) PSK Setup ✕

Zabbix Agent pre-shared key configuration

Please enter the PSK information for configure Zabbix Agent

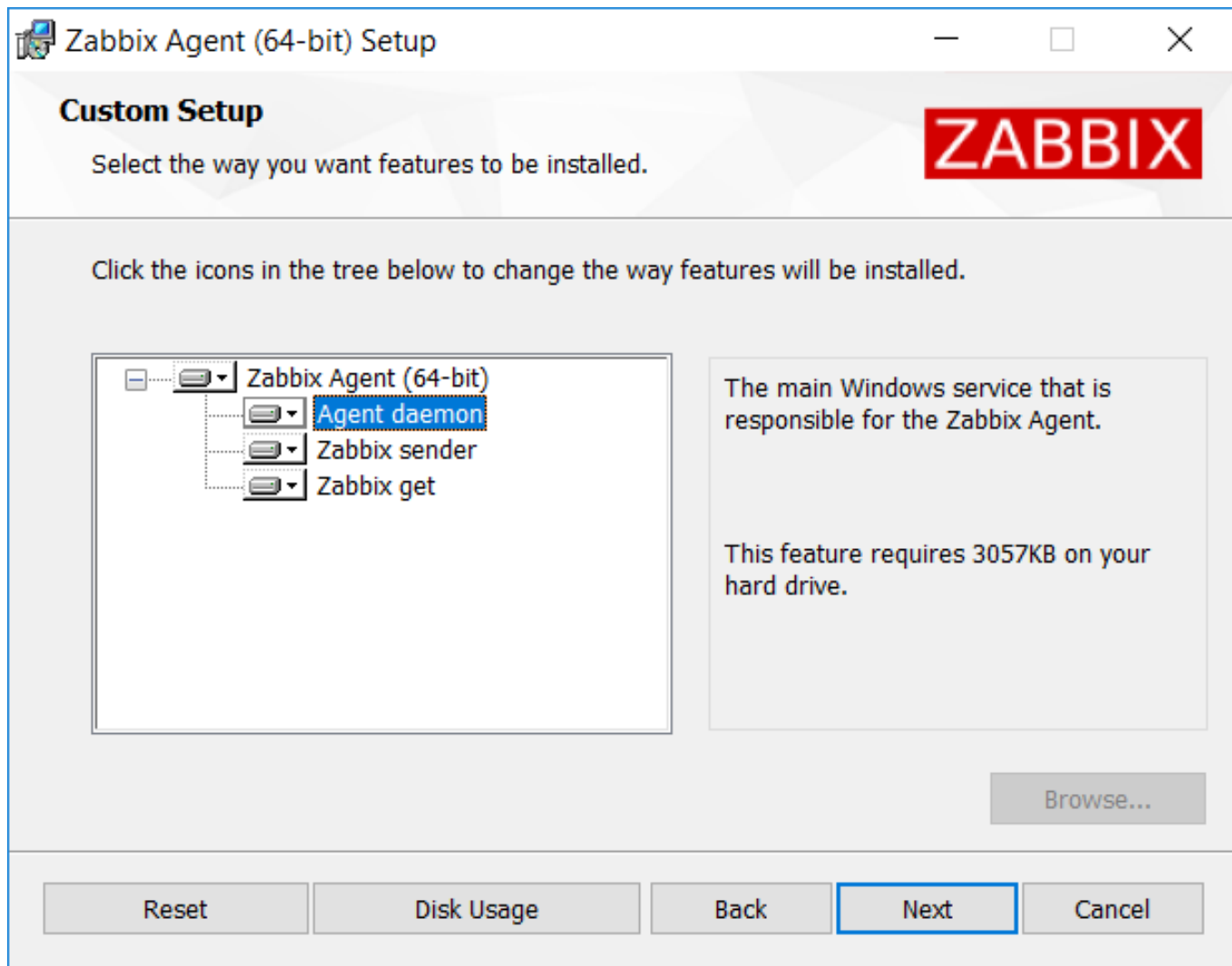


Pre-shared key identity:

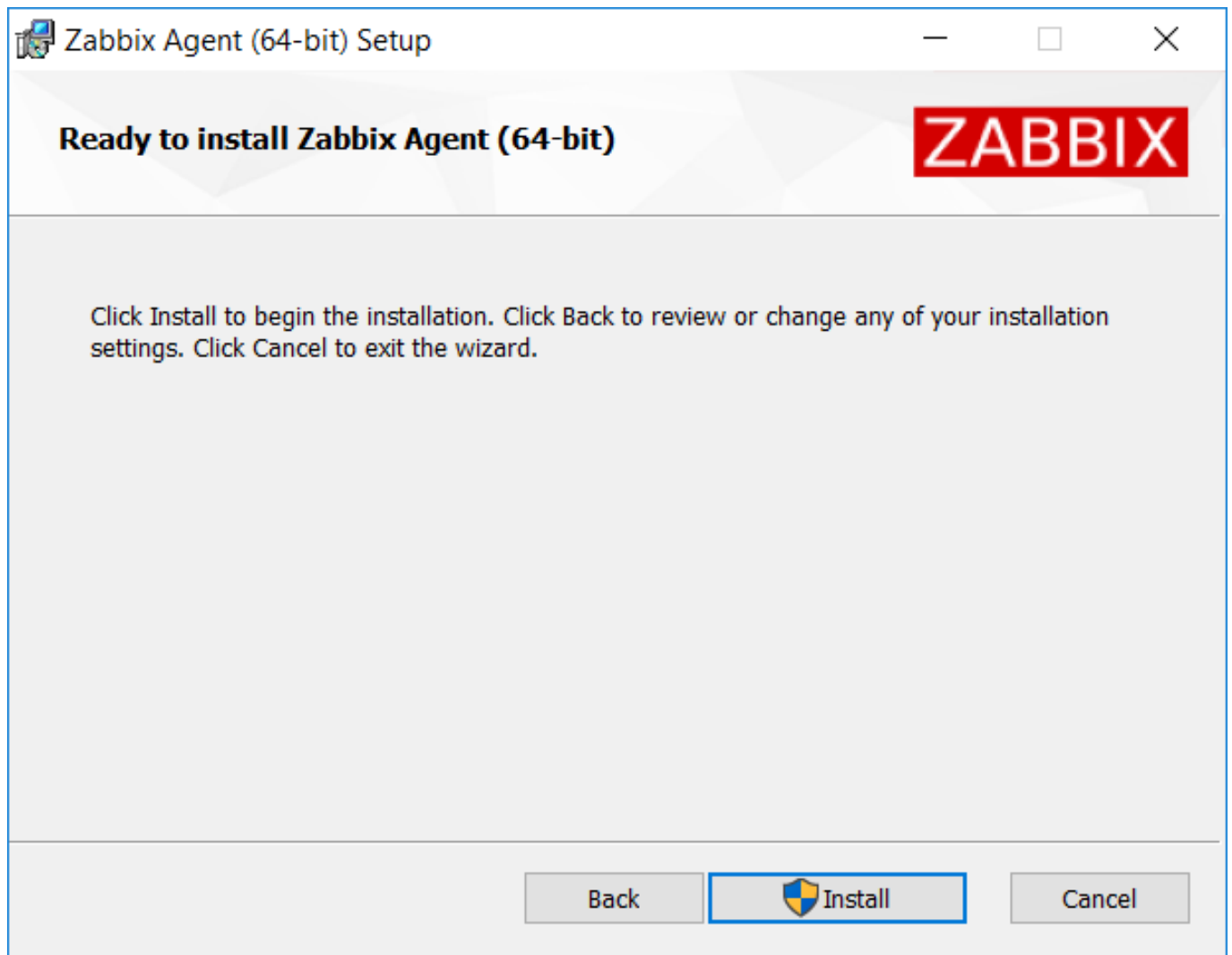
Pre-shared key value:

Please, set minimum required permission to access the psk.key file

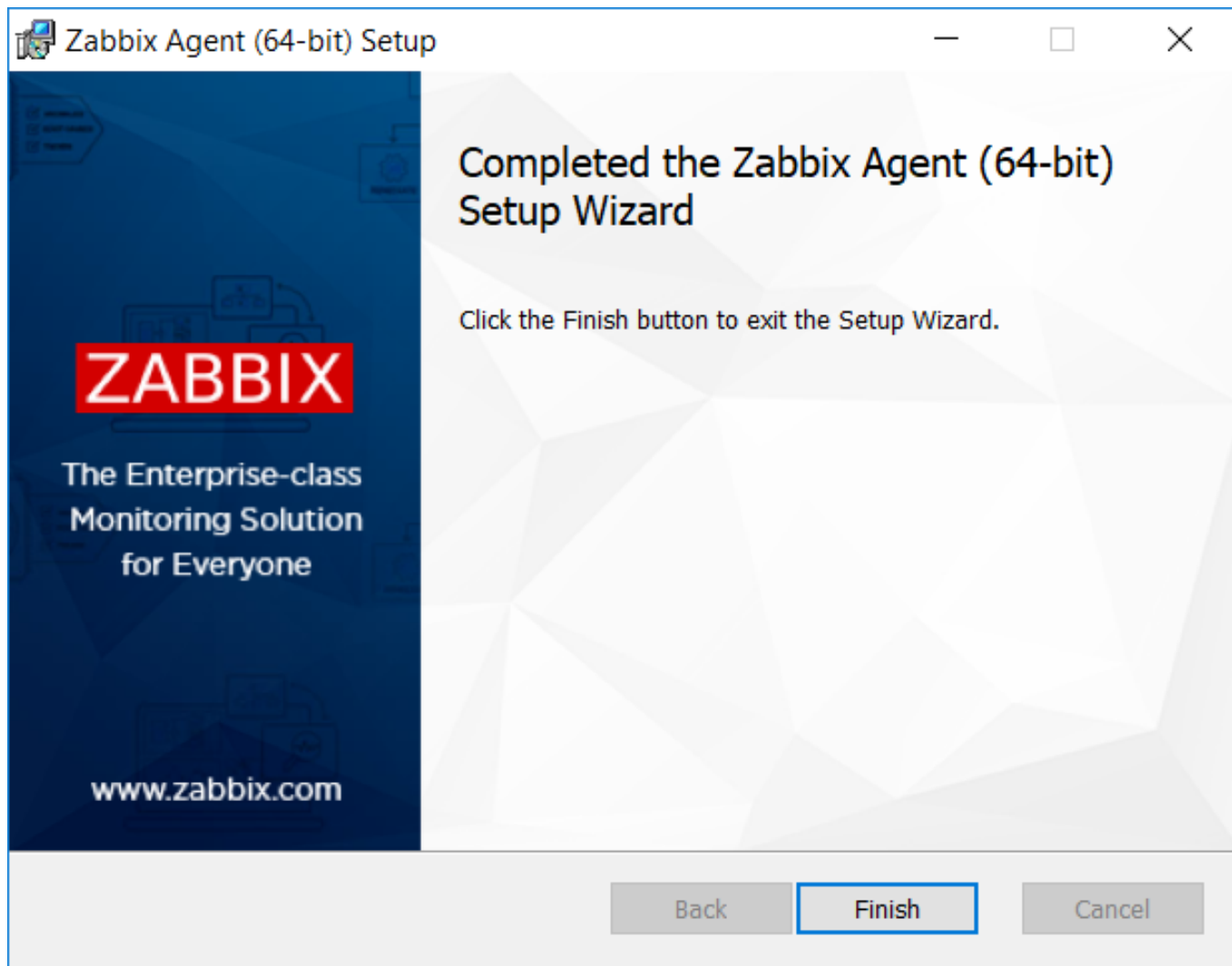
Enter pre-shared key identity and value. This step is only available if you checked *Enable PSK* in the previous step.



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	ENABLEREMOTECOMMANDS	
4	SERVER	
5	LISTENPORT	
6	SERVERACTIVE	
7	HOSTNAME	
8	TIMEOUT	
9	TLSCONNECT	
10	TLSACCEPT	
11	TLSPSKIDENTITY	
12	TLSPSKFILE	
13	TLSPSKVALUE	
14	TLSCAFILE	
15	TLSCRLFILE	
16	TLSSERVERCERTISSUER	
17	TLSSERVERCERTSUBJECT	
18	TLSCERTFILE	
19	TLSKEYFILE	
20	INSTALLFOLDER	
21	ENABLEPATH	
22	SKIP	SKIP=fw - do not install firewall exception rule

To install you may run, for example:

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

msiexec /l*v log.txt /i zabbix_agent-4.2.0-x86.msi /qn^
LOGTYPE=file^
LOGFILE="%INSTALLFOLDER%\za.log"^
ENABLEREMOTECOMMANDS=1^
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
```

or

```
msiexec /l*v log.txt /i zabbix_agent-4.2.0-x86.msi /qn^
SERVER=192.168.6.76^
TLSCONNECT=psk^
TLSACCEPT=psk^
TLSPSKIDENTITY=MyPSKID^
TLSPSKVALUE=1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c5473194952
```

4 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-4.2.1-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/logs/zabbix
```

Forget that Zabbix agent has been installed:

```
sudo pkgutil --forget com.zabbix.pkg.ZabbixAgent
```

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	3.4
ubuntu	trusty
centos	latest

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

Note:

Zabbix agent, Zabbix proxy, Zabbix server and Zabbix appliance images are based on Alpine Linux version 3.4, all other images are based on Alpine Linux 3.7.

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 appliance with MySQL database support and Nginx web-server - [zabbix/zabbix-appliance](#)
- 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-4.0-latest - latest minor version of a Zabbix 4.0 component based on Alpine Linux image
- ubuntu-4.0-latest - latest minor version of a Zabbix 4.0 component based on Ubuntu image
- alpine-4.0.* - different minor versions of a Zabbix 4.0 component based on Alpine Linux image, where * is the minor version of Zabbix component
- ubuntu-4.0.* - different minor versions of a Zabbix 4.0 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	Zabbix database name.
	Proxy	By default, value is 'zabbix' for Zabbix server and 'zabbix_proxy' for Zabbix proxy.
	Web interface	
POSTGRES_USER	Server	PostgreSQL database user.
POSTGRES_PASSWORD	Web interface	By default, value is 'zabbix'.
	Server	PostgreSQL database password.
POSTGRES_DB	Web interface	By default, value is 'zabbix'.
	Server	Zabbix database name.
PHP_TZ	Web interface	By default, value is 'zabbix' for Zabbix server and 'zabbix_proxy' for Zabbix proxy.
		Timezone in PHP format. Full list of supported timezones are available on php.net .
ZBX_SERVER_NAME	Web interface	By default, value is 'Europe/Riga'. Visible Zabbix installation name in right top corner of the web interface.
ZBX_JAVAGATEWAY_ENABLE	Server	By default, value is 'Zabbix Docker'
	Proxy	Enables communication with Zabbix Java gateway to collect Java related checks.
ZBX_ENABLE_SNMP_TRAPS	Server	By default, value is "false"
	Proxy	Enables SNMP trap feature. It requires zabbix-smnptraps instance and shared volume <code>/var/lib/zabbix/smnptraps</code> 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	
<code>/etc/zabbix/zabbix_agentd.d</code>	The volume allows to include *.conf files and extend Zabbix agent using the <code>UserParameter</code> feature
<code>/var/lib/zabbix/modules</code>	The volume allows to load additional modules and extend Zabbix agent using the <code>LoadModule</code> feature
<code>/var/lib/zabbix/enc</code>	The volume is used to store TLS-related files. These file names are specified using <code>ZBX_TLSCAFILE</code> , <code>ZBX_TLSCRLFILE</code> , <code>ZBX_TLSKEY_FILE</code> and <code>ZBX_TLSPSKFILE</code> environment variables
Zabbix server	
<code>/usr/lib/zabbix/alertscripts</code>	The volume is used for custom alert scripts. It is the <code>AlertScriptsPath</code> parameter in <code>zabbix_server.conf</code>
<code>/usr/lib/zabbix/externalscripts</code>	The volume is used by <code>external checks</code> . It is the <code>ExternalScripts</code> parameter in <code>zabbix_server.conf</code>
<code>/var/lib/zabbix/modules</code>	The volume allows to load additional modules and extend Zabbix server using the <code>LoadModule</code> feature
<code>/var/lib/zabbix/enc</code>	The volume is used to store TLS related files. These file names are specified using <code>ZBX_TLSCAFILE</code> , <code>ZBX_TLSCRLFILE</code> , <code>ZBX_TLSKEY_FILE</code> and <code>ZBX_TLSPSKFILE</code> environment variables
<code>/var/lib/zabbix/ssl/certs</code>	The volume is used as location of SSL client certificate files for client authentication. It is the <code>SSLCertLocation</code> parameter in <code>zabbix_server.conf</code>
<code>/var/lib/zabbix/ssl/keys</code>	The volume is used as location of SSL private key files for client authentication. It is the <code>SSLKeyLocation</code> parameter in <code>zabbix_server.conf</code>

<i>/var/lib/zabbix/ssl/ssl_ca</i>	The volume is used as location of certificate authority (CA) files for SSL server certificate verification. It is the <code>SSLCALocation</code> parameter in <code>zabbix_server.conf</code>
<i>/var/lib/zabbix/snmptraps</i>	The volume is used as location of <code>snmptraps.log</code> file. It could be shared by <code>zabbix-snmptraps</code> container and inherited using the <code>volumes_from</code> Docker option while creating a new instance of Zabbix server. SNMP trap processing feature could be enabled by using shared volume and switching the <code>ZBX_ENABLE_SNMP_TRAPS</code> environment variable to 'true'
<i>/var/lib/zabbix/mibs</i>	The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in <code>/var/lib/zabbix/mibs</code>
Zabbix proxy	
<i>/usr/lib/zabbix/externalscripts</i>	The volume is used by <code>external checks</code> . It is the <code>ExternalScripts</code> parameter in <code>zabbix_proxy.conf</code>
<i>/var/lib/zabbix/modules</i>	The volume allows to load additional modules and extend Zabbix server using the <code>LoadModule</code> feature
<i>/var/lib/zabbix/enc</i>	The volume is used to store TLS related files. These file names are specified using <code>ZBX_TLSCAFILE</code> , <code>ZBX_TLSCRLFILE</code> , <code>ZBX_TLSKEY_FILE</code> and <code>ZBX_TLSPSKFILE</code> environment variables
<i>/var/lib/zabbix/ssl/certs</i>	The volume is used as location of SSL client certificate files for client authentication. It is the <code>SSLCertLocation</code> parameter in <code>zabbix_proxy.conf</code>
<i>/var/lib/zabbix/ssl/keys</i>	The volume is used as location of SSL private key files for client authentication. It is the <code>SSLKeyLocation</code> parameter in <code>zabbix_proxy.conf</code>
<i>/var/lib/zabbix/ssl/ssl_ca</i>	The volume is used as location of certificate authority (CA) files for SSL server certificate verification. It is the <code>SSLCALocation</code> parameter in <code>zabbix_proxy.conf</code>
<i>/var/lib/zabbix/snmptraps</i>	The volume is used as location of <code>snmptraps.log</code> file. It could be shared by the <code>zabbix-snmptraps</code> container and inherited using the <code>volumes_from</code> Docker option while creating a new instance of Zabbix server. SNMP trap processing feature could be enabled by using shared volume and switching the <code>ZBX_ENABLE_SNMP_TRAPS</code> environment variable to 'true'
<i>/var/lib/zabbix/mibs</i>	The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in <code>/var/lib/zabbix/mibs</code>
Zabbix web interface based on Apache2 web server	
<i>/etc/ssl/apache2</i>	The volume allows to enable HTTPS for Zabbix web interface. The volume must contain the two <code>ssl.crt</code> and <code>ssl.key</code> files prepared for Apache2 SSL connections
Zabbix web interface based on Nginx web server	
<i>/etc/ssl/nginx</i>	The volume allows to enable HTTPS for Zabbix web interface. The volume must contain the two <code>ssl.crt</code> , <code>ssl.key</code> files and <code>dhparam.pem</code> prepared for Nginx SSL connections
Zabbix snmptraps	
<i>/var/lib/zabbix/snmptraps</i>	The volume contains the <code>snmptraps.log</code> log file named with received SNMP traps
<i>/var/lib/zabbix/mibs</i>	The volume allows to add new MIB files. It does not support subdirectories, all MIBs must be placed in <code>/var/lib/zabbix/mibs</code>

For additional information use Zabbix official repositories in Docker Hub.

Usage examples

** Example 1 **

The example demonstrates how to run Zabbix appliance with built-in MySQL database, Zabbix server, Zabbix web interface based on the Nginx web server and Zabbix Java gateway.

```
# docker run --name zabbix-appliance -t \
```

```
-p 10051:10051 \
-p 80:80 \
--restart unless-stopped \
-d zabbix/zabbix-appliance:latest
```

Note:

Zabbix appliance instance exposes 10051/TCP port (Zabbix trapper) and 80/TCP port (HTTP) to host machine.

**** Example 2 ****

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. 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" \
-d mysql:5.7 \
--restart unless-stopped \
--character-set-server=utf8 --collation-server=utf8_bin
```

2. Start Zabbix Java gateway instance

```
# docker run --name zabbix-java-gateway -t \
--restart unless-stopped \
-d zabbix/zabbix-java-gateway:latest
```

3. 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" \
--link mysql-server:mysql \
--link zabbix-java-gateway:zabbix-java-gateway \
-p 10051:10051 \
--restart unless-stopped \
-d zabbix/zabbix-server-mysql:latest
```

Note:

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

4. 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 DB_SERVER_HOST="mysql-server" \
-e MYSQL_DATABASE="zabbix" \
-e MYSQL_USER="zabbix" \
-e MYSQL_PASSWORD="zabbix_pwd" \
-e MYSQL_ROOT_PASSWORD="root_pwd" \
--link mysql-server:mysql \
--link zabbix-server-mysql:zabbix-server \
-p 80:80 \
--restart unless-stopped \
-d zabbix/zabbix-web-nginx-mysql:latest
```

Note:

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

**** Example 3 ****

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. Start empty PostgreSQL server instance

```
# docker run --name postgres-server -t \
  -e POSTGRES_USER="zabbix" \
  -e POSTGRES_PASSWORD="zabbix" \
  -e POSTGRES_DB="zabbix_pwd" \
  --restart unless-stopped \
  -d postgres:latest
```

2. Start Zabbix snmptraps instance

```
# docker run --name zabbix-snmptaps -t \
  -v /zbx_instance/snmptaps:/var/lib/zabbix/snmptaps:rw \
  -v /var/lib/zabbix/mibs:/usr/share/snmp/mibs:ro \
  -p 162:162/udp \
  --restart unless-stopped \
  -d zabbix/zabbix-snmptaps:latest
```

Note:

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

3. 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" \
  -e POSTGRES_DB="zabbix_pwd" \
  -e ZBX_ENABLE_SNMP_TRAPS="true" \
  --link postgres-server:postgres \
  -p 10051:10051 \
  --volumes-from zabbix-snmptaps \
  --restart unless-stopped \
  -d zabbix/zabbix-server-pgsql:latest
```

Note:

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

4. 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 DB_SERVER_HOST="postgres-server" \
  -e POSTGRES_USER="zabbix" \
  -e POSTGRES_PASSWORD="zabbix" \
  -e POSTGRES_DB="zabbix_pwd" \
  --link postgres-server:postgres \
  --link zabbix-server-pgsql:zabbix-server \
  -p 443:443 \
  -v /etc/ssl/nginx:/etc/ssl/nginx:ro \
  --restart unless-stopped \
  -d zabbix/zabbix-web-nginx-pgsql: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.

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
<code>docker-compose_v3_alpine_mysql_latest.yaml</code>	The compose file runs the latest version of Zabbix 4.0 components on Alpine Linux with MySQL database support.
<code>docker-compose_v3_alpine_mysql_local.yaml</code>	The compose file locally builds the latest version of Zabbix 4.0 and runs Zabbix components on Alpine Linux with MySQL database support.
<code>docker-compose_v3_alpine_pgsql_latest.yaml</code>	The compose file runs the latest version of Zabbix 4.0 components on Alpine Linux with PostgreSQL database support.
<code>docker-compose_v3_alpine_pgsql_local.yaml</code>	The compose file locally builds the latest version of Zabbix 4.0 and runs Zabbix components on Alpine Linux with PostgreSQL database support.
<code>docker-compose_v3_centos_mysql_latest.yaml</code>	The compose file runs the latest version of Zabbix 4.0 components on CentOS 7 with MySQL database support.
<code>docker-compose_v3_centos_mysql_local.yaml</code>	The compose file locally builds the latest version of Zabbix 4.0 and runs Zabbix components on CentOS 7 with MySQL database support.
<code>docker-compose_v3_centos_pgsql_latest.yaml</code>	The compose file runs the latest version of Zabbix 4.0 components on CentOS 7 with PostgreSQL database support.
<code>docker-compose_v3_centos_pgsql_local.yaml</code>	The compose file locally builds the latest version of Zabbix 4.0 and runs Zabbix components on CentOS 7 with PostgreSQL database support.
<code>docker-compose_v3_ubuntu_mysql_latest.yaml</code>	The compose file runs the latest version of Zabbix 4.0 components on Ubuntu 14.04 with MySQL database support.
<code>docker-compose_v3_ubuntu_mysql_local.yaml</code>	The compose file locally builds the latest version of Zabbix 4.0 and runs Zabbix components on Ubuntu 14.04 with MySQL database support.
<code>docker-compose_v3_ubuntu_pgsql_latest.yaml</code>	The compose file runs the latest version of Zabbix 4.0 components on Ubuntu 14.04 with PostgreSQL database support.
<code>docker-compose_v3_ubuntu_pgsql_local.yaml</code>	The compose file locally builds the latest version of Zabbix 4.0 and runs Zabbix components on Ubuntu 14.04 with PostgreSQL database support.

Attention:

Available Docker compose files support both versions 2 and 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

**** Example 1 ****

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

The command will download latest Zabbix 4.0 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 ****

```
# docker-compose -f ./docker-compose_v3_ubuntu_mysql_local.yaml up -d
```

The command will download base image Ubuntu 14.04, then build Zabbix 4.0 components locally and run them in detach mode.

6 Upgrade procedure

Overview

This section provides upgrade information for Zabbix 4.2:

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

Direct upgrade to Zabbix 4.2.x is possible from Zabbix 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.

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

1 Red Hat Enterprise Linux/CentOS

Overview

This section provides the steps required for a successful upgrade from Zabbix 4.0.x to Zabbix 4.2.x using official Zabbix packages for Red Hat Enterprise Linux/CentOS.

Attention:

Please note, that support for Jabber notifications was removed from official Zabbix server packages starting from version 4.2.0-1 due to the lack of support for and vulnerable libiksemel library. If you still need it, see this section for instructions.

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 4.2 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
4.0.x LTS	For: Zabbix 4.2	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

Upgrade from	Read full upgrade notes	Most important changes between versions
3.4.x	For: Zabbix 4.0 Zabbix 4.2	'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	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	Database upgrade may be slow, depending on the history table size

You may also want to check the [requirements](#) for 4.2.

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, yet unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled when upgrading to 4.2 (or later) from any version before 4.2, 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/doc/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/4.2/rhel/7/x86_64/zabbix-release-4.2-1.el7.noarch.rpm
```

5 Upgrade Zabbix components

To upgrade Zabbix components you may run something like:

```
# yum upgrade zabbix-server-mysql zabbix-web-mysql zabbix-agent
```

If using PostgreSQL, substitute mysql with postgresql in the command. If upgrading the proxy, substitute server with proxy in the command.

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
```

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 4.2.x (for example, from 4.2.1 to 4.2.3). Upgrading between minor versions is easy.

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

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

To execute Zabbix server minor version upgrade run:

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

To execute Zabbix agent minor version upgrade run:

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

Note that you may also use 'update' instead of 'upgrade' in these commands. While 'upgrade' will delete obsolete packages, 'update' will preserve them.

Adding Jabber notifications support to Zabbix server package

The recommended way of adding [Jabber notifications](#) support to Zabbix server is rebuilding from source package.

```
$ sudo yum -y install iksemel-devel gcc
$ rpm -ivh http://repo.zabbix.com/zabbix/4.2/rhel/7/SRPMS/zabbix-4.2.1-1.el7.src.rpm
$ cd ~/rpmbuild/SPECS
$ sudo yum-builddep -y zabbix.spec
$ sed -i.bak -r 's/^(build_flags=.*--enable-server)"$/\1 --with-jabber"/;s/^(Release\s*:\s*[~1-9]*[1-9] [0-9]
$ rpmbuild -ba zabbix.spec
[.]
Wrote: /root/rpmbuild/SRPMS/zabbix-4.2.1-1.1.el7.src.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/zabbix-agent-4.2.1-1.1.el7.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/zabbix-get-4.2.1-1.1.el7.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/zabbix-sender-4.2.1-1.1.el7.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/zabbix-proxy-mysql-4.2.1-1.1.el7.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/zabbix-proxy-postgresql-4.2.1-1.1.el7.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/zabbix-proxy-sqlite3-4.2.1-1.1.el7.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/zabbix-java-gateway-4.2.1-1.1.el7.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/zabbix-server-mysql-4.2.1-1.1.el7.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/zabbix-server-postgresql-4.2.1-1.1.el7.x86_64.rpm
Wrote: /root/rpmbuild/RPMS/noarch/zabbix-web-4.2.1-1.1.el7.noarch.rpm
Wrote: /root/rpmbuild/RPMS/noarch/zabbix-web-mysql-4.2.1-1.1.el7.noarch.rpm
Wrote: /root/rpmbuild/RPMS/noarch/zabbix-web-postgresql-4.2.1-1.1.el7.noarch.rpm
```



```

Wrote: /root/rpmbuild/RPMS/noarch/zabbix-web-japanese-4.2.1-1.1.el7.noarch.rpm
Wrote: /root/rpmbuild/RPMS/x86_64/zabbix-debuginfo-4.2.1-1.1.el7.x86_64.rpm
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.XFsa0b
+ umask 022
+ cd /root/rpmbuild/BUILD
+ cd zabbix-4.2.1
+ rm -rf /root/rpmbuild/BUILDROOT/zabbix-4.2.1-1.1.el7.x86_64
+ exit 0

```

2 Debian/Ubuntu

Overview

This section provides the steps required for a successful **upgrade** from Zabbix **4.0.x** to Zabbix **4.2.x** using official Zabbix packages for Debian/Ubuntu.

Attention:

Please note, that support for **Jabber notifications** was **removed from official Zabbix server packages** starting from version 4.2.0-1 due to the lack of support for and vulnerable libiksemel library. If you still need it, see [this section](#) for instructions.

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.

Attention:

It is no longer possible to start the upgraded server and have older, yet unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled when upgrading to 4.2 (or later) from any version before 4.2, 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 4.2 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	Important notes/changes between versions
4.0.x	For: Zabbix For 4.2	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	'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	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

Upgrade from	Read full upgrade notes	Important notes/changes between versions
3.0.x LTS	For: Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2	Database upgrade may be slow, depending on the history table size

You may also want to check the [requirements](#) for 4.2.

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
```

Attention:

It is no longer possible to start the upgraded server and have older, yet unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled when upgrading to 4.2 (or later) from any version before 4.2, 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/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/doc/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 9** run:

```
# wget https://repo.zabbix.com/zabbix/4.2/debian/pool/main/z/zabbix-release/zabbix-release_4.2-1+stretch_all.deb
# dpkg -i zabbix-release_4.2-1+stretch_all.deb
```

On **Debian 8** run:

```
# wget https://repo.zabbix.com/zabbix/4.2/debian/pool/main/z/zabbix-release/zabbix-release_4.2-1+jessie_all.deb
# dpkg -i zabbix-release_4.2-1+jessie_all.deb
```

On **Ubuntu 18.04** run:

```
# wget https://repo.zabbix.com/zabbix/4.2/ubuntu/pool/main/z/zabbix-release/zabbix-release_4.2-1+bionic_all.deb
# dpkg -i zabbix-release_4.2-1+bionic_all.deb
```

On **Ubuntu 16.04** run:

```
# wget https://repo.zabbix.com/zabbix/4.2/ubuntu/pool/main/z/zabbix-release/zabbix-release_4.2-1+xenial_all.deb
# dpkg -i zabbix-release_4.2-1+xenial_all.deb
```

On **Ubuntu 14.04** run:

```
# wget https://repo.zabbix.com/zabbix/4.2/ubuntu/pool/main/z/zabbix-release/zabbix-release_4.2-1+trusty_all.deb
# dpkg -i zabbix-release_4.2-1+trusty_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.

6 Review component configuration parameters

See the upgrade notes for details on [mandatory changes](#).

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
```

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 4.2.x (for example, from 4.2.1 to 4.2.3). 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.*'
```

Adding Jabber notifications support to Zabbix server package

The recommended way of adding [Jabber notifications](#) support to Zabbix server is rebuilding from source package.

```
$ sudo apt -y install dpkg-dev devscripts libiksemel-dev
$ sudo apt-get -y build-dep zabbix
$ mkdir /tmp/zabbix-server-build
$ cd !$
$ apt-get source zabbix
$ cd zabbix-*
$ sed -i.bak -r 's/^(COMMON_CONFIG\s*=\.*)\\/\1--with-jabber \\/' debian/rules
$ debchange --rebuild "rebuild with Jabber notifications support"
$ dpkg-buildpackage
[...]
```

```
dpkg-deb: building package 'zabbix-java-gateway' in '../zabbix-java-gateway_4.2.1-1+stretch_all.deb'.
dpkg-deb: building package 'zabbix-proxy-pgsql' in '../zabbix-proxy-pgsql_4.2.1-1+stretch_amd64.deb'.
dpkg-deb: building package 'zabbix-proxy-mysql' in '../zabbix-proxy-mysql_4.2.1-1+stretch_amd64.deb'.
```

```
dpkg-deb: building package 'zabbix-proxy-sqlite3' in '../zabbix-proxy-sqlite3_4.2.1-1+stretch_i_amd64.deb'.
dpkg-deb: building package 'zabbix-server-mysql' in '../zabbix-server-mysql_4.2.1-1+stretch_i_amd64.deb'.
dpkg-deb: building package 'zabbix-server-pgsql' in '../zabbix-server-pgsql_4.2.1-1+stretch_i_amd64.deb'.
dpkg-genbuildinfo
dpkg-genchanges >../zabbix_4.2.1-1+stretch_i_amd64.changes
dpkg-genchanges: info: not including original source code in upload
dpkg-source --after-build zabbix-4.2.1
dpkg-source: info: using options from zabbix-4.2.1/debian/source/options: --extend-diff-ignore=*.bat|*.exe
dpkg-buildpackage: info: binary and diff upload (original source NOT included)
$ sudo dpkg -i ../zabbix-server-mysql_*.deb
(Reading database ... 44692 files and directories currently installed.)
Preparing to unpack ../zabbix-server-mysql_4.2.1-1+stretch_i_amd64.deb ...
Unpacking zabbix-server-mysql (1:4.2.1-1+stretch) over (1:4.2.1-1+stretch) ...
Setting up zabbix-server-mysql (1:4.2.1-1+stretch) ...
```

Upgrade from sources

Overview

This section provides the steps required for a successful **upgrade** from Zabbix **4.0.x** to Zabbix **4.2.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, yet unupgraded proxies report data to a newer server. This approach, which was never recommended nor supported by Zabbix, now is officially disabled when upgrading to 4.2 (or later) from any version before 4.2, 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 4.2 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	Important notes/changes between versions
4.0.x LTS	For: Zabbix 4.2	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	'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	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

Upgrade from	Read full upgrade notes	Important notes/changes between versions
3.0.x LTS	For: Zabbix 3.2 Zabbix 3.4 Zabbix 4.0 Zabbix 4.2	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	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	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	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 4.2.

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 5.4.0. 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](#). For new optional parameters, see the [What's new](#) section.

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.

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 to agent [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 4.2.x (for example from 4.2.1 to 4.2.3) 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.

7 Known issues

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.

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.

ODBC checks

Zabbix server or proxy that uses MySQL as its database may or may not work correctly with MySQL ODBC library due to an [upstream bug](#). Please 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.

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](#).

HTTPS checks

Web scenarios and HTTP agent items using the https protocol, Zabbix agent checks `net.tcp.service[https...]` and `net.tcp.service.perf[https...]` may fail if the target server is configured to disallow TLS v1.0 protocol or below. Please see [ZBX-9879](#) for more information and available workarounds.

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.

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.

Flipping frontend locales

It has been observed that frontend locales may flip without apparent logic. A known workaround to this is to disable multithreading in PHP and Apache. Please see [ZBX-10911](#) for more information.

Compatibility issue with PHP 7.0

It has been observed that with PHP 7.0 importing a template with web monitoring triggers may fail due to incorrectly added double quotes to the web monitoring items in the trigger expressions. The issue goes away when upgrading PHP to 7.1.

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

Changes to Daylight Saving Time (DST) result in irregularities when displaying X axis labels (date duplication, date missing, etc).

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

The **output** parameter does not work properly with the `history.get` method (fixed in 4.2.5).

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

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.

IE11 issue with map resizing in dashboard widgets

Maps, in Internet Explorer 11, are cut off on the right side if the map content is larger than the dashboard widget area (instead of being resized proportionately). This is intentional because of an IE11-related issue with proper resizing of SVG images.

Known issues in 4.2.0, 4.2.1

- IPMI pollers are unable to process more than 60 items per minute. Fixed in 4.2.2.
- You may encounter the issue of 100% CPU usage by IPMI pollers if using OpenIPMI version 2.0.26 and above. Fixed in 4.2.2.

Known issues in 4.2.2

- This version requires *write* permissions to the `assets` directory for the web server user. This requirement, however, would break those installations of the frontend where the web server has no write permissions to the document root (see [ZBX-16180](#) for details). As a result, this requirement is rolled back in 4.2.3.

8 Template changes

This page lists all changes to the stock templates that are shipped with Zabbix. It is suggested to modify these templates in existing installations - depending on the changes, it can be done either by:

- importing the latest official version (while in *Configuration* → *Templates* you can import the updated template manually from the `templates` directory of the downloaded latest Zabbix version);
- performing the change manually.

Changes in 4.2.0 Zabbix server template

- New low-level discovery manager and worker processes have been added
- The new `zabbix[lld_queue]` item has been added to monitor low-level discovery queue
- More descriptions for triggers related to caching parameters have been added (e.g. "Consider increasing `HistoryCacheSize` in the `zabbix_server.conf` configuration file" for the *More than 75% used in the history cache* trigger, etc.).

Zabbix proxy template

Description of the *More than 75% used in the configuration cache* trigger has been fixed to refer to the `zabbix_proxy.conf` file (instead of `zabbix_server.conf`).

More descriptions for triggers related to caching parameters have been added (e.g. "Consider increasing `HistoryCacheSize` in the `zabbix_proxy.conf` configuration file" for *More than 75% used in the history cache* trigger, etc.).

Zabbix agent template

Minor naming inconsistency in items and triggers has been fixed in *Template App Zabbix Agent* (e.g. *Version of zabbix_agent(d) running* → *Version of Zabbix agent running*)

Changes in 4.2.1 SNMP interface templates

Update intervals have been changed from 5m to 3m for these LLD item prototypes:

- Interface {#IFNAME}({#IFALIAS}): Inbound packets with errors
- Interface {#IFNAME}({#IFALIAS}): Outbound packets with errors

in:

- *Template Module Interfaces Simple SNMPv1*
- *Template Module Interfaces Simple SNMPv2*
- *Template Module Interfaces SNMPv1*
- *Template Module Interfaces SNMPv2*
- *Template Module Interfaces Windows SNMPv2*

These update intervals have been changed to avoid creation of excess internal events caused by trigger state flapping.

Changes in 4.2.7 Juniper template

The more correct OID of 'jnxOperatingCPU' (1.3.6.1.4.1.2636.3.1.13.1.8) is now used instead of 'jnxOperating5MinLoadAvg' (1.3.6.1.4.1.2636.3.1.13.1.21) in these LLD item and trigger prototypes:

- {#SNMPVALUE}: CPU utilization
- {#SNMPVALUE}: High CPU utilization

in:

- *Template Net Juniper SNMPv2*

Changes in 4.2.8 New Cisco UCS server template

A Cisco UCS server monitoring template - *Template Server Cisco UCS SNMPv2* has been added.

It will be available in *Configuration* → *Templates* in new installations. If you are upgrading from previous versions, you can find this template in the `templates` directory of the downloaded latest Zabbix version. Then, while in *Configuration* → *Templates* you can import it manually into Zabbix.

HP iLO template

The counter for overall server health check has been fixed in the *Template Server HP iLO SNMPv2* template.

HP Comware template

The *Template Net HP Comware HH3C SNMPv2* now contains the correct value mapping.

9 Upgrade notes for 4.2.0

These notes are for upgrading from Zabbix 4.0.x to Zabbix 4.2.0. 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 4.2.0 from versions before Zabbix 4.0.0. See the [upgrade procedure](#) section for all relevant information about upgrading from previous Zabbix versions.

Critical Preprocessing support on Zabbix proxy

For hosts monitored by proxies, all item (including low-level discovery rules, dependent items) value preprocessing will now be done on the proxy.

Given new preprocessing options such as Javascript, extensive validation and throttling options, preprocessing may become a server bottleneck thus preprocessing by proxies offers the necessary scalability. Note that since this is not a configurable setting the affected proxies will require more resources after the upgrade, especially if heavier preprocessing is involved.

Warning:

Because of this change, keeping proxies running during the server upgrade process will bring no 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.

It is also no longer possible to use the officially unsupported possibility to start the upgraded server and have older, yet unupgraded proxies report data to a newer server. This approach now is officially disabled when upgrading to 4.2 (or later) from any version before 4.2, as the server will ignore data from unupgraded proxies.

Low-level discovery rules

In order to support discovery item value preprocessing and custom paths to low-level discovery (LLD) macro values in a JSON document, 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. Instead, low-level discovery will now accept a normal JSON containing an array.

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.

Warning:

As a result of the changes above, newer agents no longer will be able to work with an older Zabbix server.

Requirement changes

PHP GD library minimum version has been **upped** from 2.0 to 2.0.28 to support animated GIFs in maps.

Informational Low-level discovery processing

Processing low-level discovery has been split from data gathering processes into its own processing. Due to this, depending on how heavy low-level discovery is in your environment, you may:

- Need to increase the default number of the new LLD processors (StartLLDProcessors server **parameter**, 2 by default)
- Consider decreasing the number of pollers, trappers and proxypollers (if they were heavily loaded with LLD processing before)

The new `zabbix[lld_queue]` internal **item** can be used to monitor if the default number of StartLLDProcessors is enough.

Deprecated API features removed

Several API features that have been deprecated since Zabbix 3.4, are now completely removed:

- `httptest.create`, `httptest.update` methods: support of string format for 'headers' and 'variables' fields in both web scenario and web scenario step objects has been dropped;
- `map.create`, `map.update` methods: support of 'elementid' in map element object has been dropped.

Changed URL for global search

As the global search page has been moved to MVC, its URL has changed from `https://localhost/zabbix/search.php?search=Zabbix` to `https://localhost/zabbix/zabbix.php?action=search&search=Zabbix`.

Host sorting in host group map element

Host sorting if the 'Host group' map option is selected, is now done non-casesensitively.

Zabbix sender messages

Zabbix sender response messages have been changed from:

```
info from server: "processed: 100; failed: 0; total: 100; seconds spent: 0.013683"
sent:100; skipped: 0; total: 100
```

to:

```
Response from "127.0.0.1:10051": "processed: 100; failed: 0; total: 100; seconds spent: 0.000247"
sent: 100; skipped: 0; total: 100
```

or, when sent to multiple servers:

```
Response from "127.0.0.1:10051": "processed: 100; failed: 0; total: 100; seconds spent: 0.000247"
Response from "192.168.0.1:10052": "processed: 100; failed: 0; total: 100; seconds spent: 0.000391"
Response from "localhost:10053": "processed: 100; failed: 0; total: 100; seconds spent: 0.000086"
sent: 100; skipped: 0; total: 100
```

See also

- [Template changes](#)

10 Upgrade notes for 4.2.1

Items

- Trailing whitespace is no longer allowed after the `maxdelay` parameter in `log[]`, `log.count[]`, `logrt[]`, `logrt.count[]` [items](#).

11 Upgrade notes for 4.2.2

Frontend

- In this version only, *write* permissions to the `assets` directory are required for the web server user. (See also [known issues](#).)
- The `styles` folder has been moved to the `assets` directory, which is important if creating a [custom theme](#).
- Media types in the *Action log* dashboard [widget](#)/[screen element](#) are now sorted by media type ID instead of media type name.

12 Upgrade notes for 4.2.3

This minor version does not have any upgrade notes.

13 Upgrade notes for 4.2.4

This minor version does not have any upgrade notes.

14 Upgrade notes for 4.2.5

This minor version does not have any upgrade notes.

15 Upgrade notes for 4.2.6

Global notifications

Default audio clips are now in MP3 (it used to be wave files). An embedded media player is not used any more (it was used for Internet Explorer only), so now, in case of custom audio files, you must be sure that your browser can play them (or just use the MP3 format).

Leading and trailing spaces disallowed in host names

Leading and trailing spaces are no longer allowed in host and template names.

16 Upgrade notes for 4.2.7

This minor version does not have any upgrade notes.

17 Upgrade notes for 4.2.8

This minor version does not have any upgrade notes.

18 Upgrade notes for 4.2.9

This minor version does not have any upgrade notes.

5. Quickstart

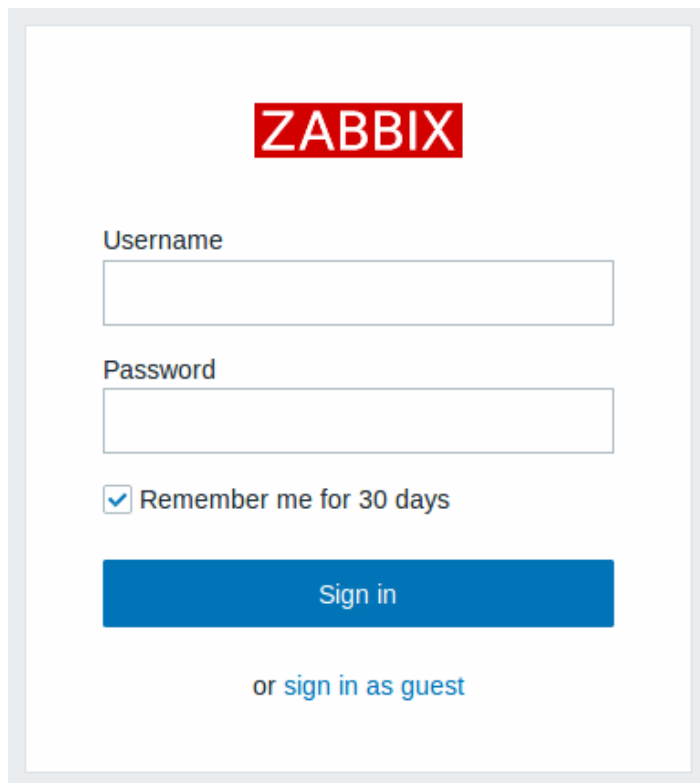
Please use the sidebar to access content in the Quickstart section.

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

The image shows the Zabbix login interface. At the top, the word "ZABBIX" is displayed in white capital letters on a red rectangular background. Below this, the label "Username" is positioned above a white text input field. Underneath the username field, the label "Password" is positioned above another white text input field. Below the password field, there is a checked checkbox followed by the text "Remember me for 30 days". At the bottom of the form area, there is a blue rectangular button with the text "Sign in" in white. Below the button, the text "or sign in as guest" is displayed in a smaller, blue font.

This is the Zabbix "Welcome" screen. Enter the user name **Admin** with password **zabbix** to log in as a **Zabbix superuser**.

When logged in, you will see 'Connected as Admin' in the lower right corner of the page. 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*.

Users

User group

All

Create user

☐	ALIAS ▲	NAME	SURNAME	USER TYPE	GROUPS	IS ONLINE?	LOGIN	FRONTEND ACCESS	DEBUG MODE	STATUS
☐	Admin	Zabbix	Administrator	Zabbix Super Admin	Zabbix administrators	Yes (2015-08-05 17:25:44)	Ok	System default	Disabled	Enabled
☐	guest			Zabbix User	Guests	Yes (2015-08-05 17:16:38)	Ok	System default	Disabled	Enabled

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'.

Users

User Media Permissions

* Alias

Name

Surname

* Groups

* Password

* Password (once again)

Language

Theme

Auto-login ☐

Auto-logout (min 90 seconds) ☐

* Refresh (in seconds)

* Rows per page

URL (after login)

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*.

Media



Type Email

* Send to

[Remove](#)

Add

* When active

Use if severity

☒ Not classified

☒ Information

☒ Warning

☒ Average

☒ High

☒ Disaster

Enabled ☒

Add

Cancel

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 description of the format), by default a medium is always active. You can also customise [trigger severity](#) levels for which the medium will be active, but leave all of them enabled for now.

Click on *Add*, then click *Add* in the user properties form. The new user appears in the userlist.

Users

User groupAll

Create user

☐	ALIAS ▲	NAME	SURNAME	USER TYPE	GROUPS	IS ONLINE?	LOGIN	FRONTEND ACCESS	DEBUG MODE	STATUS
☐	Admin	Zabbix	Administrator	Zabbix Super Admin	Zabbix administrators	Yes (2015-11-05 07:26:26)	Ok	System default	Disabled	Enabled
☐	guest			Zabbix User	Guests	Yes (2015-11-05 07:25:22)	Ok	System default	Disabled	Enabled
☐	user	New	User	Zabbix User	Zabbix administrators	No	Ok	System default	Disabled	Enabled

Adding permissions

By default, a new user has no permissions to access hosts. 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 *Permissions* tab.

User groups

User group

Permissions

Permissions

Host group

*

Permissions

None

Select

Read-write

Read

Deny

None

Add

This user is to have read-only access to *Linux servers* group, so click on *Select* next to the user group selection field.

Host groups

☐ Name

☐ Discovered hosts

☐ Hypervisors

☒ Linux servers

☐ Templates

☐ Templates/Applications

☐ Virtual machines

☐ Zabbix servers

Select

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 permission level and then *Add* to add the group to the list of permissions. In the user group properties form, click *Update*.

Attention:

In Zabbix, access rights to hosts 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*. 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.

Hosts

[Host](#)
[Templates](#)
[IPMI](#)
[Macros](#)
[Host inventory](#)
[Encryption](#)

* Host name

Visible name

* Groups

Linux servers ✕ Zabbix servers ✕

type here to search

Select

* At least one interface must exist.

Agent interfaces

IP address	DNS name	Connect to	Port
127.0.0.1		IP DNS	10050

Add

SNMP interfaces

Add

JMX interfaces

Add

IPMI interfaces

Add

Description

Monitored by proxy

(no proxy)

Enabled

☒

Add

Cancel

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.

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.

IP address

- 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 hostlist.

Note:

If the ZBX icon in the *Availability* column is red, there is some error with communication - move your mouse cursor over it to see the error message. If that icon is gray, no status update has happened so far. 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 data to get off of 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.

The *Items* link in the row of 'New host' should display a count of '0'. Click on the link, and then click on *Create item*. This will present us with an item definition form.

The screenshot shows the 'Create item' form in Zabbix. The form is divided into two tabs: 'Item' and 'Preprocessing'. The 'Item' tab is active. The form contains the following fields and sections:

- Name:** CPU Load (mandatory, marked with a red asterisk).
- Type:** Zabbix agent (dropdown).
- Key:** system.cpu.load (mandatory, marked with a red asterisk). There is a 'Select' button next to it.
- Host interface:** 127.0.0.1 : 10050 (mandatory, marked with a red asterisk).
- Type of information:** Numeric (float) (dropdown).
- Units:** (empty text field).
- Update interval:** 30s (mandatory, marked with a red asterisk).
- Custom intervals:** A table with columns: Type, Interval, Period, Action.
 - Row 1: Flexible, Scheduling, 50s, 1-7,00:00-24:00, Remove.There is an 'Add' link below the table.
- History storage period:** 90d (mandatory, marked with a red asterisk).
- Trend storage period:** 365d (mandatory, marked with a red asterisk).
- Show value:** As is (dropdown). There is a 'show value mappings' link next to it.
- New application:** (empty text field).
- Applications:** A list box showing '-None-'.
- Populates host inventory field:** -None- (dropdown).
- Description:** (empty text area).
- Enabled:** A checkbox that is checked.
- Buttons:** 'Add' and 'Cancel'.

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 a 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

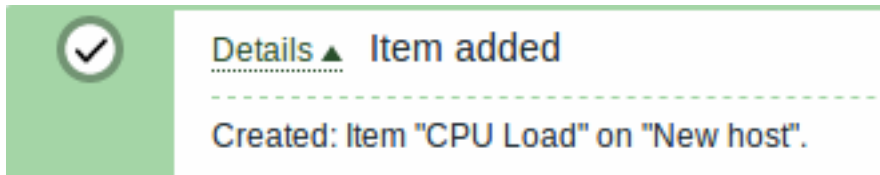
- Select *Numeric (float)* here. This attribute defines the format of expected data.

Note:

You may also want to reduce the amount 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 itemlist. 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*.

Then click on the **+** before **- other -** and expect your item to be there and displaying data.

▼ ☐ HOST	NAME ▲	LAST CHECK	LAST VALUE	CHANGE
▼ New host	- other - (1 Item)			
☐	CPU Load	2015-08-08 16:00:49	2.24	-0.26 Graph

With that said, first data may take up to 60 seconds 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.

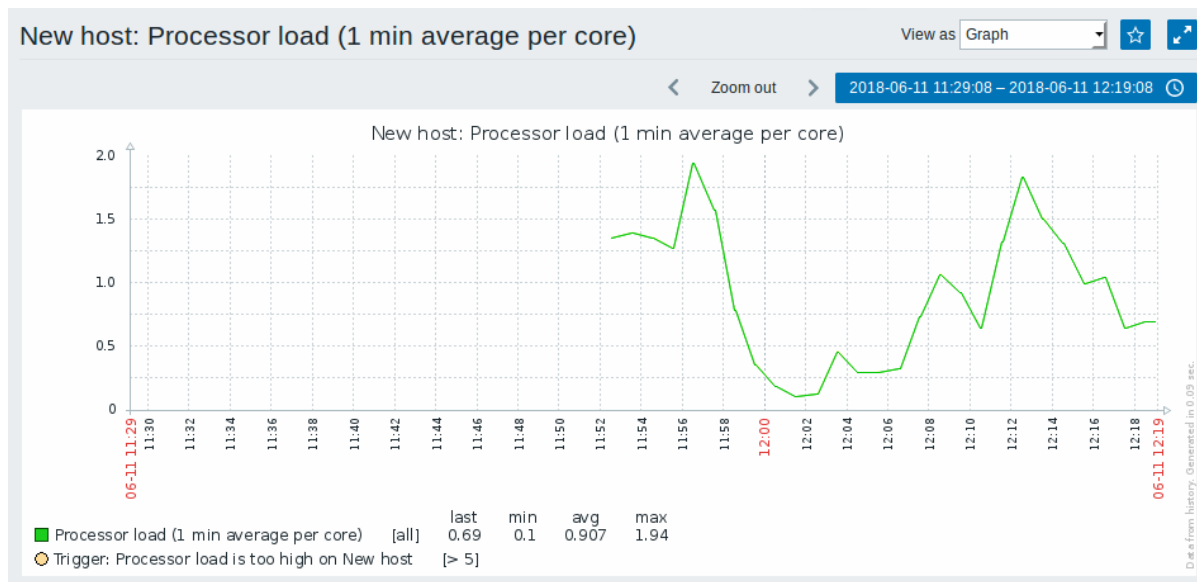
If you do not see information about the item as in the screenshot, make sure that:

- you entered item 'Key' and 'Type of information' fields exactly as in the screenshot
- both agent and server are running
- host status is 'Monitored' and its availability icon is green
- host is selected in the host dropdown, 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.

Trigger
Dependencies

* Name

CPU load too high on "New host" for 3 minutes

Severity

Not classified
Information
Warning
Average
High

* Expression

{New host:system.cpu.load.avg(3m)}>2

[Expression constructor](#)

OK event generation

Expression
Recovery expression
None

PROBLEM event generation mode

Single
Multiple

OK event closes

All problems
All problems if tag values match

Tags

tag

value

[Add](#)

Allow manual close

☐

URL

Description

Enabled

☒

Add

Cancel

All mandatory input fields are marked with a red asterisk.

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: {New host:system.cpu.load.avg(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*.

Time ▲	☐ Severity	Recovery time	Status	Info	Host	Problem	Duration
09:48:39	☐ Not classified		PROBLEM		New host	CPU load too high on New host for 3 minutes	1m 23s

The flashing 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.

Media types					Create media type
☐ NAME ▲	TYPE	STATUS	USED IN ACTIONS	DETAILS	
☐ Email	Email	Enabled		SMTP server: "mail.company.com", SMTP helo: "company.com", SMTP email: "zabbix@company.com"	
☐ Jabber	Jabber	Enabled		Jabber identifier: "jabber@company.com"	
☐ SMS	SMS	Enabled		GSM modem: "dev/ttyS0"	

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

Media types

Media type

Options

*

Name

Email

Type

Email

*

SMTP server

mail.zabbix.com

SMTP server port

25

*

SMTP helo

zabbix.com

*

SMTP email

zabbix@zabbix.com

Connection security

None

STARTTLS

SSL/TLS

Authentication

None

Username and password

Message format

HTML

Plain text

Enabled

☒

Add

Cancel

All mandatory input fields are marked with a red asterisk.

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.

Press *Update* when ready.

Now you have configured 'Email' as a working media type. A media type must 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*.

Actions

Action

Operations

Recovery operations

Update operations

* Name

Test action

Conditions

Label

Name

Action

New condition

Trigger name

like

[Add](#)

Enabled ☒

* At least one operation, recovery operation or update operation must exist.

Add

Cancel

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 *New* in the Operations block, which opens a new operation form.

Actions

Action Operations **Recovery operations** Update operations

* Default operation step duration 1h

Default subject Problem: {EVENT.NAME}

Default message
Problem started at {EVENT.TIME} on {EVENT.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME}
Severity: {EVENT.SEVERITY}

Original problem ID: {EVENT.ID}
{TRIGGER.URL}

Pause operations for suppressed problems ☒

Operations	Steps	Details	Start in	Duration	Action
1	Send message to users:	user (New user) via Email	Immediately	Default	Edit Remove

Operation details

Steps 1 - 1 (0 - infinitely)

Step duration 0 (0 - use action default)

Operation type Send message

* At least one user or user group must be selected.

Send to User groups	User group	Action
	Add	

Send to Users	User	Action
	user (New user)	Remove
	Add	

Send only to Email

Default message ☒

Conditions	Label	Name	Action
	New		

[Update](#) [Cancel](#)

* At least one operation, recovery operation or update operation must exist.

[Add](#)

[Cancel](#)

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* in the operation detail block.

{TRIGGER.STATUS} and {TRIGGER.NAME} macros (or variables), visible in the *Default subject* and *Default message* fields, will be replaced with the actual trigger status and trigger name values.

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.

Templates

Template Linked templates Macros

* Template name

Visible name

* Groups
type here to search

Description

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.

Groups

- Select one or several groups by clicking *Select* button. The template must belong to a group.

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 go to the **Templates** tab.

There, click on *Select* next to *Link new templates*. In the pop-up window click on the name of template we have created ('New template'). As it appears in the *Link new templates* field, click on *Add*. The template should appear in the *Linked templates* list.

Hosts

All hosts / New host Enabled ZBX SNMP JMX IPMI Applications Items 1 Triggers 1 Graphs D

Host Templates IPMI Macros Host inventory

Linked templates

Name	Action
New template	Unlink

Link new templates

Add

Update Clone Full clone Delete Cancel

Click *Update* in the form to save the changes. The template is now added to the host, with all entities that it holds.

As you may have guessed, 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 Zabbix appliance installation CD image. Zabbix appliance installation CD could be used for instant deployment of Zabbix server (MySQL), Zabbix server (PostgreSQL), Zabbix proxy (MySQL) and Zabbix proxy (SQLite 3).

Zabbix appliance virtual machines have prepared Zabbix server with MySQL support. It is built using Zabbix appliance installation CD.

|<|<|<|

|<|<|<|

Zabbix appliance and installation CD versions are based upon the following Ubuntu versions:

Zabbix appliance version	Ubuntu version
4.0.0	16.04.3

Zabbix appliance is available in the following formats:

- vmdk (VMware/Virtualbox)
- OVF (Open Virtualisation Format)
- KVM
- HDD/flash image, USB stick
- Live CD/DVD

- Xen guest
- Microsoft VHD (Azure)
- Microsoft VHD (Hyper-V)

To get started, boot the appliance and point your browser at the IP it has received over DHCP: `http://<host_ip>/zabbix`
It has Zabbix server configured and running on MySQL, as well as frontend available.

The appliance has been built using standard Ubuntu/Debian feature called Preseed files.

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 Ubuntu configuration There are some changes applied to the base Ubuntu configuration.

1.1 Repositories

Official Zabbix **repository** has been added to `/etc/apt/sources.list`:

```
## Zabbix repository
deb https://repo.zabbix.com/zabbix/4.0/ubuntu xenial main
deb-src https://repo.zabbix.com/zabbix/4.0/ubuntu xenial main
```

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 Additional packages

Various basic utilities have been added that could make working with Zabbix and monitoring in general easier:

- iptables-persistent
- mc
- htop
- snmptrapfmt
- snmp-mibs-downloader

Some of these packages are used by Zabbix, some of them are installed to help users to configure/manage appliance settings.

1.4 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 file `/etc/network/interfaces` in your favourite editor;
- `iface eth0 inet dhcp → iface eth0 inet static`
- Add the following lines after `iface eth0 inet static`:
 - `address <IP address of the appliance>`
 - `netmask <network mask>`
 - `gateway <your gateway address>`
- Run the commands **`sudo ifdown eth0 && sudo ifup eth0`**.

Note:

For more information about other possible options see the official Ubuntu [documentation](#).

To configure DNS, add nameserver entries in `/etc/resolv.conf`, specifying each nameserver on its own line: **nameserver 192.168.1.2**.

1.5 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
```

1.6 Locale changes

The appliance contains a few locale changes:

- Contains languages: *en_US.UTF-8*, *ru_RU.UTF-8*, *ja_JP.UTF-8*, *cs_CZ.UTF-8*, *ko_KR.UTF-8*, *it_IT.UTF-8*, *pt_BR.UTF-8*, *sk_SK.UTF-8*, *uk_UA.UTF-8*, *fr_FR.UTF-8*, *pl.UTF-8*;
- Default locale is *en_US.UTF-8*.

These changes are required to support a multilingual Zabbix web-interface.

1.7 Other changes

- Network is configured to use DHCP to obtain IP address;
- Utility **fping** is set to have permissions 4710 and is owned by group **zabbix** - suid and only allowed to be used by zabbix group;
- ntpd configured to synchronise to the public pool servers: *ntp.ubuntu.com*;
- LVM volume is used with ext4 filesystem.
- "*UseDNS no*" is added to SSH server configuration file */etc/ssh/sshd_config* to avoid long SSH connection waits;
- Daemon snmpd is disabled using */etc/default/snmpd* configuration file.

2 Zabbix configuration Appliance Zabbix setup has the following passwords and other configuration changes:

2.1 Credentials (login:password)

System:

- appliance:zabbix

Attention:

Use "sudo su" command with "appliance" user name password to get root access rights.

Attention:

LiveCD comes without any password and uses the **ubuntu** user to log in into the console.

Database:

- root:<random>
- zabbix:<random>

Note:

Database passwords are randomly generated during the installation process.

Root password is stored to */root/.my.cnf* file, it is not required to input a password under the "root" account.

Zabbix frontend:

- Admin:zabbix

To change the database user password it has to be changed in the following locations:

- MySQL;
- */etc/zabbix/zabbix_server.conf*;
- */etc/zabbix/web/zabbix.conf.php*.

2.2 File locations

- Configuration files are placed in **/etc/zabbix**.
- Zabbix server, proxy and agent logfiles are placed in **/var/log/zabbix**.
- Zabbix frontend is placed in **/usr/share/zabbix**.
- Home directory for user **zabbix** is **/var/lib/zabbix**.

2.3 Changes to Zabbix configuration

- Server name for Zabbix frontend is set to "Zabbix Appliance";
- Frontend timezone is set to Europe/Riga (this can be modified in */etc/apache2/conf-available/zabbix.conf*);

2.4 Preserving configuration

If you are running a Live CD/DVD version of the appliance or for some other reason cannot have persistent storage, you can create a backup of the whole database, including all configuration and gathered data.

To create the backup, run:

```
sudo mysqldump zabbix | bzip2 -9 > dbdump.bz2
```

Now you can transfer the **dbdump.bz2** file to another machine.

To restore from the backup, transfer it to the appliance and execute:

```
bzcat dbdump.bz2 | sudo mysql zabbix
```

Attention:

Make sure that Zabbix server is stopped while performing the restore.

3 Frontend access Access to frontend by default is allowed from everywhere.

The frontend can be accessed `http://<host>/zabbix`.

This can be customised in `/etc/apache2/conf-available/zabbix.conf`. You have to restart the webserver after modifying this file. To do so, log in using SSH as **root** user and execute:

```
service apache2 restart
```

4 Firewall By default, only the ports listed in changes are open. To open additional ports just modify `"/etc/iptables/rules.v4"` or `"/etc/iptables/rules.v6"` files and reload firewall rules:

```
service iptables-persistent reload
```

5 Monitoring capabilities Zabbix installation is provided with the support for the following:

- SNMP
- IPMI
- Web monitoring
- VMware monitoring
- Jabber notifications
- EZ Texting notifications
- ODBC
- SSH2
- IPv6
- SNMP Traps
- Zabbix Java Gateway

6 SNMP traps Zabbix appliance uses `snmptrapfmt` to handle SNMP traps. It is configured to receive all traps from everywhere.

Authentication is not required. If you would like to enable authentication, you need to change the `/etc/snmp/snmptrapd.conf` file and specify required auth settings.

All traps are stored in the `/var/log/zabbix/snmptrapfmt.log` file. It is rotated by logrotate before reaching 2GB file size.

7 Upgrading The appliance Zabbix packages may be upgraded. To do so, run:

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

8 Naming, init and other scripts Appropriate init scripts are provided. To control Zabbix server, use any of these:

```
service zabbix-server status
```

Replace **server** with **agent** for Zabbix agent daemon or with **proxy** for Zabbix proxy daemon.

8.1 Increasing available diskpace

Warning:

Create a backup of all data before attempting any of the steps.

Available diskpace on the appliance might not be sufficient. In that case it is possible to expand the disk. To do so, first expand the block device in your virtualization environment, then follow these steps.

Start `fdisk` to change the partition size. As **root**, execute:

```
fdisk /dev/sda
```

This will start *fdisk* on disk *sda*. Next, switch to sectors by issuing:

```
u
```

Attention:

Don't disable DOS compatibility mode by entering *c*. Proceeding with it disabled will damage the partition.

Then delete the existing partition and create a new one with the desired size. In the majority of cases you will accept the available maximum, which will expand the filesystem to whatever size you made available for the virtual disk. To do so, enter the following sequence in *fdisk* prompt:

```
d
n
p
1
(accept default 63)
(accept default max)
```

If you wish to leave some space for additional partitions (swap etc), you can enter another value for *last sector*. When done, save the changes by issuing:

```
w
```

After partition creation (new disk or extended existing) create physical volume:

```
pvcreate /dev/sdb1
```

Warning:

Partition name */dev/sdb1* is used in the example; in your case disk name and partition number could be different. You can check partition number using *fdisk -l /dev/sdb* command.

Check newly created physical volume:

```
pvdisplay /dev/sdb1
```

Check available physical volumes. There must be 2 volumes *zabbix-vg* and newly created:

```
pvs
```

Extend your existing volume group with the newly created physical volume:

```
vgextend zabbix-vg /dev/sdb1
```

Check "zabbix-vg" volume group:

```
vgdisplay
```

Now extend your logical volume with the free PE space:

```
lvextend -l +100%FREE /dev/mapper/zabbix--vg-root
```

Resize your root volume (can be done on a live sysyem):

```
resize2fs /dev/mapper/zabbix--vg-root
```

Reboot the virtual machine (as the partition we modified is in use currently). That's it, filesystem should be grown to the partition size now. Check *"/dev/mapper/zabbix--vg-root"* volume:

```
df -h
```

9 Format-specific notes 9.1 Xen

Converting image for XenServer

To use Xen images with Citrix Xenserver you have to convert the disk image. To do so:

- Create a virtual disk, which is at least as large as the image
- Find out the UUID for this disk

```
xe vdi-list params=all
```

- If there are lots of disks, they can be filtered by the name parameter *name-label*, as assigned when creating the virtual disk
- Import the image

```
xe vdi-import filename="image.raw" uuid="<UUID>"
```


Instructions from Brian Radford blog.

9.2 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](#).

9.3 HDD/flash image (raw)

```
dd if=./zabbix_appliance_4.0.0_x86_64.raw of=/dev/sdc bs=4k conv=fdatasync
```

Replace */dev/sdc* with your Flash/HDD disk device.

10 Known issues

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 **Template** tab contains general template attributes.

The screenshot shows the 'Template' configuration form in Zabbix. At the top, there are four tabs: 'Template' (selected), 'Linked templates', 'Tags', and 'Macros'. The form contains the following fields:

- * Template name**: A text input field containing 'Template OS Linux'.
- Visible name**: An empty text input field.
- * Groups**: A dropdown menu showing 'Templates/Operating systems' with a search icon. Below the dropdown is a text input field with the placeholder 'type here to search'.
- Description**: A large text area for entering a description.

At the bottom of the form, there are two buttons: 'Add' (in blue) and 'Cancel' (in white with a blue border).

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. Note that leading and trailing spaces are disallowed since Zabbix 4.2.6.
Visible name	If you set this name, it will be the one visible in lists, maps, etc.
Groups	Host/template groups the template belongs to.
Description	Enter the template description.

The **Linked templates** tab allows you to 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 in the *Link new templates* field until a list of templates corresponding to the entered letter(s) appear. Scroll down to select. When all templates to be linked are selected, click on *Add*.

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

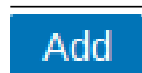
The **Tags** tab allows you to define template-level tags. All problems of hosts linked to this template will be tagged with the values entered here.

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**. 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.

Buttons:



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.
Delete and clear	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 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 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 screens

To add screens to a template in *Configuration → Templates*, do the following:

- Click on *Screens* in the row of the template
- Configure a screen following the usual method of [configuring screens](#)

Attention:

The elements that can be included in a template screen are: simple graph, custom graph, clock, plain text, URL.

Note:

For details on accessing host screens that are created from template screens, see the [host screen](#) 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](#)

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.

Attention:

Templates are linked directly to individual hosts and not to host groups. Simply adding a template to a host group will not link it. Host groups are used only for logical grouping of hosts and templates.

Linking a template

To link a template to the host, do the following:

- Go to *Configuration* → *Hosts*
- Click on the required host and switch to the *Templates* tab
- Click on *Add* next to *Link new templates*
- Select one or several templates in the 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
- 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 grey 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
- for applications - application name

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 in the *Templates* tab select to link additional templates:

The screenshot shows the 'Link templates' dialog in Zabbix. The 'Templates' tab is active. A search input field contains the text 'Template Module Interfaces SNMPv2'. Below the search field, there are two checkboxes: 'Replace' (which is checked) and 'Clear when unlinking' (which is unchecked). At the bottom of the dialog, there are two buttons: 'Update' and 'Cancel'.

Select *Link templates* and 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 *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 your 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.

Unlinking a template

To unlink a template from a host, do the following:

- Go to *Configuration* → *Hosts*
- Click on the required host and switch to the *Templates* tab
- 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 on individual templates entities 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 one "nested" template.

The benefit of nesting is that then you have to link only the one template to the host and the host will inherit all entities of the linked templates automatically.

Configuring a nested template

If you want to link some templates, to begin with you can take an existing template or a new one, then:

- Open the template properties form
- Look for the *Linked templates* tab
- Click on *Select* to select templates in the popup window
- Click on *Add* to list selected templates
- Click on *Add/Update* in the template properties form

Now the template should have all the entities (items, triggers, custom graphs etc.) of the linked templates.

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

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

Choosing the *Unlink and clear* option will remove both the association with the other 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*, *Linked templates* or *Tags*)
- Mark the checkboxes of any attribute to update and enter a new value for them

The screenshot shows a web interface for mass updating templates. At the top, there are three tabs: 'Template' (selected), 'Linked templates', and 'Tags'. Below the tabs, there are two rows of attributes for updating. The first row is 'Host groups' with a checked checkbox, followed by three buttons: 'Add', 'Replace', and 'Remove'. Below these buttons is a search input field with the placeholder text 'type here to search'. The second row is 'Description' with a checked checkbox, followed by a large text area for entering new values. At the bottom of the form, there are two buttons: 'Update' and 'Cancel'.

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 templates.
- *Replace* - will remove the template 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 templates.

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.

Template **Linked templates** Tags

Link templates ☒

type here to search

☐ Replace

☐ Clear when unlinking

Update Cancel

To update template linkage in the **Linked templates** tab, select *Link templates* and 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 templates before. 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.).

Template **Linked templates** **Tags**

Tags ☒

Add Replace Remove

Name	Value
tag	value

Add

Update Cancel

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 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"?

Typical Zabbix hosts are the devices you wish to monitor (servers, workstations, switches, etc).

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*

- 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:

Host
Templates
IPMI
Tags
Macros
Inventory
Encryption

* Host name

Visible name

* Groups

Discovered hosts

Linux servers

Select

type here to search

* At least one interface must exist.

Agent interfaces

IP address	DNS name	Connect to	Port
192.168.6.87		IP DNS	10050
Add			

SNMP interfaces

127.0.0.1		IP DNS	161
☒ Use bulk requests			
Add			

JMX interfaces

Add

IPMI interfaces

Add

Description

Monitored by proxy

Remote proxy

Enabled

☒

Add

Cancel

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Host name</i>	Enter a unique host name. Alphanumerics, spaces, dots, dashes and underscores are allowed. Note that leading and trailing spaces are disallowed since Zabbix 4.2.6. <i>Note:</i> With Zabbix agent running on the host you are configuring, the agent configuration file parameter <i>Hostname</i> must have the same value as the host name entered here. The name in the parameter is needed in the processing of active checks .

Parameter	Description
<i>Visible name</i>	If you set this name, it will be the one visible in lists, maps, etc. This attribute has UTF-8 support.
<i>Groups</i>	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.
<i>Interfaces</i>	Several host interface types are supported for a host: <i>Agent</i> , <i>SNMP</i> , <i>JMX</i> and <i>IPMI</i> . To add a new interface, click on <i>Add</i> in the <i>Interfaces</i> block and enter <i>IP/DNS</i> , <i>Connect to</i> and <i>Port</i> info. <i>Note</i> : Interfaces that are used in any items cannot be removed and link <i>Remove</i> is greyed out for them. <i>Use bulk requests</i> option for <i>SNMP</i> interfaces allows to enable/disable bulk processing of <i>SNMP</i> requests per interface.
<i>IP address</i>	Host IP address (optional).
<i>DNS name</i>	Host DNS name (optional).
<i>Connect to</i>	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
<i>Port</i>	TCP/UDP port number. Default values are: 10050 for Zabbix agent, 161 for <i>SNMP</i> agent, 12345 for <i>JMX</i> and 623 for <i>IPMI</i> .
<i>Default</i>	Check the radio button to set the default interface.
<i>Description</i>	Enter the host description.
<i>Monitored by proxy</i>	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"
<i>Enabled</i>	Mark the checkbox to make the host active, ready to be monitored. If unchecked, the host is not active, thus not monitored.

The **Templates** tab allows you to link **templates** to the host. All entities (items, triggers, graphs and applications) will be inherited from the template.

To link a new template, start typing in the *Link new templates* field until a list of matching templates appear. Scroll down to select. When all templates to be linked are selected, click on *Add*.

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.

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

Parameter	Description
<i>Authentication algorithm</i>	Select the authentication algorithm.
<i>Privilege level</i>	Select the privilege level.
<i>Username</i>	User name for authentication.
<i>Password</i>	Password for authentication.

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**. You may also view here template-level and global 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.

The **Encryption** tab allows you to require **encrypted** connections with the host.

Parameter	Description
<i>Connections to host</i>	How Zabbix server or proxy connects to Zabbix agent on a host: no encryption (default), using PSK (pre-shared key) or certificate.
<i>Connections from host</i>	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".
<i>Issuer</i>	Allowed issuer of certificate. Certificate is first validated with CA (certificate authority). If it is valid, signed by the CA, then the <i>Issuer</i> 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.
<i>Subject</i>	Allowed subject of certificate. Certificate is first validated with CA. If it is valid, signed by the CA, then the <i>Subject</i> field can be used to allow only one value of <i>Subject</i> string. If this field is empty then any valid certificate signed by the configured CA is accepted.
<i>PSK identity</i>	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.

Parameter	Description
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

Creating a host group

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

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

Host groups

* Group name

Add

Cancel

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Group name</i>	Enter a unique host group name. 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. Nested representation of host groups is supported since Zabbix 3.2.0.
<i>Apply permissions to all subgroups</i>	Checkbox is available to Zabbix Super Admin users only and only when editing an existing host group. Mark this checkbox and click on <i>Update</i> to apply the same level of permissions 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. This option is supported since Zabbix 3.4.0.

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[package,manager,short|full] - default is [all,all,full], package is regexp

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 auto registration 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.

Host inventory

[Overview](#) [Details](#)

Host name Zabbix server 1

Visible name Zabbix server

Agent interfaces

IP address	DNS name	Connect to	Port	Default
192.168.3.220		☒ IP ☐ DNS	10050	☒

SNMP interfaces

127.0.0.1		☐ IP ☐ DNS	161	☒

OS Linux linux-qvvt 3.11.10-21-default #1 SMP Mon Jul 21 15:28:46 U

Description Added on 2015-07-28.

Monitoring [Web](#) [Latest data](#) [Triggers](#) [Problems](#) [Graphs](#) [Screens](#)

Configuration [Host](#) [Applications 13](#) [Items 81](#) [Triggers 47](#) [Graphs 12](#) [Discovery 3](#) [Web 1](#)

[Cancel](#)

The **Overview** tab shows:

Parameter	Description
<i>Host name</i>	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.
<i>Visible name</i>	Visible name of the host (if defined).
<i>Host (Agent, SNMP, JMX, IPMI)
interfaces</i>	This block provides details of the interfaces configured for the host.
<i>OS</i>	Operating system inventory field of the host (if defined).
<i>Hardware</i>	Host hardware inventory field (if defined).
<i>Software</i>	Host software inventory field (if defined).
<i>Description</i>	Host description.
<i>Monitoring</i>	Links to monitoring sections with data for this host: <i>Web, Latest data, Triggers, Problems, Graphs, Screens.</i>
<i>Configuration</i>	Links to configuration sections for this host: <i>Host, Applications, Items, Triggers, Graphs, Discovery, Web.</i> 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, Templates, IPMI, Inventory or Encryption*)
- Mark the checkboxes of any attribute to update and enter a new value for them

Host Templates IPMI Tags Inventory Encryption

Host groups ☒ Add Replace Remove

Description ☐ Original

Monitored by proxy ☒ (no proxy)

Status ☐ Original

Update Cancel

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.

Host Templates IPMI Tags Inventory Encryption

Link templates ☒

☐ Replace

☐ Clear when unlinking

To update template linkage in the **Templates** tab, select *Link templates* and 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 *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.).

Host Templates IPMI Tags Inventory Encryption

Authentication algorithm ☐ Original

Privilege level ☒ Operator

Username ☐ Original

Password ☐ Original

Host
Templates
IPMI
Tags
Inventory
Encryption

Tags ☒

Add
Replace
Remove

Name

Value

tag

value

Add

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

Host
Templates
IPMI
Tags
Inventory
Encryption

Inventory mode ☒

Disabled
Manual
Automatic

Type ☐ Original

Type (Full details) ☐ Original

Name ☐ Original

Alias ☐ Original

OS ☐ Original

OS (Full details) ☐ Original

To be able to mass update inventory fields, the *Inventory mode* should be set to 'Manual' or 'Automatic'.

Host
Templates
IPMI
Tags
Inventory
Encryption

Connections ☒

Connections to host

No encryption
PSK
Certificate

Connections from host ☒ No encryption

☐ PSK
☐ Certificate

* PSK identity

* PSK

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.

133

Parameter	Description
<i>Type</i>	Item type. See individual item type sections.
<i>Key</i>	Item key. 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)', 'Simple check' or 'Zabbix aggregate', the key value must be supported by Zabbix agent or Zabbix server. See also: the correct key format.
<i>Host interface</i>	Select the host interface. This field is available when editing an item on the host level.
<i>Type of information</i>	Type of data as stored in the database after performing conversions, if any. Numeric (unsigned) - 64bit unsigned integer Numeric (float) - floating point number Negative values can be stored. Allowed range: -999999999999.9999 to 999999999999.9999. Starting with Zabbix 2.2, receiving values in scientific notation is also supported. E.g. 1e+7, 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.
<i>Units</i>	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 <i>bps</i> and receive a value of 881764, it will be displayed as 881.76 Kbps. Special processing is used for 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 <i>Numeric (unsigned)</i> 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. <i>Note</i> that if a unit is prefixed with !, then no unit prefixes/processing is applied to item values. See unit blacklisting.

Parameter	Description
<i>Update interval</i>	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. <i>Note:</i> If set to "0", the item will not be polled. However, if a custom interval (flexible/scheduling) also exists with a non-zero value, the item will be polled during the custom interval duration. <i>Note</i> that the first item poll after the item became active or after update interval change might occur earlier than the configured value. An existing passive item can be polled for value immediately by pushing the <i>Check now</i> button.
<i>Custom intervals</i>	You can create custom rules for checking the item: Flexible - create an exception to the <i>Update interval</i> (interval with different frequency) Scheduling - create a custom polling schedule. For detailed information see Custom intervals. Time suffixes are supported in the <i>Interval</i> 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. <i>Note:</i> Not available for Zabbix agent active items.
<i>History storage period</i>	Select either: Do not keep history - item history is not stored. Useful for master items if only dependent items need to keep history. <i>Note:</i> Before Zabbix 4.2.5, not keeping history had to be specified by entering "0" in the storage period field and it could be overridden by global housekeeper settings (now it cannot). 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 <i>Storage period</i> value can be overridden globally in <i>Administration</i> → <i>General</i> → <i>Housekeeper</i>. 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. <i>Overridden by global housekeeper settings (1d)</i>. 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.

Parameter	Description
<i>Trend storage period</i>	Select either: Do not keep trends - trends are not stored. <i>Note:</i> Before Zabbix 4.2.5, not keeping trends had to be specified by entering "0" in the storage period field and it could be overridden by global housekeeper settings (now it cannot). 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 <i>Storage period</i> value can be overridden globally in <i>Administration</i> → <i>General</i> → <i>Housekeeper</i>. 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. <i>Overridden by global housekeeper settings (7d)</i>. <i>Note:</i> Keeping trends is not available for non-numeric data - character, log and text. See also History and trends.
<i>Show value</i>	Apply value mapping to this item. Value mapping does not change received values, it is for displaying data only. It works with <i>Numeric(unsigned)</i>, <i>Numeric(float)</i> and <i>Character</i> items.
<i>Log time format</i>	For example, "Windows service states". 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 "pppppp:yyyyMMdd:hhmmss". Note that "p" and ":" chars are just placeholders and can be anything but "yMdhms".
<i>New application</i>	Enter the name of a new application for the item.
<i>Applications</i>	Link item to one or more existing applications.
<i>Populates host inventory field</i>	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 <i>Type of information</i> is set to 'Log'.
<i>Description</i>	Enter an item description.
<i>Enabled</i>	Mark the checkbox to enable the item so it will be processed.

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.

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
IBM DB2	255 bytes	2048 bytes	2048 bytes

Unit blacklisting

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'.

Any unit, however, can be prevented from being converted by using a ! prefix, for example !B. To better illustrate how the conversion works with and without the blacklisting, 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
```

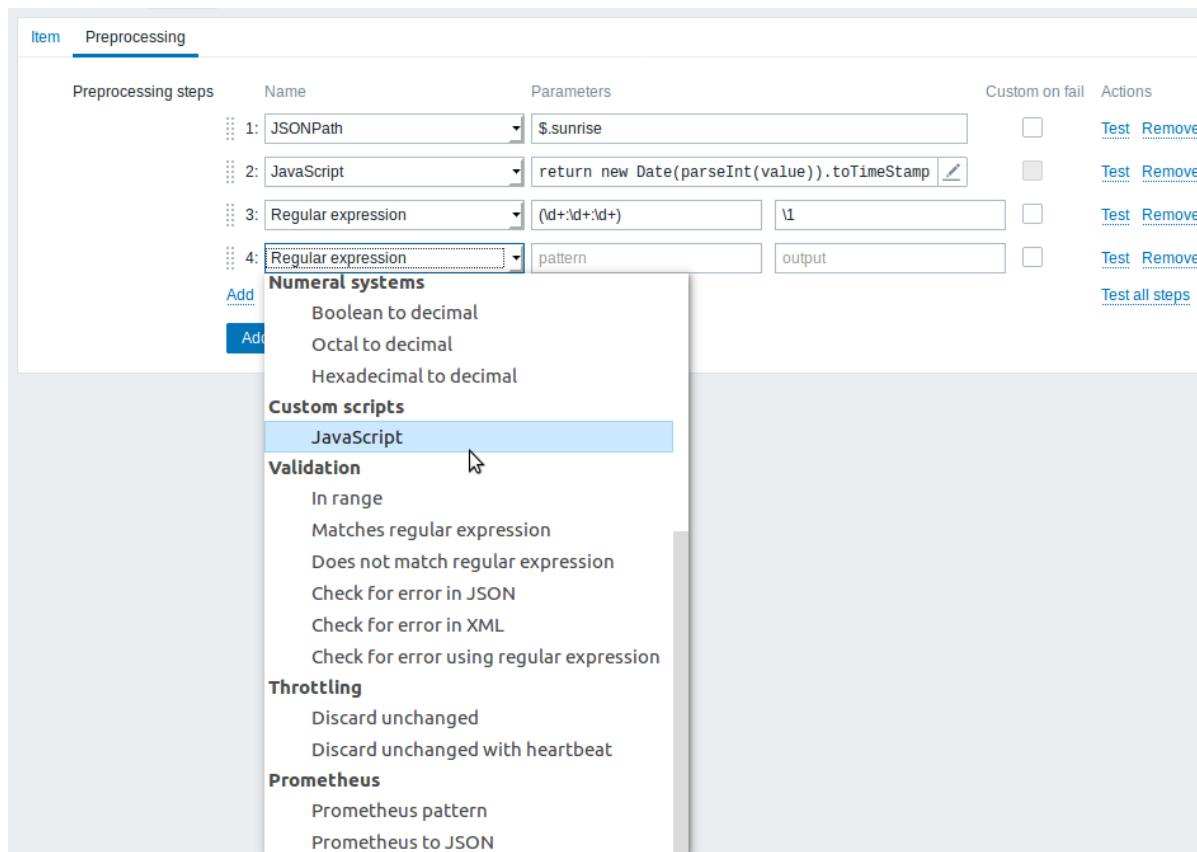
Note:

Before Zabbix 4.0, there was a hardcoded unit blacklist consisting of `ms`, `rpm`, `RPM`, `%`. This blacklist has been deprecated, thus the correct way of blacklisting such units is `!ms`, `!rpm`, `!RPM`, `!%`.

Item value preprocessing

The **Preprocessing** tab allows to define transformation rules for the received values. One or several transformations are possible before saving values to the database. Transformations are executed in the order in which they are defined. Preprocessing is done either by Zabbix server or by Zabbix proxy (for items monitored by proxy).

See also: [Preprocessing details](#)



User macros and user macros with context are supported in item value preprocessing parameters.

Note:

An item will become **unsupported** if any of the preprocessing steps fails except if custom error handling is specified using the *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.

Type	Transformation	Description
Text	<i>Regular expression</i>	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. Supported since 3.4.0. Please refer to regular expressions section for some existing examples. If you mark the <i>Custom on fail</i> 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. Remove specified characters from the beginning and end of the value.
	<i>Trim</i>	

Type	Transformation	Description
Structured data	<i>Right trim</i>	Remove specified characters from the end of the value.
	<i>Left trim</i>	Remove specified characters from the beginning of the value.
	<i>XML XPath</i>	Extract value or fragment from XML data using XPath functionality. For this option to work, Zabbix server must be compiled with libxml support. Examples: <code>number(/document/item/value)</code> will extract 10 from <pre><document><item><value>10</value></item></do</pre> <code>number(/document/item/@attribute)</code> will extract 10 from <code><document><item attribute="10"></item></document></code> <code>/document/item</code> will extract <pre><item><value>10</value></item></pre> from <pre><document><item><value>10</value></item></do</pre> Note that namespaces are not supported. Supported since 3.4.0. If you mark the <i>Custom on fail</i> 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.
	<i>JSON Path</i>	Extract value or fragment from JSON data using JSONPath functionality. (Note that only a simple subset of JSONPath functionality is supported in Zabbix 4.2.0-4.2.4.) If you mark the <i>Custom on fail</i> 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	<i>Custom multiplier</i>	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). <i>Note</i> that if the item type of information is <i>Numeric (unsigned)</i>, incoming values with a fractional part will be trimmed (i.e. '0.9' will become '0') before the custom multiplier is applied. Starting with Zabbix 2.2, using scientific notation is also supported. E.g. 1e+70. If you mark the <i>Custom on fail</i> 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	Transformation	Description
Numeral systems	<i>Simple change</i>	Calculate difference between the current and previous value. Evaluated as value-prev_value, where <i>value</i> - current value; <i>prev_value</i> - 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 <i>Custom on fail</i> 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.
	<i>Change per second</i>	Calculate the value change (difference between the current and previous value) speed per second. Evaluated as (value-prev_value)/(time-prev_time), where <i>value</i> - current value; <i>prev_value</i> - previously received value; <i>time</i> - current timestamp; <i>prev_time</i> - 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. <i>Note:</i> As this calculation may produce floating point numbers, it is recommended to set the 'Type of information' to <i>Numeric (float)</i>, 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 <i>Numeric (unsigned)</i> and thus trim only the decimal part. Only one change operation per item is allowed. If you mark the <i>Custom on fail</i> 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	Transformation	Description
	<i>Boolean to decimal</i>	Convert the value from boolean format to decimal. 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: <i>TRUE</i> - true, t, yes, y, on, up, running, enabled, available, ok, master <i>FALSE</i> - 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. Following values are supported since 4.0.0: ok, master, err, slave. If you mark the <i>Custom on fail</i> 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.
	<i>Octal to decimal</i>	Convert the value from octal format to decimal. If you mark the <i>Custom on fail</i> 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.
	<i>Hexadecimal to decimal</i>	Convert the value from hexadecimal format to decimal. If you mark the <i>Custom on fail</i> 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	<i>Javascript</i>	Enter JavaScript code in the block that appears when clicking in the parameter field or on <i>Open</i>. Note that available JavaScript length depends on the database used. For more information, see: Preprocessing by Javascript details.
Validation	<i>In range</i>	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. Minimum value should be less than the maximum. At least one value must exist. If you mark the <i>Custom on fail</i> 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	Transformation	Description
	<i>Matches regular expression</i>	Specify a regular expression that a value must match. If you mark the <i>Custom on fail</i> 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.
	<i>Does not match regular expression</i>	Specify a regular expression that a value must not match. If you mark the <i>Custom on fail</i> 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.
	<i>Check for error in JSON</i>	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. No error will be reported in case of failing to parse invalid JSON.
	<i>Check for error in XML</i>	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.
	<i>Check for error using a regular expression</i>	Check for an application-level error message using a regular expression. 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. 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.

Throttling

Type	Transformation	Description
Prometheus	<i>Discard unchanged</i>	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. Trigger functions will work only based on data that is actually saved in 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.
	<i>Discard unchanged with heartbeat</i>	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. Trigger functions will work only based on data that is actually saved in 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.
	<i>Prometheus pattern</i>	Use the following query to extract required data from Prometheus metrics. See Prometheus checks for more details.
	<i>Prometheus to JSON</i>	Convert required Prometheus metrics to JSON. See Prometheus checks for more details.

For change and throttling preprocessing steps Zabbix has to remember the last value to calculate/compare the new value as required. If Zabbix server is restarted or there is any change 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.

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 integer.

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
IBM DB2	2048	2048
SQLite (only Zabbix proxy)	65535	not limited

Testing preprocessing steps

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.

Each preprocessing step can be tested individually as well as all steps can be tested together. When you click on *Test* or *Test all steps* button respectively in the Actions block, a testing window is opened.

Parameter	Description
<i>Value</i>	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.
<i>Time</i>	Time of the input value is displayed: <i>now</i> (read-only).
<i>Previous value</i>	Enter a previous input value to compare to. Only for <i>Change</i> and <i>Throttling</i> preprocessing steps.
<i>Previous time</i>	Enter the previous input value time to compare to. Only for <i>Change</i> and <i>Throttling</i> preprocessing steps. The default value is based on the 'Update interval' field value of the item (if '1m', then this field is filled with <i>now-1m</i>). If nothing is specified or user has no access to host, the default is <i>now-30s</i> .
<i>Macros</i>	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. If non-existing or non-accessible (because of permissions) macro names are used, the macro values are editable within the testing context as well.
<i>End of line sequence</i>	Select the end of line sequence for multiline input values: LF - LF (line feed) sequence CRLF - CRLF (carriage-return line-feed) sequence. This option is supported since Zabbix 4.2.2.
<i>Preprocessing steps</i>	Preprocessing steps are listed; the testing result is displayed for each step after the <i>Test</i> 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).

Parameter	Description
<i>Result</i>	The final result of testing preprocessing steps is displayed in all cases when all steps are tested together (when you click on the <i>Test all steps</i> button). The type of conversion to the value type of the item is also displayed, for example <code>Result</code> converted to <code>Numeric (unsigned)</code>. An error icon is displayed in case of errors. The error description is displayed on mouseover. This field is available since Zabbix 4.2.4.

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 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.

Technical connection errors are displayed as error box at the top of the testing window.

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.
Check 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.
Clear history and trends	Delete the item history and trends.
Delete	Delete the item.
Cancel	Cancel the editing of item properties.

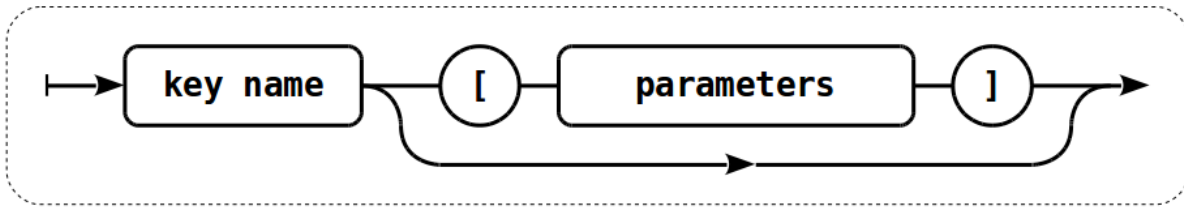
Unsupported items

An item can become unsupported if its value cannot be retrieved for some reason. Such items are still rechecked at a fixed interval, configurable in [Administration section](#).

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.

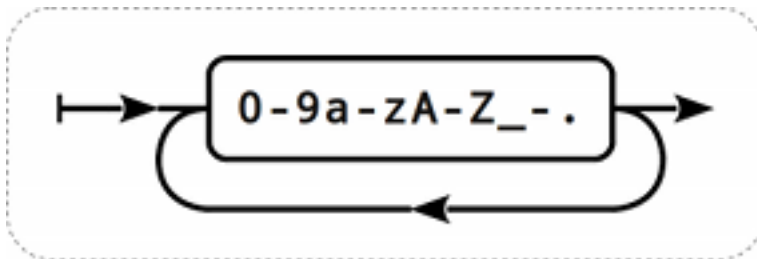
Key name

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

0-9a-zA-Z_-. .

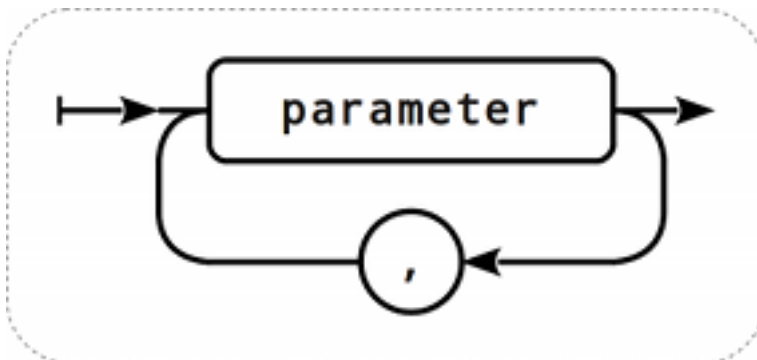
Which means:

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

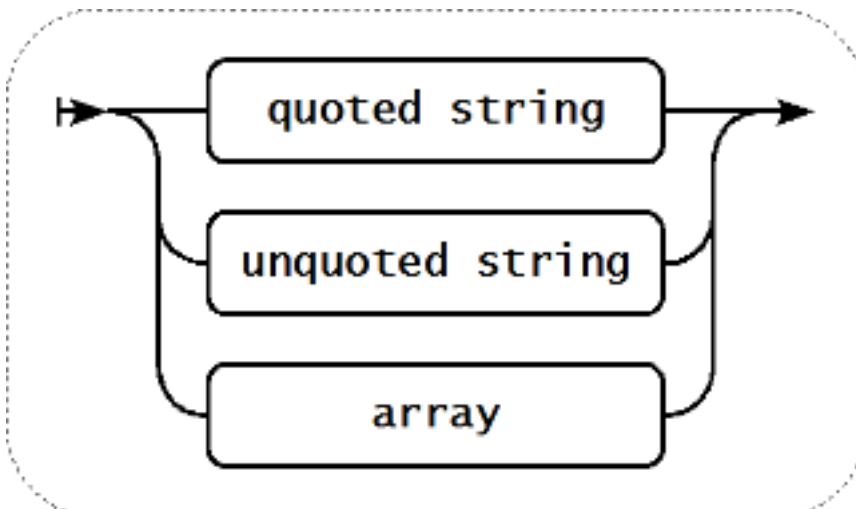


Key parameters

An item key can have multiple parameters that are comma separated.



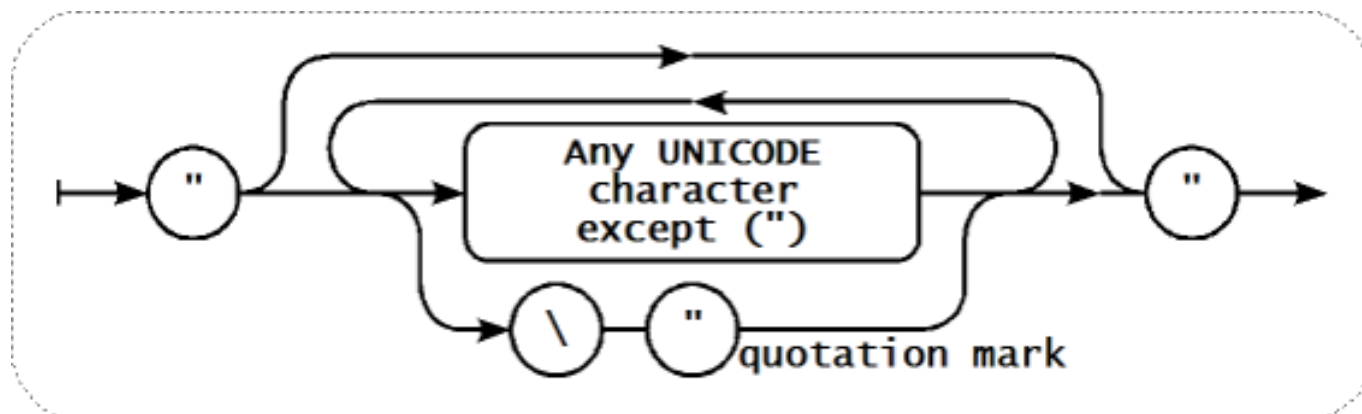
Each key parameter can be either a quoted string, an unquoted string or an array.



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, and included double quotes must be backslash escaped.

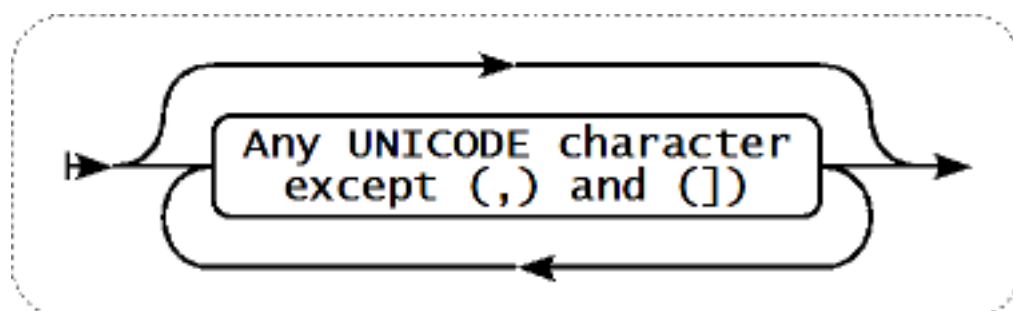


Warning:

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

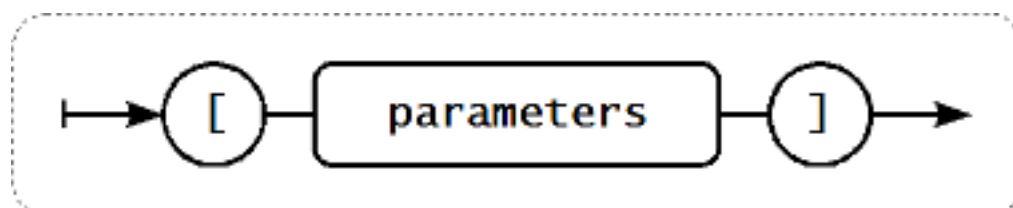
Parameter - unquoted string

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)

Up to seven flexible intervals can be defined. 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	<code><from></code>	<code><to></code>
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
<code>m0-59</code>	every minute
<code>h9-17/2</code>	every 2 hours starting with 9:00 (9:00, 11:00 ...)
<code>m0,30</code> or <code>m/30</code>	hourly at hh:00 and hh:30
<code>m0,5,10,15,20,25,30,35,40,45,50,55</code> or <code>m/5</code>	every five minutes
<code>wd1-5h9</code>	every Monday till Friday at 9:00
<code>wd1-5h9-18</code>	every Monday till Friday at 9:00,10:00,...,18:00
<code>h9,10,11</code> or <code>h9-11</code>	every day at 9:00, 10:00 and 11:00
<code>md1h9m30</code>	every 1st day of each month at 9:30
<code>md1wd1h9m30</code>	every 1st day of each month at 9:30 if it is Monday
<code>h9m/30</code>	every day at 9:00, 9:30
<code>h9m0-59/30</code>	every day at 9:00, 9:30
<code>h9,10m/30</code>	every day at 9:00, 9:30, 10:00, 10:30
<code>h9-10m30</code>	every day at 9:30, 10:30
<code>h9m10-40/30</code>	every day at 9:10, 9:40
<code>h9,10m10-40/30</code>	every day at 9:10, 9:40, 10:10, 10:40
<code>h9-10m10-40/30</code>	every day at 9:10, 9:40, 10:10, 10:40
<code>h9m10-40</code>	every day at 9:10, 9:11, 9:12, ... 9:40
<code>h9m10-40/1</code>	every day at 9:10, 9:11, 9:12, ... 9:40
<code>h9-12,15</code>	every day at 9:00, 10:00, 11:00, 12:00, 15:00
<code>h9-12,15m0</code>	every day at 9:00, 10:00, 11:00, 12:00, 15:00
<code>h9-12,15m0s30</code>	every day at 9:00:30, 10:00:30, 11:00:30, 12:00:30, 15:00:30
<code>h9-12s30</code>	every day at 9:00:30, 9:01:30, 9:02:30 ... 12:58:30, 12:59:30
<code>h9m/30;h10</code> (<i>API-specific syntax</i>)	every day at 9:00, 9:30, 10:00
<code>h9m/30</code>	every day at 9:00, 9:30, 10:00
<code>h10</code> (<i>add this as another row in frontend</i>)	

2 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
- Zabbix internal checks
- SSH checks
- Telnet checks
- External checks
- Aggregate checks
- Trapper items
- JMX monitoring

- [ODBC checks](#)
- [Dependent items](#)
- [HTTP checks](#)

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

These checks use the 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

Supported item keys

The table provides details on the item keys that you can use with Zabbix agent items.

See also:

- [Items supported by platform](#)
- [Item keys specific for Windows agent](#)
- [Minimum permission level for Windows agent items](#)

**** Mandatory and optional parameters ****

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

Key	Description	Return value	Parameters	Comments
agent.hostname	Agent host name.	String		Returns the actual value of the agent hostname from a configuration file.
agent.ping	Agent availability check.	Nothing - unavailable 1 - available		Use the nodata() trigger function to check for host unavailability.
agent.version	Version of Zabbix agent.	String		Example of returned value: 1.8.2
kernel.maxfiles				

Key		
	Maximum number of opened files supported by OS.	Integer
kernel.maxproc	Maximum number of processes supported by OS.	Integer
log[file,<regex>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>]		

Log file monitoring.	Log	file - full path and name of log file regexp - regular expression ⁴ 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 - possible values: <i>all</i> (default), <i>skip</i> - skip processing of older data (affects only newly created items). output - an optional output formatting template. The \0 escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an \N (where N=1...9) escape sequence is replaced with Nth matched group (or an empty string if the N exceeds the number of captured	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,, <i>Using output parameter for extracting a number from log record:</i> => 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,

log.count[file,<regex>,<encoding>,<maxproclines>,<mode>,<maxdelay>]

Count of
matched lines
in log file
monitoring.

Integer

file - full path
and name of
log file

regex -
regular
expression⁴
describing the
required
pattern

encoding -
code page
identifier

maxproclines
- maximum
number of new
lines per
second the
agent will
analyze.

Default value is
10*'MaxLines-
PerSecond' in
zab-

bix_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!

The item must
be configured
as an **active**
check.

If file is missing
or permissions
do not allow
access, item
turns
unsupported.

See also
additional
information on
log monitoring.

This item is not
supported for
Windows Event
Log.

Supported
since Zabbix
3.2.0.

logrt[file_regex,<regex>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>]

Log file monitoring with log rotation support.	Log	file_regexp - absolute path to file and regular expression⁴ describing the file name pattern regexp - regular expression⁴ 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 - possible values: <i>all</i> (default), <i>skip</i> - skip processing of older data (affects only newly created items). output - an optional output formatting template. The \0 escape sequence is replaced with the matched part of text (from the first character where match begins until the character where match ends) while an \N (where N=1...9) escape sequence is replaced with Nth matched	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:

Key

logrt.count[file_regexp,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>]

Count of matched lines in log file monitoring with log rotation support.	Integer	file_regexp - absolute path to file and regular expression⁴ describing the file name pattern regexp - regular expression⁴ describing the required content pattern encoding - code page identifier maxproclines - maximum number of new lines per second the agent will analyze. Default value is 10*'MaxLines-PerSecond' in zabbix_agentd.conf. mode - possible values: <i>all</i> (default), <i>skip</i> - 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 - type of log file rotation. Possible values: <i>rotate</i> (default),	The item must be configured as an active check. Log rotation is based on the last modification time of files. See also additional information on log monitoring. The options parameter is supported since Zabbix 4.0. This item is not supported for Windows Event Log. Supported since Zabbix 3.2.0.

Key

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)
name - DNS
name to query
type - record
type to be
queried
(default is *SOA*)
timeout
(ignored on
Windows) -
timeout for the
request in
seconds
(default is 1
second)
count (ignored
on Windows) -
number of tries
for the request
(default is 2)
protocol - the
protocol used
to perform DNS
queries: *udp*
(default) or *tcp*

Example:
=>

net.dns[8.8.8.8,zabbix.com,M]

The possible
values for
type are:
*ANY, A, NS,
CNAME, MB,
MG, MR, PTR,
MD, MF, MX,
SOA, NULL,
WKS* (except
for Windows),
*HINFO, MINFO,
TXT, SRV*

Internationalized
domain names
are not
supported,
please use
IDNA encoded
names instead.

The protocol
parameter is
supported
since Zabbix
3.0.
SRV record
type is
supported
since Zabbix
agent versions
1.8.6 (Unix)
and 2.0.0
(Windows).

Naming before
Zabbix 2.0 (still
supported):
net.tcp.dns

net.dns.record[<ip>,name,<type>,<timeout>,<count>,<protocol>]

Key	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) name - DNS name to query type - record type to be queried (default is <i>SOA</i>) timeout (ignored on Windows) - timeout for the request in seconds (default is 1 second) count (ignored on Windows) - number of tries for the request (default is 2) protocol - the protocol used to perform DNS queries: <i>udp</i> (default) or <i>tcp</i>	Example: => <i>net.dns.record[8.8.8.8,zabbix.</i> The possible values for type are: <i>ANY, A, NS, CNAME, MB, MG, MR, PTR, MD, MF, MX, SOA, NULL, WKS</i> (except for Windows), <i>HINFO, MINFO, TXT, SRV</i> Internationalized domain names are not supported, please use IDNA encoded names instead. The protocol parameter is supported since Zabbix 3.0. SRV record type is supported since Zabbix agent versions 1.8.6 (Unix) and 2.0.0 (Windows). Naming before Zabbix 2.0 (still supported): <i>net.tcp.dns.query</i>
<i>net.if.collisions[if]</i>	Number of out-of-window collisions.	Integer	if - network interface name	
<i>net.if.discovery</i>				

Key

	List of network interfaces. Used for low-level discovery.	JSON object	Supported since Zabbix agent version 2.0. On FreeBSD, OpenBSD and NetBSD supported since Zabbix agent version 2.2. 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>]			

Key	Description	Value	Unit
<code>net.if.in[if,<mode>]</code>	Incoming traffic statistics on network interface.	Integer	if - network interface name (Unix); network interface full description or IPv4 address (Windows) mode - possible values: <i>bytes</i> - number of bytes (default) <i>packets</i> - number of packets <i>errors</i> - number of errors <i>dropped</i> - number of dropped packets <i>overruns (fifo)</i> - the number of FIFO buffer errors <i>frame</i> - the number of packet framing errors <i>compressed</i> - the number of compressed packets transmitted or received by the device driver <i>multicast</i> - the number of multicast frames received by the device driver 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 since Zabbix agent version 1.8.6. Examples: => <code>net.if.in[eth0,errors]</code> => <code>net.if.in[eth0]</code> You may obtain network interface descriptions on Windows with <code>net.if.discovery</code> or <code>net.if.list</code> items. You may use this key with the <i>Change per second</i> preprocessing step in order to get bytes per second statistics.

`net.if.out[if,<mode>]`

Key	Description	Unit	Notes
net.if.out[if,<mode>]	Outgoing traffic statistics on network interface.	Integer	if - network interface name (Unix); network interface full description or IPv4 address (Windows) mode - possible values: <i>bytes</i> - number of bytes (default) <i>packets</i> - number of packets <i>errors</i> - number of errors <i>dropped</i> - number of dropped packets <i>overruns (fifo)</i> - the number of FIFO buffer errors <i>collisions</i> (<i>colls</i>) - the number of collisions detected on the interface <i>carrier</i> - the number of carrier losses detected by the device driver <i>compressed</i> - the number of compressed packets transmitted by the device driver 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 since Zabbix agent 1.8.6 version. Examples: => net.if.out[eth0,errors] => net.if.out[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. You may use this key with the <i>Change per second</i> preprocessing step in order to get bytes per second statistics.

net.if.total[if,<mode>]

	Sum of incoming and outgoing traffic statistics on network interface.	Integer	if - network interface name (Unix); network interface full description or IPv4 address (Windows) mode - possible values: <i>bytes</i> - number of bytes (default) <i>packets</i> - number of packets <i>errors</i> - number of errors <i>dropped</i> - number of dropped packets <i>overruns (fifo)</i> - the number of FIFO buffer errors <i>compressed</i> - the number of compressed packets transmitted or received by the device driver	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. Examples: => net.if.total[eth0,errors] => net.if.total[eth0] You may obtain network interface descriptions on Windows with net.if.discovery or net.if.list items. You may use this key with the <i>Change per second</i> 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]				

Key

	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	Example: => net.tcp.listen[80] On Linux supported since Zabbix agent version 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 address (default is 127.0.0.1) port - port number	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). Old naming: check_port[*]
net.tcp.service[service,<ip>,<port>]				

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)

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 by Windows agent is currently not supported.

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.

Old naming:
`check_service[*]`

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)

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>], for checks like these.

Checking of LDAP and HTTPS by Windows agent is currently not supported.

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.

Old naming:
check_service_perf[]*

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

Example:
=>
net.udp.listen[68]

On Linux supported since Zabbix agent version 1.8.4

net.udp.service[service,<ip>,<port>]

Key

	Checks if service is running and responding to UDP requests.	0 - service is down 1 - service is running	service - <i>ntp</i> (see details) ip - IP address (default is 127.0.0.1) 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 3.0.0, but <i>ntp</i> 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 - <i>ntp</i> (see details) ip - IP address (default is 127.0.0.1) 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 3.0.0, but <i>ntp</i> service was available for net.tcp.service[] item in prior versions.
proc.cpu.util[<name>,<user>,<type>,<cmdline>,<mode>,<zone>]				

Process CPU utilisation percentage.	Float	name - process name (default is <i>all processes</i>) user - user name (default is <i>all users</i>) type - CPU utilisation type: <i>total</i> (default), <i>user</i>, <i>system</i> cmdline - filter by command line (it is a regular expression⁴) mode - data gathering mode: <i>avg1</i> (default), <i>avg5</i>, <i>avg15</i> zone - target zone: <i>current</i> (default), <i>all</i>. This parameter is supported on Solaris only.	Examples: => proc.cpu.util[,root] → CPU utilisation of all processes running under the "root" user => proc.cpu.util[zabbix_server,za → CPU utilisation of all zabbix_server processes running under the zabbix user The returned value is based on single CPU core utilisation percentage. For example CPU utilisation of a process fully using two cores is 200%. The process CPU utilisation 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. <i>Note</i> that when setting the zone parameter to <i>current</i> (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 NOT-SUPPORTED (the agent

Key

proc.mem[<name>,<user>,<mode>,<cmdline>,<memtype>]

Memory used by process in bytes.	Integer - with mode as <i>max</i> , <i>min</i> , <i>sum</i> Float - with mode as <i>avg</i>	name - process name (default is <i>all processes</i>) user - user name (default is <i>all users</i>) mode - possible values: <i>avg</i> , <i>max</i> , <i>min</i> , <i>sum</i> (default) cmdline - filter by command line (it is a regular expression ⁴) memtype - type of memory used by process	Examples: => proc.mem[,root] → memory used by all processes running under the "root" user => proc.mem[zabbix_server,zabbix_server] → memory used by all zabbix_server processes running under the zabbix user => proc.mem[,oracle,max,oracle2] → memory used by the most memory-hungry process running under oracle having oracleZABBIX in its command line <i>Note:</i> 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

Key

proc.num[<name>,<user>,<state>,<cmdline>,<zone>]

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
expression⁴)
zone - target
zone: *current*
(default), *all*.
This parameter
is supported on
Solaris only.

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,oracle]
→ 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).

On Windows,
only the **name**
and **user**
parameters are
supported.

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

Key

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).

Reads
/proc/sys/dev/sensors
on Linux 2.4.

Example:
=> sen-
sor[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.
Reads the
hw.sensors MIB
on OpenBSD.

Examples:
=> sen-
sor[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.boottime

System boot
time.

Integer (Unix
timestamp)

system.cpu.discovery

List of detected
CPUs/CPU
cores. Used for
low-level
discovery.

JSON object

Supported on
all platforms
since 2.4.0.

system.cpu.intr

Device
interrupts.

Integer

system.cpu.load[<cpu>,<mode>]

	CPU load.	Float	cpu - possible values: <i>all</i> (default), <i>percpu</i> (total load divided by online CPU count) mode - possible values: <i>avg1</i> (one-minute average, default), <i>avg5</i> , <i>avg15</i>	Example: => sys-tem.cpu.load[,avg5] <i>percpu</i> is supported since Zabbix 2.0.0. Old naming: sys-tem.cpu.loadX
system.cpu.num[<type>]	Number of CPUs.	Integer	type - possible values: <i>online</i> (default), <i>max</i>	Example: => sys-tem.cpu.num
system.cpu.switches	Count of context switches.	Integer		Old naming: sys-tem[switches]
system.cpu.util[<cpu>,<type>,<mode>]				

	CPU utilisation percentage.	Float	cpu - <CPU <i>number</i> > or <i>all</i> (default) type - possible values: <i>idle, nice, user</i> (default), <i>system</i> (default for Windows), <i>iowait,</i> <i>interrupt,</i> <i>softirq, steal,</i> <i>guest</i> (on Linux kernels 2.6.24 and above), <i>guest_nice</i> (on Linux kernels 2.6.33 and above). Parameters <i>user</i> and <i>nice</i> time no longer include <i>guest</i> time and <i>guest_nice</i> time since Zabbix 3.0.14, 3.4.5 and 4.0.0. mode - possible values: <i>avg1</i> (one-minute average, default), <i>avg5,</i> <i>avg15</i>	Example: => sys- tem.cpu.util[0,user,avg5] Old naming: sys- tem.cpu.idleX, sys- tem.cpu.niceX, sys- tem.cpu.systemX, sys- tem.cpu.userX
system.hostname[<type>]				

Key				
	System host name.	String	type (Windows only, must not be used on other systems) - possible values: <i>netbios</i> (default) or <i>host</i>	The value is acquired by either GetComputerName() (for netbios) or gethostname() (for host) functions on Windows and by "hostname" command on other systems. Examples of returned values: <i>on Linux:</i> => sys-tem.hostname → linux-w7x1 => sys-tem.hostname → www.zabbix.com <i>on Windows:</i> => sys-tem.hostname → WIN-SERV2008-I6 => sys-tem.hostname[host] → Win-Serv2008-I6LonG The type parameter for this item is supported since Zabbix 1.8.6. See also a more detailed description.
system.hw.chassis[<info>]				

Key	Chassis information.	String	info - one of full (default), model, serial, type or vendor	Example: system.hw.chassis[full] Hewlett-Packard HP Pro 3010 Small Form Factor PC CZXXXXXXXXX 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 agent version 2.0.
system.hw.cpu[<cpu>,<info>]				

	CPU information.	String or integer	cpu - <CPU number> or <i>all</i> (default) info - possible values: <i>full</i> (default), <i>curfreq</i> , <i>maxfreq</i> , <i>model</i> or <i>vendor</i>	Example: => sys-tem.hw.cpu[0,vendor] → AuthenticAMD Gathers info from /proc/cpuinfo and /sys/devices/system/cpu/cpu If a CPU number and <i>curfreq</i> or <i>maxfreq</i> is specified, a numeric value is returned (Hz). Supported since Zabbix agent version 2.0.
system.hw.devices[<type>]	Listing of PCI or USB devices.	Text	type - <i>pci</i> (default) or <i>usb</i>	Example: => sys-tem.hw.devices[pci] → 00:00.0 Host bridge: Advanced Micro Devices [AMD] RS780 Host Bridge [...] Returns the output of either <i>lspci</i> or <i>lsusb</i> utility (executed without any parameters) Supported since Zabbix agent version 2.0.
system.hw.macaddr[<interface>,<format>]				

Key				
	Listing of MAC addresses.	String	interface - <i>all</i> (default) or a regular expression ⁴ format - <i>full</i> (default) or <i>short</i>	Lists MAC addresses of the interfaces whose name matches the given interface regular expression⁴ (<i>all</i> lists for all interfaces). Example: => sys- tem.hw.macaddr["eth0\$",full] → [eth0] 00:11:22:33:44:55 If <i>format</i> is specified as <i>short</i>, interface names and identical MAC addresses are not listed. Supported since Zabbix agent version 2.0.
system.localtime[<type>]	System time.	Integer - with type as <i>utc</i> String - with type as <i>local</i>	type - possible values: <i>utc</i> - (default) the time since the Epoch (00:00:00 UTC, January 1, 1970), measured in seconds. <i>local</i> - the time in the 'yyyy-mm-dd,hh:mm:ss.nnn, +mm:nn' format	Must be used as a passive check only. Parameters for this item are supported since Zabbix agent version 2.0. Example: => sys- tem.localtime[local] → create an item using this key and then use it to display host time in the Clock screen element.
system.run[command,<mode>]				

Run specified command on the host.	Text result of the command	command - command for execution mode - possible values: <i>wait</i> - wait end of execution (default), <i>nowait</i> - do not wait	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. <i>Note:</i> To enable this functionality, agent configuration file must contain EnableRemoteCommands=1 option. 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 .
1 - with mode as <i>nowait</i> (regardless of command result)			

system.stat[resource,<type>]

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

				Comments
				This item is supported on AIX only, since Zabbix 1.8.1. Take note of the following limitations in these items: => sys-tem.stat[cpu,app] - supported only on AIX LPAR of type "Shared" => sys-tem.stat[cpu,ec] - supported on AIX LPAR of type "Shared" and "Dedicated" ("Dedicated" always returns 100 (percent)) => sys-tem.stat[cpu,lbusy] - supported only on AIX LPAR of type "Shared" => sys-tem.stat[cpu,pc] - supported on AIX LPAR of type "Shared" and "Dedicated" => sys-tem.stat[ent] - supported on AIX LPAR of type "Shared" and "Dedicated"
system.sw.arch	Software architecture information.	String		Example: => system.sw.arch → i686 Info is acquired from uname() function. Supported since Zabbix agent version 2.0.
system.sw.os[<info>]				

Key				
	Operating system information.	String	info - possible values: <i>full</i> (default), <i>short</i> or <i>name</i>	Example: => sys-tem.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 (<i>full</i>) /proc/version_signature (<i>short</i>) PRETTY_NAME parameter from /etc/os-release on systems supporting it, or /etc/issue.net (<i>name</i>) Supported since Zabbix agent version 2.0.
system.sw.packages[<package>,<manager>,<format>]				

Listing of installed packages.	Text	package - <i>all</i> (default) or a regular expression⁴ manager - <i>all</i> (default) or a package manager format - <i>full</i> (default) or <i>short</i> Lists (alphabetically) installed packages whose name matches the given package regular expression⁴ (<i>all</i> lists them all). Example: <pre>=> system.sw.packages[mini,dpkg,sl → python-minimal, python2.6-minimal, ubuntu-minimal</pre> Supported package managers (executed command): <pre>dpkg (dpkg --get-selections) pkgtool (ls /var/log/packages) rpm (rpm -qa) pacman (pacman -Q)</pre> If format is specified as <i>full</i>, packages are grouped by package managers (each manager on a separate line beginning with its name in square brackets). If format is specified as <i>short</i>, packages are not grouped and are listed on a single line. Supported since Zabbix agent version 2.0.
system.swap.in[<device>,<type>]		

	Swap in (from device into memory) statistics.	Integer	device - device used for swapping (default is <i>all</i>) type - possible values: <i>count</i> (number of swapins), <i>sectors</i> (sectors swapped in), <i>pages</i> (pages swapped in). See supported by platform for details on defaults.	Example: => sys-tem.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)
system.swap.out[<device>,<type>]	Swap out (from memory onto device) statistics.	Integer	device - device used for swapping (default is <i>all</i>) type - possible values: <i>count</i> (number of swapouts), <i>sectors</i> (sectors swapped out), <i>pages</i> (pages swapped out). See supported by platform for details on defaults.	Example: => sys-tem.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 - device used for swapping (default is <i>all</i>) type - possible values: <i>free</i> (free swap space, default), <i>pfree</i> (free swap space, in percent), <i>pusd</i> (used swap space, in percent), <i>total</i> (total swap space), <i>used</i> (used swap space)	Example: => <code>sys-tem.swap.size[,pfree]</code> → free swap space percentage If <i>device</i> is not specified Zabbix agent will only take into account swap devices (files), physical memory will be ignored. For example, on Solaris systems <code>swap -s</code> command includes a portion of physical memory and swap devices (unlike <code>swap -l</code>). 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 <code>perf_counter[\700(_Total</code> key to obtain correct swap space percentage. Old naming: <code>sys-tem.swap.free</code> , <code>sys-tem.swap.total</code>
system.uname				

Identification of the system.	String	Example of returned value
		(Unix): FreeBSD localhost 4.2-RELEASE FreeBSD 4.2-RELEASE #0: Mon Nov i386
		Example of returned value (Windows): Windows ZABBIX-WIN 6.0.6001 Microsoft® Windows Server® 2008 Standard Service Pack 1 x86
		On Unix since Zabbix 2.2.0 the value for this item is obtained with <code>uname()</code> system call. Previously it was obtained by invoking <code>"uname -a"</code> . The value of this item might differ from the output of <code>"uname -a"</code> and does not include additional information that <code>"uname -a"</code> prints based on other sources.
		On Windows since Zabbix 3.0 the value for this item is obtained from <code>Win32_OperatingSystem</code> and <code>Win32_Processor</code> WMI classes. Previously it was obtained from volatile Windows APIs and undocumented registry keys.

Key

system.uptime	System uptime in seconds.	Integer		In item configuration , use s or uptime units to get readable values.
system.users.num	Number of users logged in.	Integer		who command is used on the agent side to obtain the value.
vfs.dev.read[<device>,<type>,<mode>]	Disk read statistics.	Integer - with type in <i>sectors, operations, bytes</i> Float - with type in <i>sps, ops, bps</i> <i>Note</i>: if using an update interval of three hours or more², will always return '0'	device - disk device (default is <i>all</i>³) type - possible values: <i>sectors, operations, bytes, sps, ops, bps</i> This parameter must be specified, since defaults differ under various OSes. <i>sps, ops, bps</i> stand for: sectors, operations, bytes per second, respectively. mode - possible values: <i>avg1</i> (one-minute average, default), <i>avg5</i>, <i>avg15</i>. This parameter is supported only with type in: <i>sps, ops, bps</i>.	You may use relative device names (for example, <i>sda</i>) as well as an optional <i>/dev/</i> prefix (for example, <i>/dev/sda</i>). LVM logical volumes are supported. Default values of 'type' parameter for different OSes: AIX - operations FreeBSD - <i>bps</i> Linux - <i>sps</i> OpenBSD - operations Solaris - bytes Example: => <code>vfs.dev.read[,operations]</code> <i>sps, ops</i> and <i>bps</i> on supported platforms used to be limited to 8 devices (7 individual and one <i>all</i>). Since Zabbix 2.0.1 this limit is 1024 devices (1023 individual and one for <i>all</i>). Old naming: <i>io[*]</i>

`vfs.dev.write[<device>,<type>,<mode>]`

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 more², will always return '0'

device - disk device (default is *all*³)

type - possible values: *sectors, operations, bytes, sps, ops, bps*

This parameter must be specified, since defaults differ under various OSes.

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*.

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 used to be limited to 8 devices (7 individual and one *all*). Since Zabbix 2.0.1 this limit is 1024 (1023 individual and one for *all*).

Old naming:
io[]*

`vfs.dir.count[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>,<max_age>]`

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: <i>file</i> - regular file, <i>dir</i> - subdirectory, <i>sym</i> - symbolic link, <i>sock</i> - socket, <i>bdev</i> - block device, <i>cdev</i> - character device, <i>fifo</i> - FIFO, <i>dev</i> - synonymous with "bdev,cdev", <i>all</i> - 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	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. On Windows, directory symlinks are skipped and hard links are counted only once. 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, not directories, 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

Key

vfs.dir.size[dir,<regex_incl>,<regex_excl>,<mode>,<max_depth>,<regex_excl_dir>]

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: <i>apparent</i> (default) - gets apparent file sizes rather than disk usage (acts as <code>du -sb dir</code>), <i>disk</i> - gets disk usage (acts as <code>du -s -B1 dir</code>). Unlike <code>du</code> command, <code>vfs.dir.size</code> item takes hidden files in account when calculating directory size (acts as <code>du -sb . [^.] * *</code> 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	Only directories with at least read permission for <i>zabbix</i> user are calculated. On Windows any symlink is skipped and hard links are taken into account only once. 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: ⇒ <code>vfs.dir.size[/tmp,log]</code> - calculates size of all files in /tmp which contain 'log' ⇒ <code>vfs.dir.size[/tmp,log,^[^.]old\$]</code> - 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.

Key

vfs.file.cksum[file]

File checksum, calculated by the UNIX cksum algorithm.

Integer

file - full path to file

Example:
=>
vfs.file.cksum[/etc/passwd]

Example of returned value:
1938292000

Old naming:
cksum

The file size limit depends on **large file support**.

vfs.file.contents[file,<encoding>]

Retrieving contents of a file.

Text

file - full path to file
encoding - code page identifier

Returns an empty string if the file is empty or contains LF/CR characters only.

Example:
=>
vfs.file.contents[/etc/passwd]

This item is limited to files no larger than 64 Kbytes.

Supported since Zabbix agent version 2.0.

vfs.file.exists[file]

Checks if file exists.

0 - not found

1 - regular file or a link (symbolic or hard) to regular file exists

file - full path to file

Example:
=>
vfs.file.exists[/tmp/application]

The return value depends on what S_ISREG POSIX macro returns.

The file size limit depends on **large file support**.

vfs.file.md5sum[file]

Key				
	MD5 checksum of file.	Character string (MD5 hash of the file)	file - full path to file	Example: => vfs.file.md5sum[/usr/local/etc/ Example of returned value: b5052dec b577e0ff d622d6d 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.regexp[file,regexp,<encoding>,<start line>,<end line>,<output>]				

Key	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 ⁴ 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).	Only the first matching line is returned. An empty string is returned if no line matched the expression. 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,zab => vfs.file.regexp[/path/to/some/f 9]+)\$" „3,5,\1] => vfs.file.regexp[/etc/passwd,"^ 9]+)" „,\1] → getting the ID of user <i>zabbix</i>
	vfs.file.regmatch[file,regexp,<encoding>,<start line>,<end line>]			

Key

	Find string in a file.	0 - match not found 1 - found	file - full path to file regexp - regular expression ⁴ 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).	The start line and end line parameters are supported from version 2.2. Example: => vfs.file.regmatch[/var/log/app.
vfs.file.size[file]	File size (in bytes).	Integer	file - full path to file	The file must have read permissions for user <i>zabbix</i> . Example: => vfs.file.size[/var/log/syslog] The file size limit depends on large file support .
vfs.file.time[file,<mode>]	File time information.	Integer (Unix timestamp)	file - full path to the file mode - possible values: <i>modify</i> (default) - last time of modifying file content, <i>access</i> - last time of reading file, <i>change</i> - last time of changing file properties	Example: => vfs.file.time[/etc/passwd,modi The file size limit depends on large file support .
vfs.fs.discovery				

Key				
	List of mounted filesystems. Used for low-level discovery.	JSON object		Supported since Zabbix agent version 2.0. {#FSDRIVETYPE} macro is supported on Windows since Zabbix agent version 3.0.
vfs.fs.inode[fs,<mode>]	Number or percentage of inodes.	Integer - for number Float - for percentage	fs - filesystem mode - possible values: <i>total</i> (default), <i>free</i> , <i>used</i> , <i>//pfree</i> // (free, percentage), <i>pused</i> (used, percentage)	Example: => vfs.fs.inode[/,pfree] Old naming: <i>vfs.fs.inode.free[*]</i> , <i>vfs.fs.inode.pfree[*]</i> , <i>vfs.fs.inode.total[*]</i>
vfs.fs.size[fs,<mode>]	Disk space in bytes or in percentage from total.	Integer - for bytes Float - for percentage	fs - filesystem mode - possible values: <i>total</i> (default), <i>free</i> , <i>used</i> , <i>pfree</i> (free, percentage), <i>pused</i> (used, percentage)	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 <i>free</i> mode. Old naming: <i>vfs.fs.free[*]</i> , <i>vfs.fs.total[*]</i> , <i>vfs.fs.used[*]</i> , <i>vfs.fs.pfree[*]</i> , <i>vfs.fs.pused[*]</i>
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*,
pusd (used, percentage),
available,
pavailable
 (available, percentage)

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*, *pusd*, *available*, *pavailable*.

See a more detailed description of `vm.memory.size` [parameters](#).

Old naming:
vm.memory.buffers,
vm.memory.cached,
vm.memory.free,
vm.memory.shared,
vm.memory.total

`web.page.get[host,<path>,<port>]`

Key				
	Get content of web page.	Web page source as text (including headers)	host - hostname or (since 4.2.4) URL (as <code>scheme://host:port/path</code> where only <code>host</code> is mandatory). Allowed URL schemes: <code>http</code>, <code>https</code>⁵. Missing scheme will be treated as <code>http</code>. If URL is specified <code>path</code> and <code>port</code> must be empty. Specifying user name/password when connecting to servers that require authentication, for example: <code>http://user:password@www.example.com</code> is only possible with cURL support⁵. Punycode is supported in hostnames (since 4.2.4). path - path to HTML document (default is <code>/</code>) port - port number (default is 80 for HTTP)	This item turns unsupported if the resource specified in <code>host/port</code> 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: => <code>web.page.get[www.zabbix.com]</code> => <code>web.page.get[https://www.zabbix.com]</code> <code>web.page.get[https://blog.zabbix.com]</code> => <code>web.page.get[localhost:80]</code> => <code>web.page.get["::1]/server-status"]</code>
	<code>web.page.perf[host,<path>,<port>]</code>			

Key	Loading time of full web page (in seconds).	Float	host - hostname or (since 4.2.4) URL (as <code>scheme://host:port/path</code> where only <code>host</code> is mandatory). Allowed URL schemes: <code>http</code>, <code>https</code>⁵. Missing scheme will be treated as <code>http</code>. If URL is specified <code>path</code> and <code>port</code> must be empty. Specifying user name/password when connecting to servers that require authentication, for example: <code>http://user:password@www.example.com</code> is only possible with cURL support⁵. Punycode is supported in hostnames (since 4.2.4). path - path to HTML document (default is /) port - port number (default is 80 for HTTP)	This item turns unsupported if the resource specified in <code>host/port</code> 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: => <code>web.page.perf[www.zabbix.co</code> => <code>web.page.perf[https://www.za</code> <code>http://user:password@www.example.com</code>

`web.page.regexp[host,<path>,<port>,regexp,<length>,<output>]`

Find string on a web page.	The matched string, or as specified by the optional output parameter	host - hostname or (since 4.2.4) URL (as <code>scheme://host:port/path</code> where only <i>host</i> is mandatory). Allowed URL schemes: <i>http</i>, <i>https</i>⁵. Missing scheme will be treated as <i>http</i>. If URL is specified <i>path</i> and <i>port</i> must be empty. Specifying user name/password when connecting to servers that require authentication, for example: <code>http://user:password@example.com</code> is only possible with cURL support⁵. Punycode is supported in hostnames (since 4.2.4). path - path to HTML document (default is /) port - port number (default is 80 for HTTP) regexp - regular expression⁴ describing the required pattern length - maximum number of characters to return output - an optional output formatting template. The <code>\0</code> escape sequence is replaced with the matched part of text (from the first character where match begins until the	This item turns unsupported if the resource specified in <i>host</i> does not exist or is unavailable. <i>host</i> 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: => <code>web.page.regexp[www.zabbix]</code> => <code>web.page.regexp[https://www</code>

Key

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)

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 4.0.5.

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)

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 4.0.5.

Footnotes

¹A Linux-specific note. Zabbix agent must have read-only access to filesystem */proc*. Kernel patches from www.grsecurity.org limit access rights of non-privileged users.

² `vfs.dev.read[]`, `vfs.dev.write[]`: Zabbix agent will terminate "stale" device connections if the item values are not accessed for more than 3 hours. This may happen if a system has devices with dynamically changing paths or if a device gets manually removed. Note also that these items, if using an update interval of 3 hours or more, will always return '0'.

³ `vfs.dev.read[]`, `vfs.dev.write[]`: If default *all* is used for the first parameter then the key will return summary statistics, including all block devices like `sda`, `sdb` and their partitions (`sda1`, `sda2`, `sdb3`...) and multiple devices (MD raid) based on those block devices/partitions and logical volumes (LVM) based on those block devices/partitions. In such cases returned values should be considered only as relative value (dynamic in time) but not as absolute values.

⁴ [Perl Compatible Regular Expression](#) (PCRE) since Zabbix 3.4; POSIX-extended regular expression before that. See also: [Regular expressions supported by location](#).

⁵ SSL (HTTPS) is supported only if agent is compiled with cURL support. Otherwise the item will turn unsupported.

Available encodings

The `encoding` parameter is used to specify encoding for processing corresponding item checks, so that data acquired will not be corrupted. For a list of supported encodings (code page identifiers), please consult respective documentation, such as documentation for [libiconv](#) (GNU Project) or Microsoft Windows SDK documentation for "Code Page Identifiers".

If empty `encoding` is passed, then UTF-8 (default locale for newer Unix/Linux distributions, see your system's settings) or ANSI with system-specific extension (Windows) is used by default.

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-specific item keys

Item keys

The table provides details on the item keys that you can use with Zabbix Windows agent only.

See also: [Minimum permission level for Windows agent items](#)

Key	Description	Return value	Parameters	Comments
eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>,<mode>]				

Event log monitoring.	Log		
		name - name of event log regex - regular expression describing the required pattern severity - regular expression describing severity This parameter accepts the following values: <i>"Information", "Warning", "Error", "Critical", "Verbose"</i> (since Zabbix 2.2.0 running on Windows Vista or newer) source - regular expression describing source identifier (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.config mode - possible values: <i>all</i> (default), <i>skip</i> - skip processing of	The item must be configured as an active check . Examples: => event-log[Application] => event-log[Security,"Failure Audit" ,"^(529 680)\$] => event-log[System,"Warning Error"] => event-log[System,,,,^1\$] => event-log[System,,,,@TWOSHORT] - here a custom regular expression named TWOSHORT is referenced (defined as a <i>Result is TRUE</i> type, the expression itself being <i>^1\$\ ^70\$</i>). <i>Note</i> 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. <i>Note</i> 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

Key

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 en-
abling/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>]

	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 <code>interval</code> 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 <code>\System\Threads</code>). It will not work as expected for counters that require more than one sample - like CPU utilisation. Since 1.6, <code>interval</code> is used, so the check returns an average value for last "interval" seconds every time. See also: Windows performance counters .
<code>perf_counter_en[counter,<interval>]</code>	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 <code>interval</code> 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: <code>HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perfl</code> Supported since Zabbix agent version 4.2.7.
<code>proc_info[process,<attribute>,<type>]</code>				

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 - size
(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

service.discovery

List of Windows services. Used for **low-level discovery**.

JSON object

Supported since Zabbix agent version 3.0.

service.info[service,<param>]

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 - <i>all</i> (default), <i>automatic</i> , <i>manual</i> or <i>disabled</i> state - <i>all</i> (default), <i>stopped</i> , <i>started</i> , <i>start_pending</i> , <i>stop_pending</i> , <i>running</i> , <i>continue_pending</i> , <i>pause_pending</i> or <i>paused</i> exclude - services to exclude from the result. Excluded services should be listed in double quotes, separated by comma, without spaces.	Examples: => <i>services[,started]</i> - list of started services => <i>services[automatic, stopped]</i> - list of stopped services, that should be run => <i>services[automatic, stopped, "service1,service2,service3"]</i> - list of stopped services, that should be run, excluding services with names <i>service1</i> , <i>service2</i> and <i>service3</i> The <i>exclude</i> parameter is supported since Zabbix 1.8.1.
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	Example: => <i>wmi.get[root\cimv2,select status from Win32_DiskDrive where Name like '%PHYSICALDRIVE0%']</i> - returns the status of the first physical disk. This key is supported since Zabbix 2.2.0.
vm.vmemory.size[<type>]				

Key				
	Virtual memory size in bytes or in percentage from total.	Integer - for bytes Float - for percentage	type - possible values: <i>available</i> (available virtual memory), <i>pavailable</i> (available virtual memory, in percent), <i>used</i> (used virtual memory, in percent), <i>total</i> (total virtual memory, default), <i>used</i> (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.

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.

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.

Since Zabbix 2.2.3 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. Please see the **technical detail section** below on how it works internally. Since Zabbix 2.4 there is also a "Use bulk requests" setting for each interface that allows to disable bulk requests for devices that cannot handle them properly.

Since Zabbix 2.2.7 and Zabbix 2.4.2 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.

Since Zabbix 2.2 Zabbix server and proxy daemons correctly use the Timeout configuration parameter when performing SNMP checks. Additionally the daemons do not perform retries after a single unsuccessful SNMP request (timeout/wrong credentials). Previously the SNMP library default timeout and retry values (1 second and 5 retries respectively) were actually used.

Since Zabbix 2.2.8 and Zabbix 2.4.2 Zabbix server and proxy daemons will always retry at least one time: 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.

Configuring SNMP monitoring

To start monitoring a device through SNMP, the following steps have to be performed:

Step 1

Create a host for the device with an SNMP interface.

Enter the IP address. 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.

Note:

SNMP checks do not use *Agent port*, it is ignored.

Step 2

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 10.62.1.22 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_INTEGER, ASN_UNSIGNED64, 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 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'. Make sure the 'Host interface' field has your switch/router in it and change the 'Type' field to "SNMPv* agent". Enter the community (usually public) and 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

Enter the SNMP 'Port' as 161 and the 'Key' as something meaningful, e.g. SNMP-InOctets-Bps. Set the 'Type of information' to *Numeric (float)* and the preprocessing step as *Change per second* (important, otherwise you will get cumulative values from the SNMP device instead of the latest change). Choose a custom multiplier if you want one and enter an 'Update interval' and 'History storage period' if you want them to be different from the default.

Items

All hosts / Zabbix server Enabled ZBX SNMP JMX IPMI Applications 13 Items 81 Triggers 47

Item Preprocessing

* Name	SNMP: InOctets (Bps)
Type	SNMPv3 agent
* Key	SNMP-InOctets-Bps
* Host interface	127.0.0.1 : 161
* SNMP OID	.1.3.6.1.2.1.2.2.1.10.3
Context name	
Security name	
Security level	authPriv
Authentication protocol	MD5 SHA
Authentication passphrase	
Privacy protocol	DES AES
Privacy passphrase	
Port	161
Type of information	Numeric (float)

All mandatory input fields are marked with a red asterisk.

Now save the item and go to *Monitoring* → *Latest data* for your SNMP data!

Take note of specific options available for SNMPv3 items:

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 or SHA.
Authentication passphrase	Enter authentication passphrase. User macros are resolved in this field.

Parameter	Description
<i>Privacy protocol</i>	Select privacy protocol - <i>DES</i> or <i>AES</i> .
<i>Privacy passphrase</i>	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.

Warning:

Server/proxy restart is required for changes in *Authentication protocol*, *Authentication passphrase*, *Privacy protocol* or *Privacy passphrase* to take effect.

Example 1

General example:

Parameter	Description
Community	public
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
Community	public
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.

Special OID	Identifier	Description
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:

```
# yum 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 reenables
# 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:

```
# yum 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 is to keep in mind that Zabbix processes do not get aware 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/local/share/snmp/mibs" >> /etc/snmp/snmp.conf
# 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 a 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 built-in mechanisms for passing the traps to Zabbix - either a perl script or SNMPPTT.

The workflow of receiving a trap:

1. **snmptrapd** receives a trap
2. snmptrapd passes the trap to SNMPPTT or calls Perl trap receiver
3. SNMPPTT or Perl trap 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.)

1 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 snmptrap[regexp] Catches all SNMP traps that match the regular expression specified in regexp . If regexp is unspecified, catches any trap.	Return value SNMP trap	Comments This item can be set only for SNMP interfaces. This item is supported since Zabbix 2.0.0 . <i>Note:</i> Starting with Zabbix 2.0.5, user macros and global regular expressions are supported in the parameter of this item key.
snmptrap.fallback Catches all SNMP traps that were not caught by any of the snmptrap[] items for that interface.	SNMP trap	This item can be set only for SNMP interfaces. This item is supported since Zabbix 2.0.0 .

Note:

Multi-line regexp matching is not supported at this time.

Set the **Type of information** to be '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 correctly set up.

2 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 perl trap receiver. To do that, edit the configuration file (`zabbix_server.conf` or `zabbix_proxy.conf`):

1. `StartSNMPTrapper=1`
2. `SNMPTrapperFile=[TRAP FILE]`

Warning:

If systemd parameter `PrivateTmp` is used, this file is unlikely to work in `/tmp`.

Configuring SNMPTT

At first, `snmptrapd` should be configured to use SNMPTT.

Note:

For the best performance, SNMPTT should be configured as a daemon using `snmpthandler-embedded` to pass the traps to it. See instructions for configuring SNMPTT in its homepage:

<http://snmptt.sourceforge.net/docs/snmptt.shtml>

When SNMPTT is configured to receive the traps, configure SNMPTT to log the traps:

1. log traps to the trap file which will be read by Zabbix:
 - `log_enable = 1`
 - `log_file = [TRAP FILE]`
2. set the date-time format:
 - `date_time_format = %H:%M:%S %Y/%m/%d = [DATE TIME FORMAT]`

Now format the traps for Zabbix to recognise 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 recognise them. Unknown traps can be handled by defining a general event in `snmptt.conf`:

`EVENT general .* "General event" Normal`

Configuring Perl trap receiver

Requirements: Perl, Net-SNMP compiled with `--enable-embedded-perl` (done by default since Net-SNMP 5.4)

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 `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 script name is not quoted, snmptrapd will refuse to start up with messages, similar to these:

Regexp modifiers "/" and "/"a" are mutually exclusive at (eval 2) line 1, at end of line
 Regexp modifier "/" may not appear twice at (eval 2) line 1, at end of line

SNMP trap format

All customised 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] - 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 on interface 2.
Admin state: 1. Operational state: 2
```

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.
```

3 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 define 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).

4 Setup example

This example uses snmptrapd + SNMPTT to pass traps to Zabbix server. Setup:

1. **zabbix_server.conf** - configure Zabbix to start SNMP trapper and set the trap file:
 StartSNMPTrapper=1
 SNMPTrapperFile=/tmp/my_zabbix_traps.tmp
2. **snmptrapd.conf** - add SNMPTT as the trap handler:
 traphandle default snmptt
3. **snmptt.ini** - configure output file and time format:
 log_file = /tmp/my_zabbix_traps.tmp
 date_time_format = %H:%M:%S %Y/%m/%d
4. **snmptt.conf** - define a default trap format:
 EVENT general .* "General event" Normal
 FORMAT ZBXTRAP \$aA \$ar
5. Create an SNMP item TEST:
 Host's SNMP interface IP: 127.0.0.1
 Key: snmptrap["General"]
 Log time format: hh:mm:ss yyyy/MM/dd

This results in:

1. Command used to send a trap:
`snmptrap -v 1 -c public 127.0.0.1 '.1.3.6.1.6.3.1.1.5.3' '0.0.0.0' 6 33 '55' .1.3.6.1.6.3.1.1.5.3 s "teststring000"`
2. The received trap:
`15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal "General event" localhost - ZBXTRAP 127.0.0.1 127.0.0.1`
3. Value for item TEST:
`15:48:18 2011/07/26 .1.3.6.1.6.3.1.1.5.3.0.33 Normal "General event" localhost - 127.0.0.1`

Note:

This simple example uses SNMPPTT as **traphandle**. For better performance on production systems, use embedded Perl to pass traps from snmptrapd to SNMPPTT or directly to Zabbix.

5 See also

- [CentOS based SNMP trap tutorial on zabbix.org](#)

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:

- For *Host interface* select the IPMI IP and port
- Select 'IPMI agent' as the *Type*
- Specify the *IPMI sensor* (for example 'FAN MOD 1A RPM' on Dell Poweredge). By default, 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.
- Enter an item **key** that is unique within the host (say, `ipmi.fan.rpm`)
- 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

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 a function **band** with mask 1 can be used. The trigger expression could be like

```
{www.zabbix.com:Power Unit Stat.band(#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: <i>min</i> , <i>max</i> , <i>avg</i> (default)	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: <i>ssh, ldap, smtp, ftp, http, pop, nntp, imap, tcp, https, telnet</i> (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 <i>tcp</i> 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. <i>https</i> and <i>telnet</i> services are supported since Zabbix 2.0.

net.tcp.service.perf[service,<ip>,<port>]

	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: <i>ssh, ldap, smtp, ftp, http, pop, nntp, imap, tcp, https, telnet</i> (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 <i>tcp</i> 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>], for checks like these. <i>https</i> and <i>telnet</i> 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: <i>ntp</i> (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 <i>ntp</i> service was available for net.tcp.service[] item in prior versions.
net.udp.service.perf[service,<ip>,<port>]				

Key				
	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: <i>ntp</i> (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 <i>ntp</i> service was available for net.tcp.service[] item in prior versions.

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

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.

1 VMware monitoring item keys

Item keys

The table provides details on the simple checks that can be used to monitor **VMware environments**.

Key	Description	Return value	Parameters	Comments
vmware.cluster.discovery[<url>]	Discovery of VMware clusters.	JSON object	url - VMware service URL	
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.datastore.discovery[<url>]	Discovery of VMware datastores.	JSON object	url - VMware service URL	This item is supported since Zabbix 4.2.4.
vmware.datastore.hv.list[<url>,<datastore>]	List of datastore hypervisors.	JSON object	url - VMware service URL datastore - datastore name	This item is supported since Zabbix 4.2.4.
vmware.datastore.read[<url>,<datastore>,<mode>]	Amount of time for a read operation from the datastore (milliseconds).	Integer ²	url - VMware service URL datastore - datastore name mode - latency (average value, default), maxlatency (maximum value)	This item is supported since Zabbix 4.2.4.
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	This item is supported since Zabbix 4.2.4.
vmware.datastore.write[<url>,<datastore>,<mode>]				

	Amount of time for a write operation to the datastore (milliseconds).	Integer ²	url - VMware service URL datastore - datastore name mode - latency (average value, default), maxlatency (maximum value)	This item is supported since Zabbix 4.2.4.
vmware.eventlog[<url>,<mode>]	VMware event log.	Log	url - VMware service URL mode - <i>all</i> (default), <i>skip</i> - skip processing of older data	
vmware.fullname[<url>]	VMware service full name.	String	url - VMware service URL	
vmware.hv.cluster.name[<url>,<uuid>]	VMware hypervisor cluster name.	String	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.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	
vmware.hv.datastore.list[<url>,<uuid>]	List of VMware hypervisor datastores.	JSON object	url - VMware service URL uuid - VMware hypervisor host name	This item is supported since Zabbix 4.2.4.
vmware.hv.datastore.read[<url>,<uuid>,<datastore>,<mode>]				

	Average amount of time for a read operation from the datastore (milliseconds).	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	Available since Zabbix versions 3.0.6, 3.2.2
vmware.hv.datastore.write[<url>,<uuid>,<datastore>,<mode>]	Average amount of time for a write operation to the datastore (milliseconds).	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.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	
vmware.hv.hw.cpu.num[<url>,<uuid>]				

Key

vmware.hv.hw.cpu.threads[<url>,<uuid>]	Number of processor cores on VMware hypervisor.	Integer	url - VMware service URL uuid - VMware hypervisor host name
vmware.hv.hw.memory[<url>,<uuid>]	Number of processor threads on VMware hypervisor.	Integer	url - VMware service URL uuid - VMware hypervisor host name
vmware.hv.hw.model[<url>,<uuid>]	VMware hypervisor total memory size (bytes).	Integer	url - VMware service URL uuid - VMware hypervisor host name
vmware.hv.hw.uuid[<url>,<uuid>]	VMware hypervisor model.	String	url - VMware service URL uuid - VMware hypervisor host name
vmware.hv.hw.vendor[<url>,<uuid>]	VMware hypervisor BIOS UUID.	String	url - VMware service URL uuid - VMware hypervisor host name
vmware.hv.memory.size.ballooned[<url>,<uuid>]	VMware hypervisor vendor name.	String	url - VMware service URL uuid - VMware hypervisor host name
vmware.hv.memory.used[<url>,<uuid>]	VMware hypervisor ballooned memory size (bytes).	Integer	url - VMware service URL uuid - VMware hypervisor host name
vmware.hv.network.in[<url>,<uuid>,<mode>]	VMware hypervisor used memory size (bytes).	Integer	url - VMware service URL uuid - VMware hypervisor host name
vmware.hv.network.out[<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)

Key

	VMware hypervisor network output statistics (bytes per second).	Integer ²	url - VMware service URL uuid - VMware hypervisor host name mode - bps (default)	
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)	Available since Zabbix versions 2.2.9, 2.4.4
vmware.hv.sensor.health.state[<url>,<uuid>]	VMware hypervisor health state rollout sensor.	Integer: 0 - gray; 1 - green; 2 - yellow; 3 - red	url - VMware service URL uuid - VMware hypervisor host name	Available since Zabbix 2.2.16, 3.0.6, 3.2.2
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 host system overall status property since Zabbix 2.2.16, 3.0.6, 3.2.2
vmware.hv.uptime[<url>,<uuid>]	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	
vmware.version[<url>]	VMware service version.	String	url - VMware service URL	
vmware.vm.cluster.name[<url>,<uuid>]				

Key				
	VMware virtual machine name.	String	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.ready[<url>,<uuid>]	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	Available since Zabbix version 3.0.0
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.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.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.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.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>]	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.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

Key

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.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)	Available since Zabbix versions 2.2.9, 2.4.4
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.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.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.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 - to-tal/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]`

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.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.

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](#).

* Name	Log item		
Type	Zabbix agent (active)		
* Key	log[/var/log/syslog,error]	Select	
Type of information	Log		
* Update interval	30s		
* History storage period	3600		
Log time format	ppppddphh:mm:ss		

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: <code>log[/var/log/syslog,error]</code>. 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	Select: For log[] or logrt[] items - Log; For log.count[] or logrt.count[] items - Numeric (unsigned). If optionally using the output parameter, you may 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: <pre>" 23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211)."</pre> 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 <code>"pppppp:yyyyMMdd:hhmmss"</code>. 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)
 - * On UNIX/GNU/Linux systems it is assumed that the file systems where log files are stored report inode
 - * 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 ta
 - * The inode numbers, file indexes and MD5 sums are internally collected by Zabbix agent. They are not t
 - * Do not modify the last modification time of log files with 'touch' utility, do not copy a log file wi
 - * If there are several matching log files for 'logrt[]' item and Zabbix agent is following the most r
- * 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) ar
- * Whenever the log file becomes smaller than the log size counter known by the agent, the counter is reset
- * If there are several matching files with the same last modification time in the directory, then the agen
- * 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.
- * To find the required string Zabbix will process 10 times more new lines than set in `MaxLinesPerSecond`.
- * Additionally, `log` and `log.count` values are always limited to 50% of the agent send buffer size, even if
- * In the absence of log items all agent buffer size is used for non-log values. When log values come in the
- * For log file records longer than 256kB, only the first 256kB are matched against the regular expression
- * Special note for "\" path separators: if `file_format` is `"file\log"`, then there should not be a `"file"` d
- * Regular expressions for `'logrt'` are supported in filename only, directory regular expression matching
- * On UNIX platforms a `'logrt[]'` item becomes `NOTSUPPORTED` if a directory where the log files are expected
- * On Microsoft Windows, if a directory does not exist the item will not become `NOTSUPPORTED` (for example,
- * An absence of log files for `'logrt[]'` item does not make it `NOTSUPPORTED`. Errors of reading log files
- * Zabbix agent log file can be helpful to find out why a `'log[]'` or `'logrt[]'` item became `NOTSUPPORTED`

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 subgroup 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
```

The reason why Zabbix will return only the number is because 'output' here is defined by **\1** referring to the first and only subgroup of interest: **([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 4*'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.

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). Behaviour 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 behaviour 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 you can create calculations on the basis of other items.

Thus, calculated items are a way of creating virtual data sources. The values will be periodically calculated based on an arithmetical expression. All calculations are done by the Zabbix server - nothing related to calculated items is performed on Zabbix agents or proxies.

The resulting data will be stored in the Zabbix database as for any other item - this means storing both history and trend values for fast graph generation. Calculated items may be used in trigger expressions, 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 formula in any way.

The correct syntax of a simple formula is:

```
func(<key>|<hostname:key>,<parameter1>,<parameter2>,...)
```

Where:

ARGUMENT	DEFINITION
func	One of the functions supported in trigger expressions: last, min, max, avg, count, etc
key	The key of another item whose data you want to use. It may be defined as key or hostname:key . <i>Note:</i> Putting the whole key in double quotes ("...") is strongly recommended to avoid incorrect parsing because of spaces or commas within the key. If there are also quoted parameters within the key, those double quotes must be escaped by using the backslash (\). See Example 5 below.
parameter(s)	Function parameter(s), if required.

Note:

All items that are referenced from the calculated item formula must exist and be collecting data (exceptions in **functions and unsupported items**). Also, if you change the item key of a referenced item, you have to manually update any formulas using that key.

Attention:

User macros in the formula will be expanded if used to reference a function 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. Note that the syntax is slightly different, however 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.

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)*.

A calculated item may become unsupported in several cases:

1. referenced item(s)
 - is not found
 - is disabled
 - belongs to a disabled host
 - is not supported (see exceptions in **functions and unsupported items**, **Expressions with unsupported items and unknown values** and **Operators**)
2. no data to calculate a function
3. division by zero
4. incorrect syntax used

Support for calculated items was introduced in Zabbix 1.8.1.

Starting from Zabbix 3.2 calculated items in some cases may involve unsupported items as described in **functions and unsupported items**, **Expressions with unsupported items and unknown values** and **Operators**.

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]",600)
```

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]"))
```

Example 5

Using aggregated items correctly within a calculated item.

Take note of how double quotes are escaped within the quoted key:

```
last("grpsum[\"video\\\", \"net.if.out[eth0,bytes]\\\", \"last\\\"]") / last("grpsum[\"video\\\", \"nginx_stat.sh[act
```

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 (since Zabbix 2.4.0).

To use this item, choose the **Zabbix internal** item type.

Note:

Internal checks are processed by Zabbix pollers.

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[history]			

Key			
	Number of values stored in the HISTORY table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! <i>(not supported on proxy)</i>
zabbix[history_log]			
	Number of values stored in the HIS-TORY_LOG table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! This item is supported since Zabbix 1.8.3. <i>(not supported on proxy)</i>
zabbix[history_str]			
	Number of values stored in the HIS-TORY_STR table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! <i>(not supported on proxy)</i>
zabbix[history_text]			
	Number of values stored in the HIS-TORY_TEXT table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! This item is supported since Zabbix 1.8.3. <i>(not supported on proxy)</i>
zabbix[history_uint]			
	Number of values stored in the HIS-TORY_UINT table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! This item is supported since Zabbix 1.8.3. <i>(not supported on proxy)</i>

Key			
zabbix[host,,items]	Number of enabled items (supported and not supported) on the host.	Integer.	This item is supported since Zabbix 3.0.0.
zabbix[host,,items_unsupported]	Number of enabled unsupported items on the host.	Integer.	This item is supported since Zabbix 3.0.0.*
zabbix[host,,maintenance]	Current maintenance status of a host.	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. This item is supported since Zabbix 2.4.0.
zabbix[host,discovery,interfaces]	Details of all configured interfaces of the host in Zabbix frontend.	JSON object.	This item can be used in low-level discovery . This item is supported since Zabbix 3.4.0. (<i>not supported on proxy</i>)
zabbix[host,<type>,available]			

Key			
	Availability of a particular type of checks on the host. The value of this item corresponds to availability icons in the host list.	0 - not available, 1 - available, 2 - unknown.	Valid types are: <i>agent, snmp, ipmi, jmx</i> The item value is calculated according to configuration parameters regarding host unreachability/unavailability . This item is supported since Zabbix 2.0.0.
zabbix[hosts]	Number of monitored hosts.	Integer.	This item is supported since Zabbix 2.2.0.
zabbix[items]	Number of enabled items (supported and not supported).	Integer.	
zabbix[items_unsupported]	Number of not supported items.	Integer.	
zabbix[java,,<param>]	Information about Zabbix Java gateway.	If <param> is <i>ping</i> , "1" is returned. Can be used to check Java gateway availability using <code>nodata()</code> trigger function. If <param> is <i>version</i> , version of Java gateway is returned. Example: "2.0.0".	Valid values for param are: <i>ping, version</i> Second parameter must be empty and is reserved for future use. This item is supported since Zabbix 2.0.0.

Key

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>]

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> is specified to be 6), such an item will turn into unsupported state. Minimum and maximum refers to the usage percentage for a single process. So if in a group of 3 pollers usage percentages per process were 2, 18 and 66, min would return 2 and max would return 66.

Processes report what they are doing in shared memory and the self-monitoring process summarizes that data

Percentage of time. Float.

Supported **types** of **server** **processes**:
alert manager, alerter, configuration syncer, discoverer, escalator, history syncer, house-keeper, http poller, icmp pinger, ipmi manager, ipmi poller, java poller, lld manager, lld worker, poller, pre-processing manager, preprocess-ing worker, proxy poller, self-monitoring, snmp trapper, task manager, timer, trapper, unreachable poller, vmware collector

Supported **types** of **proxy** **processes**:
configuration syncer, data sender, discoverer, heartbeat sender, history syncer, house-keeper, http poller, icmp pinger, ipmi manager, ipmi poller, java poller, poller, self-monitoring, snmp trapper, task manager,

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

Example:
=> zab-
bix[proxy,"Germany",lastacc

fuzzytime()
trigger
function can
be used to
check
availability
of proxies.
Since Zabbix
2.4.0 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.

This item is
supported
since Zabbix
2.2.0.
(*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. Parameters from and to are supported since Zabbix 1.8.3.
zabbix[rcache,<cache>,<mode>]	Availability statistics of Zabbix configuration cache.	Integer (for size); float (for percentage).	cache: <i>buffer</i> Valid modes are: <i>total</i> - total size of buffer <i>free</i> - size of free buffer <i>pfree</i> - percentage of free buffer <i>used</i> - size of used buffer <i>pused</i> - percentage of used buffer <i>pused</i> 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 <i>Reports</i> → System information . This item is supported since Zabbix 1.6.2.
zabbix[stats,<ip>,<port>]			

Key			
	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 4.0.5.
zabbix[stats,<ip>,<port>,queue,<from>,<to>]			

Key			
	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 4.0.5.
zabbix[trends]	Number of values stored in the TRENDS table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! (<i>not supported on proxy</i>)
zabbix[trends_uint]			

Key			
	Number of values stored in the TRENDS_UINT table.	Integer.	Do not use if MySQL InnoDB, Oracle or PostgreSQL is used! This item is supported since Zabbix 1.8.3. <i>(not supported on proxy)</i>
zabbix[triggers]	Number of enabled triggers in Zabbix database, with all items enabled on enabled hosts.	Integer.	<i>(not supported on proxy)</i>
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: <i>total</i> - total size of buffer <i>free</i> - size of free buffer <i>pfree</i> - percentage of free buffer <i>used</i> - size of used buffer <i>pused</i> - percentage of used buffer This item is supported since Zabbix 2.2.0. <i>(not supported on proxy)</i>
zabbix[vcache,cache,<parameter>]			

Key			
	Effectiveness statistics of Zabbix value cache.	Integer. With the <i>mode</i> parameter: 0 - normal mode, 1 - low memory mode	Valid parameter values are: <i>requests</i> - total number of requests <i>hits</i> - number of cache hits (history values taken from the cache) <i>misses</i> - number of cache misses (history values taken from the database) <i>mode</i> - value cache operating mode This item is supported since Zabbix 2.2.0 and the <i>mode</i> parameter since Zabbix 3.0.0. (<i>not supported on proxy</i>) You may use this key with the <i>Change per second</i> preprocessing step in order to get values per second statistics.
zabbix[vmware,buffer,<mode>]			

zabbix[wcache,<cache>,<mode>]	Availability statistics of Zabbix vmware cache.			Integer (for size); float (for percentage).	Valid modes are: <i>total</i> - total size of buffer <i>free</i> - size of free buffer <i>pfree</i> - percentage of free buffer <i>used</i> - size of used buffer <i>pusd</i> - percentage of used buffer This item is supported since Zabbix 2.2.0.
	Statistics and availability of Zabbix write cache.				Specifying <cache> is mandatory.
	Cache values	Mode all (<i>default</i>)	Total number of values processed by Zabbix server or Zabbix proxy, except unsupported items.	Integer.	Counter. You may use this key with the <i>Change per second</i> 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.
		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. <i>Not supported</i> mode is supported since Zabbix 1.8.6.
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.	
	pusd	Percentage of used history buffer.	Float.	<i>pusd</i> 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. <i>Index</i> cache is supported since Zabbix 3.0.0.
	free	Size of free history index buffer.	Integer.	
	total	Total size of history index buffer.	Integer.	
	used	Size of used history index buffer.	Integer.	
	pusd	Percentage of used history index buffer.	Float.	<i>pusd</i> mode is supported since Zabbix 4.0.0.

Key					
	trend	pfree (default)	Percentage of free trend cache.	Float.	Trend cache stores aggregate for the current hour for all items that receive data. (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)
		puused	Percentage of used trend buffer.	Float.	(not supported on proxy)
					puused 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.

Attention:

The minimum supported libssh2 library version is 1.0.0.

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 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 the *zabbix* user account with a home directory in not very well-known places (as for system accounts). For example, for CentOS it's */var/lib/zabbix*, for Debian it's */var/run/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):
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 multi lined.

* Name	SSH test check (whithout passphrase)
Type	SSH agent
* Key	ssh.run[clear]
* Host interface	127.0.0.1 : 10051
Authentication method	Public key
* User name	root
* Public key file	id_rsa.pub
* Private key file	id_rsa
Key passphrase	
* Executed script	service mysql-server status
Type of information	Numeric (unsigned)
Units	
* Update interval	30s

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 <i>Authentication method</i> is "Public key". Required	Example: <i>id_rsa.pub</i> - default public key file name generated by a command ssh-keygen
Private key file	File name of private key if <i>Authentication method</i> is "Public key". Required	Example: <i>id_rsa</i> - default private key file name
Password or Key passphrase	Password to authenticate or Passphrase if it was used for the private key	Leave the <i>Key passphrase</i> 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: <i>date +%s</i> <i>service mysql-server status</i> <i>ps auxww grep httpd wc -l</i>

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 multi lined.

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	Comments
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 Aggregate checks

Overview

In aggregate checks Zabbix server collects aggregate information from items by doing direct database queries.

Aggregate checks do not require any agent running on the host being monitored.

Syntax

The syntax of the aggregate item key is:

```
groupfunc["host group","item key",itemfunc,timeperiod]
```

Supported group functions (groupfunc) are:

Group function	Description
<i>grpavg</i>	Average value
<i>grpmax</i>	Maximum value
<i>grpmin</i>	Minimum value
<i>grpsum</i>	Sum of values

Multiple host groups may be included by inserting a comma-delimited array. Specifying a parent host group will include the parent group and all nested host groups with their items.

All items that are referenced from the aggregate item key must exist and be collecting data. Only enabled items on enabled hosts are included in the calculations.

Attention:

The key of the aggregate item must be updated manually, if the item key of a referenced item is changed.

Supported item functions (itemfunc) are:

Item function	Description
<i>avg</i>	Average value
<i>count</i>	Number of values
<i>last</i>	Last value
<i>max</i>	Maximum value
<i>min</i>	Minimum value
<i>sum</i>	Sum of values

The **timeperiod** parameter specifies a time period of latest collected values. **Supported unit symbols** can be used in this parameter for convenience, for example '5m' (minutes) instead of '300' (seconds) or '1d' (day) instead of '86400' (seconds).

Warning:

An amount of values (prefixed with #) is not supported in the timeperiod.

Timeperiod is ignored by the server if the third parameter (item function) is *last* and can thus be omitted:

```
groupfunc["host group","item key",last]
```

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 item 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 checks:

Example 1

Total disk space of host group 'MySQL Servers'.

```
grpsum["MySQL Servers","vfs.fs.size[/,total]",last]
```

Example 2

Average processor load of host group 'MySQL Servers'.

```
grpavg["MySQL Servers","system.cpu.load[,avg1]","last"]
```

Example 3

5-minute average of the number of queries per second for host group 'MySQL Servers'.

```
grpavg["MySQL Servers",mysql.qps,avg,5m]
```

Example 4

Average CPU load on all hosts in multiple host groups.

```
grpavg[["Servers A","Servers B","Servers C"],system.cpu.load,last]
```

13 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

* Name	Trapper item
Type	Zabbix trapper
* Key	trap
Type of information	Text
* History storage period	3600
Allowed hosts	

All mandatory input fields are marked with a red asterisk.

The fields that require specific information for trapper items are:

Type

Key

Type of information

Select **Zabbix trapper** here.

Enter a key that will be used to recognize the item when sending in data.

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*:

▼ ☐ HOST	NAME ▼	LAST CHECK	LAST VALUE	CHANGE
▼ New host	- other - (2 items)			
☐	Trapper item	2015-08-11 18:50:53	test value	History

14 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 \
-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 \
-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.

Hosts

- Host
- Templates
- IPMI
- Macros
- Host inventory
- Encryption

* Host name

Visible name

* Groups

In groups

Other groups

Zabbix servers

Anna group
Annas group
bypass
calendarian
data poolers
Discovered hosts
group 1
group 2
Hypervisors
Linux servers

New group

* At least one interface must exist.

Agent interfaces

IP address	DNS name	Connect to	Port	Default
127.0.0.1		☒ IP ☐ DNS	10050	☒ Remove
Add				

SNMP interfaces [Add](#)

JMX interfaces

IP address	DNS name	Connect to	Port	Default
127.0.0.1		☒ IP ☐ DNS	12345	☒ Remove
Add				

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"]`.

<i>JMX endpoint</i>	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.
<i>User name</i>	Specify the user name, if you have configured authentication on your Java application. User macros are supported.
<i>Password</i>	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]
```

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.

Custom endpoint example 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.

The screenshot shows the Zabbix web interface for configuring a host named 'jboss'. The 'Host' tab is selected, and the 'JMX' interface type is chosen. The 'Agent interfaces' section shows a default interface with IP 127.0.0.1 and port 10050. The 'JMX interfaces' section, highlighted with a red box, shows a new interface with IP 127.0.0.1 and port 9999. A red arrow points to the port field in the JMX interface section.

As we know that this version of JBoss uses the the JBoss Remoting protocol instead of RMI, we may mass update the JMX endpoint parameter in our JMX template accordingly:

```
service:jmx:remoting-jmx://{HOST.CONN}:{HOST.PORT}
```


Items

All templates / Template App Generic Java JMX-remoting Applications 8 Items 55 Triggers 26

Type ☐ Original

JMX endpoint ☒ `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$1.run(ZabbixJMXConnectorFactory.java:76) ~[zabbix-java-gateway-3.4.2.jar:na]
    at java.util.concurrent.Executors$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$1.run(ZabbixJMXConnectorFactory.java:76) ~[zabbix-java-gateway-3.4.2.jar:na]
    at java.util.concurrent.Executors$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:

Latest data				
	Filter ▼			
▼ ☐ Name ▲		Last check	Last value	Change
▼ Classes (3 items)				
☐ cl Loaded Class Count		2017-11-07 14:08:10	7866	+2
☐ cl Total Loaded Class Count		2017-11-07 14:08:09	7865	+2
☐ cl Unloaded Class Count		2017-11-07 14:08:10	0	
▼ Compilation (2 items)				
☐ comp Accumulated time spent in compilation		2017-11-07 14:08:10	46s 759ms	+1s 440ms
☐ comp Name of the current JIT compiler		2017-11-07 14:00:39	HotSpot 64-Bit Tiered Compilers	
▼ Garbage Collector (4 items)				
☐ gc Copy accumulated time spent in collection		2017-11-07 14:08:09	0	
☐ gc Copy number of collections per second		2017-11-07 14:08:09	0	
☐ gc MarkSweepCompact accumulated time spent in collection		2017-11-07 14:08:10	372ms	
☐ gc MarkSweepCompact number of collections per second		2017-11-07 14:08:10	0	
▼ Memory (6 items)				
☐ mem Heap Memory committed		2017-11-07 14:08:10	1.23 GB	
☐ mem Heap Memory max		2017-11-07 14:00:39	1.23 GB	
☐ mem Heap Memory used		2017-11-07 14:08:09	271.07 MB	+4.01 MB
☐ mem Non-Heap Memory committed		2017-11-07 14:08:10	66.39 MB	+384 KB
☐ mem Non-Heap Memory used		2017-11-07 14:08:10	59.5 MB	+128.1 KB
☐ mem Object Pending Finalization Count		2017-11-07 14:08:10	0	
▼ Memory Pool (6 items)				
☐ mp Code Cache committed		2017-11-07 14:08:09	12.31 MB	+128 KB
☐ mp Code Cache max		2017-11-07 14:00:40	240 MB	
☐ mp Code Cache used		2017-11-07 14:08:09	12.23 MB	+145.44 KB
☐ mp Tenured Gen committed		2017-11-07 14:08:10	869.38 MB	
☐ mp Tenured Gen max		2017-11-07 14:00:40	869.38 MB	
☐ mp Tenured Gen used		2017-11-07 14:08:09	32.25 MB	

15 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.

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> yum -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> yum install mysql-connector-odbc
```

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

```
zypper in MySQL-connector-odbc
```

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
<i>mysql</i>	Database driver name.
<i>Description</i>	Database driver description.
<i>Driver</i>	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
<i>test</i>	Data source name (DSN).
<i>Description</i>	Data source description.
<i>Driver</i>	Database driver name - as specified in odbcinst.ini
<i>Server</i>	Database server IP/DNS.
<i>User</i>	Database user for connection.
<i>Password</i>	Database user password.
<i>Port</i>	Database connection port.
<i>Database</i>	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](#).

* Name

Type

* Key

User name

Password

* SQL query

select count(*) from hosts

Type of information

All mandatory input fields are marked with a red asterisk.

Specifically for database monitoring items you must enter:

Type	Select <i>Database monitor</i> here.
Key	Enter db.odbc.select [unique_description,data_source_name] The unique description will serve to identify the item in triggers etc. The data source name (DSN) must be set as specified in <i>odbc.ini</i> .
User name	Enter the database user name (optional if user is specified in <i>odbc.ini</i>)
Password	Enter the database user password (optional if password is specified in <i>odbc.ini</i>)
SQL query	Enter the SQL query
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 <i>type of information</i> the item will turn unsupported.

Important notes

- 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 query must return one value only.
- 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.
- 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.
- See also **known issues** for ODBC checks

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 executing

Native error code	Native error message
SQLState	

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**:

```
# yum 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_SOCKET]
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**:

```
# yum 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
# Booleans as Char
# Booleans are mapped to SQL_CHAR, otherwise to SQL_BIT.
BooleansAsChar = Yes
# SSL mode
```

```
SSLmode = Yes
# Send to backend on connection
ConnSettings =

3 Recommended UnixODBC settings for Oracle
```

Installation

Please refer to [Oracle documentation](#) for all the necessary instructions.

For some additional information please refer to: [Installing unixODBC](#).

4 Recommended UnixODBC settings for MSSQL

Installation

*** Red Hat Enterprise Linux/CentOS**:

```
# yum -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
```

16 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 999 (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

The screenshot shows the 'Item Preprocessing' configuration form in Zabbix. The form has a tabbed interface with 'Item' selected. The fields are as follows:

- Name** (mandatory, marked with a red asterisk): Apache server status
- Type**: Zabbix agent
- Key** (mandatory, marked with a red asterisk): web.page.get[127.0.0.1,/server-status]
- Host interface** (mandatory, marked with a red asterisk): 127.0.0.1 : 10050
- Type of information**: Text
- Update interval** (mandatory, marked with a red asterisk): 30s

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**.

The screenshot shows the 'Item Preprocessing' configuration form in Zabbix for a dependent item. The form has a tabbed interface with 'Item' selected. The fields are as follows:

- Name** (mandatory, marked with a red asterisk): Apache server uptime
- Type**: Dependent item
- Key** (mandatory, marked with a red asterisk): apache.server.uptime
- Master item** (mandatory, marked with a red asterisk): Apache server status: web.page.get[127.0.0.1,/server-status]
- Type of information**: Text

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.

Item

Preprocessing

Preprocessing steps

Name

Parameters

Regular expression

<dt>Server uptime: (.*)<Vdt>

1

Add

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 is to use the wizard in the item list:

Wizard

Name ▲

...

Apache server status

...

...

ITEM "APACHE SERVER STATUS"

Create trigger

Edit trigger

Create dependent item

Display

In the item list dependent items are displayed with their master item name as prefix.

Wizard	Name ▲	Triggers	Key
☐	...	Apache server status	web.page.get[192.168.3.31/server-status]
☐	...	Apache server status: Apache server uptime	apache.server.uptime

If a master item is deleted, so are all its dependent items.

17 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

Item Preprocessing

Name

HTTP agent item

Type

HTTP agent

Key

http_value_search

Select

URL

http://localhost:9200/_search

Parse

Query fields

Name	Value	
scroll	10s	Remove

Add

Request type

POST

Timeout

3s

Request body type

Raw data

JSON data

XML data

Request body

```
{
  "query": {
    "bool": {
      "must": [
        {
          "match": {
            "itemid": 28275
          }
        }
      ]
    }
  }
}
```

Headers

Name	Value	
name	value	Remove

Add

Required status codes

200

Follow redirects

☐

Retrieve mode

Body

Headers

Body and headers

Convert to JSON

☐

HTTP proxy

[protocol://[user[:password]@]proxy.example.com[:port]]

HTTP authentication

None

SSL verify peer

☐

SSL verify host

☐

SSL certificate file

SSL key file

SSL key password

Host interface

127.0.0.1 : 10050

Type of information

Numeric (unsigned)

Units

Update interval

30s

Custom intervals

Type	Interval	Period
Flexible	Scheduling	50s
		1-7,00:00-24:00

Add

History storage period

90d

Trend storage period

365d

Show value

As is

show value mapp

Enable trapping

☒

Allowed hosts

104.24.103.152

New application

Applications

-None-

CPU

Filesystems

General

Memory

Network interfaces

OS

Performance

Processes

Security

Populates host inventory field

-None-

Description

Enabled

☒

Add

Cancel

All mandatory input fields are marked with a red asterisk.

The fields that require specific information for HTTP items are:

Type	Select HTTP agent here.
Key	Enter a unique item key.
URL	URL to connect to and retrieve data. For example: https://www.google.com http://www.zabbix.com/download Domain names can be specified in Unicode characters. They are automatically punycode-converted to ASCII when executing the HTTP check. The <i>Parse</i> button can be used to separate optional query fields (like ?name=Admin&password=mypassword) from the URL, moving the attributes and values into <i>Query fields</i> for automatic URL-encoding. Limited to 2048 characters. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros.
Query fields	This sets the CURLOPT_URL cURL option. 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}, user macros, low-level discovery macros.
Request type	This sets the CURLOPT_URL cURL option.
Timeout	Select request method type: <i>GET</i> , <i>POST</i> , <i>PUT</i> or <i>HEAD</i> Zabbix will not spend more than the set amount of time on processing the URL (maximum is 1 minute). 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.
Request body type	This sets the CURLOPT_TIMEOUT cURL option. 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"
Request body	<i>Note that selecting XML data requires libxml2.</i> Enter the request body. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros.

<i>Headers</i>	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}, user macros, low-level discovery macros. This sets the CURLOPT_HTTPHEADER cURL option.
<i>Required status codes</i>	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.
<i>Follow redirects</i>	Mark the checkbox to follow HTTP redirects. This sets the CURLOPT_FOLLOWLOCATION cURL option.
<i>Retrieve mode</i>	Select the part of response that must be retrieved: Body - body only Headers - headers only Body and headers - body and headers
<i>Convert to JSON</i>	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: <pre>{ "header": { "<key>": "<value>", "<key2>": "<value>" }, "body": <body> }</pre>
<i>HTTP proxy</i>	You can specify an HTTP proxy to use, using the format [protocol://] [username[:password]@]proxy.mycompany.com[:port]. The optional protocol:// prefix may be used to specify alternative proxy protocols (the protocol prefix support was added in cURL 7.21.7). With no protocol specified, the proxy will be treated as an HTTP proxy. By default, 1080 port will be used. If specified, the proxy will overwrite proxy related environment variables like http_proxy, HTTPS_PROXY. If not specified, the proxy will not overwrite proxy-related environment variables. The entered value is passed on "as is", no sanity checking takes place. You may also enter a SOCKS proxy address. If you specify the wrong protocol, the connection will fail and the item will become unsupported. <i>Note</i> 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}, user macros, low-level discovery macros. This sets the CURLOPT_PROXY cURL option.

HTTP authentication

SSL verify peer

SSL verify host

SSL certificate file

SSL key file

SSL key password

Enable trapping

Allowed hosts

Authentication type:

None - no authentication used.

Basic authentication - basic authentication is used.

NTLM authentication - NTLM ([Windows NT LAN Manager](#)) 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.

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.

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.

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. 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}, user macros, low-level discovery macros.

This sets the [CURLOPT_SSLCERT](#) cURL option.

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 Zabbix server or proxy configuration parameter SSLKeyLocation. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros.

This sets the [CURLOPT_SSLKEY](#) cURL option.

SSL private key file password.

Supported macros: user macros, low-level discovery macros.

This sets the [CURLOPT_KEYPASSWD](#) cURL option.

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.

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": "DnF1ZXJ5VGhlbkZldGNoBQAAAAAAAAAAkF1lRMlZBWS1UU1pxTmdEeGVwQjRBTfEAAAAAAAAAJRZZUTJWQVktVFN",
  "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:

Item
Preprocessing

* Name

Check Zabbix API version

Type

HTTP agent ▼

* Key

check_zabbix_api_apiinfo.version

* URL

http://zabbix-web-apache-mysql/api_jsonrpc.php

Query fields

Name	Value
Add	

Request type

POST ▼

Timeout

3s

Request body type

Raw data JSON data XML data

Request body

```
{
  "jsonrpc": "2.0",
  "method": "apiinfo.version",
  "params": [],
  "id": 1
}
```

Headers

Name	Value
Content-Type	⇒ application/json-rpc
Add	

Required status codes

Follow redirects

☐

Retrieve mode

Body Headers Body and headers

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:

Item
Preprocessing

Preprocessing steps

Name	Parameters
Regular expression ▼	HTTPV1.1 ([0-9]+) \1
Add	

- Checking the result in *Latest data*:

Latest data

Filter ▲

Host groups

type here to search

Select

Hosts

Zabbix server ✕ nginx ✕

Select

Application

type here to search

Select

Name

Check Zabbix API

Show items without data

☐

Show details

☐

Apply

Reset

▼ ☐ Host	Name ▲	Last check	Last value	Change
▼ Zabbix server	- other - (1 item)			
☐	Check Zabbix API version	2018-05-16 23:50:34	OK (200)	Graph

Example 4

Retrieving weather information by connecting to the Openweathermap public service.

- Configure a master item for bulk data collection in a single JSON:

Item
Preprocessing

Parent items
Template Weather

* Name
Get weather

Type
HTTP agent

* Key
get_weather.http

* URL
http://api.openweathermap.org/data/2.5/weather

Query fields

Name	Value
units	metric
lat	{SLAT}
lon	{SLON}
APPID	{SWEATHER_APIKEY}
lang	{SWEATHER_LANG}

Add

Request type
GET

Timeout
3s

Request body type
Raw data
JSON data
XML data

Request body

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:

The screenshot shows the 'Preprocessing' configuration interface. It includes the following fields:

- Name:** Humidity
- Type:** Dependent item
- Key:** humidity
- Master item:** Template Weather: Get weather
- Type of information:** Numeric (float)

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

- Sample dependent item value preprocessing with JSONPath:

The screenshot shows the 'Preprocessing' configuration interface with a table of preprocessing steps:

Preprocessing steps	Name	Parameters	Action
...	JSON Path	\$.body.main.humidity	Remove

Below the table is an 'Add' button.

- 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:

Item

Preprocessing

* Name

Get NGINX status page

Type

HTTP agent ▼

* Key

get_nginx

* URL

http://{HOST.CONN}/nginx_status

Query fields

Name

Value

name

⇒

value

Add

Request type

GET ▼

Timeout

3s

Request body type

Raw data

JSON data

XML data

Request body

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:

Item **Preprocessing**

* Name

Type

* Key

* Master item

Type of information

- Sample dependent item value preprocessing with regular expression:

Item **Preprocessing**

Preprocessing steps

Name	Parameters
Regular expression	server accepts handled requests\s+([0-9]+) ([0-9]+) ([0-9]+)
Change per second	\3

[Add](#)

- Check the complete result from stub module in *Latest data*:

Host	Name	Last check	Last value
nginx	Nginx (8 Items)		
☐	Accepted client connections	2018-05-18 17:54:53	568
☐	Active connections	2018-05-18 17:54:53	1
☐	Client requests per second	2018-05-18 17:54:53	0 rps
☒	Get Nginx stub status	2018-05-18 17:54:53	HTTP/1.1 200 OK Se...
☐	Handled connections per second	2018-05-18 17:54:53	0
☐	Reading	2018-05-18 17:54:53	0
☐	Waiting	2018-05-18 17:54:53	0
☐	Writing	2018-05-18 17:54:53	1

18 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.

Configuration

Providing you have the HTTP master item configured, you need to create a **dependent item** that uses 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*)

Item

Preprocessing

Preprocessing steps

1:

Prometheus pattern

cpu_usage_system{cpu="cpu-total"}

<label name>

Add

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.	<pre>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</pre>
Output	Define label name (optional). In this case the value corresponding to the label name is returned. This field is only available for the <i>Prometheus pattern</i> option.	

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	
Query Prometheus server target	Plain text in Prometheus exposition format
Return instant vector	Metric or label value (Prometheus pattern) Array of metrics for single value in JSON (Prometheus to JSON)

PromQL instant vector selector	Zabbix Prometheus preprocessing
Label, !=, =~, !~	=, =~
match- ing op- er- a- tors	
Regular	PCRE
expression used in la- bel or met- ric name match- ing	
Comparison	Only == (equal) is supported for value filtering
Similarities Selecting by met- ric name that equals string	<metric name> or {__name__=<metric name>}
Selecting by met- ric name that matches reg- u- lar ex- pres- sion	{__name__=~<regex>}
Selecting by <la- bel name> value that equals string	{<label name>=<label value>,...}

PromQL instant vector selector	Zabbix Prometheus preprocessing
<pre> Select <label name>=~"<regex>",...} by <label name> value that matches reg- u- lar ex- pres- sion Select <time__=~".*"> == <value> by value that equals string </pre>	<pre> {<label name>=~"<regex>",...} {__name__=~".*" } == <value> </pre>

3 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 trends into the database (where frontend can find them) in these situations:

- a new hour has started and server receives a new value for the item;
- a new hour is about to end in less than 5 minutes (no new values)
- 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.

4 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.

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.
Command	Parameters are given when configuring the item. Command to be executed to evaluate value of the key. <i>For flexible user parameters only:</i> 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

Restart Zabbix agent.

Agent will reload configuration file.

Test this 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.

5 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.

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 (since 4.2.3) 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**.

```
/*
** Zabbix
** Copyright (C) 2001-2016 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 "sysinc.h"
####include "module.h"

/* the variable keeps timeout setting for item processing */
static int  item_timeout = 0;

/* module SHOULD define internal functions as static and use a naming pattern different from Zabbix intern
/* symbols (zbx_*) and loadable module API functions (zbx_module_*) to avoid conflicts
static int  dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result);
static int  dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result);
static int  dummy_random(AGENT_REQUEST *request, AGENT_RESULT *result);

static ZBX_METRIC keys[] =
/* KEY          FLAG          FUNCTION    TEST PARAMETERS */
{
    {"dummy.ping",      0,        dummy_ping, NULL},
    {"dummy.echo",      CF_HAVEPARAMS,  dummy_echo, "a message"},
    {"dummy.random",    CF_HAVEPARAMS,  dummy_random, "1,1000"},
    {NULL}
};

/*****
*
* Function: zbx_module_api_version
*
*****/
```



```

* Purpose: returns version number of the module interface *
* *
* Return value: ZBX_MODULE_API_VERSION - version of module.h module is *
*             compiled with, in order to load module successfully Zabbix *
*             MUST be compiled with the same version of this header file *
* *
*****/
int zbx_module_api_version(void)
{
    return ZBX_MODULE_API_VERSION;
}

/*****
* *
* Function: zbx_module_item_timeout *
* *
* Purpose: set timeout value for processing of items *
* *
* Parameters: timeout - timeout in seconds, 0 - no timeout set *
* *
*****/
void zbx_module_item_timeout(int timeout)
{
    item_timeout = timeout;
}

/*****
* *
* Function: zbx_module_item_list *
* *
* Purpose: returns list of item keys supported by the module *
* *
* Return value: list of item keys *
* *
*****/
ZBX_METRIC *zbx_module_item_list(void)
{
    return keys;
}

static int dummy_ping(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    SET_UI64_RESULT(result, 1);

    return SYSINFO_RET_OK;
}

static int dummy_echo(AGENT_REQUEST *request, AGENT_RESULT *result)
{
    char *param;

    if (1 != request->nparam)
    {
        /* set optional error message */
        SET_MSG_RESULT(result, strdup("Invalid number of parameters."));
        return SYSINFO_RET_FAIL;
    }

    param = get_rparam(request, 0);

    SET_STR_RESULT(result, strdup(param));
}

```

```

    return SYSINFO_RET_OK;
}

/*****
 *
 * Function: dummy_random
 *
 * Purpose: a main entry point for processing of an item
 *
 * Parameters: request - structure that contains item key and parameters
 *              request+key - item key without parameters
 *              request+nparam - number of parameters
 *              request+timeout - processing should not take longer than
 *                              this number of seconds
 *              request+params[N-1] - pointers to item key parameters
 *
 *              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.
 *****/
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 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.

6 Windows performance counters

Overview

You can effectively monitor Windows performance counters using the *perf_counter[]* key.

For example:

```
perf_counter["\Processor(0)\Interrupts/sec"]
```

or

```
perf_counter["\Processor(0)\Interrupts/sec", 10]
```

For more information on using this key or its English-only equivalent *perf_counter_en*, see [Windows-specific item keys](#).

In order to get a full list of performance counters available for monitoring, you may run:

```
typeperf -qx
```

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, starting with Zabbix 4.2.7 you can use the *perf_counter_en* key or *PerfCounterEn* configuration parameter which allow to use English names regardless of system locale.

To find out the numeric equivalents, run **regedit**, then find *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Perflib\00*

The registry entry contains information like this:

```
1
1847
2
System
4
Memory
6
% Processor Time
10
File Read Operations/sec
12
File Write Operations/sec
14
File Control Operations/sec
16
File Read Bytes/sec
18
File Write Bytes/sec
....
```

Here you can find the corresponding numbers for each string part of the performance counter, like in '*System\% Processor Time*':

System → 2

% Processor Time → 6

Then you can use these numbers to represent the path in numbers:

\2\6

Performance counter parameters

You can deploy some PerfCounter parameters for the monitoring of Windows performance counters.

For example, you can add these to the Zabbix agent configuration file:

```
PerfCounter=UserPerfCounter1,"\Memory\Page Reads/sec",30  
or  
PerfCounter=UserPerfCounter2,"4\24",30
```

With such parameters in place, you can then simply use *UserPerfCounter1* or *UserPerfCounter2* as the keys for creating the respective items.

Remember to restart Zabbix agent after making changes to the configuration file.

7 Mass update

Overview

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

Using mass update

To mass-update some items, do the following:

- Mark the checkboxes of the items to update in the list
- Click on *Mass update* below the list
- Navigate to the tab with required attributes (*Item* or *Preprocessing*)
- Mark the checkboxes of the attributes to update
- Enter new values for the attributes

Type	☐	Original
Host interface	☐	Original
JMX endpoint	☐	Original
URL	☐	Original
Request body type	☐	Original
Request body	☐	Original
Headers	☐	Original
SNMP community	☐	Original
Context name	☐	Original
Security name	☐	Original
Security level	☐	Original
Authentication protocol	☐	Original
Authentication passphrase	☐	Original
Privacy protocol	☐	Original
Privacy passphrase	☐	Original
Port	☐	Original
Type of information	☐	Original
Units	☐	Original
Authentication method	☐	Original
User name	☐	Original
Public key file	☐	Original
Private key file	☐	Original
Password	☐	Original
Update interval	☐	Original
History storage period	☒	7d
Trend storage period	☐	Original
Status	☐	Original
Log time format	☐	Original
Show value	☐	Original
Enable trapping	☐	Original
Allowed hosts	☐	Original
Applications	☒	AddReplaceRemove type here to search
Master item	☐	Original
Description	☐	Original

Update

Cancel

The *Applications* option allows to:

- *Add* - add new or existing applications to the items by specifying application name
- *Replace* - replace existing applications of the items with the one(s) specified in this field
- *Remove* - only remove specified applications from the items

The field for specifying applications is auto-complete - starting to type in it offers a dropdown of matching applications. If the application is new, it also appears in the dropdown and it is indicated by *(new)* after the string. Just scroll down to select.

The screenshot shows the 'Preprocessing' tab for an item. On the left, there is a checkbox labeled 'Preprocessing steps' which is checked. To the right, there is a table with two columns: 'Name' and 'Parameters'. The table contains two rows: the first row has 'JSONPath' in the 'Name' column and '\$.path.to.node' in the 'Parameters' column; the second row has 'JavaScript' in the 'Name' column and 'script' in the 'Parameters' column. Below the table, there is a blue 'Add' link. At the bottom of the configuration area, there are two buttons: 'Update' and 'Cancel'.

	Name	Parameters
1:	JSONPath	\$.path.to.node
2:	JavaScript	script

[Add](#)

Update Cancel

When done, click on *Update*.

8 Value mapping

Overview

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

Value mappings can be used in both the Zabbix frontend and notifications sent by email/SMS/jabber etc.

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'

Thus, 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 *Show value* field.

Note:

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

Value mappings, starting with Zabbix 3.0, can be exported/imported, either separately, or with the respective template or host.

Configuration

To define a value map:

- Go to: *Administration* → *General*
- Select *Value mapping* from the dropdown
- Click on *Create value map* (or on the name of an existing map)

* Name

Windows service state

* Mappings

Value		Mapped to
0	⇒	Running
1	⇒	Paused
2	⇒	Start pending
3	⇒	Pause pending
4	⇒	Continue pending
5	⇒	Stop pending
6	⇒	Stopped
7	⇒	Unknown
255	⇒	No such service

Add

Add

Cancel

Parameters of a value map:

Parameter	Description
<i>Name</i>	Unique name of a set of value mappings.
<i>Mappings</i>	Individual mappings - pairs of numeric values and their string representations.

All mandatory input fields are marked with a red asterisk.

To add a new individual mapping, click on *Add*.

How this works

For example, one of the predefined agent items 'Ping to the server (TCP)' uses an existing value map called 'Service state' to display its values.

*** Name**

*** Mappings**

Value		Mapped to
0	⇒	Down
1	⇒	Up

[Add](#)

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

Show value [show value mappings](#)

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

NAME	LAST CHECK	LAST VALUE
Zabbix agent (1 item)		
☐ Agent ping	2015-08-11 22:01:07	Up (1)

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 parenthesis).

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:

NAME	LAST CHECK	LAST VALUE
Zabbix agent (1 item)		
☐ Agent ping	2015-08-11 22:09:21	1

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

9 Applications

Overview

Applications are used to group items in logical groups.

For example, the *MySQL Server* application can hold all items related to the MySQL server: availability of MySQL, disk space, processor load, transactions per second, number of slow queries, etc.

Applications are also used for grouping web scenarios.

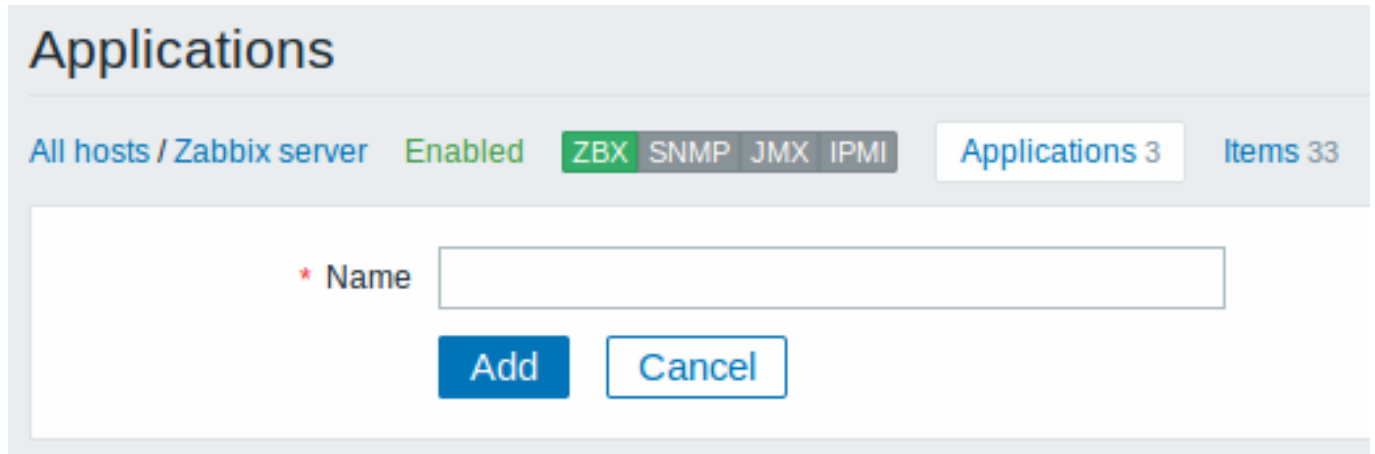
If you are using applications, then in *Monitoring* → *Latest data* you will see items and web scenarios grouped under their respective applications.

Configuration

To work with applications you must first create them and then link items or web scenarios to them.

To create an application, do the following:

- Go to *Configuration* → *Hosts* or *Templates*
- Click on *Applications* next to the required host or template
- Click on *Create application*
- Enter the application name and click on *Add* to save it



The screenshot shows the 'Applications' configuration page in Zabbix. At the top, there's a header 'Applications'. Below it, a navigation bar shows 'All hosts / Zabbix server' followed by a green 'Enabled' status. To the right are tabs for 'ZBX', 'SNMP', 'JMX', and 'IPMI'. Further right, it says 'Applications 3' and 'Items 33'. The main form area has a label '* Name' next to a text input field. Below the input field are two buttons: 'Add' and 'Cancel'.

You can also create a new application directly in the item properties form.

Items are linked to applications in the item properties form. Select one or more applications the item will belong to.

Web scenarios are linked to applications in the web scenario definition form. Select the application the scenario will belong to.

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*. *Overview* should be selected in the dropdown to the right.

Queue of items to be updated Overview						
Items	5 seconds	10 seconds	30 seconds	1 minute	5 minutes	More than 10 minutes
Zabbix agent	0	6	0	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
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

The picture here is generally "ok" so we may assume that the server is doing fine.

The queue shows six items waiting for 10 seconds. It would be great to know what items these are.

To do just that, select *Details* in the dropdown in the upper right corner. Now you can see a list of those delayed items.

Scheduled check	Delayed by	Host	Name
2018-11-01 10:22:45	20s	Remote proxy: My host	Incoming network traffic on eth0
2018-11-01 10:22:46	19s	Remote proxy: My host	Outgoing network traffic on eth0
2018-11-01 10:22:47	18s	Remote proxy: My host	Free inodes on / (percentage)
2018-11-01 10:22:48	17s	Remote proxy: My host	Free disk space on /
2018-11-01 10:22:49	16s	Remote proxy: My host	Free disk space on / (percentage)
2018-11-01 10:22:51	14s	Remote proxy: My host	Used disk space on /

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 of items to be updated 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

Is the agent down?

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/aggregate items and some macros much faster, since Zabbix 2.2 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 Check 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
- Zabbix aggregate
- External check
- Database monitor
- JMX agent
- SSH agent
- Telnet
- Calculated
- HTTP agent

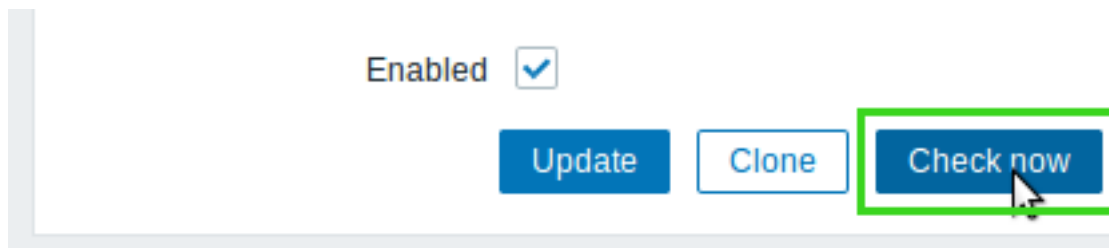
Attention:

The check must be present in 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 has been created just now.

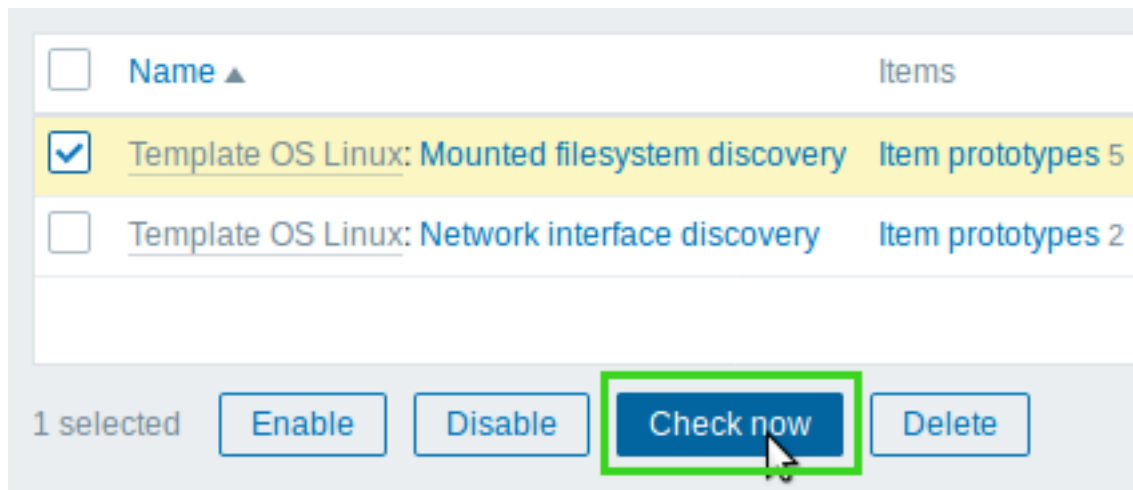
Configuration

To execute a passive check immediately:

- click on *Check now* in an existing item (or discovery rule) configuration form:



- click on *Check now* for selected items/rules in the list of items/discovery rules:



In the latter case several items/rules can be selected and "checked now" at once.

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 status to PROBLEM.

A trigger may have the following status:

VALUE	DESCRIPTION
OK	This is a normal trigger state. Called FALSE in older Zabbix versions.
PROBLEM	Normally means that something happened. For example, the processor load is too high. Called TRUE in older Zabbix versions.

Trigger status (the expression) is recalculated every time Zabbix server receives a new value that is part of the expression.

Triggers are evaluated based on **history** data only; trend data are never considered.

If time-based functions (**nodata()**, **date()**, **dayofmonth()**, **dayofweek()**, **time()**, **now()**) are used in the expression, the trigger is recalculated every 30 seconds by a Zabbix *history syncer* process. If both time-based and non-time-based functions are used in an expression, it is recalculated when a new value is received **and** every 30 seconds.

You can **build trigger expressions** with different degrees of complexity.

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

Configuration

The **Trigger** tab contains all the essential trigger attributes.

Parameter	Description
<i>Name</i>	Trigger name. The name may contain the supported macros: {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {ITEM.VALUE}, {ITEM.LASTVALUE} and {\$MACRO}. \$1, \$2...\$9 macros can be used to refer to the first, second...ninth constant of the expression. <i>Note</i>: \$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 {New host:system.cpu.load[percpu,avg1].last()}>5
<i>Severity</i>	Set the required trigger severity by clicking the buttons.
<i>Expression</i>	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 <i>Recovery expression</i>.
<i>OK event generation</i>	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. Supported since Zabbix 3.2.0.
<i>Recovery expression</i>	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 <i>OK event generation</i>. Supported since Zabbix 3.2.0.
<i>PROBLEM event generation mode</i>	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.
<i>OK event closes</i>	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 Supported since Zabbix 3.2.0.
<i>Tag for matching</i>	Enter event tag name to use for event correlation. This field is displayed if 'All problems if tag values match' is selected for the <i>OK event closes</i> property and is mandatory in this case. Supported since Zabbix 3.2.0.
<i>Allow manual close</i>	Check to allow manual closing of problem events generated by this trigger. Manual closing is possible when acknowledging problem events. Supported since Zabbix 3.2.0.
<i>URL</i>	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 <i>Monitoring</i> → <i>Problems</i> (URL option in the <i>Trigger</i> menu) and <i>Problems</i> dashboard widget. Supported macros: {ITEM.VALUE}, {ITEM.LASTVALUE}, {TRIGGER.ID}, several {HOST.*} macros, user macros.

Parameter	Description
<i>Description</i>	Text field used to provide more information about this trigger. May contain instructions for fixing specific problem, contact detail of responsible staff, etc. <i>Starting with Zabbix 2.2</i> , the description may contain the same set of macros as trigger name.
<i>Enabled</i>	Unchecking this box will disable the trigger if required.

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
<i>Name/Value</i>	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. User macros, user macro context, low-level discovery macros and macro functions with <code>{ITEM.VALUE}</code> , <code>{ITEM.LASTVALUE}</code> and low-level discovery macros are supported in event tags. Low-level discovery macros can be used inside macro context <code>{TRIGGER.ID}</code> macro is supported in trigger tag values since Zabbix 4.2.8. 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 <i>Monitoring → Problems</i> or the <i>Problems</i> widget. Supported since Zabbix 3.2.0.

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.

Following expression from an official template is taken as an example:

```
{Template Net Cisco IOS SNMPv2:sensor.temp.value[ciscoEnvMonTemperatureValue.{#SNMPINDEX}].avg(5m)}>{$TEMP}
or
{Template Net Cisco IOS SNMPv2:sensor.temp.status[ciscoEnvMonTemperatureState.{#SNMPINDEX}].last(0)}={$TEMP}
```

To test the expression, click on *Expression constructor* under the expression field.

Trigger

Dependencies

Name

Template OS CISCO: Temperature too high

Severity

Not classified

Information

Warning

Average

High

Expression

{Template Net Cisco IOS
SNMPv2:sensor.temp.value[ciscoEnvMonTemperatureValue.
{#SNMPINDEX}].avg(5m)}>{\$TEMP_WARN}
or
{Template Net Cisco IOS
SNMPv2:sensor.temp.status[ciscoEnvMonTemperatureState.
{#SNMPINDEX}].last(0)}={\$TEMP_WARN_STATUS}

Expression constructor

In the Expression constructor, all individual expressions are listed. To open the testing window, click on *Test* below the expression list.

Target	Expression	Action	Info
☒	Or		Remove
☐	A {Template Net Cisco IOS SNMPv2:sensor.temp.value[ciscoEnvMonTemperatureValue.{#SNMPINDEX}].avg(5m)}>{\$TEMP_WARN}	Remove	
☐	B {Template Net Cisco IOS SNMPv2:sensor.temp.status[ciscoEnvMonTemperatureState.{#SNMPINDEX}].last(0)}={\$TEMP_WARN_STATUS}	Remove	
Test			

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.

Test

Test data

Expression Variable Elements	Result type	Value
{Template Net Cisco IOS SNMPv2:sensor.temp.value[ciscoEnvMonTemperatureValue.{#SNMPINDEX}].avg(5m)}	Numeric (float)	80
{\$TEMP_WARN}	Numeric (float)	70
{Template Net Cisco IOS SNMPv2:sensor.temp.status[ciscoEnvMonTemperatureState.{#SNMPINDEX}].last(0)}	Numeric (integer)	0
{\$TEMP_WARN_STATUS}	Numeric (float)	1

Result

Expression	Result
Or	TRUE
A {Template Net Cisco IOS SNMPv2:sensor.temp.value[ciscoEnvMonTemperatureValue.{#SNMPINDEX}].avg(5m)}>{\$TEMP_WARN}	TRUE
B {Template Net Cisco IOS SNMPv2:sensor.temp.status[ciscoEnvMonTemperatureState.{#SNMPINDEX}].last(0)}={\$TEMP_WARN_STATUS}	FALSE
A or B	TRUE

Test

Cancel

The result of the individual expressions as well as the whole expression can be seen.

"TRUE" result means the specified expression is correct. In this particular case A, "80" is greater than {\$TEMP_WARN} specified value, "70" in this example. Respectively, "TRUE" result appears.

"FALSE" result means the specified expression is incorrect. In this particular case B, {\$TEMP_WARN_STATUS}, "1" in this example, needs to be equal with specified "0" value, that is wrong. Respectively, "FALSE" result appears.

Chosen Expression type is "OR"/"TRUE". If at least one of the specified conditions (A or B in this case) is TRUE, overall result will be TRUE as well. That means, 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 useful expression might look like:

```
{<server>:<key>.<function>(<parameter>)}<operator><constant>
```

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 avoid trigger flapping in **hysteresis**.

Functions

Trigger functions allow to reference the collected values, current time and other factors.

A complete list of **supported functions** is available.

Function parameters

Most of numeric functions accept the number of seconds as a parameter.

You may use the prefix **#** to specify that a parameter has a different meaning:

FUNCTION CALL	MEANING
sum(600)	Sum of all values in no more than the latest 600 seconds
sum(#5)	Sum of all values in no more than the last 5 values

The function **last** uses a different meaning for values when prefixed with the hash mark - it makes it choose the n-th previous value, so given the values 3, 7, 2, 6, 5 (from most recent to least recent), **last(#2)** would return 7 and **last(#5)** would return 5.

Several functions support an additional, second `time_shift` parameter. This parameter allows to reference data from a period of time in the past. For example, **avg(1h,1d)** will return the average value for an hour one day ago.

You can use the supported **unit symbols** in trigger expressions, for example '5m' (minutes) instead of '300' seconds or '1d' (day) instead of '86400' seconds. '1K' will stand for '1024' bytes.

Numbers with a '+' sign are not supported.

Operators

The following operators are supported for triggers (**in descending priority of execution**):

PRIORITY	OPERATOR	DEFINITION	Notes for unknown values
1	-	Unary minus	-Unknown → Unknown
2	not	Logical NOT	not Unknown → Unknown
3	*	Multiplication	0 * Unknown → Unknown (yes, Unknown, not 0 - to not lose Unknown in arithmetic operations) 1.2 * Unknown → Unknown
	/	Division	Unknown / 0 → error Unknown / 1.2 → Unknown 0.0 / Unknown → Unknown
4	+	Arithmetic plus	1.2 + Unknown → Unknown
	-	Arithmetic minus	1.2 - Unknown → Unknown

5	<	Less than. The operator is defined as: $A < B \Leftrightarrow (A < B - 0.000001)$	1.2 < Unknown $\rightarrow$ Unknown
	<=	Less than or equal to. The operator is defined as: $A \leq B \Leftrightarrow (A \leq B + 0.000001)$	Unknown <= Unknown $\rightarrow$ Unknown
	>	More than. The operator is defined as: $A > B \Leftrightarrow (A > B + 0.000001)$	
	>=	More than or equal to. The operator is defined as: $A \geq B \Leftrightarrow (A \geq B - 0.000001)$	
6	=	Is equal. The operator is defined as: $A = B \Leftrightarrow (A \geq B - 0.000001 \text{ and } A \leq B + 0.000001)$	

PRIORITY OPERATOR		DEFINITION	Notes for unknown values
	<>	Not equal. The operator is defined as: $A <> B \Leftrightarrow (A < B \vee A > B)$	
7	and	Logical AND	0 and Unknown → 0 1 and Unknown → Unknown Unknown and Unknown → Unknown
8	or	Logical OR	1 or Unknown → 1 0 or Unknown → Unknown Unknown or Unknown → Unknown

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

Processor load is too high on www.zabbix.com

```
{www.zabbix.com:system.cpu.load[all,avg1].last()}>5
```

'www.zabbix.com:system.cpu.load[all,avg1]' gives a short name of the monitored parameter. It specifies that the server is 'www.zabbix.com' and the key being monitored is 'system.cpu.load[all,avg1]'. By using the function 'last()', we are referring to the most recent value. Finally, '>5' means that the trigger is in the PROBLEM state whenever the most recent processor load measurement from www.zabbix.com is greater than 5.

Example 2

www.zabbix.com is overloaded

```
{www.zabbix.com:system.cpu.load[all,avg1].last()}>5 or {www.zabbix.com:system.cpu.load[all,avg1].min(10m)}>2
```

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

Use of function diff:

```
{www.zabbix.com:vfs.file.cksum[/etc/passwd].diff()}=1
```

The expression is true when the previous value of checksum of /etc/passwd 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:

```
{www.zabbix.com:net.if.in[eth0,bytes].min(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:

```
{smtp1.zabbix.com:net.tcp.service[smtp].last()}=0 and {smtp2.zabbix.com:net.tcp.service[smtp].last()}=0
```

The expression is true when both SMTP servers are down on both smtp1.zabbix.com and smtp2.zabbix.com.

Example 6

Zabbix agent needs to be upgraded

Use of function str():

```
{zabbix.zabbix.com:agent.version.str("beta8")}=1
```

The expression is true if Zabbix agent has version beta8 (presumably 1.0beta8).

Example 7

Server is unreachable

```
{zabbix.zabbix.com:icmpping.count(30m,0)}>5
```

The expression is true if host "zabbix.zabbix.com" is unreachable more than 5 times in the last 30 minutes.

Example 8

No heartbeats within last 3 minutes

Use of function nodata():

```
{zabbix.zabbix.com:tick.nodata(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():

```
{zabbix:system.cpu.load[all,avg1].min(5m)}>2 and {zabbix:system.cpu.load[all,avg1].time()}>000000 and {zab
```

The trigger may change its status to true, only at night (00:00-06:00) time.

Example 10

Check if client local time is in sync with Zabbix server time

Use of function fuzzytime():

```
{MySQL_DB:system.localtime.fuzzytime(10)}=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 11

Comparing average load today with average load of the same time yesterday (using a second time_shift parameter).

```
{server:system.cpu.load.avg(1h)}/{server:system.cpu.load.avg(1h,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 12

Using the value of another item to get a trigger threshold:

```
{Template PfSense:hrStorageFree[{#SNMPVALUE}].last()}<{Template PfSense:hrStorageSize[{#SNMPVALUE}].last()}
```

The trigger will fire if the free storage drops below 10 percent.

Example 13

Using **evaluation result** to get the number of triggers over a threshold:

```
({server1:system.cpu.load[all,avg1].last()}>5) + ({server2:system.cpu.load[all,avg1].last()}>5) + ({server3:system.cpu.load[all,avg1].last()}>5)
```

The trigger will fire if at least two of the triggers in the expression are over 5.

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:

```
{server:temp.last()}>20
```

Recovery expression:

```
{server:temp.last()}<=15
```

Example 2

Free disk space is too low.

Problem expression: it is less than 10GB for last 5 minutes

```
{server:vfs.fs.size[/,free].max(5m)}<10G
```

Recovery expression: it is more than 40GB for last 10 minutes

```
{server:vfs.fs.size[/,free].min(10m)}>40G
```

Expressions with unsupported items and unknown values

Versions before Zabbix 3.2 are very strict about unsupported items in a trigger expression. Any unsupported item in the expression immediately renders trigger value to **Unknown**.

Since Zabbix 3.2 there is a more flexible approach to unsupported items by admitting unknown values into expression evaluation:

- For some functions their values are not affected by whether an item is supported or unsupported. Such functions are now evaluated even if they refer to unsupported items. See the list in **functions and unsupported items**.
- Logical expressions with OR and AND can be evaluated to known values in two cases regardless of unknown operands:
 - "1 or Unsupported_item1.some_function() or Unsupported_item2.some_function() or ..." can be evaluated to '1' (True),

- "0 and Unsupported_item1.some_function() and Unsupported_item2.some_function() and ..." can be evaluated to '0' (False).

Zabbix tries to evaluate logical expressions taking unsupported items as `Unknown` values. In the two cases mentioned above a known value will be produced; in other cases trigger value will be `Unknown`.

- If a function evaluation for supported item results in error, the function value is `Unknown` and it takes part in further expression evaluation.

Note that unknown values may "disappear" only in logical expressions as described above. In arithmetic expressions unknown values always lead to result `Unknown` (except division by 0).

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 dependants 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 dependant 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.

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.

Trigger
Dependencies

Dependencies

NAME
New host: Zabbix agent on {HOST.NAME} is unreachable for 5 minutes
Add

Click *Update*. Now the trigger has an indication of its dependency in the list.

{HOST.NAME} is unreachable
Depends on:
New host: Zabbix agent on {HOST.NAME} is unreachable for 5 minutes

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	COLOUR
Not classified	Unknown severity.	Grey
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 colours 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 **customise trigger severity names and colours**.

5 Customising trigger severities

Trigger severity names and colours for severity related GUI elements can be configured in *Administration* → *General* → *Trigger severities*. Colours are shared among all GUI themes.

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, 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

Trigger Tags Dependencies

Severity ☒

Not classifiedInformationWarningAverageHighDis

Allow manual close ☒

NoYes

UpdateCancel

Trigger Tags Dependencies

Tags ☒

AddReplaceRemove

NameValue

tagvalue

Add

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.

The screenshot shows the 'Dependencies' tab in the Zabbix web interface. On the left, there is a checkbox labeled 'Replace dependencies' which is checked. To the right, there is a list of dependencies. The first two are 'My host: Zabbix agent on {HOST.NAME} is unreachable for 5 minutes' and 'My host: {HOST.NAME} has just been restarted'. Below the list is an 'Add' button with a plus icon. At the bottom of the panel are 'Update' and 'Cancel' buttons.

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 behaviour 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 signalling about a potential unwanted situation. First: trigger must fire when the system after "time to act" is expected to be in problem state. Second: 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 analyse to come up with prediction. You do it in a familiar way by means of `sec` or `#num` 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 analysed item has to reach, no difference if from above or from below. Once we have determined $f(t)$ (see below) we should solve equation $f(t) = \text{threshold}$ and return the root which is closer to now and to the right from now or 9999999999.9999 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

^aAccording 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	$x = f(t)$
<i>linear</i>	$x = a + b*t$
<i>polynomialN</i> ¹	$x = a_0 + a_1*t + a_2*t^2 + \dots + a_n*t^n$
<i>exponential</i>	$x = a*\exp(b*t)$
<i>logarithmic</i>	$x = a + b*\log(t)$
<i>power</i>	$x = a*t^b$

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 analysed 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
<i>value</i>	$f(\text{now} + \text{time})$
<i>max</i>	$\max_{\text{now} \leq t \leq \text{now} + \text{time}} f(t)$
<i>min</i>	$\min_{\text{now} \leq t \leq \text{now} + \text{time}} f(t)$
<i>delta</i>	$\text{max} - \text{min}$
<i>avg</i>	average of $f(t)$ ($\text{now} \leq t \leq \text{now} + \text{time}$) 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 defined²;
- 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:

```
{host:vfs.fs.size[/,free].timeleft(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

¹Polynomial degree can be from 1 to 6, *polynomial1* is equivalent to *linear*. However, use higher degree polynomials [with caution](#). If the evaluation period contains less points than needed to determine polynomial coefficients, polynomial degree will be lowered (e.g *polynomial5* is requested, but there are only 4 points, therefore *polynomial3* will be fitted).

²For 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.

³For *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).

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 expression⁴:

```
{host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h and {host:vfs.fs.size[/,free].timeleft(1h,,0)}<>-1
```

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 `{host:item.forecast(...)}<...` or like `{host:item.forecast(...)}>...`

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** operator = works). So add `...` or `{host:item.forecast(...)}=-1` or `...` and `{host:item.forecast(...)}<>-1` if you want or don't want to treat -1 as a problem respectively.

See also

1. [Predictive trigger functions \(pdf\)](#)

Technical test (please ignore): [Test file \(docx\)](#)

4 Events

Overview

There are several types of events generated in Zabbix:

- trigger events - whenever a trigger changes its status (*OK*→*PROBLEM*→*OK*)
- discovery events - when hosts or services are detected
- auto registration 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**

⁴ But in this case -1 can cause your trigger to recover from the problem state. To be fully protected use: `{host:vfs.fs.size[/,free].timeleft(1h,,0)}<1h and ({TRIGGER.VALUE}=0 and {host:vfs.fs.size[/,free].timeleft(1h,,0)}<>-1 or {TRIGGER.VALUE}=1)`

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

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 auto-registration creates events in Zabbix.

If configured, active agent auto-registration 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 auto-registration](#) 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.

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 a new value arrives 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 screen

If a problem arises for a trigger with the *Manual close* flag, you can go to the [problem update](#) screen of that problem and close the problem manually.

To close the problem, check the *Close problem* option in the form and click on *Update*.

Update problem

Message	Fixed, closing.					
History	Time User User action Message					
Scope	☒ Only selected problem ☐ Selected and all other problems of related triggers 1 event					
Change severity	☐ <table><tr><td>Not classified</td><td>Information</td><td>Warning</td><td>Average</td><td>High</td></tr></table>	Not classified	Information	Warning	Average	High
Not classified	Information	Warning	Average	High		
Acknowledge	☐					
Close problem	☒					

* At least one update operation or message must exist.

Update

Cancel

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 the **event tags**. 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]
```

The screenshot shows the Zabbix web interface for configuring a new item. The 'Preprocessing' tab is selected. The configuration fields are as follows:

- Name:** Syslog item
- Type:** Zabbix agent (active)
- Key:** log[/var/log/syslog]
- Type of information:** Text
- Update interval:** 30s

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:

```
{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":

```
{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

The screenshot shows the Zabbix Trigger configuration interface with the 'Dependencies' tab selected. The configuration is as follows:

- Name:** Service `{{ITEM.VALUE}.regsub("^. * service ([a-zA-Z]*) .*", "1")}` stopped
- Severity:** Not classified, Information, Warning, Average, **High**, Disaster
- Expression:** `{Template Services:log[/var/log/messages].regexp("stopped")}=1` [Add]
- Expression constructor:** (Link)
- OK event generation:** Expression, **Recovery expression**, None
- Recovery expression:** `{Template Services:log[/var/log/messages].regexp("started")}=1` [Add]
- Expression constructor:** (Link)
- PROBLEM event generation mode:** Single, **Multiple**
- OK event closes:** All problems, **All problems if tag values match**
- Tag for matching:** Service
- Tags:**
 - Service: `{{ITEM.VALUE}.regsub("^. * service ([a-zA-Z]*) .*", "1")}` [Remove]
 - Datacenter: value [Remove]
- [Add] (Link)

If configured successfully you will be able to see problem events tagged by application and matched to their resolution in *Monitoring* → *Problems*.

Problems

Export to CSV

Filter

Time	☐	Severity	Recovery time	Status	Info	Host	Problem	Duration	Ack	Actions	Tags
08:38:18	☐	High	08:38:18	RESOLVED	Zabbix server	Service Apache stopped		0	No		Service: Apache Webserver

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.

Event tags

Overview

There is an option to define custom event tags in Zabbix. Tags can be defined on template, host and trigger levels.

After the tags are defined, corresponding new events get marked with tag data:

- with template level tags - host problems that are created by triggers from this template will be marked
- with host level tags - all problems of the host will be marked
- with trigger level tags - problem of this trigger will be marked

An event inherits all tags from the whole chain of templates, hosts, 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. Most importantly, events can be **correlated** based on event tags. In other uses, actions can be defined based on event tags.

Event tags are realized as a pair of the *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

Note:

Tags are not supported for host prototypes and hosts created from prototypes.

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//`.
- 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.
- 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//`.
- 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}`
 - * In trigger configuration, have multiple problem event generation mode;
 - * In trigger configuration, use `[[:manual/config/event_correlation|event correlation]]`: select the option
 - * See problem events created with a tag and closed individually.
- 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.
- 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.
- 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.
- 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.
- 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

Event tags can be defined in:

- template configuration - affecting all triggers from the template when linked to hosts
- host configuration - affecting all triggers of the host
- individual trigger configuration:

Severity	Not classified	Information	Warning	Average	High
Tags					
Cloud	value	Remove			
Host	{{ITEM.VALUE2}.iregsub(	Remove			
Service	MySQL	Remove			
Customers	value	Remove			
Add					

Event tags can be defined for triggers, template triggers and trigger prototypes.

Macro support

The following macros may be used in trigger-level 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 (supported since 4.0.0).
- **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.

{EVENT.TAGS} and {EVENT.RECOVERY.TAGS} macros can be used in trigger-based notifications and they will resolve to a comma separated list of event tags or recovery event tags.

The following macros may be used in template and host-level tags:

- {HOST.HOST}, {HOST.NAME}, {HOST.CONN}, {HOST.DNS}, {HOST.IP}, {HOST.PORT} and {HOST.ID} macros
- {INVENTORY.*} **macros**
- **User macros**

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

Event tags, if defined, can be seen with new events in:

- *Monitoring → Problems*
- *Monitoring → Problems → Event details*
- *Monitoring → Dashboard → Problems* widget (in popup window that opens when rolling the mouse over problem name)

Status	Info	Host	Problem	Duration	Ack	Actions	Tags
PROBLEM		New host	Nodata on 'New host' for two minutes	39s	No		Cloud Customers Host: HP-Pro
							Cloud Customers Host: HP-Pro Service: MySQL

Only the first three tag entries are 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.

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 Zabbix Super Admin level users only.

Attention:

Event correlation must be configured very carefully, as it can negatively affect event processing performance or, if mis-configured, 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

Correlation
Operations

*

Name

Close old event

Type of calculation

And

▼

A and (B and C) and D

*

Conditions

Label	Name	Action
A	Old event tag <i>Application</i> equals new event tag <i>Application</i>	Remove
B	Old event tag <i>Application</i> equals <i>ABC</i>	Remove
C	Old event tag <i>State</i> equals <i>Down</i>	Remove
D	New event tag <i>State</i> equals <i>Up</i>	Remove

New condition

New event tag value

▼

tag

equals

▼

value

[Add](#)

Description

Close old events for Application ABC if an event with State=Up happens.

Enabled

☒

Add

Cancel

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Name</i>	Unique correlation rule name.
<i>Type of calculation</i>	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).
<i>Conditions</i>	List of conditions, as selected from the <i>New condition</i> field.

Parameter	Description
<i>New condition</i>	Select conditions for correlating events and click on <i>Add</i>. <i>Note</i> 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 <i>names</i> 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: <i>equals</i> - has the old event tag value <i>does not equal</i> - does not have the old event tag value <i>contains</i> - has the string in the old event tag value <i>does not contain</i> - 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: <i>equals</i> - has the new event tag value <i>does not equal</i> - does not have the new event tag value <i>contains</i> - has the string in the new event tag value <i>does not contain</i> - does not have the string in the new event tag value
<i>Description</i>	Correlation rule description.
<i>Enabled</i>	If you mark this checkbox, the correlation rule will be enabled.

- Select the operation of the correlation rule in the form

Parameter	Description
<i>Operations</i>	List of operations, selected from the <i>New operation</i> field.
<i>New operation</i>	Select operation to perform when event is correlated and click on <i>Add</i>. 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 <i>Close old events</i> operation or all existing problems could be closed. Close new event - close the new event when it happens

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.

CorrelationOperations

* Name

Correlate network port problems

Type of calculation

And

A and B

* Conditions

Label	Name	Action
A	Old event tag <i>Port</i> equals new event tag <i>Port</i>	Remove
B	Old event tag <i>Host</i> equals new event tag <i>Host</i>	Remove

New condition

Event tag pair

old event tag

equals

new event tag

[Add](#)

Description

Keep only one problem per port. No need to report all of them.

Enabled

☒

Add

Cancel

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.

CorrelationOperations

* Operations

Details	Action
Close new event	Remove

This operation will close new problem events on the same network port, keeping only the original problem open.

6 Visualisation

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 **customised graphs**
- access to a comparison of several items quickly in **ad-hoc graphs**
- modern customisable **vector graphs**

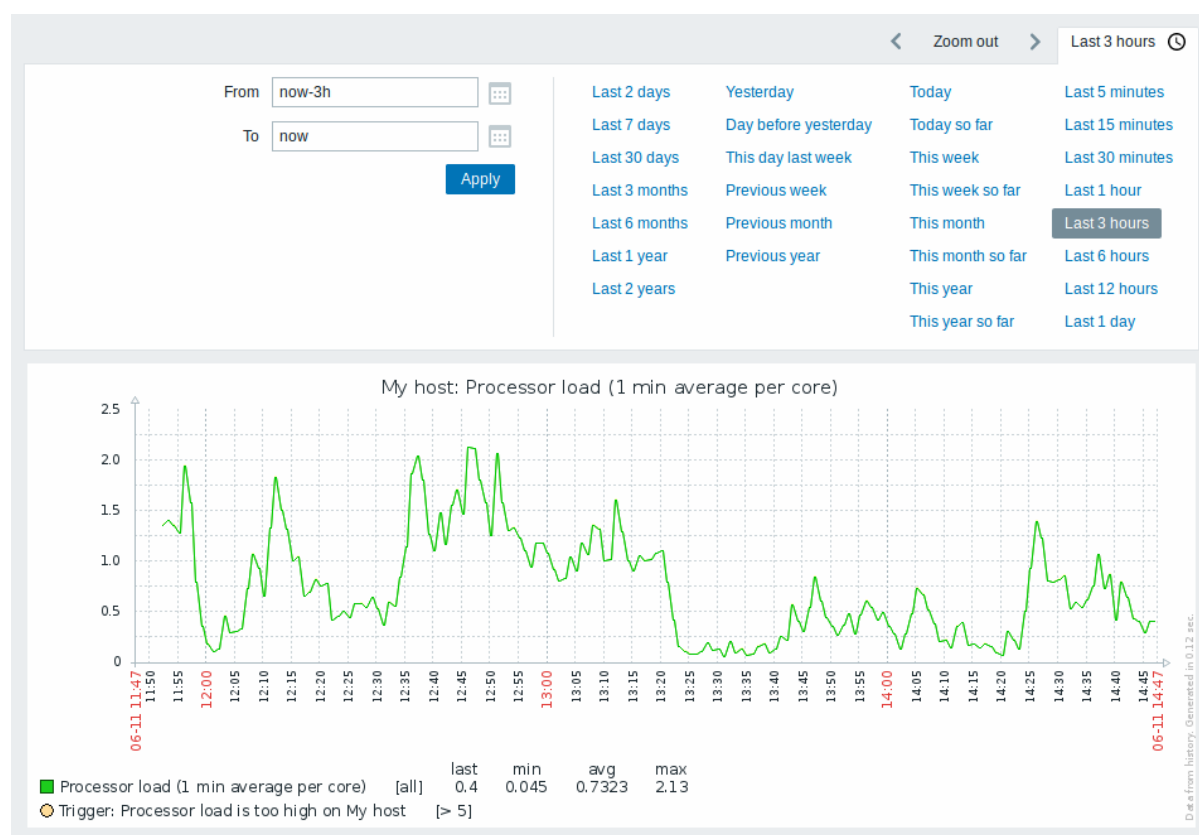
1 Simple graphs

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:

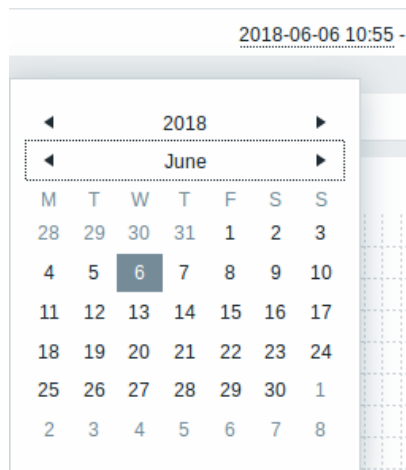
- absolute time syntax in format `Y-m-d H:i:s`

- 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)

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.

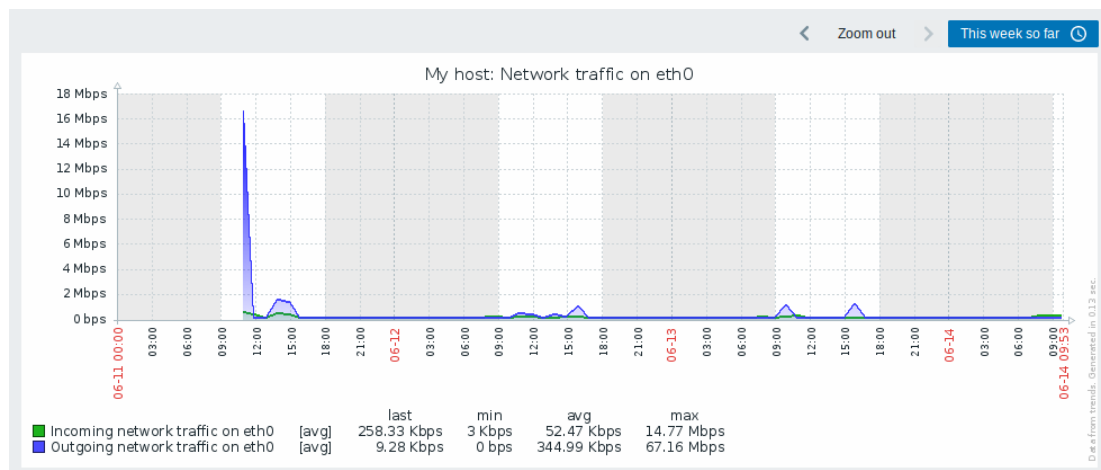
In case no time value is specified or field is left blank, time value will be set to "00:00:00". This doesn't apply to today's date selection: in that case time will be set to current value.

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 grey (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.

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 grey, 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 customisation 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

Graph

Preview

* Name

Network utilization

* Width

900

* Height

200

Graph type

Normal

Show legend

☒

Show working time

☒

Show triggers

☒

Percentile line (left)

☐

Percentile line (right)

☐

Y axis MIN value

Fixed

0

Y axis MAX value

Calculated

* Items

Name	Function	Draw style	Y axis side	Colour	Action
1: My host: Outgoing network traffic on eth0	avg	Filled region	Left	 00C800	Remove
2: My host: Incoming network traffic on eth0	avg	Bold line	Left	 C80000	Remove
Add					

Add

Cancel

All mandatory input fields are marked with a red asterisk.

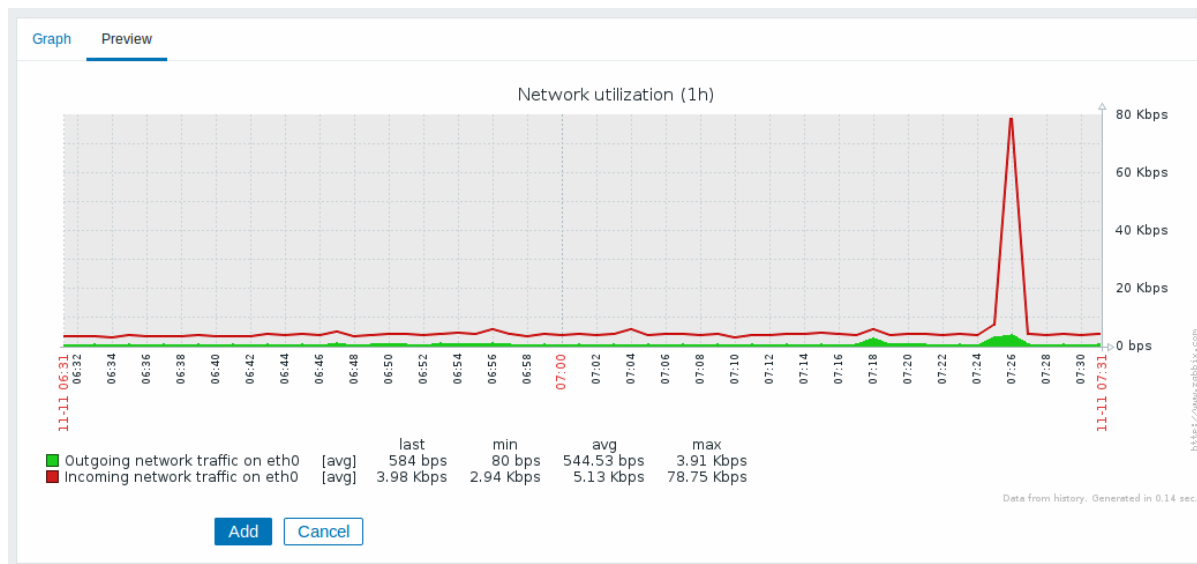
Graph attributes:

Parameter	Description
<i>Name</i>	Unique graph name. Starting with Zabbix 2.2, item values can be referenced in the name by using simple macros with the standard <code>{host:key.func(param)}</code> syntax. Only avg , last , max and min as functions with seconds as parameter are supported within this macro. <code>{HOST.HOST<1-9>}</code> macros are supported for the use within this macro, referencing the first, second, third, etc. host in the graph, for example <code>{HOST.HOST1}:key.func(param)}</code> .
<i>Width</i>	Graph width in pixels (for preview and pie/exploded graphs only).
<i>Height</i>	Graph height in pixels.
<i>Graph type</i>	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
<i>Show legend</i>	Checking this box will set to display the graph legend.
<i>Show working time</i>	If selected, non-working hours will be shown with gray background. Not available for pie and exploded pie graphs.
<i>Show triggers</i>	If selected, simple triggers will be displayed as red lines. Not available for pie and exploded pie graphs.
<i>Percentile line (left)</i>	Display percentile for left Y axis. If, for example, 95% percentile is set, then the percentile line will be at the level where 95 per cent of the values fall under. Displayed as a bright green line. Only available for normal graphs.
<i>Percentile line (right)</i>	Display percentile for right Y axis. If, for example, 95% percentile is set, then the percentile line will be at the level where 95 per cent of the values fall under. Displayed as a bright red line. Only available for normal graphs.

Parameter	Description
<i>Y axis MIN value</i>	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
<i>Y axis MAX value</i>	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
<i>3D view</i>	Enable 3D style. For pie and exploded pie graphs only.
<i>Items</i>	Items, data of which are to be displayed in this graph. Click on <i>Add</i> to select items. You can also select various displaying options (function, draw style, left/right axis display, colour).
<i>Sort or-order</i>	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 in the beginning of (0→100) line to set the sort order or which item is displayed in front of the other.
<i>Name</i>	Name of the selected item is displayed as a link. Clicking on the link opens the list of other available items.
<i>Type</i>	Type (only available for pie and exploded pie graphs): Simple - value of the item is represented proportionally on the pie Graph sum - value of the item represents the whole pie Note that colouring of the "graph sum" item will only be visible to the extent that it is not taken up by "proportional" items.
<i>Function</i>	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 <i>Normal</i> 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 <i>Pie/Exploded pie</i> is selected as graph type. max - display the maximum values min - display the minimum values
<i>Draw style</i>	Select the draw style (only available for normal graphs; for stacked graphs filled region is always used) to apply to the item data - <i>Line, Bold line, Filled region, Dot, Dashed line, Gradient line</i> .
<i>Y axis side</i>	Select the Y axis side to show the item data - <i>Left, Right</i> .
<i>Colour</i>	Select the colour 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:

3 triggers is the hard-coded limit for the number of triggers displayed in the legend.
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 customisation 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

Latest data

Host groups

Name

Hosts

Show items without data ☒

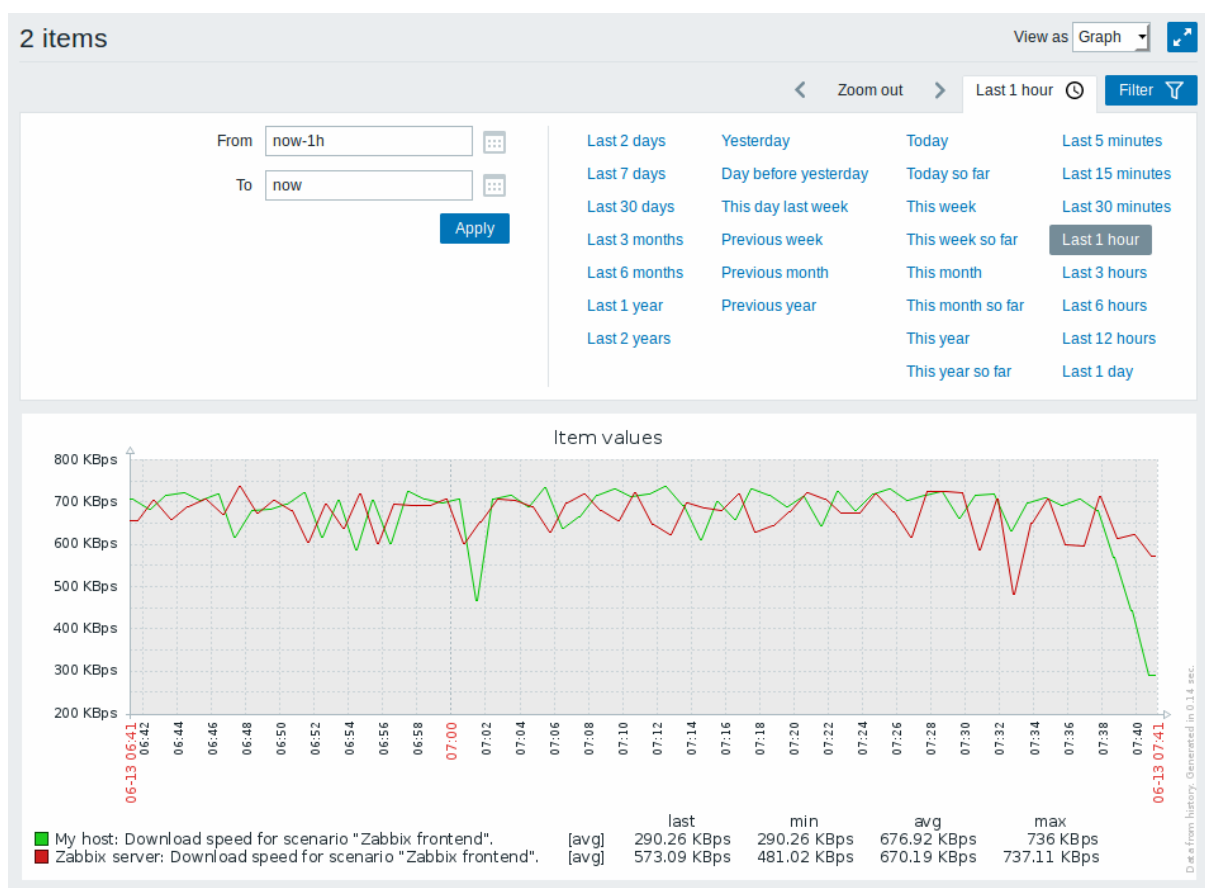
Application

Show details ☐

☐ Host	Name ▲	Last check	Last value	Change
▼ My host	Zabbix frontend (1 item)			
☒	Download speed for scenario "Zabbix frontend".	2018-06-13 07:42:52	502.11 KBps	-58.14 KBps Graph
▼ Zabbix server	Zabbix frontend (1 item)			
☒	Download speed for scenario "Zabbix frontend".	2018-06-13 07:43:01	593.04 KBps	-54.14 KBps Graph

2 selected

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).



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 member 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 greyed out icon and all textual information on the element is hidden. However, 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 it.

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:

Map

Sharing

*

Owner

Admin (Zabbix Administrator) X

Select

*

Name

Local network

*

Width

680

*

Height

600

Background image

No image

Automatic icon mapping

<manual>

show icon mappings

Icon highlight

☒

Mark elements on trigger status change

☒

Display problems

Expand single problem

Number of problems

Number of problems and expand most critical one

Advanced labels

☒

Host group label type

Label

Host label type

Label

Trigger label type

Status only

Map label type

Label

Image label type

Nothing

Map element label location

Bottom

Problem display

All

Minimum severity

Not classified

Information

Warning

Average

High

Disaster

Show suppressed problems

☐

URLs

Name

URL

Element

Latest data

https://localhost/zabbix/latest.php

Host

Add

Add

Cancel

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.

Parameter	Description
<i>Background image</i>	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.
<i>Automatic icon mapping</i>	You can set to use an automatic icon mapping, configured in <i>Administration</i> → <i>General</i> → <i>Icon mapping</i>. Icon mapping allows to map certain icons against certain host inventory fields.
<i>Icon highlighting</i>	If you check this box, map elements will receive highlighting. Elements with an active trigger will receive a round background, in the same colour 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
<i>Mark elements on trigger status change</i>	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.
<i>Display problems</i>	Select how problems are displayed with a map element: Expand single problem - name of the most critical problem is displayed Number of problems - the total number of problems is displayed Number of problems and expand most critical one - 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). (Note that trigger ID instead of problem event ID was evaluated before 4.2.1.) For 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.
<i>Advanced labels</i>	If you check this box you will be able to define separate label types for separate element types.
<i>Map element label type</i>	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
<i>Map element label location</i>	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
<i>Problem display</i>	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
<i>Minimum trigger severity</i>	Problems below the selected minimum severity level will not be displayed in the map. For example, with <i>Warning</i> selected, changes with <i>Information</i> and <i>Not classified</i> level triggers will not be reflected in the map. This parameter is supported starting with Zabbix 2.2.
<i>Show suppressed problems</i>	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.

Parameter	Description
<i>URLs</i>	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
<i>Type</i>	Select map type: Private - map is visible only to selected user groups and users Public - map is visible to all
<i>List of user group shares</i>	Select user groups that the map is accessible to. You may allow read-only or read-write access.
<i>List of user shares</i>	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

Type: Host

Label: New element

Label location: Default

* Host: My host Select

Application: Select

Automatic icon selection: ☐

Icons

Default	Server_(96)
Problem	Default
Maintenance	Default
Disabled	Default

Coordinates X: 89 Y: 127

URLs	Name	URL	Action
			Remove

[Add](#)

Apply Remove Close

Parameter	Description
<i>Type</i>	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
<i>Label</i>	Icon label, any string. Macros and multi-line strings can be used in labels. For a full list of supported macros, see supported macros and search for 'map element labels'.
<i>Label location</i>	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
<i>Host</i>	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.
<i>Map</i>	Select the map, if the element type is 'Map'.
<i>Triggers</i>	If the element type is 'Trigger', select one or more triggers in the <i>New triggers</i> field below and click on <i>Add</i>. 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 <i>Expand single problem</i> 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 <i>Number of problems and expand most critical one</i> mode is chosen the first displayed {HOST.*} macros will be resolved depending on the trigger severity.
<i>Host group</i>	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.
<i>Application</i>	You can select an application, allowing to only display problems of triggers that belong to the given application. This field is available for host and host group element types, and supported since Zabbix 2.4.0.
<i>Automatic icon selection</i>	In this case an icon mapping will be used to determine which icon to display.
<i>Icons</i>	You can choose to display different icons for the element in these cases: default, problem, maintenance, disabled.
<i>Coordinate X</i>	X coordinate of the map element.
<i>Coordinate Y</i>	Y coordinate of the map element.
<i>URLs</i>	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 (option available since Zabbix 2.0).

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 (such as, say, {HOST.NAME} for the element label).

Map element: [Add / Remove](#) Shape: [Add / Remove](#) Link: [Add / Remove](#) Expand macros: [Off](#) Grid: [Shown / On](#) 50x50 [Align map elements](#) [Update](#)

Y X: 50 100 150 200 250 300 350 400 450 500 550 600 650 700

(MAP.NAME)

New element

(HOST.NAME)
(HOST.CONN)

Mass update elements

Selected elements

Type	Name
Host	My host
Host	vcenter.zabbix.lan

☒ Label

{HOST.NAME}
{HOST.CONN}

☒ Label location

Top

☐ Automatic icon selection

☐ Icon (default)

Cloud_(24)

☐ Icon (problem)

Default

☐ Icon (maintenance)

Default

☐ Icon (disabled)

Default

[Apply](#)

[Remove](#)

[Close](#)

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.

Map element: [Add](#) / [Remove](#) Shape: [Add](#) / [Remove](#) Link: [Add](#) / [Remove](#) Expand macros: [Off](#) Grid: [Shown](#) / [On](#) 50x50 [Align map elements](#) [Update](#)

Map element

Type: Host

Label: New element

Label location: Default

* Host: My host X Select

Application: Select

Automatic icon selection ☐

Icons:

Default	Server_(96)
Problem	Default
Maintenance	Default
Disabled	Default

Coordinates X: 89 Y: 127

URLs

Name	URL	Action
		Remove

Add

Apply Remove Close

Links

Element name	Link indicators	Action
vcenter.zabbix.lan		Edit

Label: 100Mbps

Connect to: vcenter.zabbix.lan

Type (OK): Bold line

Colour (OK): 00CC00

Link indicators

Trigger	Type	Colour	Action
Add			

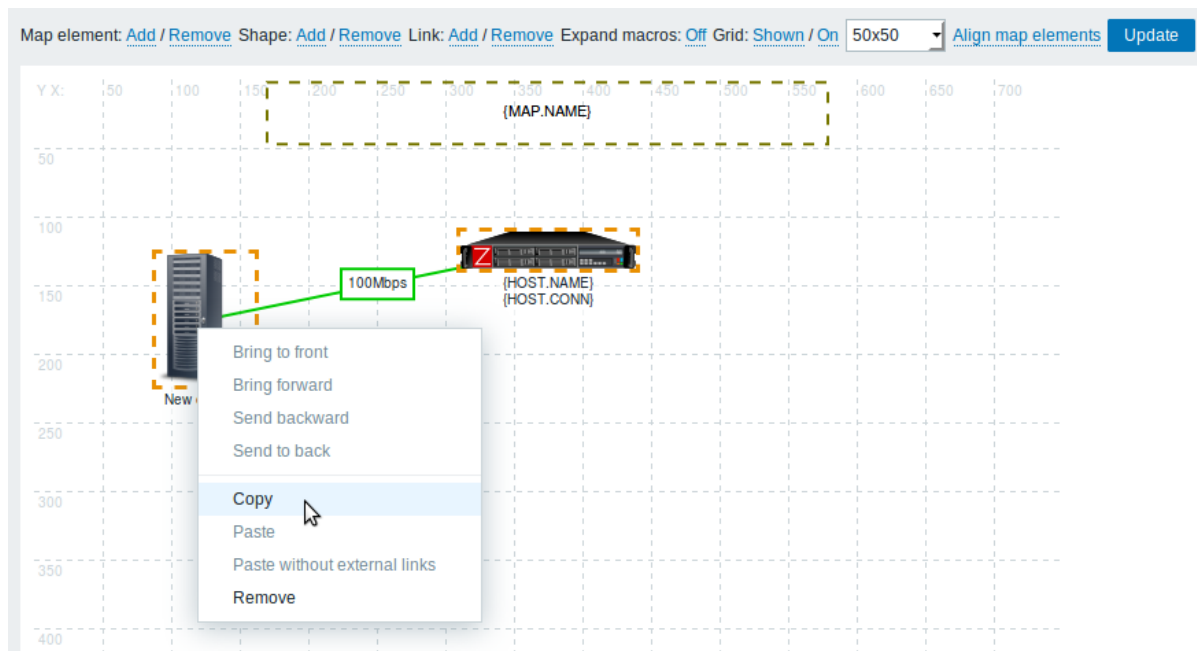
Apply Remove Close

Parameter	Description
<i>Label</i>	Label that will be rendered on top of the link. The <code>{host:key.func(param)}</code> macro is supported in this field, but only with avg, last, min and max trigger functions, with seconds as parameter.
<i>Connect to Type (OK)</i>	The element that the link connects to. Default link style: Line - single line Bold line - bold line Dot - dots Dashed line - dashed line
<i>Colour (OK)</i>	Default link colour.
<i>Link indicators</i>	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.

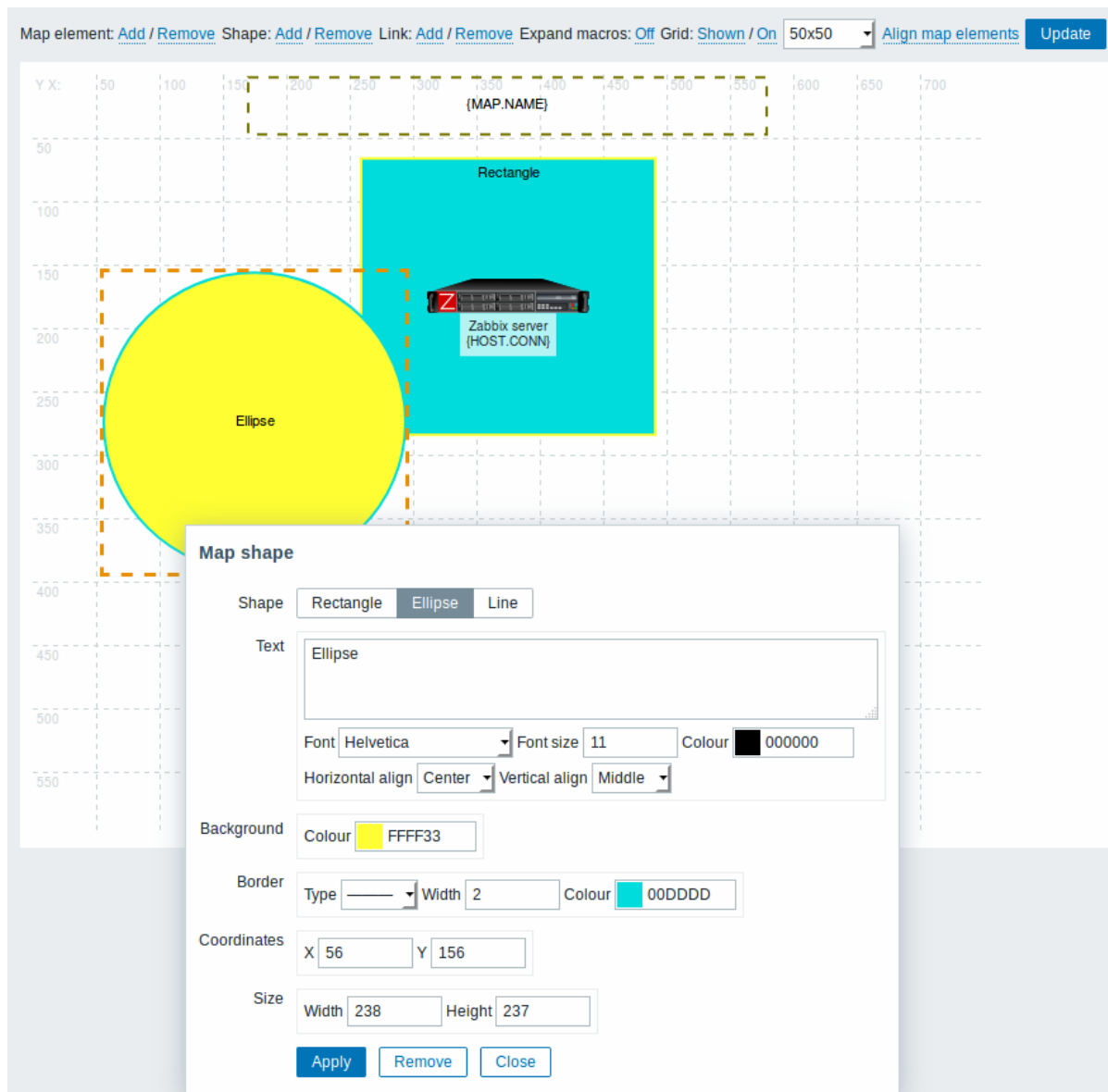
Copy-pasting works within the same browser window. Keyboard shortcuts are not supported.

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.

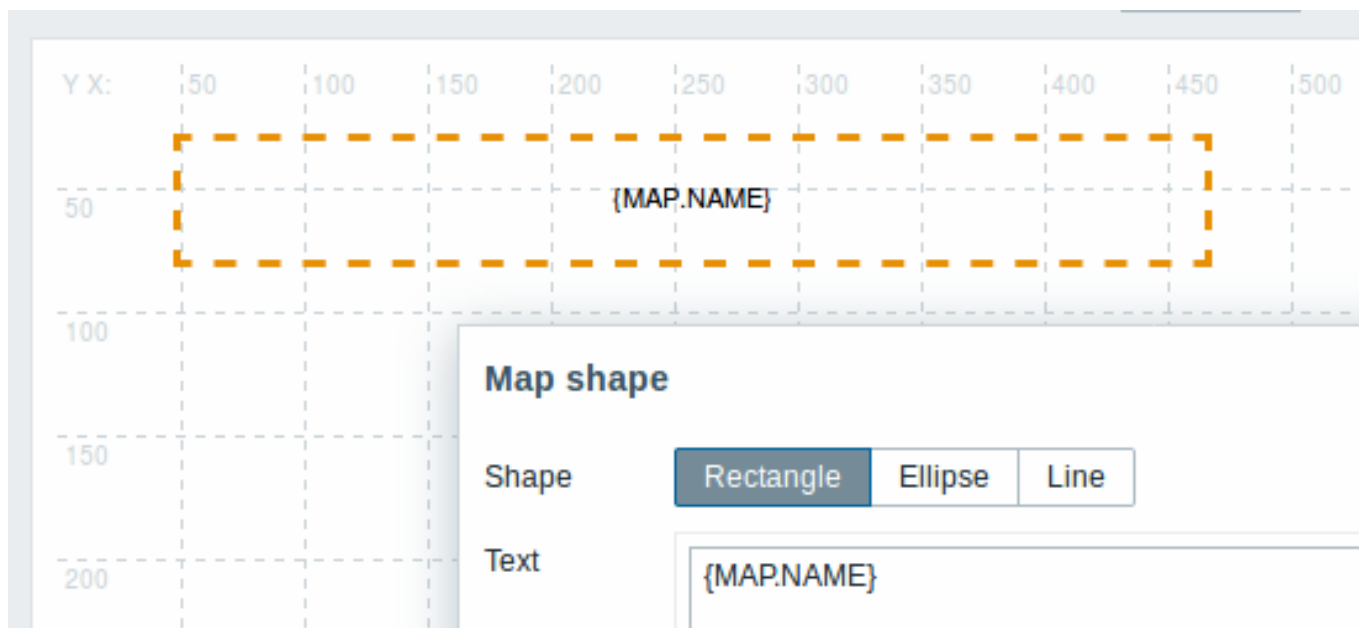
To add a shape, click on *Add* next to Shape. The new shape will appear at the top left corner of the map. Drag and drop it wherever you like.

A new shape is added with default colours. 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. 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.

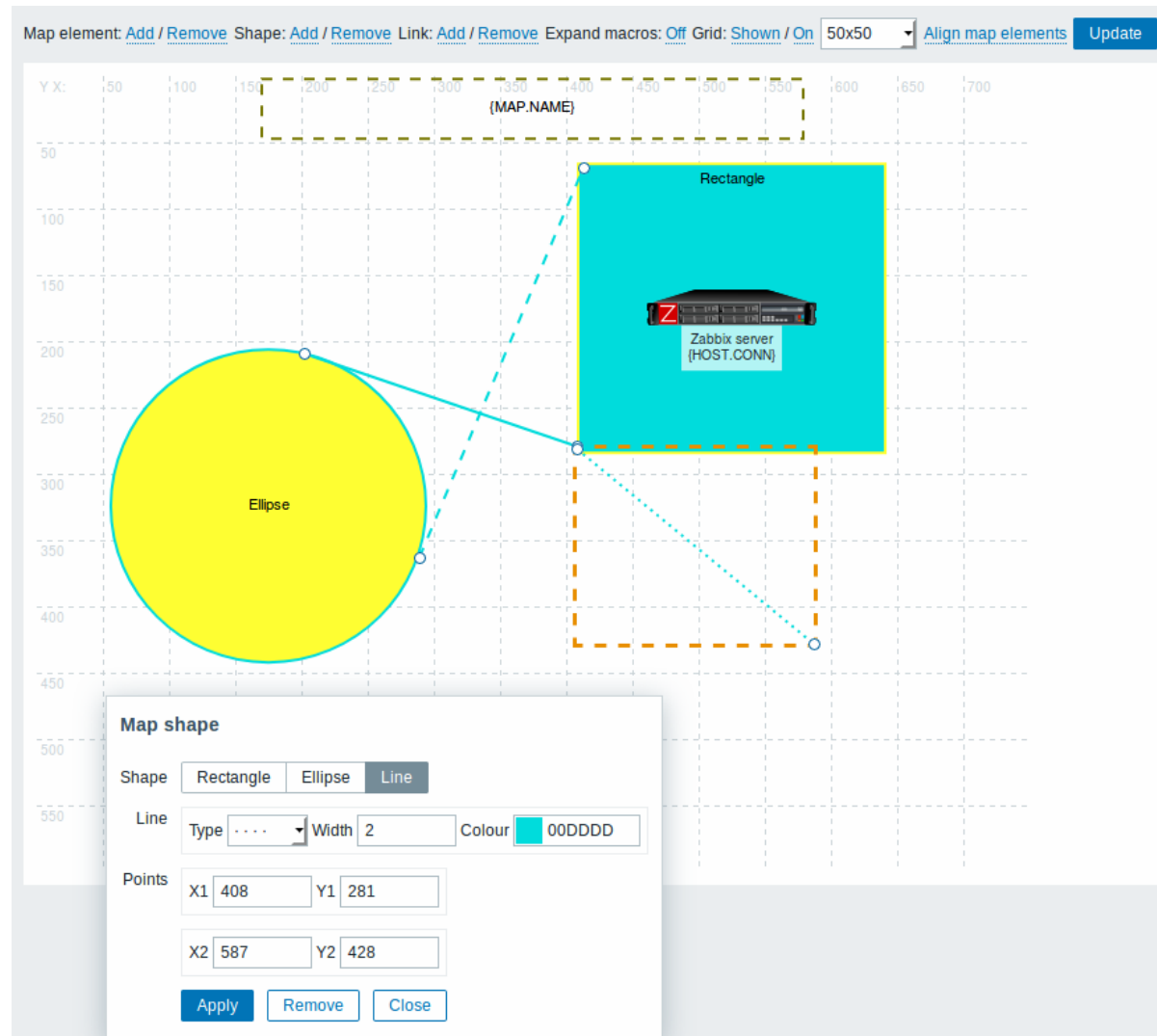
If hyperlinks are used in the text, they become clickable when viewing the map.

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

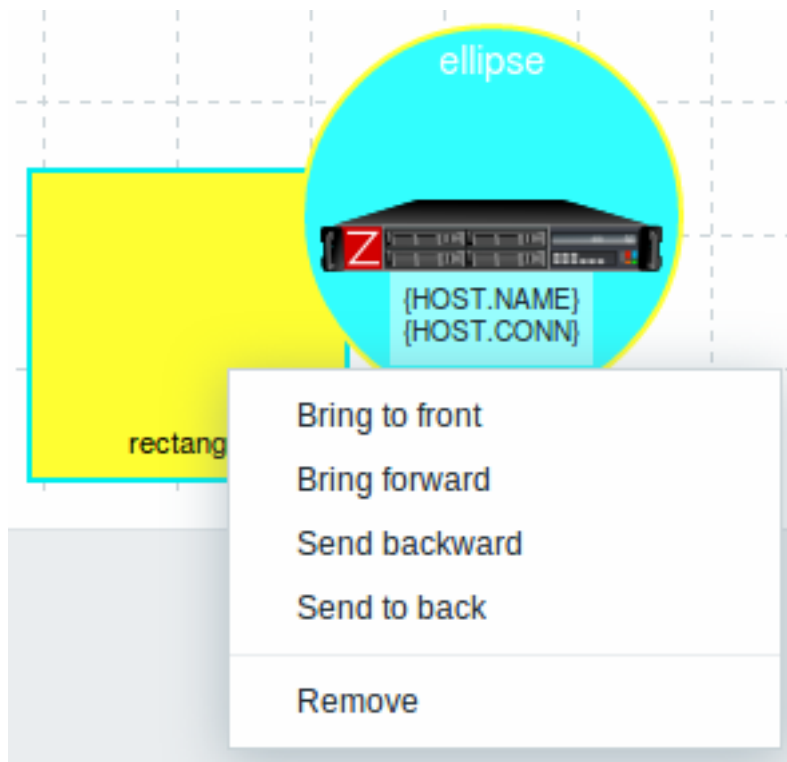
In addition to shapes, it is also possible to add some lines. Lines can be used to link elements or shapes in a map.

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, colour, 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

Map element: [Add / Remove](#) Shape: [Add / Remove](#) Link: [Add / Remove](#) Expand macros: [On](#) Grid: [Shown / On](#) 50x50 [Align map elements](#)

Map element

Type:

Show:

Area type:

Area size: Width Height

Placing algorithm:

Label:

Label location:

* Host group:

Application:

All mandatory input fields are marked with a red asterisk.

This table consists of parameters typical for *Host group* element type:

Parameter	Description
<i>Type</i>	Select Type of the element: Host group - icon representing status of all triggers of all hosts belonging to the selected group
<i>Show</i>	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 each single element (host) of the certain host group
<i>Area type</i>	This setting is available if "Host group elements" parameter is selected: Fit to map - all host group elements are equally placed within the map Custom size - manual setting of the map area for all the host group elements to be displayed

Parameter	Description
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 multi-line strings can be used in labels. If the type of the map element is “Host group” specifying certain macros has impact on the map view displaying corresponding information about each single host. For example, if {HOST.IP} macro is used, 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 “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:

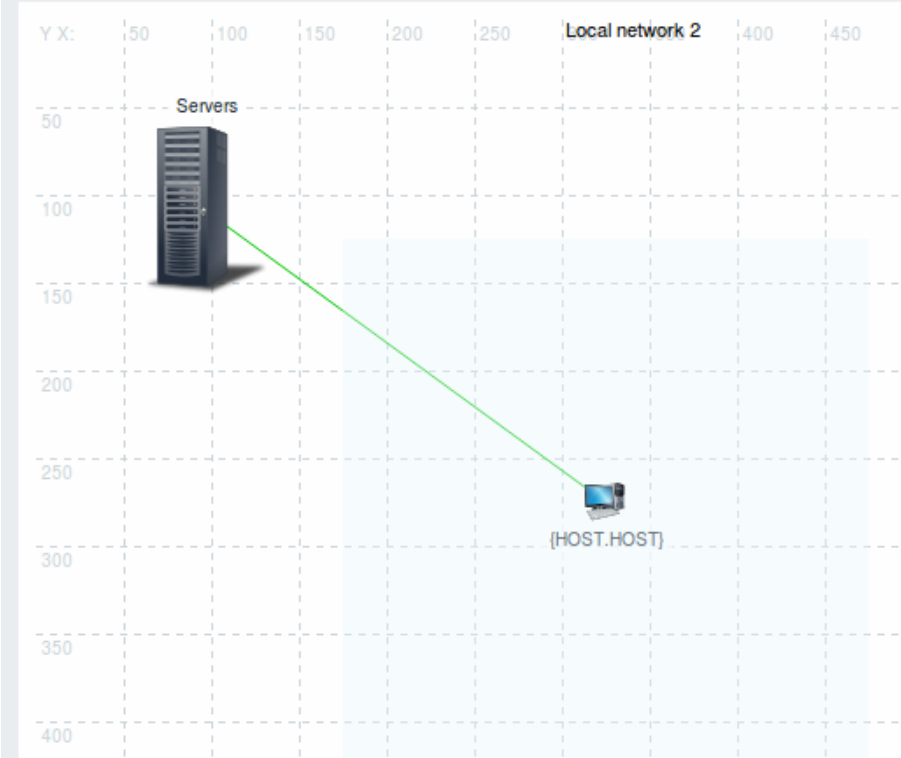
Map editing view

Network maps

Map element: [Add](#) / [Remove](#) Shape: [Add](#) / [Remove](#) Link: [Add](#) / [Remove](#) Expand macros: [On](#)

Y X: 50 100 150 200 250 300 350 400

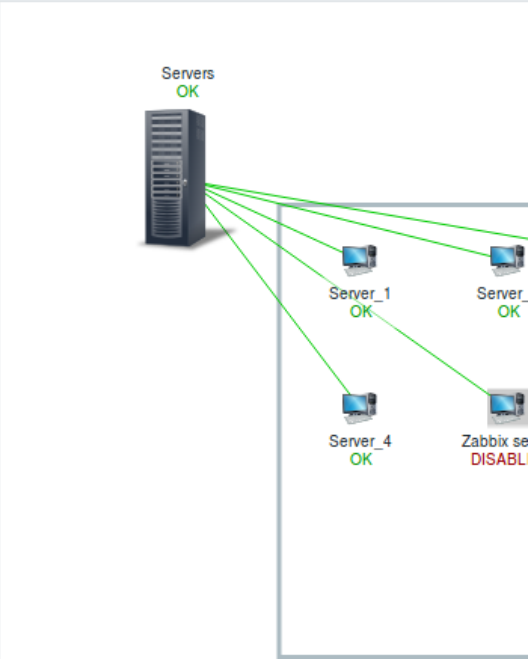
Local network 2 400 450



Map view

Maps

[All maps](#) / [Local network 2](#)



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.

361

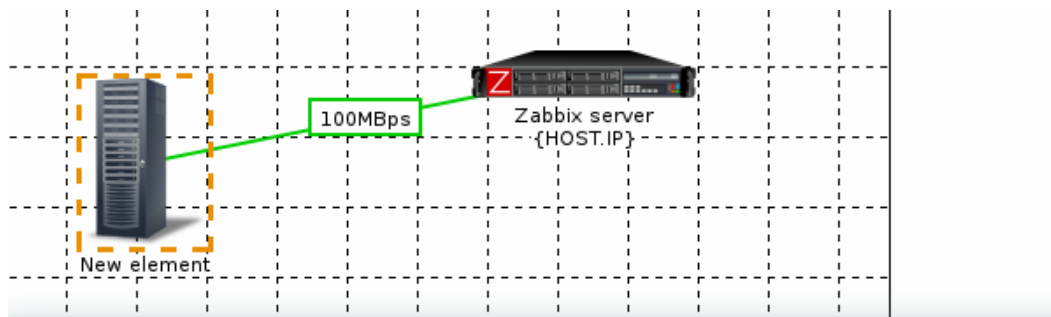
When you configure a link, you set the default link type and color. When you assign triggers to a link, you can assign different link types and colors with these triggers.

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 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



Map element

Type Host

Label

Label location Default

*Host New host x Select

Application Select

Automatic icon selection ☐

Icons

Default	Server_(96)
Problem	Server_(128)
Maintenance	Server_(24)
Disabled	Default

Coordinates X Y

URLs

NAME	URL

[Add](#)

Apply Remove Close

Links

ELEMENT NAME	LINK INDICATORS
Zabbix server	New host: Zabbix agent on New host is unreachable for 5 minutes

Label

Connect to Zabbix server

Type (OK) Bold line

Colour (OK) 00CC00

Link indicators

TRIGGER	TYPE	COLOUR
New host: Zabbix agent on New host is unreachable for 5 minutes	Line	00CC00

[Add](#)

Apply Remove Close

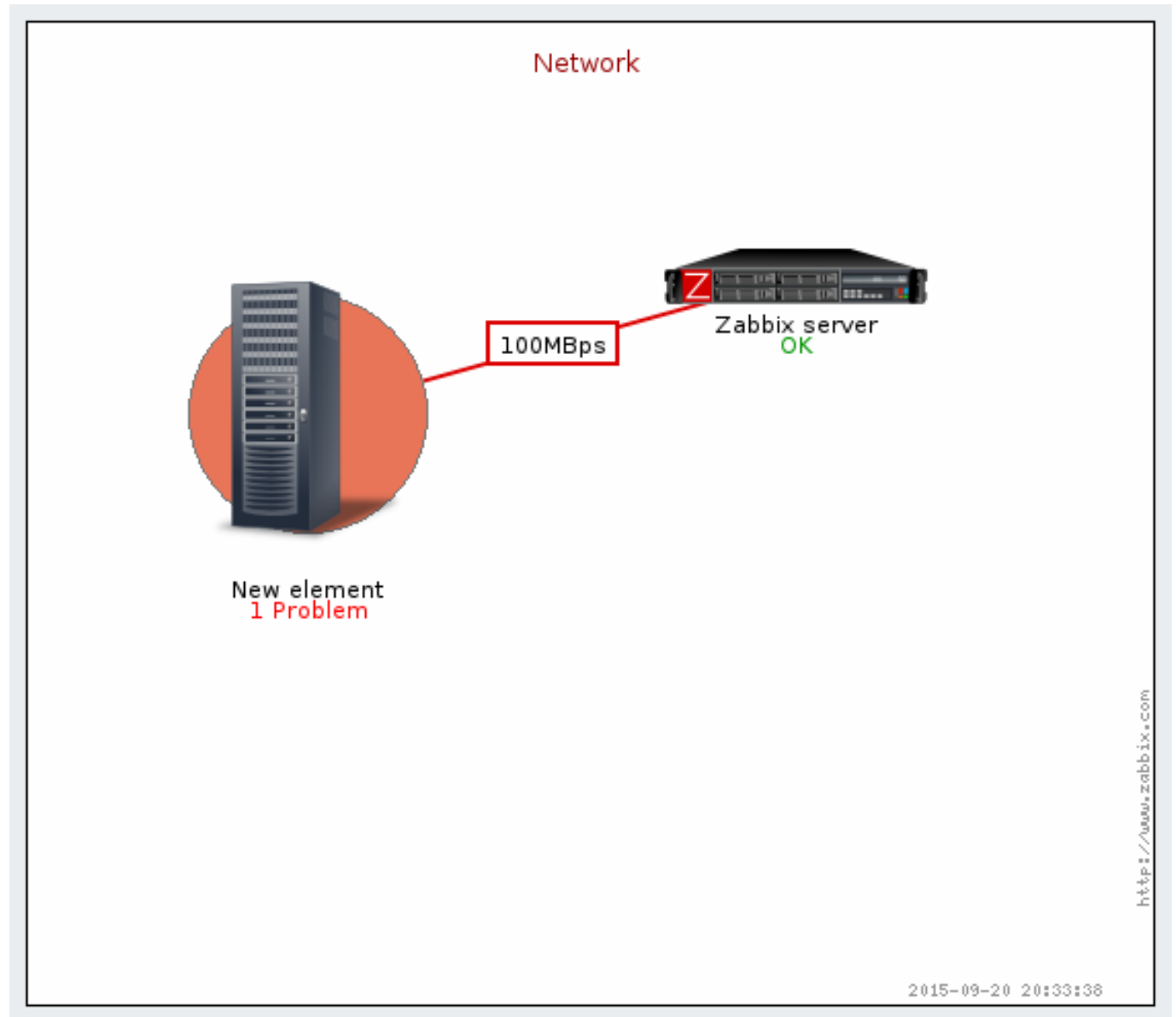
All mandatory input fields are marked with a red asterisk.

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 case with triggers with multiple problems (multiple problem generation), each problem may have severity that differs from trigger severity (changed manually), may have different tags (due to macros) and may be suppressed.

3 Screens

Overview

On Zabbix screens you can group information from various sources for a quick overview on a single screen. Building the screens is quite easy and intuitive.

Essentially a screen is a table. You choose how many cells per table and what elements to display in the cells. The following elements can be displayed:

- simple graphs
- simple graph prototypes
- user-defined custom graphs
- custom graph prototypes
- maps
- other screens
- plain text information
- server information (overview)
- host information (overview)
- trigger information (overview)
- host/host group issues (status of problems)
- problems by severity
- data overview
- clock
- history of events
- history of recent actions
- URL (data taken from another location)

Global screens are managed in *Monitoring* → *Screens*, where they can be configured, managed and viewed. They can also be added to the favourites section of *Monitoring* → *Dashboard*.

Host-level screens are configured on template level and then generated for hosts once the template is linked to the hosts.

To configure a screen you must first create it by defining its general properties and then add individual elements in the cells.

All users in Zabbix (including non-admin users) can create screens. Screens have an owner - the user who created them.

Screens can be made public or private. Public screens are visible to all users.

Private screens are visible only to their owner. Private screens can be shared by the owner to other users and user groups. Regular (non-Super admin) users can only share with the groups and users they are member of. Private screens will be visible to their owner and the users the screen is shared with as long as they have read permissions to all screen elements. Admin level users, as long as they have read permissions to all screen elements, can see and edit private screens regardless of being the owner or belonging to the shared user list.

Warning:

For both public and private screens a user must have at least read permissions to all screen elements in order to see the screen. To add an element to a screen a user must also have at least read permission to it.

Creating a screen

To create a screen, do the following:

- Go to *Monitoring* → *Screens*
- Go to the view with all screens
- Click on *Create Screen*

The **Screen** tab contains general screen attributes:

Screen
Sharing

* Owner
Admin (Zabbix Administrator) x
Select

* Name
Zabbix server

* Columns
2

* Rows
2

Add
Cancel

All mandatory input fields are marked with a red asterisk.

Give your screen a unique name and set the number of columns (vertical cells) and rows (horizontal cells).

The **Sharing** tab contains the screen type as well as sharing options (user groups, users) for private screens:

Screen
Sharing

Type
Private
Public

List of user group shares

USER GROUPS
PERMISSIONS
ACTION

Zabbix administrators
Read-only
Read-write
Remove

Add

List of user shares

USERS
PERMISSIONS
ACTION

user (New User)
Read-only
Read-write
Remove

Add

Add
Cancel

Parameter	Description
Owner	Select the screen owner.
Type	Select screen type: Private - screen is visible only to selected user groups and users Public - screen is visible to all
List of user group shares	Select user groups that the screen is accessible to. You may allow read-only or read-write access.
List of user shares	Select users that the screen is accessible to. You may allow read-only or read-write access.

Click on *Add* to save the screen.

Adding elements

To add elements to the screen, click on *Constructor* next to the screen name in the list.

On a new screen you probably only see links named *Change*. Clicking those links opens a form whereby you set what to display in each cell.

On an existing screen you click on the existing elements to open the form whereby you set what to display.

Screens: Zabbix server

Resource

Graph

* Graph

Zabbix server 1: CPU load

Select

Width

300

Height

80

Horizontal align

Left

Center

Right

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Dynamic item

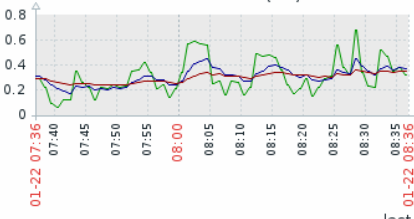
☐

Update

Delete

Cancel

New host: CPU load (1h)



Processor load (1 min average per core)

[avg]

0.32

last

Processor load (5 min average per core)

[avg]

0.37

Processor load (15 min average per core)

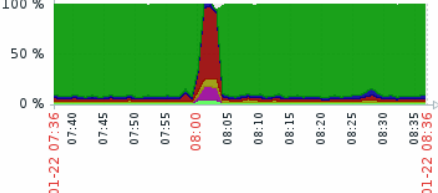
[avg]

0.355

Data from history. Generated in 0.30 sec.

Change

Zabbix server 1: CPU utilization (1h)



CPU idle time

[avg]

92.42 %

last

0 %

min

87.56 %

avg

CPU user time

[avg]

3.5 %

3.5 %

1.97 %

2.71 %

CPU system time

[avg]

2.8 %

2.8 %

1.87 %

6.34 %

CPU iowait time

[avg]

1.3 %

1.3 %

0.93 %

2.06 %

CPU nice time

[avg]

0 %

0 %

0 %

0.73 %

CPU interrupt time

[avg]

0 %

0 %

0 %

0.000573 %

CPU softirq time

[avg]

0.48 %

0.48 %

0.27 %

0.58 %

CPU steal time

[avg]

0 %

0 %

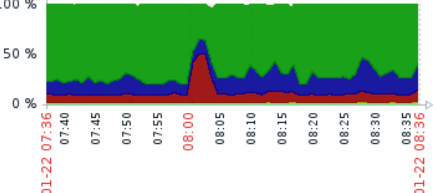
0 %

0 %

Data from history. Generated in 0.50 sec.

Change

New host: CPU utilization (1h)



CPU idle time

[avg]

62.4 %

last

35.73 %

min

71.3 %

avg

CPU user time

[avg]

26.39 %

26.39 %

10.72 %

17.46 %

CPU system time

[avg]

11.3 %

11.3 %

6.68 %

10.31 %

CPU iowait time

[avg]

1.29 %

1.29 %

0.59 %

0.94 %

CPU nice time

[avg]

0 %

0 %

0 %

0.000695 %

CPU interrupt time

[avg]

0 %

0 %

0 %

0.007013 %

CPU softirq time

[avg]

0.09 %

0.09 %

0.02 %

0.07 %

CPU steal time

[avg]

0 %

0 %

0 %

0 %

Data from history. Generated in 0.45 sec.

Change

All mandatory input fields are marked with a red asterisk.

Screen element attributes:

367

Parameter	Description
<i>Resource</i>	Information displayed in the cell: Action log - history of recent actions Clock - digital or analog clock displaying current server or local time Data overview - latest data for a group of hosts Graph - single custom graph Graph prototype - custom graph from low-level discovery rule History of events - latest events Host group issues - status of triggers filtered by the host group (includes triggers without events, since Zabbix 2.2) Host info - high level host related information Host issues - status of triggers filtered by the host (includes triggers without events, since Zabbix 2.2) Map - single map Plain text - plain text data Screen - screen (one screen may contain other screens inside) Simple graph - single simple graph Simple graph prototype - simple graph based on item generated by low-level discovery System information - high-level information about Zabbix server Problems by severity - displays problems by severity (similar to the Dashboard) Trigger info - high level trigger related information Trigger overview - status of triggers for a host group URL - include content from an external resource See also more information on configuring each resource.
<i>Horizontal align</i>	Possible values: Center Left Right
<i>Vertical align</i>	Possible values: Middle Top Bottom
<i>Column span</i>	Extend cell to a number of columns, same way as HTML column spanning works.
<i>Row span</i>	Extend cell to a number of rows, same way as HTML row spanning works.

Take note of the '+' and '-' controls on each side of the table.

Clicking on '+' above the table will add a column. Clicking on '-' beneath the table will remove a column.

Clicking on '+' on the left side of the table will add a row. Clicking on '-' on the right side of the table will remove a row.

Attention:

If graph height is set as less than 120 pixels, no trigger will be displayed in the legend.

Dynamic elements

For some of the elements there is an extra option called *Dynamic item*. Checking this box at first does not seem to change anything.

However, once you go to *Monitoring → Screens*, you may realize that now you have extra dropdowns there for selecting the host. Thus you have a screen where some elements display the same information while others display information depending on the currently selected host.

The benefit of this is that you do not need to create extra screens just because you want to see the same graphs containing data from various hosts.

Dynamic item option is available for several screen elements:

- Graphs (custom graphs)
- Graph prototypes

- Simple graphs
- Simple graph prototypes
- Plain text
- URL

Note:

Clicking on a dynamic graph opens it in full view; although with custom graphs and graph prototypes that is currently supported with the default host only (i.e. with host 'not selected' in the dropdown). When selecting another host in the dropdown, the dynamic graph is created using item data of that host and the resulting graph is not clickable.

Note:

Dynamic URL elements will not be displayed in *Monitoring → Screens*, unless a host is selected. Without a selected host the "No host selected" message will be visible only.

1 Screen elements

Overview

This section lists available **screen** elements and provides details for screen element configuration.

1 Action log

In the action log element you can display details of action operations (notifications, remote commands). It replicates information from *Reports → Audit*.

To configure, select *Action log* as resource:

Resource

Action log

* Show lines

25

Sort entries by

Time (descending)

Vertical align

TopMiddleBottom

* Column span

1

* Row span

1

Add

Cancel

All mandatory input fields are marked with a red asterisk.

You may set the following specific options:

Show lines	Set how many action log lines will be displayed in the screen cell.
Sort entries by	Sort entries by: Time (descending or ascending) Type (descending or ascending) Status (descending or ascending) Recipient (descending or ascending).

2 Clock

In the clock element you may display local, server or specified host time.

To configure, select *Clock* as resource:

Resource

Clock

Time type

Local time

Width

500

Height

100

Horizontal align

Left

Center

Right

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Add

Cancel

You may set the following specific options:

Time type	Select local, server or specified host time.
Item	Select the item for displaying time. To display host time, use the <code>system.localtime[local]</code> item. This item must exist on the host.
Width	This field is available only when <i>Host time</i> is selected.
Height	Select clock width. Select clock height.

3 Data overview

In the data overview element you can display the latest data for a group of hosts. It replicates information from *Monitoring → Overview* (when *Data* is selected as Type there).

To configure, select *Data overview* as resource:

Resource

Data overview

* Group

Discovered hosts X

Select

Application

CPU

Hosts location

Left

Top

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Add

Cancel

You may set the following specific options:

<i>Group</i>	Select host group.
<i>Application</i>	Enter application name.
<i>Hosts location</i>	Select host location - left or top.

4 Graph

In the graph element you can display a single custom graph.

To configure, select *Graph* as resource:

Resource

Graph

* Graph

Zabbix server: CPU utilization

Select

Width

500

Height

100

Horizontal align

Left

Center

Right

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Dynamic item

☐

Add

Cancel

You may set the following specific options:

<i>Graph</i>	Select the graph to display.
<i>Width</i>	Select graph width. Note that a line graph may actually take up more space due to legend text.
<i>Height</i>	Select graph height. Note that a line graph may actually take up more space due to legend text.
<i>Dynamic item</i>	Set graph to display different data depending on the selected host.

5 Graph prototype

In the graph prototype element you can display a custom graph from a low-level discovery rule.

To configure, select *Graph prototype* as resource:

Resource

Graph prototype

* Graph prototype

Zabbix server: Network traffic on {#IFNAME}

Select

* Max columns

3

Width

500

Height

100

Horizontal align

Left

Center

Right

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Dynamic item

☐

Add

Cancel

You may set the following specific options:

<i>Graph prototype</i>	Select the graph prototype to display.
<i>Max columns</i>	In how many columns generated graphs should be displayed in the screen cell.
<i>Width</i>	Useful when there are many LLD-generated graphs. Select graph width. Note that a line graph may actually take up more space due to legend text.
<i>Height</i>	Select graph height. Note that a line graph may actually take up more space due to legend text.
<i>Dynamic item</i>	Set graph to display different data depending on the selected host.

6 History of events

In the history of events element you can display latest events.

To configure, select *History of events* as resource:

Resource

History of events

* Show lines

25

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Add

Cancel

You may set the following specific option:

Show lines	Set how many event lines will be displayed in the screen cell.

7 Host group issues

In the host group issue element you can display problem details filtered by the selected host group.

The problem severity colour displayed is originally from the underlying trigger, but can be adjusted in the [problem update](#) screen.

To configure, select *Host group issues* as resource:

Resource

Host group issues

Group

Linux servers X

Select

* Show lines

25

Sort triggers by

Last change (descending)

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Add

Cancel

You may set the following specific options:

Group	Select host group.
Show lines	Set how many problem lines will be displayed in the screen cell.

Sort triggers by	Select from the dropdown to sort problems by last change, severity (both descending) or host (ascending).

8 Host info

In the host information element you can display high-level information about host availability.

To configure, select *Host info* as resource:

Resource

Host info

Group

Linux servers

Select

Style

Horizontal

Vertical

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Add

Cancel

You may set the following specific options:

Group	Select host group(s).
Style	Select vertical or horizontal display.

9 Host issues

In the host issue element you can display problem details filtered by the selected host.

The problem severity colour displayed is originally from the underlying trigger, but can be adjusted in the [problem update](#) screen.

To configure, select *Host issues* as resource:

Resource

Host issues

Host

New host x

Select

* Show lines

25

Sort triggers by

Last change (descending)

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Add

Cancel

You may set the following specific options:

Host	Select the host.
Show lines	Set how many problem lines will be displayed in the screen cell.
Sort triggers by	Select from the dropdown to sort problems by last change, severity (both descending) or host (ascending).

10 Map

In the map element you can display a configured network map.

To configure, select *Map* as resource:

Resource

Map

* Map

Network

Select

Horizontal align

Left

Center

Right

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Add

Cancel

You may set the following specific options:

Map	Select the map to display.

11 Plain text

In the plain text element you can display latest item data in plain text.

To configure, select *Plain text* as resource:

Resource

Plain text

* Item

New host: Checksum of /etc/passwd

Select

* Show lines

25

Show text as HTML

☐

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Dynamic item

☐

Add

Cancel

You may set the following specific options:

Item	Select the item.
Show lines	Set how many latest data lines will be displayed in the screen cell.
Show text as HTML	Set to display text as HTML.
Dynamic item	Set to display different data depending on the selected host.

12 Screen

In the screen element you can display another Zabbix screen. One screen may contain other screens inside.

To configure, select *Screen* as resource:

Resource

Screen

* Screen

Zabbix server2

Select

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Add

Cancel

You may set the following specific options:

Screen Select the screen to display.

13 Simple graph

In the simple graph element you can display a single simple graph.

To configure, select *Simple graph* as resource:

Resource

Simple graph

* Item

New host: Incoming network traffic on eth0

Select

Width

500

Height

100

Horizontal align

Left

Center

Right

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Dynamic item

☐

Add

Cancel

You may set the following specific options:

<i>Item</i>	Select the item for the simple graph.
<i>Width</i>	Select graph width. Note that a line graph may actually take up more space due to legend text.
<i>Height</i>	Select graph height. Note that a line graph may actually take up more space due to legend text.
<i>Dynamic item</i>	Set graph to display different data depending on the selected host.

14 Simple graph prototype

In the simple graph prototype element you can display a simple graph based on an item generated by low-level discovery.
To configure, select *Simple graph prototype* as resource:

Resource

Simple graph prototype

* Item prototype

New host: Incoming network traffic on {#IFNAME}

Select

* Max columns

3

Width

500

Height

100

Horizontal align

Left

Center

Right

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Dynamic item

☐

Add

Cancel

You may set the following specific options:

<i>Item prototype</i>	Select the item prototype for the simple graph.
<i>Max columns</i>	In how many columns generated graphs should be displayed in the screen cell. Useful when there are many LLD-generated graphs.
<i>Width</i>	Select graph width. Note that a line graph may actually take up more space due to legend text.
<i>Height</i>	Select graph height. Note that a line graph may actually take up more space due to legend text.

15 System information

In the system information element you can display high-level Zabbix and Zabbix server information.

To configure, select *System information* as resource:

Resource	System information
Vertical align	Top <input checked="" type="button" value="Middle"/> Bottom
* Column span	1
* Row span	1
	Add Cancel

16 Problems by severity

In this element you can display problems by severity similarly as in the Dashboard widget.

To configure, select *Problems by severity* as resource:

Resource	Problems by severity
Vertical align	Top <input checked="" type="button" value="Middle"/> Bottom
* Column span	1
* Row span	1
	Add Cancel

17 Trigger info

In the trigger info element you can display high-level information about trigger states.

To configure, select *Trigger info* as resource:

Resource

Trigger info

Group

Linux servers X

Select

Style

Horizontal

Vertical

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Add

Cancel

You may set the following specific options:

<i>Group</i>	Select the host group(s).
<i>Style</i>	Select vertical or horizontal display.

18 Trigger overview

In the trigger overview element you can display the trigger states for a group of hosts. It replicates information from *Monitoring → Overview* (when *Triggers* is selected as Type there).

To configure, select *Trigger overview* as resource:

Resource

Trigger overview

* Group

Linux servers X

Select

Application

Hosts location

Left

Top

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Add

Cancel

You may set the following specific options:

<i>Group</i>	Select the host group(s).
<i>Application</i>	Enter the application name.
<i>Hosts location</i>	Select host location - left or top.

19 URL

In the URL element you can display a URL content from an external resource.

To configure, select *URL* as resource:

Resource

URL

* URL

Width

500

Height

100

Horizontal align

Left

Center

Right

Vertical align

Top

Middle

Bottom

* Column span

1

* Row span

1

Dynamic item

☐

Add

Cancel

You may set the following specific options:

<i>URL</i>	Enter the URL to display.
<i>Width</i>	Select window width.
<i>Height</i>	Select window width.
<i>Dynamic item</i>	Set to display different URL content depending on the selected host.

Attention:

Browsers might not load an HTTP page included in a screen (using URL element), if Zabbix frontend is accessed over HTTPS.

4 Slide shows

Overview

In a slide show you can configure that a number of **screens** are displayed one after another at set intervals.

Sometimes you might want to switch between some configured screens. While that can be done manually, doing that more than once or twice may become very tedious. This is where the slide show function comes to rescue.

All users in Zabbix (including non-admin users) can create slide shows. Slide shows have an owner - the user who created them.

Slide shows can be made public or private. Public slide shows are visible to all users, however, they must have at least read permissions to all slide show elements (screens) to see it. To add a screen to the slide show the user must also have at least read permission to it.

Private slide shows are visible only to their owner. Private slide shows can be shared by the owner to other users and user groups. Regular (non-Super admin) users can only share with the groups and users they are member of. Private slide shows will be visible to their owner and the users the slide show is shared with as long as they have read permissions to all included screens. Admin level users, as long as they have read permissions to all included screens, can see and edit private slide shows regardless of being the owner or belonging to the shared user list.

Configuration

To create a slide show, do the following:

- Go to *Monitoring* → *Screens*
- Select *Slide shows* in the dropdown
- Go to the view with all slide shows
- Click on *Create slide show*

The **Slide** tab contains general slide show attributes:

The screenshot shows the Zabbix Slide Show configuration interface. It has two tabs: 'Slide' and 'Sharing'. The 'Slide' tab is active. It contains the following fields:

- * Owner:** A dropdown menu showing 'Admin (Zabbix Administrator)' with a 'Select' button.
- * Name:** A text input field containing 'Zabbix administrators'.
- * Default delay:** A text input field containing '30s'.
- * Slides:** A table with columns 'Screen', 'Delay', and 'Action'.

Screen	Delay	Action
1 Zabbix server	default	Remove
2 Zabbix server2	15	Remove

 Below the table is an 'Add' link.

At the bottom of the 'Slides' section are 'Add' and 'Cancel' buttons.

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Owner</i>	Select the slide show owner. Specifying owner is mandatory.
<i>Name</i>	Unique name of the slide show.
<i>Default delay</i>	How long one screen is displayed by default, before rotating to the next. Time suffixes are supported, e.g. 30s, 5m, 2h, 1d.
<i>Slides</i>	List of screens to be rotated. Click on <i>Add</i> to select screens. The <i>Up/Down</i> arrow before the screen allows to drag a screen up and down in the sort order of display. If you want to display only, say, a single graph in the slide show, create a screen containing just that one graph.
<i>Screen</i>	Screen name.
<i>Delay</i>	A custom value for how long the screen will be displayed, in seconds. If set to 0, the <i>Default delay</i> value will be used.
<i>Action</i>	Click on <i>Remove</i> to remove a screen from the slide show.

The slide show in this example consists of two screens which will be displayed in the following order:

Zabbix server ⇒ Displayed for 30 seconds ⇒ Zabbix server2 ⇒ Displayed for 15 seconds ⇒ Zabbix server ⇒ Displayed for 30 seconds ⇒ Zabbix server2 ⇒ ...

The **Sharing** tab contains the slide show type as well as sharing options (user groups, users) for private slide shows:

Slide

Sharing

Type

Private

Public

List of user group shares

USER GROUPS	PERMISSIONS	ACTION
Linux administrators	Read-only	Read-write
		Remove
Add		

List of user shares

USERS	PERMISSIONS	ACTION
guest	Read-only	Read-write
		Remove
Add		

Add

Cancel

Parameter	Description
Type	Select slide show type: Private - slide show is visible only to selected user groups and users Public - slide show is visible to all
List of user group shares	Select user groups that the slide show is accessible to. You may allow read-only or read-write access.
List of user shares	Select users that the slide show is accessible to. You may allow read-only or read-write access.

Click on *Add* to save the slide show.

Display

Slide shows that are ready can be viewed in *Monitoring → Screens*, then choosing *Slide shows* from the dropdown and clicking on the slide show name.

With the Menu option next to the dropdown, you can accelerate or slow down the display by choosing a slide delay multiplier:

REFRESH INTERVAL MULTIPLIER

x0.25

x0.5

✓ x1

x1.5

x2

x3

x4

x5

Attention:

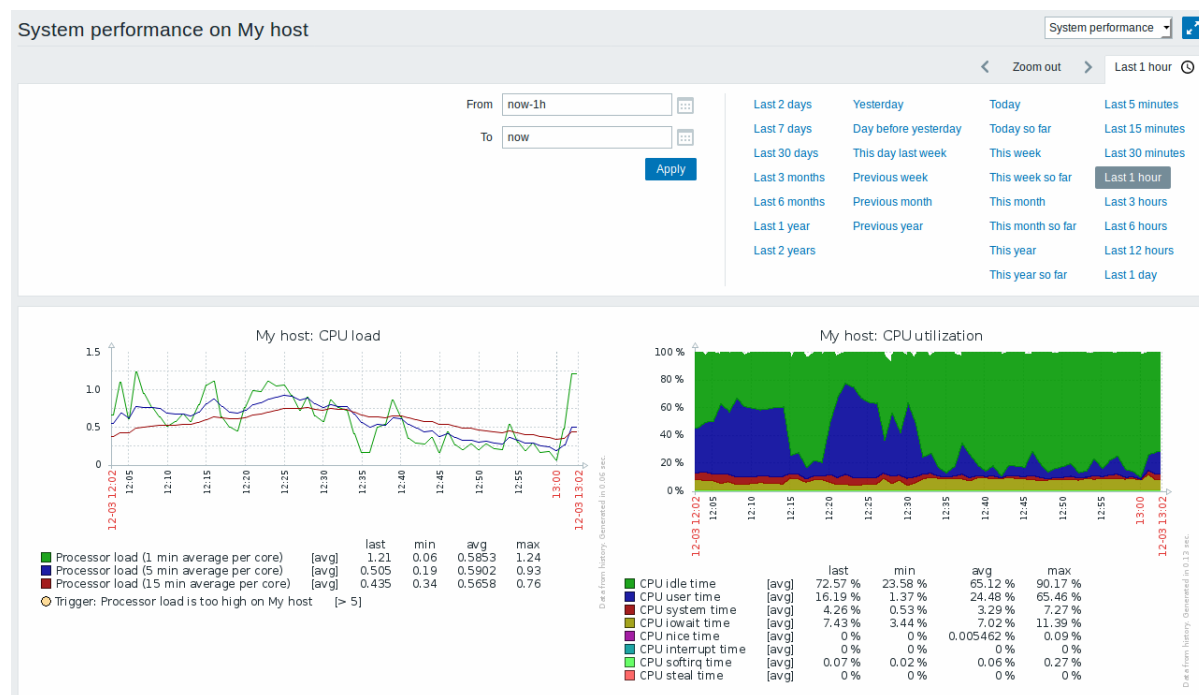
If a delay ends up as being less than 5 seconds (either by having entered a delay less than 5 seconds or by using the slide delay multiplier), a 5-second minimum delay will be used.

5 Host screens

Overview

Host screens look similar to **global screens**, however, host screens display data about the host only. Host screens are configured on the **template** level and then are generated for a host, once the template is linked to the host.

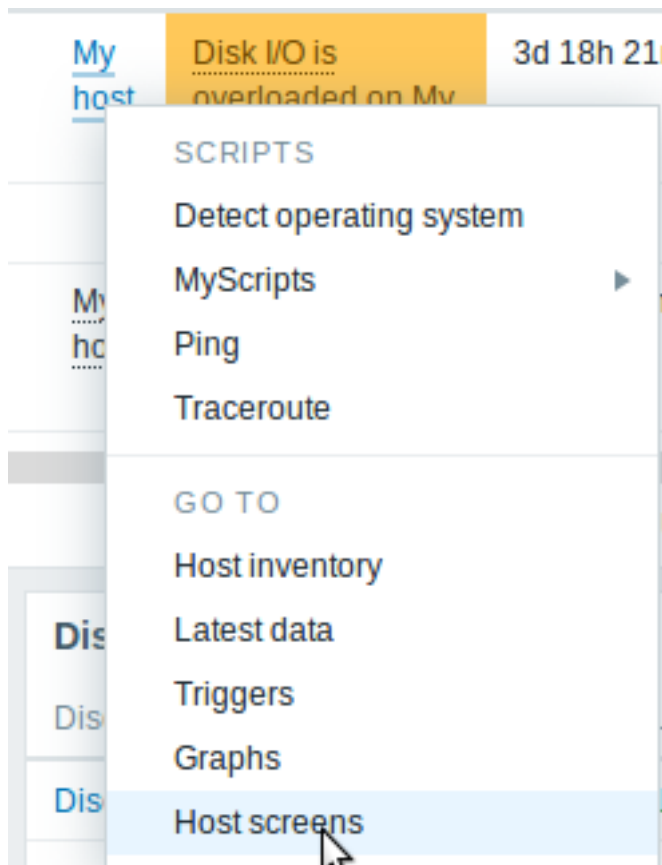
Host screens *cannot* be configured or directly accessed in the **Monitoring** → **Screens** section, which is reserved for global screens. The ways to access host screens are listed below in this section.



Accessing host screens

Access to host screens is provided:

- From the **host menu** that is available in many frontend locations:
 - click on the host name and then select **Host screens** from the drop-down menu



- When searching for a host name in **global search**:
 - click on the *Screens* link provided in search results
- When clicking on a host name in *Inventory* → **Hosts**:
 - click on the *Screens* link provided

7 Templates

Overview

A template is a set of entities that can be conveniently applied to multiple hosts.

The entities may be:

- items
- triggers
- graphs
- applications
- screens (*since Zabbix 2.0*)
- low-level discovery rules (*since Zabbix 2.0*)
- web scenarios (*since Zabbix 2.2*)

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.

Thus, the use of templates is an excellent way of reducing one's workload and streamlining the Zabbix configuration.

Proceed to [creating and configuring a template](#).

8 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.

1 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 `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)?[1L]oop[bB]ack[0-9._]*$
Result is FALSE: ^NULL[0-9._]*$
Result is FALSE: ^[1L]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	MIBs OS used	Tags
Template Net Alcatel Timetra TiMOS SNMPv2	Alcatel	Alcatel Timetra	ALCATEL SR 7750	TiMOS METRA-SYSTEM-MIB, TIMETRA-CHASSIS-MIB	Certified
Template Net Brocade FC switches SNMPv2	Brocade	Brocade FC switches	Brocade 300 SAN Switch-	- SW-MIB, ENTITY-MIB	Performance, Fault

Template name	Vendor	Device family	Known models	OS	MIBs used	Tags
<i>Template Net Brocade Foundry Stackable SNMPv2</i>	Brocade	Brocade ICX	Brocade ICX6610, Brocade ICX7250-48, Brocade ICX7450-48F		FOUNDRY-SN-AGENT-MIB, FOUNDRY-SN-STACKING-MIB	Certified
<i>Template Net Brocade Foundry Nonstackable SNMPv2</i>	Brocade Foundry	Brocade MLX, Foundry	Brocade MLXe, Foundry FLS648, Foundry FWSX424		FOUNDRY-SN-AGENT-MIB	Performance, Fault
<i>Template Net Cisco IOS SNMPv2</i>	Cisco	Cisco IOS ver > 12.2 3.5	Cisco C2950	IOS	CISCO-PROCESS-MIB, CISCO-MEMORY-POOL-MIB, CISCO-ENVMON-MIB	Certified
<i>Template Net Cisco releases later than 12.0_3_T and prior to 12.2_3.5_ SNMPv2</i>	Cisco	Cisco IOS > 12.0 3 T and 12.2 3.5	-	IOS	CISCO-PROCESS-MIB, CISCO-MEMORY-POOL-MIB, CISCO-ENVMON-MIB	Certified
<i>Template Net Cisco releases prior to 12.0_3_T SNMPv2</i>	Cisco	Cisco IOS 12.0 3 T	-	IOS	OLD-CISCO-CPU-MIB, CISCO-MEMORY-POOL-MIB	Certified
<i>Template Net D-Link DES_DGS Switch SNMPv2</i>	D-Link	DES/DGX switches	D-Link DES-xxxx/DGS-xxxx, DLINK DGS-3420-26SC	-	DLINK-AGENT-MIB, EQUIPMENT-MIB, ENTITY-MIB	Certified
<i>Template Net D-Link DES 7200 SNMPv2</i>	D-Link	DES-7xxx	D-Link DES 7206	-	ENTITY-MIB, MY-SYSTEM-MIB, MY-PROCESS-MIB, MY-MEMORY-MIB	Performance Fault Interfaces
<i>Template Net Dell Force S-Series SNMPv2</i>	Dell	Dell Force S-Series	S4810		F10-S-SERIES-CHASSIS-MIB	Certified

Template name	Vendor	Device family	Known models	MIBs OS used	Tags
Template Net Extreme Exos SNMPv2	Extreme	Extreme EXOS	X670V-48x	EXOS-EXTREME-SYSTEM-MIB,EXTREME-SOFTWARE-MONITOR-MIB	Certified
Template Net Huawei VRP SNMPv2	Huawei	Huawei VRP	S2352P-EI	- ENTITY-MIB,HUAWEI-ENTITY-EXTENT-MIB	Certified
Template Net Intel_Qlogic Infiniband SNMPv2	Intel/QLogic	Intel/QLogic Infiniband devices	Infiniband 12300	ICS-CHASSIS-MIB	Fault Inventory
Template Net Juniper SNMPv2	Juniper	MX,SRX,EX models	Juniper MX240, Juniper EX4200-24F	Juniper-OS MIB	Certified
Template Net Mellanox SNMPv2	Mellanox	Mellanox Infiniband devices	SX1036	MLNX-OS MIB,HOST-RESOURCES-MIB,ENTITY-MIB,ENTITY-SENSOR-MIB,MELLANOX-MIB	Certified
Template Net Mikrotik SNMPv2	Mikrotik	Mikrotik RouterOS devices	Mikrotik CCR1016-12G, Mikrotik RB2011UAS-2HnD, Mikrotik 912UAG-5HPnD, Mikrotik 941-2nD, Mikrotik 951G-2HnD, Mikrotik 1100AHx2	RouterOS MIB,HOST-RESOURCES-MIB	Certified
Template Net QTech QSW SNMPv2	QTech	Qtech devices	Qtech QSW-2800-28T	- QTECH-MIB,ENTITY-MIB	Performance Inventory
Template Net Ubiquiti AirOS SNMPv1	Ubiquiti	Ubiquiti AirOS wireless devices	NanoBridge, NanoStation	NanoOS-PRO-FOOTER-RESOURCES-MIB,IEEE802dot11-MIB	Performance
Template Net HP Comware HH3C SNMPv2	HP	HP (H3C) Comware	HP A5500-24G-4SFP HI Switch	HH3C-ENTITY-EXT-MIB,ENTITY-MIB	Certified

Template name	Vendor	Device family	Known models	OS	MIBs used	Tags
Template Net HP Enterprise Switch SNMPv2	HP	HP Enterprise Switch	HP ProCurve J4900B Switch 2626, HP J9728A 2920-48G Switch		STATISTICS Certified MIB,NETSWITCH- MIB,HP- ICF- CHASSIS,ENTITY- MIB,SEMI- MIB	
Template Net TP-LINK SNMPv2	TP- LINK	TP-LINK	T2600G- 28TS v2.0		TPLINK- SYSMONITOR- MIB,TPLINK- SYSINFO- MIB	Performance Inventory
Template Net Netgear Fastpath SNMPv2	Netgear	Netgear Fastpath	M5300- 28G		FASTPATH- Fault Inventory SWITCHING- MIB,FASTPATH- BOXSERVICES- PRIVATE- MIB	

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
- Templates are provided for SNMPv2. SNMPv1 is used if it is known that the majority of devices don't support SNMPv2.
- 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 controls its operational status(link).
 - If you do not want to monitor this condition for a specific interface create a user macro with context with the value 0.
For example:

The screenshot shows the Zabbix web interface with the 'Macros' tab selected. Under the 'Host macros' section, there is a table with two columns: 'Macro' and 'Value'. A single macro is listed with the name '{\$IFCONTROL: \"Gi0/0\"}' and the value '0'.

Macro	Value
{\$IFCONTROL: \"Gi0/0\"}	0

where Gi0/0 is {#IFNAME}. That way the trigger is not used any more for this specific interface.

- * You can also change the default behaviour for all triggers not to fire and activate this trigger only

Host
Templates
IPMI
Macros
Host inventory
Encryption

Host macros
Inherited and host macros

Macro	Value
{SIFCONTROL}	⇒ 0
{SIFCONTROL: "Gi0/0"}	⇒ 1
{SIFCONTROL: "Gi0/1"}	⇒ 1

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.

9 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 auto-registration 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 in Zabbix.

You can configure several media types:

- **E-mail**
- **SMS**
- **Jabber**
- **Ez Texting**
- **Custom alertscripts**

1 E-mail

Overview

Parameter	Description
<i>SMTP email</i>	The address entered here will be used as the From address for the messages sent. Adding a sender display name (like "Zabbix_info" in <i>Zabbix_info</i> <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: <i>zabbix@company.com</i> (only email address, no need to use angle brackets) <i>Zabbix_info</i> <zabbix@company.com> (display name and email address in angle brackets) <i>ΣQ-monitoring</i> <zabbix@company.com> (UTF-8 characters in display name) Invalid examples: <i>Zabbix HQ zabbix@company.com</i> (display name present but no angle brackets around email address) <i>"Zabbix\@ <H(comment)Q >"</i> <zabbix@company.com> (although valid by RFC 5322, quoted pairs and comments are not supported in Zabbix emails)
<i>Connection security</i>	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 <i>starting with Zabbix 3.0</i>.
<i>SSL verify peer</i>	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 <i>starting with Zabbix 3.0</i>.
<i>SSL verify host</i>	Mark the checkbox to verify that the <i>Common Name</i> field or the <i>Subject Alternate Name</i> field of the SMTP server certificate matches. This sets cURL option CURLOPT_SSL_VERIFYHOST. This option is supported <i>starting with Zabbix 3.0</i>.
<i>Authentication</i>	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 <i>starting with Zabbix 3.0</i>.
<i>Username</i>	User name to use in authentication. This sets the value of CURLOPT_USERNAME.
<i>Password</i>	Password to use in authentication. This sets the value of CURLOPT_PASSWORD.
<i>Message format</i>	This option is supported <i>starting with Zabbix 3.0</i>. Select message format: HTML - send as HTML Plain text - send as plain text
<i>Enabled</i>	Mark the checkbox to enable the media type.

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.

The **Options** tab in the e-mail media type **configuration** contains alert processing settings. The same set of options are configurable for other media types, too.

All media types are processed in parallel. The maximum number of concurrent sessions is configurable per media type, but the total number of alerter processes on server can only be limited by the StartAlerters **parameter**. Alerts generated by one trigger are processed sequentially.

Parameter	Description
<i>Concurrent sessions</i>	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.
<i>Attempts</i>	Number of attempts for trying to send a notification. Up to 10 attempts can be specified; default value is '3'. If '1' is specified Zabbix will send the notification only once and will not retry if the sending fails.
<i>Attempt interval</i>	Frequency of trying to resend a notification in case the sending failed, in seconds (0-60). If '0' is specified, Zabbix will retry immediately. Time suffixes are supported, e.g. 5s, 1m.

User media

To assign a specific address to the user:

- Go to *Administration* → *Users*
- Open the user properties form
- In Media tab, click on *Add*

Media

Type Email

* Send to

Recipient name <address@company.com>

Remove

address2@company.com

Remove

Add

* When active

1-7,00:00-24:00

Use if severity

☒ Not classified
☒ Information
☒ Warning
☒ Average
☒ High
☒ Disaster

Enabled

☒

Add

Cancel

User media attributes:

Parameter	Description
<i>Type</i>	Select <i>Email</i> as the type.
<i>Send to</i>	Specify e-mail addresses to send the messages to. To add more than one address click on <i>Add</i> below the address field. If multiple e-mail addresses are specified, one e-mail will be sent to all the specified recipients. You may add the recipient display name (like "Recipient name" in <i>Recipient name <address1@company.com></i> in the screenshot above) with the actual e-mail address. See examples and restrictions on display name and email address in media type attribute SMTP email description.
<i>When active</i>	You can limit the time when messages are sent, for example, the working days only (1-5,09:00-18:00). See the Time period specification page for description of the format. User macros are supported.
<i>Use if severity</i>	Mark the checkboxes of trigger severities that you want to receive notifications for. <i>Note</i> 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 colours while unselected ones will be greyed out.
<i>Status</i>	Status of the user media. Enabled - is in use. Disabled - is not being used.

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 **Media type** tab contains general media type attributes:

Parameter	Description
<i>Description</i>	Name of the media type.
<i>Type</i>	Select <i>SMS</i> as the type.
<i>GSM modem</i>	Set the serial device name of the GSM modem.

The **Options** tab contains alert **processing settings** that are common for all media types. Note that parallel processing of sending SMS notifications is not possible.

User media

To assign a phone number to the user:

- Go to *Administration* → *Users*
- Open the user properties form
- In Media tab, click on *Add*

User media attributes:

Parameter	Description
<i>Type</i>	Select <i>SMS</i> as the type.
<i>Send to</i>	Specify the phone number to send messages to.
<i>When active</i>	You can limit the time when messages are sent, for example, the working days only (1-5,09:00-18:00). See the Time period specification page for description of the format.
<i>Use if severity</i>	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 colours while unselected ones will be greyed out.
<i>Status</i>	Status of the user media. Enabled - is in use. Disabled - is not being used.

3 Jabber

Overview

Zabbix supports sending Jabber messages.

When sending notifications, Zabbix tries to look up the Jabber SRV record first, and if that fails, it uses an address record for that domain. Among Jabber SRV records, the one with the highest priority and maximum weight is chosen. If it fails, other records are not tried.

To configure Jabber as the delivery channel for messages, you need to configure Jabber as the media type and enter the respective addresses for the users.

Configuration

To configure Jabber as the media type:

- Go to *Administration* → *Media types*
- Click on *Create media type* (or click on *Jabber* in the list of pre-defined media types).

The **Media type** tab contains general media type attributes:

Parameter	Description
<i>Description</i>	Name of the media type.
<i>Type</i>	Select <i>Jabber</i> as the type.
<i>Jabber identifier</i>	Enter Jabber identifier.
<i>Password</i>	Enter Jabber password.

The **Options** tab contains alert **processing settings** that are common for all media types.

User media

To assign a Jabber address to the user:

- Go to *Administration* → *Users*
- Open the user properties form
- In Media tab, click on *Add*

User media attributes:

Parameter	Description
<i>Type</i>	Select <i>Jabber</i> as the type.
<i>Send to</i>	Specify the address to send messages to.
<i>When active</i>	You can limit the time when messages are sent, for example, the working days only (1-5,09:00-18:00). See the Time period specification page for description of the format.
<i>Use if severity</i>	Mark the checkboxes of trigger severities that you want to receive notifications for. <i>Note</i> 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 colours while unselected ones will be greyed out.
<i>Status</i>	Status of the user media. Enabled - is in use. Disabled - is not being used.

4 Ez Texting

Overview

You can use [Zabbix technological partner](#) Ez Texting for message sending.

To configure Ez Texting as the delivery channel for messages, you need to configure Ez Texting as the media type and assign recipient identification to the users.

Configuration

To configure Ez Texting as the media type:

- Go to *Administration* → *Media types*
- Click on *Create media type*

The **Media type** tab contains general media type attributes:

Media types

* Name

Type <https://app.eztexting.com>

* Username

Password

Message text limit

Enabled ☒

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Description</i>	Name of the media type.
<i>Type</i>	Select <i>Ez Texting</i> as the type.
<i>Username</i>	Enter the Ez Texting username.
<i>Password</i>	Enter the Ez Texting password.
<i>Message text limit</i>	Select the message text limit. USA (160 characters) Canada (136 characters)

The **Options** tab contains alert **processing settings** that are common for all media types.

User media

To assign Ez Texting recipient identification to the user:

- Go to *Administration* → *Users*
- Open the user properties form
- In *Media* tab, click on *Add*

User media attributes:

Parameter	Description
<i>Type</i>	Select the Ez Texting media type.
<i>Send to</i>	Specify the recipient to send the messages to.
<i>When active</i>	You can limit the time when messages are sent, for example, the working days only (1-5,09:00-18:00). See the Time period specification page for description of the format.

Parameter	Description
<i>Use if severity</i>	Mark the checkboxes of trigger severities that you want to receive notifications for. <i>Note</i> 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 colours while unselected ones will be greyed out.
<i>Status</i>	Status of the user media. Enabled - is in use. Disabled - is not being used.

5 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:

* Name

Type

* Script name

Script parameters

PARAMETER	ACTION
{ALERT.SENDTO}	Remove
{ALERT.SUBJECT}	Remove
{ALERT.MESSAGE}	Remove
	Remove
Add	

Enabled ☒

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Name</i>	Enter name of the media type.
<i>Type</i>	Select <i>Script</i> as the type.
<i>Script name</i>	Enter the name of the script.
<i>Script parameters</i>	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.

The **Options** tab contains alert **processing settings** that are common for all media types.

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

To assign custom alertscripts to the user:

- Go to *Administration* → *Users*
- Open the user properties form
- In Media tab, click on *Add*

User media attributes:

Parameter	Description
<i>Type</i>	Select the custom alertscripts media type.
<i>Send to</i>	Specify the recipient to receive the alerts.

Parameter	Description
<i>When active</i>	You can limit the time when alertscripts are executed, for example, the working days only (1-5,09:00-18:00). See the Time period specification page for description of the format.
<i>Use if severity</i>	Mark the checkboxes of trigger severities that you want to activate the alertscript for. <i>Note</i> 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 colours while unselected ones will be greyed out.
<i>Status</i>	Status of the user media. Enabled - is in use. Disabled - is not being used.

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 events - when trigger status changes from *OK* to *PROBLEM* and back
- Discovery events - when network discovery takes place
- Auto registration events - when new active agents auto-register (or host metadata changes for registered ones)
- Internal 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*
- From the *Event source* dropdown select the required source
- Click on *Create action*
- Name the action
- Choose **conditions** upon which operations are carried out
- Choose the **operations** to carry out
- Choose the **recovery operations** to carry out

General action attributes:

Actions

Action

Operations

Recovery operations

Update operations

*

Name

Report problems to Zabbix administrators

Type of calculation

And/Or

A and B

Conditions

Label

Name

A

Problem is not suppressed

B

Host group equals Zabbix servers

New condition

Host group

equals

type here to search

Add

Enabled

☒

*

At least one operation, recovery operation or update operation must exist.

Add

Cancel

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Name</i>	Unique action name.
<i>Type of calculation</i>	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.
<i>Conditions</i>	List of action conditions.
<i>New condition</i>	Select a new action condition and click on <i>Add</i> .
<i>Enabled</i>	Mark the checkbox to enable the action. Otherwise it will be disabled.

1 Conditions

Overview

An action is executed only in case an event matches a defined set of conditions. Conditions are set when configuring an **action**.

Trigger actions

The following conditions can be set for trigger-based actions:

Condition type	Supported operators	Description
<i>Application</i>	equals contains does not contain	Specify an application or an application to exclude. equals - event belongs to a trigger of the item that is linked to the specified application. contains - event belongs to a trigger of the item that is linked to an application containing the string. does not contain - event belongs to a trigger of the item that is linked to an application not containing the string.
<i>Host group</i>	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.
<i>Template</i>	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.
<i>Host</i>	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.
<i>Tag</i>	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

Condition type	Supported operators	Description
<i>Tag value</i>	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
<i>Trigger</i>	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.
<i>Trigger name</i>	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. Case sensitive. does not contain - this string cannot be found in the trigger name. Case sensitive. <i>Note:</i> Entered value will be compared to trigger name with all macros expanded.
<i>Trigger severity</i>	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
<i>Time period</i>	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.

Condition type	Supported operators	Description
<i>Problem is suppressed</i>	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 set for discovery-based events:

Condition type	Supported operators	Description
<i>Host IP</i>	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.
<i>Service type</i>	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 (<i>available since Zabbix 2.2 version</i>), POP, NNTP, IMAP, TCP, Zabbix agent, SNMPv1 agent, SNMPv2 agent, SNMPv3 agent, ICMP ping, telnet (<i>available since Zabbix 2.2 version</i>).
<i>Service port</i>	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.

Condition type	Supported operators	Description
<i>Discovery rule</i>	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.
<i>Discovery check</i>	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.
<i>Discovery object</i>	equals	Specify the discovered object. equals - equal to discovered object (a device or a service).
<i>Discovery status</i>	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
<i>Uptime/Downtime</i>	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.

Condition type	Supported operators	Description
<i>Received value</i>	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. Case sensitive 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.
<i>Proxy</i>	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.

Auto-registration actions

The following conditions can be set for actions based on active agent auto-registration:

Condition type	Supported operators	Description
<i>Host metadata</i>	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.

Condition type	Supported operators	Description
<i>Host name</i>	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.
<i>Proxy</i>	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
<i>Application</i>	equals contains does not contain	Specify an application or an application to exclude. equals - event belongs to an item that is linked to the specified application. contains - event belongs to an item that is linked to an application containing the string. does not contain - event belongs to an item that is linked to an application not containing the string.
<i>Event type</i>	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
<i>Host group</i>	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.

Condition type	Supported operators	Description
<i>Template</i>	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.
<i>Host</i>	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 (including IPMI)

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 auto-registration 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** and click on *New* in the Operations block. Edit the operation step and click on *Add* to add to the list of *Operations*.

Operation attributes:

Action
Operations
Recovery operations
Update operations

* Default operation step duration
1h

Default subject
Problem: {EVENT.NAME}

Default message
Problem started at {EVENT.TIME} on {EVENT.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME}
Severity: {EVENT.SEVERITY}

Original problem ID: {EVENT.ID}
{TRIGGER.URL}

Pause operations for suppressed problems
☒

Operations

Steps	Details	Start in	Duration	Action
1	Send message to user groups: Zabbix administrators1 via Email	Immediately	Default	Edit Remove
3	Send message to user groups: Managers via SMS	02:00:00	Default	Edit Remove
4	Run remote commands on current host	03:00:00	Default	Edit Remove

Operation details

Steps
 - (0 - infinitely)

Step duration
 (0 - use action default)

Operation type
Send message

* At least one user or user group must be selected.

Send to User groups

User group	Action
Managers	Remove
Add	

Send to Users

User	Action
Add	

Send only to
SMS

Default message
☒

Conditions

Label	Name	Action
A	Event acknowledged equals Not Ack	Remove
New		

[Update](#) [Cancel](#)

* At least one operation, recovery operation or update operation must exist.

Add

Cancel

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Default operation step duration</i>	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.
<i>Default subject</i>	User macros are supported, since Zabbix 3.4.0. Default message subject for notifications. The subject may contain macros . It is limited to 255 characters.
<i>Default message</i>	Default message for notifications. 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).

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. If you unmark this checkbox, operations will be executed without delay even during a maintenance period. This option is supported since Zabbix 3.2.0. Action operations are displayed, with these details: Steps - escalation step(s) to which the operation is assigned Details - type of operation and its recipient/target. Since Zabbix 2.2, the operation list also displays the media type (e-mail, SMS, Jabber, etc) used in sending a message as well as the name and surname (in parentheses after the alias) of a notification recipient. Start in - how long after an event the operation is performed Duration (sec) - step duration is displayed. <i>Default</i> 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. To configure a new operation, click on <i>New</i>.	
Operations		
Operation details	Steps	This block is used to configure the details of an operation. Select the step(s) to assign the operation to in an escalation schedule: From - execute starting with this step To - execute until this step (0=infinity, execution will not be limited)
	Step duration	Custom duration for these steps (0=use default step duration). Time suffixes are supported, e.g. 60s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. Several operations can be assigned to the same step. If these operations have different step duration defined, the shortest one is taken into account and applied to the step.
	Operation type	Two operation types are available for all events: Send message - send message to user Remote command - execute a remote command More operations are available for discovery and auto-registration based events (see above).
	Operation type: send message	
	Send to user groups	Click on <i>Add</i> 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.

Parameter	Description
<i>Send to users</i>	Click on <i>Add</i> to select users to send the message to. The user must have at least "read" permissions to the host in order to be notified.
<i>Send only to</i>	Send message to all defined media types or a selected one only.
<i>Default message</i>	If selected, the default message will be used (see above).
<i>Subject</i>	Subject of the custom message. The subject may contain macros. It is limited to 255 characters.
<i>Message</i>	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).
<i>Operation type: re-mote command</i>	
<i>Target list</i>	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 <i>Host group</i> setting in global script configuration .
<i>Type</i>	Select the command type: IPMI - execute an IPMI command Custom script - execute a custom set of commands SSH - execute an SSH command Telnet - execute a Telnet command Global script - execute one of the global scripts defined in <i>Administration→Scripts</i> .

Parameter		Description
	<i>Execute on</i>	Execute a custom script on: Zabbix agent - the script will be executed by Zabbix agent on the host Zabbix server (proxy) - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy Zabbix server - the script will be executed by Zabbix server only To execute scripts on the agent, it must be configured (<i>EnableRemoteCommands</i> parameter enabled) to allow remote commands from the server. To execute scripts on proxy, it must be configured (<i>EnableRemoteCommands</i> parameter enabled) to allow remote commands from the server. This field is available if 'Custom script' is selected as <i>Type</i> .
	<i>Commands</i>	Enter the command(s). Supported macros will be resolved based on the trigger expression that caused the event. For example, host macros will resolve to the hosts of the trigger expression (and not of the target list).
	<i>Conditions</i>	Condition for performing the operation: Not ack - only when the event is unacknowledged Ack - only when the event is acknowledged.

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

Warning:

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 auto-registration or internal events.

- **configure an action operation** that sends a message to one of the defined media

Attention:

Zabbix sends notifications only to those users that have at least 'read' permissions to the host that generated the event. At least one host of a trigger expression must be accessible.

You can configure custom scenarios for sending messages using **escalations**.

To successfully receive and read 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 setting the `EnableRemoteCommands` parameter to '1'.

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
IBM DB2	2048	2048
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 respective `zabbix_agentd.conf`.

Make sure that the **EnableRemoteCommands** parameter is set to **1** and uncommented. 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:

Actions

Action Operations Recovery operations Update operations

* Name	Serious problem with Apache									
Type of calculation	And/Or	A and B and C								
Conditions	<table><thead><tr><th>Label</th><th>Name</th></tr></thead><tbody><tr><td>A</td><td>Problem is not suppressed</td></tr><tr><td>B</td><td>Application contains <i>Apache</i></td></tr><tr><td>C</td><td>Trigger severity is greater than or equals <i>Disaster</i></td></tr></tbody></table>		Label	Name	A	Problem is not suppressed	B	Application contains <i>Apache</i>	C	Trigger severity is greater than or equals <i>Disaster</i>
Label	Name									
A	Problem is not suppressed									
B	Application contains <i>Apache</i>									
C	Trigger severity is greater than or equals <i>Disaster</i>									

All mandatory input fields are marked with a red asterisk.

- In the *Operations* tab, select the **Remote command** operation type
- Select the remote command type (IPMI, Custom script, SSH, Telnet, Global script)
- If *Custom script* type is selected choose the way how custom script will be executed (by Zabbix agent, Zabbix server (proxy) or Zabbix server only)
- Enter the remote command

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 remote commands are executed with timeout as set in the `TrapperTimeout` parameter of `zabbix_server.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

Example 1

Restart of Windows on certain condition.

In order to automatically restart Windows upon a problem detected by Zabbix, define the following actions:

PARAMETER	Description
Operation type	'Remote command'
Type	'Custom script'
Command	c:\windows\system32\shutdown.exe -r -f

Example 2

Restart the host by using IPMI control.

PARAMETER	Description
Operation type	'Remote command'
Type	'IPMI'
Command	reset

Example 3

Power off the host by using IPMI control.

PARAMETER	Description
Operation type	'Remote command'
Type	'IPMI'
Command	power off

3 Additional operations

Overview

In this section you may find some details of **additional operations** for discovery/auto-registration 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.

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: {zabbix.zabbix.com:system.cpu.load[,avg1].last()}

When you receive the message, the message will be replaced by something like:

Processor load is: 1.45

Example 3

Message:

Latest value: {{HOST.HOST}}:{{ITEM.KEY}}.last()}

MAX for 15 minutes: {{HOST.HOST}}:{{ITEM.KEY}}.max(900)}

MIN for 15 minutes: {{HOST.HOST}}:{{ITEM.KEY}}.min(900)}

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/events.php?triggerid={TRIGGER.ID}&filter_set=1

When you receive the message, it will contain a link to all events of the problem 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: {Myhost:system.cpu.load[percpu,avg1].last()}>5 or {Myotherhost:system.cpu.load[percpu,

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 acknowledgement: {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}

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 acknowledgement: 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

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. Recovery operations do not support escalating - all operations are assigned to a single step.

Use cases

Some use cases for recovery operations are as follows:

1. Notify all users that were notified on the problem
 - * Select 'Send recovery message' as operation type
- Have multiple operations upon recovery: send a notification and execute a remote command
 - * Add operation types for sending a message and executing a command
- 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 *Recovery operations* tab in action **configuration**
- Click on *New* in the Operations block
- Edit the operation details and click on *Add*

Several operations can be added.

Recovery operation attributes:

Action
Operations
Recovery operations
Update operations

Default subject
Resolved: {EVENT.NAME}

Default message
Problem has been resolved at {EVENT.RECOVERY.TIME} on {EVENT.RECOVERY.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME}
Severity: {EVENT.SEVERITY}

Original problem ID: {EVENT.ID}
{TRIGGER.URL}

Operations

Details
Action

Notify all involved
Edit Remove

Run remote commands on current host
Edit Remove

Operation details

Operation type
Remote command

* Target list

Target
Current host
New

Action
Remove

Type
Custom script

Execute on
Zabbix agent
Zabbix server (proxy)
Zabbix server

* Commands
sudo /etc/init.d/apache2 restart

Update Cancel

* At least one operation, recovery operation or update operation must exist.

Add Cancel

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Default subject</i>	Default message subject for recovery notifications. The subject may contain macros .
<i>Default message</i>	Default message for recovery notifications. The message may contain macros .
<i>Operations</i>	Recovery operation details are displayed. To configure a new recovery operation, click on New .
<i>Operation details</i>	This block is used to configure the details of a recovery operation.

Parameter	Description
Operation type	Three operation types are available for recovery events: Send message - send recovery message to specified user Remote command - execute a remote command Notify all involved - send recovery message to all users who were notified on the problem event Note that if the same recipient with unchanged default subject/message is defined in several operation types, duplicate notifications are not sent.
Operation type: send message	
Send to user groups	Click on <i>Add</i> 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 <i>Add</i> 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 recovery message to all defined media types or a selected one only.
Default message	If selected, the default message will be used (see above).
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
	<i>Target list</i>	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 <i>Host group</i> setting in global script configuration.
	<i>Type</i>	Select the command type: IPMI - execute an IPMI command Custom script - execute a custom set of commands SSH - execute an SSH command Telnet - execute a Telnet command Global script - execute one of the global scripts defined in <i>Administration→Scripts</i>.
	<i>Execute on</i>	Execute a custom script on: Zabbix agent - the script will be executed by Zabbix agent on the host Zabbix server (proxy) - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy Zabbix server - the script will be executed by Zabbix server only To execute scripts on the agent, it must be configured to allow remote commands from the server. This field is available if 'Custom script' is selected as <i>Type</i>.
	<i>Commands</i>	Enter the command(s). Supported macros will be resolved based on the trigger expression that caused the event. For example, host macros will resolve to the hosts of the trigger expression (and not of the target list).
	<i>Operation type:</i> no-tify all involved	
	<i>Default message</i>	If selected, the default message will be used (see above).
	<i>Subject</i>	Subject of the custom message. The subject may contain macros.

Parameter	Description
<i>Message</i>	The custom message. The message may contain macros.

4 Update operations

Overview

Update operations allow you to be notified when problems are **updated** by other users, i.e.:

- commented upon
- acknowledged
- severity changed
- closed (manually)

Update operations are available in actions with the event source as *Triggers*.

Both messages and remote commands are supported in update operations. Update operations do not support escalating - all operations are assigned to a single step.

Configuring an update operation

To configure an update operation:

- Go to the *Update operations* tab in action **configuration**
- Click on *New* in the Operations block
- Edit the operation details and click on *Add*

Several operations can be added.

Update operation attributes:

Action
Operations
Recovery operations
Update operations

Default subject
Updated problem: {EVENT.NAME}

Default message
{USER.FULLNAME} {EVENT.UPDATE.ACTION} problem at {EVENT.UPDATE.DATE} {EVENT.UPDATE.TIME}: {EVENT.UPDATE.MESSAGE}

Current problem status is {EVENT.STATUS}, acknowledged: {EVENT.ACK.STATUS}

Operations
Details
Notify all involved

Send message to user groups: Zabbix administrators via SMS

Operation details

Operation type
Notify all involved

Default media type
- All -

Default message
☒

[Update](#)
[Cancel](#)

* At least one operation, recovery operation or update operation must exist.

Add
Cancel

Parameter	Description
<i>Default subject</i>	Default message subject for update notifications. The subject may contain macros .
<i>Default message</i>	Default message for update notifications. The message may contain macros .
<i>Operations</i>	Update operation details are displayed. To configure a new update operation, click on <i>New</i> .
<i>Operation details</i>	This block is used to configure the details of an update operation.

Parameter	Description
Operation type	Three operation types are available for update operations: Send message - send update message to specified user when event is updated, for example, acknowledged Remote command - execute a remote command when event is updated, for example, acknowledged Notify all involved - send notification message to all users who received notification about the problem appearing and/or have updated the problem event. If the same recipient with unchanged default subject/message is defined in several operation types, duplicate notifications are not sent. The person who updates a problem does not receive notification about their own update.
Operation type: send mes- sage	
Send to user groups	Click on <i>Add</i> to select user groups to send the update message to.
Send to users	The user group must have at least "read" permissions to the host in order to be notified.
Send only to	Click on <i>Add</i> to select users to send the update message to.
Default message	The user must have at least "read" permissions to the host in order to be notified.
Subject	Send update message to all defined media types or a selected one only.
Message	If selected, the default message will be used (see above).
Operation type: re- mote com- mand	Subject of the custom message. The subject may contain macros.
	The custom message. The message may contain macros.

Parameter		Description
	<i>Target list</i>	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 <i>Host group</i> setting in global script configuration.
	<i>Type</i>	Select the command type: IPMI - execute an IPMI command Custom script - execute a custom set of commands SSH - execute an SSH command Telnet - execute a Telnet command Global script - execute one of the global scripts defined in <i>Administration→Scripts</i>.
	<i>Execute on</i>	Execute a custom script on: Zabbix agent - the script will be executed by Zabbix agent on the host Zabbix server (proxy) - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy Zabbix server - the script will be executed by Zabbix server only To execute scripts on the agent, it must be configured to allow remote commands from the server. This field is available if 'Custom script' is selected as <i>Type</i>.
	<i>Commands</i>	Enter the command(s). Supported macros will be resolved based on the trigger expression that caused the event. For example, host macros will resolve to the hosts of the trigger expression (and not of the target list).
	Operation type: no- tify all in- volved	

Parameter	Description
<i>Default media type</i>	Users who update a problem but have not received notifications about the problem appearing will receive notifications about further updates on the selected default media type - Email, Jabber or SMS. This field is available since Zabbix 3.4.2.
<i>Default message</i>	If selected, the default message will be used (see above).
<i>Subject</i>	Subject of the custom message. The subject may contain macros.
<i>Message</i>	The custom message. The message may contain macros.

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 behaviour

Let's consider what happens in different circumstances if an action contains several escalation steps.

Situation	Behaviour
<i>The host in question goes into maintenance after the initial problem notification is sent</i>	Depending on the <i>Pause operations for suppressed problems</i> 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.
<i>The time period defined in the Time period action condition ends after the initial notification is sent</i>	All remaining escalation steps are executed. The <i>Time period</i> condition cannot stop operations; it has effect with regard to when actions are started/not started, not operations.
<i>A problem starts during maintenance and continues (is not resolved) after maintenance ends</i>	Depending on the <i>Pause operations for suppressed problems</i> setting in action configuration , all escalation steps are executed either from the moment maintenance ends or immediately.
<i>A problem starts during a no-data maintenance and continues (is not resolved) after maintenance ends</i>	It must wait for the trigger to fire, before all escalation steps are executed.
<i>Different escalations follow in close succession and overlap</i>	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.

Situation	Behaviour
<i>During an escalation in progress (like a message being sent), based on any type of event:
- the action is disabled
- the event is deleted
Based on trigger event:
- the trigger is disabled or deleted
- the host or item is disabled
Based on internal event about triggers:
- the trigger is disabled or deleted
Based on internal event about items/low-level discovery rules:
- the item is disabled or deleted
- the host is disabled</i>	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: <i>Escalation cancelled</i>) naming the reason (for example, NOTE: <i>Escalation cancelled: action '<Action name>' disabled</i>). This way the recipient is informed that the escalation is cancelled 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 <i>Escalation cancelled</i> message is also sent if operations are finished, but recovery operations are configured and are not executed yet.
<i>During an escalation in progress (like a message being sent) the action is deleted</i>	No more messages are sent. The information is logged to the server log file (starting from Debug Level 3=Warning), for example: <code>escalation cancelled: action id:334 deleted</code>

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

ActionOperationsRecovery operationsUpdate operations

* Default operation step duration30m

Default subject

Problem: {EVENT.NAME}

Default message

Problem started at {EVENT.TIME} on {EVENT.DATE}
 Problem name: {TRIGGER.NAME}
 Host: {HOST.NAME}
 Severity: {EVENT.SEVERITY}

 Original problem ID: {EVENT.ID}
 {TRIGGER.URL}

Pause operations while in maintenance

☒

Operations

StepsDetailsStart inDuration

1 - 5 **Send message to user groups:** MySQL Administrators via all media Immediately Default

New

* At least one operation, recovery operation or update operation must exist.

All mandatory input fields are marked with a red asterisk.

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' seconds (10 hours)
- Set the escalation steps to be *From '2' To '2'*

[Action](#) [Operations](#) [Recovery operations](#) [Update operations](#)

* Default operation step duration10h

Default subjectProblem: {EVENT.NAME}

Default messageProblem started at {EVENT.TIME} on {EVENT.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME}
Severity: {EVENT.SEVERITY}

Original problem ID: {EVENT.ID}
{TRIGGER.URL}

Pause operations while in maintenance☒

Operations

Steps	Details	Start in	Duration
2	Send message to user groups: Managers via Email	10:00:00	Default

New

* At least one operation, recovery operation or update operation must exist.

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.

Action
Operations
Recovery operations
Update operations

* Default operation step duration
30m

Default subject
Problem: {EVENT.NAME}

Default message
Problem started at {EVENT.TIME} on {EVENT.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME}
Severity: {EVENT.SEVERITY}

Original problem ID: {EVENT.ID}
{TRIGGER.URL}

Pause operations for suppressed problems
☒

Operations

Steps	Details	Start in	Duration	Action
1 - 0	Send message to user groups: MySQL Administrators via Email	Immediately	Default	Edit Remove
5	Send message to users: Database manager (Mr Swift) via all media	02:00:00	Default	Edit Remove

Operation details

Steps
5 - 5 (0 - infinitely)

Step duration
0 (0 - use action default)

Operation type
Send message

* At least one user or user group must be selected.

Send to User groups

User group	Action
Add	

Send to Users

User	Action
Database manager (Mr Swift)	Remove
Add	

Send only to
- All -

Default message
☐

Subject
Unacknowledged problem: {EVENT.NAME}

Message
Problem started at {EVENT.TIME} on {EVENT.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME}
Severity: {EVENT.SEVERITY}

Original problem ID: {EVENT.ID}
{TRIGGER.URL}
{ESC.HISTORY}

Conditions

Label	Name	Action
A	Event acknowledged equals Not Ack	Remove
New		

Note the use of {ESC.HISTORY} macro in the 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.

Action
Operations
Recovery operations
Update operations

* Default operation step duration
30m

Default subject
Problem: {EVENT.NAME}

Default message
Problem started at {EVENT.TIME} on {EVENT.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME}
Severity: {EVENT.SEVERITY}

Original problem ID: {EVENT.ID}
{TRIGGER.URL}

Pause operations while in maintenance
☒

Operations

Steps	Details	Start in	Duration
1 - 0	Send message to user groups: MySQL Administrators via Email	Immediately	Default
5	Send message to users: Database manager (Mr Swift) via all media	02:00:00	Default
6	Run remote commands on current host	02:30:00	Default
7	Send message to user groups: Guests via all media	03:00:00	Default
9	Run remote commands on current host	04:00:00	Default
New			

* At least one operation, recovery operation or update operation must exist.

Example 5

An escalation with several operations assigned to one step and custom intervals used. The default operation step duration is 30 minutes.

Action
Operations
Recovery operations
Update operations

* Default operation step duration
30m

Default subject
Problem: {EVENT.NAME}

Default message
Problem started at {EVENT.TIME} on {EVENT.DATE}
Problem name: {TRIGGER.NAME}
Host: {HOST.NAME}
Severity: {EVENT.SEVERITY}

Original problem ID: {EVENT.ID}
{TRIGGER.URL}

Pause operations while in maintenance
☒

Operations

Steps	Details	Start in	Duration
1 - 4	Send message to user groups: MySQL Administrators via Email	Immediately	Default
5 - 6	Send message to users: Database manager (Mr Swift) via all media	02:00:00	1h
5 - 7	Send message to user groups: Zabbix administrators via Email	02:00:00	10m
11	Send message to user groups: Guests via Email	04:00:00	Default
New			

* At least one operation, recovery operation or update operation must exist.

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

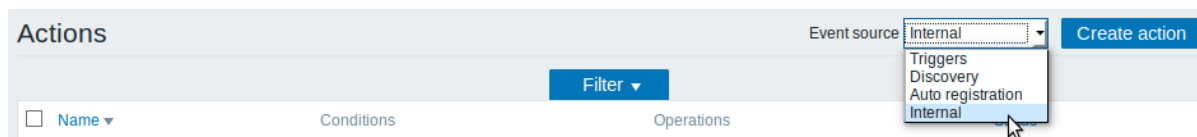
Configure **some media**, such as e-mail, SMS or Jabber, to use for the notifications. Refer to the corresponding sections of the manual to perform this task.

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* as the event source. Click on *Create action* on the upper right to open an action configuration form.



Step 3

In the **Action** tab enter a name for the action. Then select *Event type* in the New condition block and select *Item in "not supported" state* as the value.

Don't forget to click on *Add* to actually list the condition in the *Conditions* block.

Step 4

In the **Operations** tab, enter the subject/content of the problem message.

Click on *New* in the *Operations* block and select some recipients of the message (user groups/users) and the media types (or 'All') to use for delivery.

Actions

[Action](#)[Operations](#)[Recovery operations](#)

Default operation step duration

3600

(minimum 60 seconds)

Default subject

{ITEM.STATE}: {HOST.NAME}: {ITEM.NAME}

Default message

Host: {HOST.NAME}
Item: {ITEM.NAME}
Item key: {ITEM.KEY}
State: {ITEM.STATE}
Problem event: {EVENT.ID}
So far: {ESC.HISTORY}

Operations

StepsDetails

1 - 2Send message to user groups: Zabbix administrators via Email

Operation details

Steps

1

 -

2

 (0 - infinitely)

Step duration

300

 (minimum 60 seconds, 0 - use action)

Operation typeSend message

Send to User groups

User group	Action
Zabbix administrators	Remove
Add	

Send to Users

User	Action
Add	

Send only to

Email

Default message☒

[Update](#) [Cancel](#)

Click on *Add* in the *Operation details* block 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

operation.

Step 5

The **Recovery operations** tab allows to configure a recovery notification when an item goes back to the normal state.

Enter the subject/content of the recovery message.

Click on *New* in the *Operations* block and select some recipients of the message (user groups/users) and the media types (or 'All') to use for delivery.

Actions

[Action](#)[Operations](#)[Recovery operations](#)

Default subject

{ITEM.STATE}: {HOST.NAME}: {ITEM.NAME}

Default message

Host: {HOST.NAME}
Item: {ITEM.NAME}
Item key: {ITEM.KEY}
State: {ITEM.STATE}
Recovery event: {EVENT.RECOVERY.ID}

Operations

Details
Notify all who received any messages regarding the problem before

Operation details

Operation type

Send recovery message

Default message

☒

Update

Cancel

Add

Cancel

Click on *Add* in the *Operation details* block to actually list the operation in the *Operations* block.

Step 6

When finished, click on the **Add** button underneath 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.

10 Macros

Overview

Zabbix supports a number of macros which may be used in various situations. 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.

See also:

- full list of supported macros
- macro functions
- how to configure user macros

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:

```
{{ITEM.VALUE}.regsub(pattern, output)}  
{{#LLDMACRO}.regsub(pattern, output)}
```

Supported macro functions

FUNCTION	Description	Parameters	Supported for
regsub (<pattern>,<output>)	Substring extraction by a regular expression match (case sensitive).	pattern - the regular expression to match output - the output options. \1 - \9 place-holders 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)

FUNCTION

iregsub (<pattern>,<output>)

Substring
extrac-
tion by a
regular
expres-
sion
match
(case
insensi-
tive).

pattern
- the
regular
expres-
sion to
match
the
output
options.
\1 - \9
place-
holders
are sup-
ported
to
capture
groups.
\0
returns
the
matched
text.

{ITEM.VALUE}
{ITEM.LASTVALUE}
Low-
level
discov-
ery
macros
(except
in
low-level
discov-
ery 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 containing log lines as received value:

Received value	Macro	Output
123Log line	{{ITEM.VALUE}.regexsub('Problem ID: \1'), Problem)}	123 Problem ID: \1
123 Log line	{{ITEM.VALUE}.regexsub('Problem ID: \1'), "Problem")}	123 Problem ID: \1
123 Log line	{{ITEM.VALUE}.regexsub('Problem ID: \1'), "Problem ID: \1")}	123 Problem ID: \1
MySQL crashed errno 123	{{ITEM.VALUE}.regexsub('Problem ID: \1'), "Problem ID: \1")}	MySQL crashed errno 123 Problem ID: \1
123 Log line	{{ITEM.VALUE}.regexsub('Problem ID: \1'), "Problem ID: \1")}	123 Problem ID: \1
customername_1	{{#IFALIASES}.regexsub('customername_1', \1)}	customername_1
customername_1	{{#IFALIASES}.regexsub('1.*_([0-9]+)', \2)}	customername_1
customername_1	{{#IFALIASES}.regexsub('1.*_([0-9]+)', \1)}	customername_1

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.

If Zabbix is unable to find a macro, the macro will not be resolved.

Attention:

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 names
- item key parameters
- item update intervals and flexible intervals
- trigger names and descriptions
- 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 locations 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, exporting the template to XML and importing it in another system will still allow it to work as expected.

The following characters are allowed in the macro names: **A-Z** , **0-9** , **_** , **.**

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:

```
{ ca_001:system.cpu.load[,avg1].last() } > { $MAX_CPULOAD }
```

Such a trigger would be created on the template, not edited in individual hosts.

Note:

If you want to use amount of values as the function parameter (for example, **max(#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:

```
{ca_001:system.cpu.load[,avg1].min({$CPULOAD_PERIOD})}>{$MAX_CPULOAD}
```

Note that a macro can be used as a parameter of trigger function, in this example function **min()**.

Attention:

In trigger expressions user macros will resolve if referencing a parameter or constant. They will NOT resolve if referencing the host, item key, function, operator or another trigger expression.

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:

```
{ca_001:agent.ping.nodata({$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 *Administration* → *General* → *Working time*;
- use it in *User* → *Media* → *When active*;
- use it to set up more frequent item polling during working hours:

Update interval

Custom intervals

Type	Interval	Period
Flexible	Scheduling	{\$SHORT_INTERVAL} {\$WORKING_HOURS}

- use it in the *Time period* action condition;
- adjust the working time in *Administration* → *General* → *Macros*, if needed.

User macro context

An optional context can be used in user macros, allowing to override the default value with context-specific one.

User macros with context have a similar syntax:

```
{$MACRO:context}
```

Macro context is a simple text value. The common use case for macro contexts would be using a low-level discovery **macro value** as a user macro context. For example, a trigger prototype could be defined for mounted file system discovery to use a different low space limit depending on the mount points or file system types.

Only low-level discovery macros are supported in macro contexts. 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.

When context macros are processed, Zabbix looks up the macro with its context. If a macro with this context is not defined by host or linked templates, and it is not a defined as a global macro with context, then the macro without context is searched for.

See [usage example](#) of macro context in a disk space trigger prototype and take limitation clause into consideration.

3 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 file system names, network interfaces, SNMP OIDs, 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
 - names
 - key parameters
 - units
 - update intervals
 - history storage periods
 - trend storage periods
 - item value preprocessing steps
 - SNMP OIDs
 - IPMI sensor fields
 - calculated item formulas
 - SSH and Telnet scripts
 - database monitoring SQL queries
 - JMX item endpoint fields
 - descriptions
 - 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 field
- for trigger prototypes in
 - names
 - expression (only in constants and function parameters)
 - URLs
 - descriptions
 - event tag names and values
- for graph prototypes in
 - names

- for host prototypes in
 - names
 - visible names
 - host group prototype names
 - (see the [full list](#))

In all those places LLD macros can be used inside 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:

Tags		
Customer	{{#IFALIAS}.regsub("(.*)_([0-9]+)", \1)}	Remove
Interface	{{#IFALIAS}.regsub("(.*)_([0-9]+)", \2)}	Remove

For more information on macro function syntax, see: [Macro functions](#)

Macro functions are supported in low-level discovery macros since Zabbix 4.0.

11 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 access to:

- administrative Zabbix frontend functions
- monitored hosts in hostgroups

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**. As a guest, 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:

User
Media
Permissions

* Alias

Name

Surname

* Groups

Zabbix administrators
X

type here to search

Select

* Password

* Password (once again)

Language

English (en_GB)

Theme

System default

Auto-login

☒

Auto-logout

☐

15m

* Refresh

* Rows per page

URL (after login)

Add

Cancel

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Alias</i>	Unique username, used as the login name.
<i>Name</i>	User first name (optional).
	If not empty, visible in acknowledgement information and notification recipient information.
<i>Surname</i>	User second name (optional).
	If not empty, visible in acknowledgement information and notification recipient information.
<i>Groups</i>	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 <i>Select</i> 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 .
<i>Password</i>	Two fields for entering the user password.
	With an existing password, contains a <i>Password</i> button, clicking on which opens the password fields.
<i>Language</i>	Language of the Zabbix frontend.
	The php gettext extension is required for the translations to work.
<i>Theme</i>	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
<i>Auto-login</i>	Mark this checkbox to make Zabbix remember the user and log the user in automatically for 30 days. Browser cookies are used for this.

Parameter	Description
<i>Auto-logout</i>	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 <i>Remember me for 30 days</i> option checked.
<i>Refresh</i>	Set the refresh rate used for graphs, screens, plain text data, etc. Can be set to 0 to disable.
<i>Rows per page</i>	You can determine how many rows per page will be displayed in lists.
<i>URL (after login)</i>	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 media types.

Permissions

The *Permissions* tab contains information on:

- the user type (Zabbix User, Zabbix Admin, Zabbix Super Admin). Users cannot change their own type.
- host groups the user has access to. 'Zabbix User' and 'Zabbix Admin' users do not have access to any host groups and hosts by default. To get access they need to be included in user groups that have access to respective host groups and hosts.

See the **User permissions** page for details.

2 Permissions

Overview

You can differentiate user permissions in Zabbix by defining the respective user type and then by including the unprivileged users in user groups that have access to host group data.

User type

The user type defines the level of access to administrative menus and the default access to host group data.

User type	Description
<i>Zabbix User</i>	The user has access to the Monitoring menu. The user has no access to any resources by default. Any permissions to host groups must be explicitly assigned.
<i>Zabbix Admin</i>	The user has access to the Monitoring and Configuration menus. The user has no access to any host groups by default. Any permissions to host groups must be explicitly given.
<i>Zabbix Super Admin</i>	The user has access to everything: Monitoring, Configuration and Administration menus. The user has a read-write access to all host groups. Permissions cannot be revoked by denying access to specific host groups.

Permissions to host groups

Access to any host data in Zabbix are granted to **user groups** on host 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.

3 User groups

Overview

User groups allow to group users both for organizational purposes and for assigning permissions to data. Permissions to monitoring data of host 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 groups.

A user can belong to any amount 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:

User groups

User groupPermissionsTag filter

* Group name

Security specialists

Users

Admin (Zabbix Administrator) X

type here to search

Select

Frontend access

System default

Enabled

Debug mode

Add

Cancel

All mandatory input fields are marked with a red asterisk.

Parameter	Description
Group name	Unique group name.
Users	To add users to the group click <i>Select</i> button.
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.
Enabled	Disabled - access to Zabbix frontend is forbidden for this group Status of user group and group members. <i>Checked</i> - user group and users are enabled <i>Unchecked</i> - user group and users are disabled
Debug mode	Mark this checkbox to activate debug mode for the users.

The **Permissions** tab allows you to specify user group access to host group (and thereby host) data:

Permissions

Tag filter

Permissions

Host group

All groups

Discovered hosts

Hypervisors

Linux servers

Templates (including subgroups)

Templates/Servers Hardware

Templates/Virtualization

Permissions

None

Read-write Read Deny None

Read-write Read Deny None

Read-write Read Deny None

Read-write Read Deny None

Read-write Read Deny None

Read-write Read Deny None

type here to search

Select

Read-write Read Deny None

☐ Include subgroups

Add

Current permissions to host groups are displayed in the *Permissions* block.

If current permissions of the host group are inherited by all nested host groups, that is indicated by the *including subgroups* text in the parenthesis after the host group name.

You may change the level of access to a host group:

- **Read-write** - read-write access to a host group;
- **Read** - read-only access to a host group;
- **Deny** - access to a host group denied;
- **None** - no permissions are set.

Use the selection field below to select host groups and the level of access to them (note that selecting *None* will remove host group from the list if the group is already in the list). If you wish to include nested host groups, mark the *Include subgroups* checkbox. This field is auto-complete so starting to type the name of a host group will offer a dropdown of matching groups. If you wish to see all host groups, click on *Select*.

Note that it is possible for Zabbix Super Admin users in host group **configuration** to enforce the same level of permissions to the nested host groups as the parent host group.

The **Tag filter** tab allows you to set tag based permissions for user groups to see problems filtered by tag name and its value:

Tag filter

Permissions

Host group

Tags

Action

Templates/Databases

Service: MySQL

Remove

type here to search

Select

tag

value

☐ Include subgroups

Add

Update

Delete

Cancel

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. If you want 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, than specify "Service" tag name and "MySQL" value.

Templates/Databases

type here to search

Select

Service

MySQL

If "Service" tag name is specified and value field is left blank, corresponding user group will see all problems for selected host group with tag name "Service". If both tag name and value fields are left blank but host group selected, corresponding user group will see all problems for selected host group. Make sure a tag name and tag value are correctly specified otherwise a corresponding 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
<i>Tag filter</i>						
<i>Host group</i>	<i>Tag name</i>	<i>Tag value</i>	<i>Host group</i>	<i>Tag name</i>	<i>Tag value</i>	
Templates/Databases	Service	MySQL	Templates/Databases	Service	Oracle	Service: MySQL or Oracle problems visible
Templates/Databases	blank	blank	Templates/Databases	Service	Oracle	All problems visible
not selected	blank	blank	Templates/Databases	Service	Oracle	Service: Oracle problems visible

Attention:

Adding a filter (for example, all tags in a certain host group "Templates/Databases") results in not being able to see the problems of other host groups.

Host access from several user groups

A user may belong to any number of user groups. These groups may have different access permissions to hosts.

Therefore, it is important to know what hosts 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 starting with Zabbix 2.2.

- 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 *Templates* group. With *Read* access to *Templates* 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-Zabbix 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. The same applies to screens and slideshows as well. The users, regardless of permissions, will see any objects that are not directly or indirectly linked to hosts.
- Zabbix server will not send notifications to users defined as action operation recipients if access to the concerned host is explicitly "denied".

8. Service monitoring

Overview Service monitoring functionality is intended for those who want to get a high-level (business) view of monitored infrastructure. In many cases, we are not interested in low-level details, like the lack of disk space, high processor load, etc. What we are interested in is the availability of service provided by our IT department. We can also be interested in identifying weak places of IT infrastructure, SLA of various IT services, the structure of existing IT infrastructure, and other information of a higher level.

Zabbix service monitoring provides answers to all mentioned questions.

Services is 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. At the lowest level of services are triggers. The status of individual nodes is affected by the status of their triggers.

Note:
Note that triggers with a *Not classified* or *Information* severity do not impact SLA calculation.

Configuration To configure services, go to: *Configuration* → *Services*.

On this screen you can build a hierarchy of your monitored infrastructure. The highest-level parent service is 'root'. You can build your hierarchy downward by adding lower-level parent services and then individual nodes to them.

Services

Service	Action
root	Add child
▼ Servers	Add child
Server 1	Add child Delete
Server 2	Add child Delete
Server 3	Add child Delete
Server 4	Add child Delete
Server 5	Add child Delete

Click on *Add child* to add services. To edit an existing service, click on its name. A form is displayed where you can edit the service attributes.

Configuring a service

The **Service** tab contains general service attributes:

Service

Dependencies

Time

*

Name

Server 1

*

Parent service

SLA by service

Change

Status calculation algorithm

Problem, if at least one child has a problem

Calculate SLA, acceptable SLA (in %)

☐

99.9000

Trigger

New host: Zabbix agent on New host is unreachable

Select

*

Sort order (0->999)

0

Update

Delete

Cancel

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Name</i>	Service name.
<i>Parent service</i>	Parent service the service belongs to.
<i>Status calculation algorithm</i>	Method of calculating service status: Do not calculate - do not calculate service status Problem, if at least one child has a problem - problem status, if at least one child service has a problem Problem, if all children have problems - problem status, if all child services are having problems
<i>Calculate SLA</i>	Enable SLA calculation and display.
<i>Acceptable SLA (in %)</i>	SLA percentage that is acceptable for this service. Used for reporting.
<i>Trigger</i>	Linkage to trigger: None - no linkage trigger name - linked to the trigger, thus depends on the trigger status Services of the lowest level must be linked to triggers. (Otherwise their state will not be represented accurately.) When triggers are linked, their state prior to linking is not counted.
<i>Sort order</i>	Sort order for display, lowest comes first.

The **Dependencies** tab contains services the service depends on. Click on *Add* to add a service from those that are configured.

Service
Dependencies
Time

Depends on

SERVICES	SOFT	TRIGGER
Server 2	☐	
Server 3	☒	
Server 4	☒	
Add		

Update
Delete
Cancel

Hard and soft dependency

Availability of a service may depend on several other services, not just one. The first option is to add all those directly as child services.

However, if some service is already added somewhere else in the services tree, it cannot be simply moved out of there to a child service here. How to create a dependency on it? The answer is "soft" linking. Add the service and mark the *Soft* check box. That way the service can remain in its original location in the tree, yet be depended upon from several other services. Services that are "soft-linked" are displayed in grey in the tree. Additionally, if a service has only "soft" dependencies, it can be deleted directly, without deleting child services first.

The **Time** tab contains the service time specification.

Service
Dependencies
Time

Service times

Type	Interval	Note
New service time		
Period type	Uptime	
* From	Sunday	Time hh : mm
* Till	Sunday	Time hh : mm
Add		

Add
Cancel

Parameter	Description
<i>Service times</i>	By default, all services are expected to operate 24x7x365. If exceptions needed, add new service times.

Parameter	Description
<i>New service time</i>	Service times: Uptime - service uptime Downtime - service state within this period does not affect SLA. One-time downtime - a single downtime. Service state within this period does not affect SLA. Add the respective hours. <i>Note:</i> Service times affect only the service they are configured for. Thus, a parent service will not take into account the service time configured on a child service (unless a corresponding service time is configured on the parent service as well). Service times are taken into account when calculating service status and SLA by the frontend. However, information on service availability is being inserted into database continuously, regardless of service times.

Display To monitor services, go to *Monitoring → Services*.

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.

Since Zabbix 2.2 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.

Scenario
Steps
Authentication

* Name

Availability of google

Application

New application

Web checks

* Update interval

1m

* Attempts

1

Agent

Zabbix

HTTP proxy

[protocol://][user[:password]@]proxy.example.com[:port]

Variables

Name

Value

name

⇒

value

Add

Headers

Name

Value

name

⇒

value

Add

Enabled

☒

Add

Cancel

All mandatory input fields are marked with a red asterisk.

Scenario parameters:

Parameter	Description
<i>Host Name</i>	Name of the host/template that the scenario belongs to. Unique scenario name.
<i>Application</i>	User macros and {HOST.*} macros are supported, since Zabbix 2.2. Select an application the scenario will belong to. Web scenario items will be grouped under the selected application in <i>Monitoring</i> → <i>Latest data</i> .
<i>New application</i>	Enter the name of a new application for the scenario.

Parameter	Description
<i>Update interval</i>	How often the scenario will be executed. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. <i>Note</i> 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).
<i>Attempts</i>	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. <i>Note:</i> Zabbix will not repeat a step because of a wrong response code or the mismatch of a required string.
<i>Agent</i>	This parameter is supported starting with <i>Zabbix 2.2</i>. Select a client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers.
<i>HTTP proxy</i>	User macros can be used in this field, <i>starting with Zabbix 2.2</i>. You can specify an HTTP proxy to use, using the format <code>[protocol://] [username[:password]@]proxy.mycompany.com[:port]</code> This sets the CURLOPT_PROXY cURL option. The optional <code>protocol://</code> 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 <code>http_proxy</code>, <code>HTTPS_PROXY</code>. 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. <i>Note</i> that only simple authentication is supported with HTTP proxy. User macros can be used in this field.
<i>Variables</i>	This parameter is supported starting with <i>Zabbix 2.2</i>. 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: <code>{username}=Alexei</code> <code>{password}=kj3h5kj34bd</code> <code>{hostid}=regex:hostid is ([0-9]+)</code> The macros can then be referenced in the steps as <code>{username}</code>, <code>{password}</code> and <code>{hostid}</code>. Zabbix will automatically replace them with actual values. <i>Note</i> 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. Regular expression match in variables is supported <i>since Zabbix 2.2</i>. User macros and <code>{HOST.*}</code> macros are supported, since Zabbix 2.2. 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.

Parameter	Description
<i>Headers</i>	Custom 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, 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. Specifying custom headers is supported <i>starting with Zabbix 2.4</i> .
<i>Enabled</i>	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.

ScenarioStepsAuthentication

Steps

Name	Timeout	URL	Required	Status codes	Action
1: Home	15s	http://www.google.com		200	Remove
2: About	15s	http://www.google.com/intl/en/about		200	Remove
Add					

Step of web scenario

* Name

Home

* URL

http://www.google.com

Parse

Query fields

Name

Value

name

⇒

value

Remove

Add

Post type

Form data
Raw data

Post fields

Name

Value

name

⇒

value

Remove

Add

Variables

Name

Value

name

⇒

value

Remove

Add

Headers

Name

Value

name

⇒

value

Remove

Add

Follow redirects

☒

Retrieve mode

Body
Headers
Body and headers

* Timeout

15s

Required string

Required status codes

200

Add
Cancel

Configuring steps

Step parameters:

Parameter	Description
<i>Name</i>	Unique step name.
<i>URL</i>	User macros and {HOST.*} macros are supported, since Zabbix 2.2. URL to connect to and retrieve data. For example: https://www.google.com http://www.zabbix.com/download Domain names can be specified in Unicode characters since Zabbix 3.4. They are automatically punycode-converted to ASCII when executing the web scenario step. The <i>Parse</i> button can be used to separate optional query fields (like ?name=Admin&password=mypassword) from the URL, moving the attributes and values into <i>Query fields</i> 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, since Zabbix 2.2. Limited to 2048 characters <i>starting with Zabbix 2.4</i>.

Parameter	Description
<i>Query fields</i>	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.
<i>Post</i>	User macros and {HOST.*} macros are supported since Zabbix 2.2. 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()}.
<i>Variables</i>	User macros and {HOST.*} macros are supported, since Zabbix 2.2. 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. Having step-level variables is supported since Zabbix 2.2. 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.
<i>Headers</i>	Custom HTTP headers that will be sent when performing a request. Specified as attribute and value pairs. Headers on the step level will overwrite the headers specified for the scenario. For example, setting a 'User-Agent' attribute with no value will remove the User-Agent value set on scenario level. User macros and {HOST.*} macros are supported. This sets the CURLOPT_HTTPHEADER cURL option. Specifying custom headers is supported <i>starting with Zabbix 2.4</i>.
<i>Follow redirects</i>	Mark the checkbox to follow HTTP redirects. This sets the CURLOPT_FOLLOWLOCATION cURL option. This option is supported <i>starting with Zabbix 2.4</i>.
<i>Retrieve mode</i>	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 This option is supported <i>since Zabbix 4.2</i>.
<i>Timeout</i>	Zabbix will not spend more than the set amount of time on processing the URL (maximum is 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.

Parameter	Description
<i>Required string</i>	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 <i>Note:</i> Referencing regular expressions created in the Zabbix frontend is not supported in this field.
<i>Required status codes</i>	User macros and {HOST.*} macros are supported, since Zabbix 2.2. 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 since Zabbix 2.2.

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 authentication The **Authentication** tab allows you to configure scenario authentication options.

Authentication parameters:

Parameter	Description
<i>Authentication</i>	Authentication options. None - no authentication used. Basic authentication - basic authentication is used. NTLM authentication - NTLM (Windows NT LAN Manager) 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, <i>starting with Zabbix 2.2</i>.

Parameter	Description
<i>SSL verify peer</i>	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. This option is supported <i>starting with Zabbix 2.4</i>.
<i>SSL verify host</i>	Mark the checkbox to verify that the <i>Common Name</i> field or the <i>Subject Alternate Name</i> field of the web server certificate matches. This sets the CURLOPT_SSL_VERIFYHOST cURL option. This option is supported <i>starting with Zabbix 2.4</i>.
<i>SSL certificate file</i>	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 <i>SSL key file</i> field empty. If the key is encrypted, specify the password in <i>SSL key password</i> 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. This option is supported <i>starting with Zabbix 2.4</i>.
<i>SSL key file</i>	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 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. This option is supported <i>starting with Zabbix 2.4</i>.
<i>SSL key password</i>	SSL private key file password. User macros can be used in this field. This sets the CURLOPT_KEYPASSWD cURL option. This option is supported <i>starting with Zabbix 2.4</i>.

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:

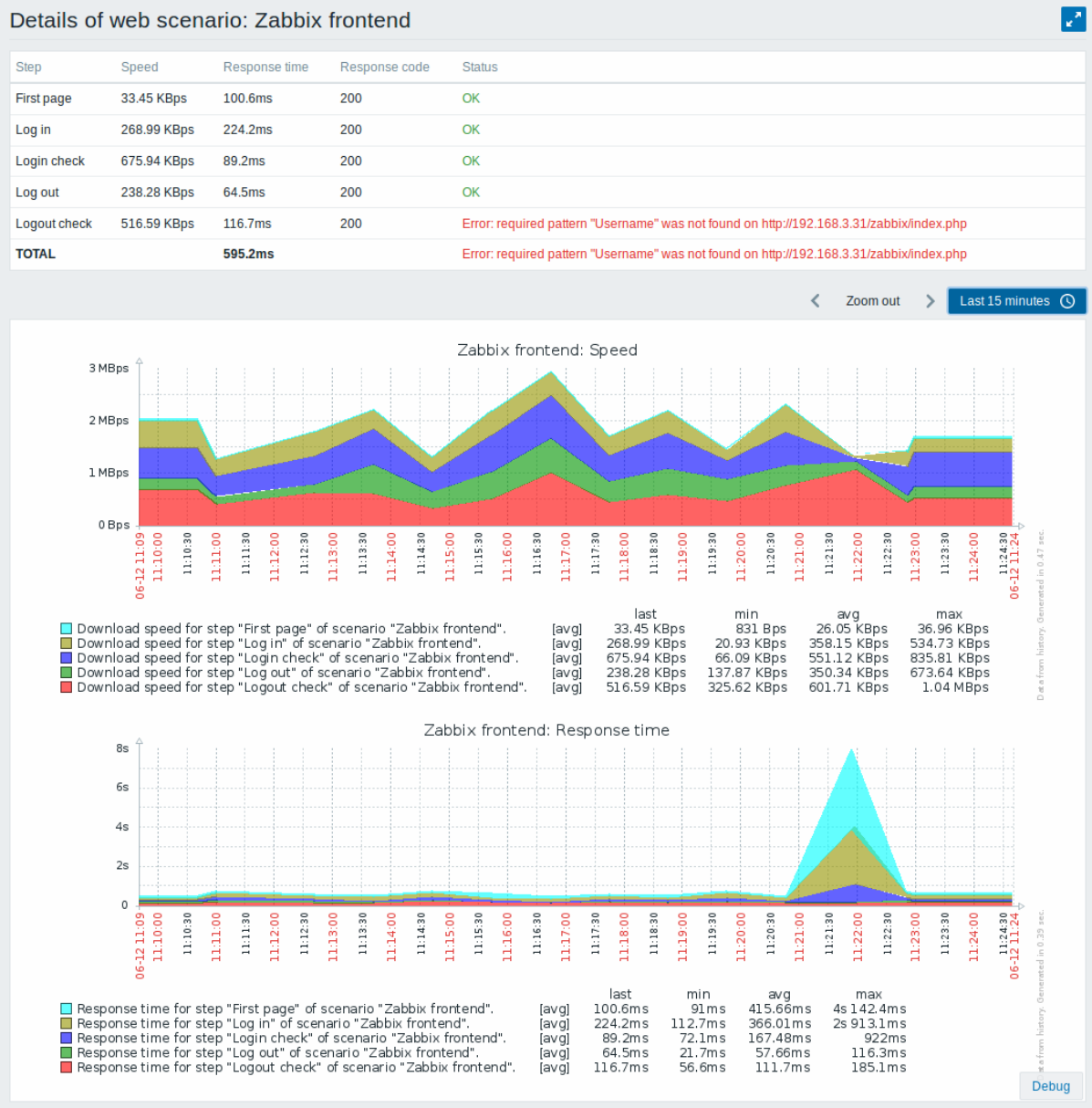
Zabbix server picks up changes in certificates without a restart.

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 detailed data of defined web scenarios, go to *Monitoring* → *Web* or *Latest data*. Click on the scenario name to see more detailed statistics.



An overview of web monitoring scenarios can be viewed in *Monitoring* → *Dashboard*.

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.

Scenario items

As soon as a scenario is created, Zabbix automatically adds the following items for monitoring, linking them to the selected application.

Item	Description
<i>Download speed for scenario</i> <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: <i>Numeric(float)</i>
<i>Failed step of scenario</i> <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: <i>Numeric(unsigned)</i>
<i>Last error message of scenario</i> <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: <i>Character</i>

The actual scenario name will be used instead of "Scenario".

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.

Example 1

To create a "Web scenario failed" trigger, you can define a trigger expression:

```
{host:web.test.fail[Scenario].last()}<>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:

```
{host:web.test.error[Scenario].strlen()}>0 and {host:web.test.fail[Scenario].last()}>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:

```
{host:web.test.in[Scenario,,bps].last()}<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, linking them to the selected application.

Item	Description
<i>Download speed for step</i> <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: <i>Numeric(float)</i>

Item	Description
<i>Response time for step <Step> of scenario <Scenario></i>	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: <i>Numeric(float)</i>
<i>Response code for step <Step> of scenario <Scenario></i>	This item will collect response codes of the step. Item key: web.test.rspcode[Scenario,Step] Type: <i>Numeric(unsigned)</i>

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:

```
{zabbix:web.test.time[ZABBIX GUI,Login,resp].last()}>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*.

Scenario
Steps
Authentication

*

Name

Zabbix frontend

Application

New application

Zabbix frontend

*

Update interval

1m

*

Attempts

1

Agent

Zabbix

HTTP proxy

[protocol://][user[:password]@]proxy.example.com[:port]

Variables

Name		Value
{password}	⇒	zabbix
{user}	⇒	Admin

Add

Headers

Name		Value
name	⇒	value

Add

Enabled

☒

Add

Cancel

All mandatory input fields are marked with a red asterisk.

In the new scenario form we will name the scenario as *Zabbix frontend* and create a new *Zabbix frontend* application for it.

Note that we will also create two variables: {user} and {password}.

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".

Step of web scenario

* Name
First page

* URL
http://localhost/zabbix/index.php
Parse

Query fields

Name
Value

name
⇒
value
Remove

Add

Post type
Form data
Raw data

Post fields

Name
Value

name
⇒
value
Remove

Add

Variables

Name
Value

name
⇒
value
Remove

Add

Headers

Name
Value

name
⇒
value
Remove

Add

Follow redirects
☒

Retrieve mode
Body
Headers
Body and headers

* Timeout
15s

Required string
Zabbix SIA

Required status codes
200

Update
Cancel

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}.

Step of web scenario

* Name

Log in

* URL

http://localhost/zabbix/index.php

Parse

Query fields

Name	Value	
name	value	Remove

Add

Post type

Form data

Raw data

Post fields

Name	Value	
name	{user}	Remove
password	{password}	Remove
enter	Sign in	Remove

Add

Variables

Name	Value	
{sid}	regex:name="csrf-token" content="([0-	Remove

Add

Headers

Name	Value	
name	value	Remove

Add

Follow redirects

☒

Retrieve mode

Body

Headers

Body and headers

* Timeout

15s

Required string

Required status codes

200

Update

Cancel

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**.

Step of web scenario

* Name
Login check

* URL
http://localhost/zabbix/index.php
Parse

Query fields

Name
Value

name
⇒
value
Remove

Add

Post type
Form data
Raw data

Post fields

Name
Value

name
⇒
value
Remove

Add

Variables

Name
Value

name
⇒
value
Remove

Add

Headers

Name
Value

name
⇒
value
Remove

Add

Follow redirects
☒

Retrieve mode
Body
Headers
Body and headers

* Timeout
15s

Required string
Administration

Required status codes
200

Update
Cancel

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.

Step of web scenario

* Name

Log out

* URL

http://localhost/zabbix/index.php

Parse

Query fields

Name	Value	
sid	{sid}	Remove
reconnect	1	Remove
Add		

Post type

Form data
Raw data

Post fields

Name	Value	
name	value	Remove
Add		

Variables

Name	Value	
name	value	Remove
Add		

Headers

Name	Value	
name	value	Remove
Add		

Follow redirects

☒

Retrieve mode

Body
Headers
Body and headers

* Timeout

15s

Required string

Required status codes

200

Update

Cancel

Web scenario step 5

We can also check that we have logged out by looking for the **Username** string.

Step of web scenario

Name

Logout check

URL

http://localhost/zabbix/index.php

Parse

Query fields

Name

Value

name

⇒

value

Remove

Add

Post type

Form data

Raw data

Post fields

Name

Value

name

⇒

value

Remove

Add

Variables

Name

Value

name

⇒

value

Remove

Add

Headers

Name

Value

name

⇒

value

Remove

Add

Follow redirects

☒

Retrieve mode

Body

Headers

Body and headers

Timeout

15s

Required string

Username

Required status codes

200

Update

Cancel

Complete configuration of steps

A complete configuration of web scenario steps should look like this:

Scenario Steps Authentication						
Steps	Name	Timeout	URL	Required	Status codes	Action
1:	First page	15s	http://localhost/zabbix/index.php	Zabbix SIA	200	Remove
2:	Log in	15s	http://localhost/zabbix/index.php		200	Remove
3:	Login check	15s	http://localhost/zabbix/index.php	Administration	200	Remove
4:	Log out	15s	http://localhost/zabbix/index.php		200	Remove
5:	Logout check	15s	http://localhost/zabbix/index.php	Username	200	Remove
	Add					

Step 3

Save the finished web monitoring scenario.

The scenario will appear in *Monitoring* → *Web*:

Web monitoring

Group **Zabbix servers** Host **all**

Host	Name ▲	Number of steps	Last check	Status
Zabbix server	Zabbix frontend	5	2017-03-24 08:32:50	OK

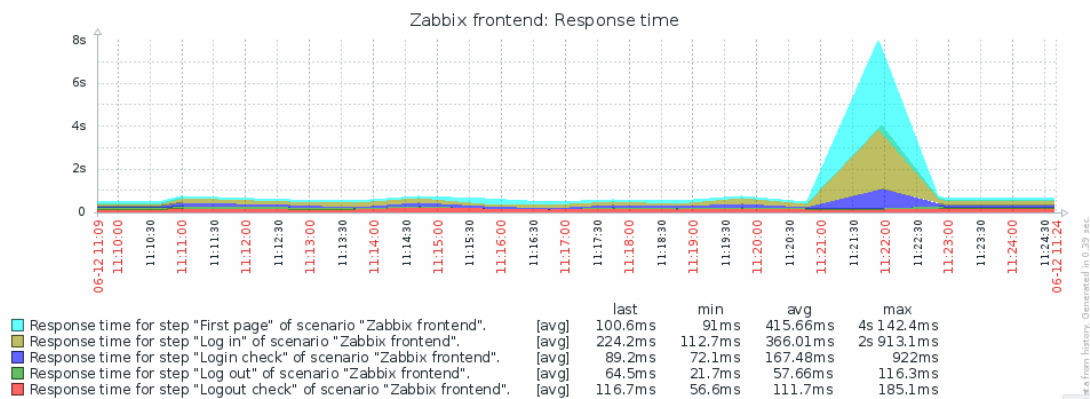
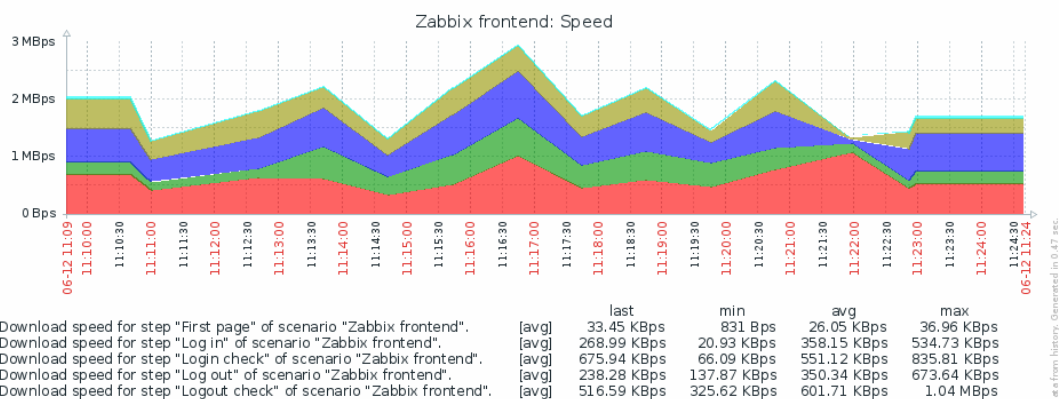
Displaying 1 of 1 found

Click on the scenario name to see more detailed statistics:

Details of web scenario: Zabbix frontend

Step	Speed	Response time	Response code	Status
First page	33.45 KBps	100.6ms	200	OK
Log in	268.99 KBps	224.2ms	200	OK
Login check	675.94 KBps	89.2ms	200	OK
Log out	238.28 KBps	64.5ms	200	OK
Logout check	516.59 KBps	116.7ms	200	Error: required pattern "Username" was not found on http://192.168.3.31/zabbix/index.php
TOTAL		595.2ms		Error: required pattern "Username" was not found on http://192.168.3.31/zabbix/index.php

Zoom out Last 15 minutes



Debug

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 4.1.

Details The virtual machine monitoring is done in two steps. First, virtual machine data is gathered by *vmware collector* Zabbix processes. Those processes obtain necessary information from VMware web services over the SOAP protocol, pre-process it and store into Zabbix server shared memory. Then, this data is retrieved by pollers using Zabbix simple check **VMware keys**.

Starting with Zabbix version 2.4.4 the collected data is divided into 2 types: VMware configuration data and VMware performance counter data. Both types are collected independently by *vmware collectors*. Because of this it is recommended to enable more collectors than the monitored VMware services. Otherwise retrieval of VMware performance counter statistics might be delayed by the retrieval of VMware configuration data (which takes a while for large installations).

Currently only datastore, network interface and disk device statistics and custom performance counter items are based on the VMware performance counter information.

Configuration For virtual machine monitoring to work, Zabbix should be **compiled** with the `--with-libxml2` and `--with-libcurl` compilation options.

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. Note that in most cases this value should not be less than 2 and should not be 2 times greater than the number of VMware services that you monitor. Also keep in mind that this value 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 `vpxd.stats.maxQueryMetrics` parameter to be at least 64. See also the VMware knowledge base [article](#).

Discovery Zabbix can use a low-level discovery rule to automatically discover VMware hypervisors and virtual machines.

Discovery rule

Filters

*

Name

Discover VMware hypervisors

Type

Simple check

*

Key

vmware.hv.discovery[{\$URL}]

User name

{\$USERNAME}

Password

{\$PASSWORD}

*

Update interval (in sec)

3600

Custom intervals

TYPE	INTERVAL	PERIOD
Flexible	Scheduling	50
Add		

*

Keep lost resources period (in days)

30

Description

Discovery of hypervisors.

Enabled

☒

All mandatory input fields are marked with a red asterisk.

Discovery rule key in the above screenshot is `vmware.hv.discovery[{$URL}]`.

Host prototypes Host prototypes can be created with the low-level discovery rule. When virtual machines are discovered, these prototypes become real hosts. Prototypes, before becoming discovered, cannot have their own items and triggers, other than those from the linked templates. Discovered hosts will belong to an existing host and will take the IP of the existing host for the host configuration.

Discovery rules

All templates / Template Virt VMware Applications 3 Items 3 Triggers Graphs Screens **Discovery**

☐ NAME ▲	ITEMS	TRIGGERS	GRAPHS	HOSTS
☐ Discover VMware clusters	Item prototypes 1	Trigger prototypes	Graph prototypes	Host protol
☐ Discover VMware hypervisors	Item prototypes	Trigger prototypes	Graph prototypes	Host protol
☐ Discover VMware VMs	Item prototypes	Trigger prototypes	Graph prototypes	Host protol

In a host prototype configuration, LLD macros are used for the host name, visible name and host group prototype fields. Linkage to existing host groups, template linkage and encryption are other options that can be set.

Host Groups Templates Host inventory Encryption

* Host name

Visible name

Create enabled ☒

Add

Cancel

If *Create enabled* is checked, the host will be added in an enabled state. If unchecked, the host will be added, but in disabled state.

Discovered hosts are prefixed with the name of the discovery rule that created them, in the host list. Discovered hosts can be manually deleted. Discovered hosts will also be automatically deleted, based on the *Keep lost resources period (in days)* value of the discovery rule. Most of the configuration options are read-only, except for enabling/disabling the host and host inventory. Discovered hosts cannot have host prototypes of their own.

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.

Note that *"Template VM VMware"* template should be used for VMware vCenter and ESX hypervisor monitoring. The *"Template VM VMware Hypervisor"* and *"Template VM VMware Guest"* templates are used by discovery and normally should not be manually linked to a host.

Templates

☐ Name ▼	Applications	Items	Triggers
☐ Template VM VMware Hypervisor	Applications 6	Items 21	Triggers
☐ Template VM VMware Guest	Applications 8	Items 19	Triggers
☐ Template VM VMware	Applications 3	Items 3	Triggers

Note:

If your server has been upgraded from a pre-2.2 version and has no such templates, you can import them manually, downloading from the community page with [official templates](#). However, these templates have dependencies from the *VMware VirtualMachinePowerState* and *VMware status* value maps, so it is necessary to create these value maps first (using an [SQL script](#), manually or importing from an XML) before importing the templates.

Host configuration To use VMware simple checks the host must have the following user macros defined:

- **{ \$URL }** - VMware service (vCenter or ESX hypervisor) SDK URL (<https://servername/sdk>)
- **{ \$USERNAME }** - VMware service user name
- **{ \$PASSWORD }** - VMware service { \$USERNAME } user password

Example The following example demonstrates how to quickly setup VMware monitoring on Zabbix:

- compile zabbix server with required options (--with-libxml2 and --with-libcurl)
- set the StartVMwareCollectors option in Zabbix server configuration file to 1 or more
- create a new host
- set the host macros required for VMware authentication:

```
{{...:assets:en:manual:vm_monitoring:vm_host_macros.png|}}
```

* Link the host to the VMware service template:

```
{{...:assets:en:manual:vm_monitoring:vm_host_templates.png|}}
```

* 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). The following examples demonstrate how extended logging can be started provided debug level 4 is already set:

Increase log level of all vmware collectors:

```
shell> zabbix_server -R log_level_increase="vmware collector"
```

Increase log level of 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 if 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.

1 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.NAME }	Cluster identifier. (Name) name.
vmware.datastore.discovery		

Item key	
Performs datastore discovery.	{#DATASTORE} Datastore name. Supported since Zabbix 4.2.4.
vmware.hv.discovery Performs hypervisor discovery.	{#HV.UUID} Unique hypervisor identifier. {#HV.ID} Hypervisor identifier (Host-System managed object name). {#HV.NAME} Hypervisor name. {#CLUSTER.NAME} name, might be empty. {#DATACENTER.NAME} Datacenter name. {#PARENT.NAME} Name of container that stores the hypervisor. Supported since Zabbix 4.0.3.

Item key

vmware.hv.datastore.discovery	Performs hypervisor datastore discovery. Note that multiple hypervisors can use the same datastore.	{#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.
vmware.vm.discovery	Performs virtual machine discovery.	{#DATASTORE} Datastore name. {#VM.UUID} Unique virtual machine identifier. {#VM.ID} Virtual machine identifier (Virtual-Machine managed object name). {#VM.NAME} Virtual machine name. {#HV.NAME} Hypervisor name. {#CLUSTER.NAME} Name, might be empty. {#DATACENTER.NAME} Datacenter name.
vmware.vm.net.if.discovery	Performs virtual machine network interface discovery.	{#IFNAME} Network interface name.
vmware.vm.vfs.dev.discovery		

Item key

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.

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.

Attention:

To ensure predictable behaviour of recurring maintenance periods (daily, weekly, monthly), it is required to use a common timezone for all parts of Zabbix.

If maintenance period, hosts, groups or tags are changed by user, the changes will only take effect after configuration cache synchronization.

Configuration To configure a maintenance period:

- Go to: *Configuration* → *Maintenance*
- Click on *Create maintenance period* (or on the name of an existing maintenance period)

The **Maintenance** tab contains general maintenance period attributes:

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Name</i>	Name of the maintenance period.
<i>Maintenance type</i>	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
<i>Active since</i>	The date and time when executing maintenance periods becomes active. <i>Note:</i> Setting this time alone does not activate a maintenance period; for that go to the <i>Periods</i> tab.
<i>Active till</i>	The date and time when executing maintenance periods stops being active.
<i>Description</i>	Description of maintenance period.

The **Periods** tab allows you to define the exact days and hours when the maintenance takes place. Clicking on *New* opens a flexible *Maintenance period* form where you can define the times - for daily, weekly, monthly or one-time maintenance.

Maintenance
Periods
Hosts and groups

* Periods

Period type	Schedule	Period	Action
Weekly	At 15:00 Monday of every week	1h	Edit

Maintenance period

Period type
Weekly

* Every week(s)
1

* Day of week
☒ Monday
☐ Tuesday
☐ Wednesday
☐ Thursday
☐ Friday
☐ Saturday
☐ Sunday

At (hour:minute)
15 : 0

* Maintenance period length
0 Days 1 Hours 0 Minutes

[Update](#)
[Cancel](#)

Add
Cancel

Daily and weekly periods have an *Every day/Every week* parameter, which defaults to 1. Setting it to 2 would make the maintenance take place every two days or every two weeks and so on. The starting day or week is the day or week that *Active since* time falls on.

For example, having *Active since* set to 2013-09-06 12:00 and an hour long daily recurrent period every two days at 23:00 will result in the first maintenance period starting on 2013-09-06 at 23:00, while the second maintenance period will start on 2013-09-08 at 23:00. Or, with the same *Active since* time and an hour long daily recurrent period every two days at 01:00, the first maintenance period will start on 2013-09-08 at 01:00, and the second maintenance period on 2013-09-10 at 01:00.

The **Hosts and groups** tab allows you to select the host groups, hosts and problem tags for maintenance.

Maintenance
Periods
Hosts and groups

* At least one host group or host must be selected.

Host groups

Hosts

Tags

And/Or
Or

Contains
Equals

Add

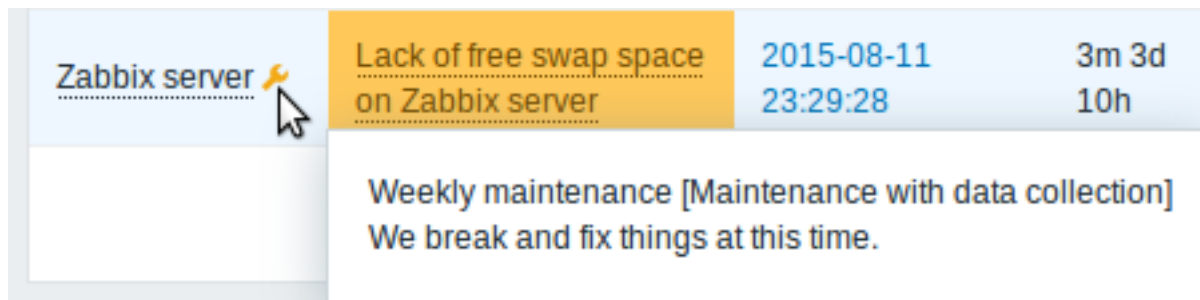
Add
Cancel

Parameter	Description
<i>Host groups</i>	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.
<i>Hosts</i>	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.
<i>Tags</i>	If maintenance tags are specified, maintenance for the selected hosts will still be activated, but problems will only be suppressed (i.e. no actions will be taken) if their tags are a match. 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)

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. Note that before 4.2.3, an orange *In maintenance* string was displayed as status instead of the icon.)



Maintenance details are displayed when the mouse pointer is positioned over the icon.

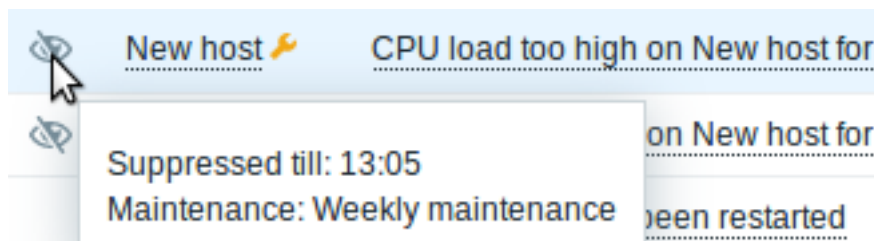
Additionally, hosts in maintenance get an orange background in *Monitoring* → *Maps*.

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* → *Overview* (in the filter; with 'Triggers' as *Type*)
- *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) 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 multi-line 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.

ExpressionsTest

Name

Network interfaces for discovery

Expressions

EXPRESSION TYPE	EXPRESSION	DELIMITER	CASE SENSITIVE	ACTION
Result is FALSE	^lo\$	☐	☒	Remove
Result is FALSE	^Software Loopback Interface	☐	☒	Remove

Add

Add

Cancel

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 <i>Add</i> in the Expressions block to add a new subexpression. 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
Delimiter	Enter substring/regular expression. 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.

Since Zabbix 2.4.0, a forward slash (/) in the expression is treated literally, rather than a delimiter. This way it is possible to save expressions containing a slash, whereas previously it would produce an error.

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).

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 ufs vxfs zfs zfs2 zfs3 zfs4 zfs5 zfs6 zfs7 zfs8 zfs9 zfs10 zfs11 zfs12 zfs13 zfs14 zfs15 zfs16 zfs17 zfs18 zfs19 zfs20 zfs21 zfs22 zfs23 zfs24 zfs25 zfs26 zfs27 zfs28 zfs29 zfs30 zfs31 zfs32 zfs33 zfs34 zfs35 zfs36 zfs37 zfs38 zfs39 zfs40 zfs41 zfs42 zfs43 zfs44 zfs45 zfs46 zfs47 zfs48 zfs49 zfs50 zfs51 zfs52 zfs53 zfs54 zfs55 zfs56 zfs57 zfs58 zfs59 zfs60 zfs61 zfs62 zfs63 zfs64 zfs65 zfs66 zfs67 zfs68 zfs69 zfs70 zfs71 zfs72 zfs73 zfs74 zfs75 zfs76 zfs77 zfs78 zfs79 zfs80 zfs81 zfs82 zfs83 zfs84 zfs85 zfs86 zfs87 zfs88 zfs89 zfs90 zfs91 zfs92 zfs93 zfs94 zfs95 zfs96 zfs97 zfs98 zfs99 zfs100 zfs101 zfs102 zfs103 zfs104 zfs105 zfs106 zfs107 zfs108 zfs109 zfs110 zfs111 zfs112 zfs113 zfs114 zfs115 zfs116 zfs117 zfs118 zfs119 zfs120 zfs121 zfs122 zfs123 zfs124 zfs125 zfs126 zfs127 zfs128 zfs129 zfs130 zfs131 zfs132 zfs133 zfs134 zfs135 zfs136 zfs137 zfs138 zfs139 zfs140 zfs141 zfs142 zfs143 zfs144 zfs145 zfs146 zfs147 zfs148 zfs149 zfs150 zfs151 zfs152 zfs153 zfs154 zfs155 zfs156 zfs157 zfs158 zfs159 zfs160 zfs161 zfs162 zfs163 zfs164 zfs165 zfs166 zfs167 zfs168 zfs169 zfs170 zfs171 zfs172 zfs173 zfs174 zfs175 zfs176 zfs177 zfs178 zfs179 zfs180 zfs181 zfs182 zfs183 zfs184 zfs185 zfs186 zfs187 zfs188 zfs189 zfs190 zfs191 zfs192 zfs193 zfs194 zfs195 zfs196 zfs197 zfs198 zfs199 zfs200 zfs201 zfs202 zfs203 zfs204 zfs205 zfs206 zfs207 zfs208 zfs209 zfs210 zfs211 zfs212 zfs213 zfs214 zfs215 zfs216 zfs217 zfs218 zfs219 zfs220 zfs221 zfs222 zfs223 zfs224 zfs225 zfs226 zfs227 zfs228 zfs229 zfs230 zfs231 zfs232 zfs233 zfs234 zfs235 zfs236 zfs237 zfs238 zfs239 zfs240 zfs241 zfs242 zfs243 zfs244 zfs245 zfs246 zfs247 zfs248 zfs249 zfs250 zfs251 zfs252 zfs253 zfs254 zfs255 zfs256 zfs257 zfs258 zfs259 zfs260 zfs261 zfs262 zfs263 zfs264 zfs265 zfs266 zfs267 zfs268 zfs269 zfs270 zfs271 zfs272 zfs273 zfs274 zfs275 zfs276 zfs277 zfs278 zfs279 zfs280 zfs281 zfs282 zfs283 zfs284 zfs285 zfs286 zfs287 zfs288 zfs289 zfs290 zfs291 zfs292 zfs293 zfs294 zfs295 zfs296 zfs297 zfs298 zfs299 zfs300 zfs301 zfs302 zfs303 zfs304 zfs305 zfs306 zfs307 zfs308 zfs309 zfs310 zfs311 zfs312 zfs313 zfs314 zfs315 zfs316 zfs317 zfs318 zfs319 zfs320 zfs321 zfs322 zfs323 zfs324 zfs325 zfs326 zfs327 zfs328 zfs329 zfs330 zfs331 zfs332 zfs333 zfs334 zfs335 zfs336 zfs337 zfs338 zfs339 zfs340 zfs341 zfs342 zfs343 zfs344 zfs345 zfs346 zfs347 zfs348 zfs349 zfs350 zfs351 zfs352 zfs353 zfs354 zfs355 zfs356 zfs357 zfs358 zfs359 zfs360 zfs361 zfs362 zfs363 zfs364 zfs365 zfs366 zfs367 zfs368 zfs369 zfs370 zfs371 zfs372 zfs373 zfs374 zfs375 zfs376 zfs377 zfs378 zfs379 zfs380 zfs381 zfs382 zfs383 zfs384 zfs385 zfs386 zfs387 zfs388 zfs389 zfs390 zfs391 zfs392 zfs393 zfs394 zfs395 zfs396 zfs397 zfs398 zfs399 zfs400 zfs401 zfs402 zfs403 zfs404 zfs405 zfs406 zfs407 zfs408 zfs409 zfs410 zfs411 zfs412 zfs413 zfs414 zfs415 zfs416 zfs417 zfs418 zfs419 zfs420 zfs421 zfs422 zfs423 zfs424 zfs425 zfs426 zfs427 zfs428 zfs429 zfs430 zfs431 zfs432 zfs433 zfs434 zfs435 zfs436 zfs437 zfs438 zfs439 zfs440 zfs441 zfs442 zfs443 zfs444 zfs445 zfs446 zfs447 zfs448 zfs449 zfs450 zfs451 zfs452 zfs453 zfs454 zfs455 zfs456 zfs457 zfs458 zfs459 zfs460 zfs461 zfs462 zfs463 zfs464 zfs465 zfs466 zfs467 zfs468 zfs469 zfs470 zfs471 zfs472 zfs473 zfs474 zfs475 zfs476 zfs477 zfs478 zfs479 zfs480 zfs481 zfs482 zfs483 zfs484 zfs485 zfs486 zfs487 zfs488 zfs489 zfs490 zfs491 zfs492 zfs493 zfs494 zfs495 zfs496 zfs497 zfs498 zfs499 zfs500 zfs501 zfs502 zfs503 zfs504 zfs505 zfs506 zfs507 zfs508 zfs509 zfs510 zfs511 zfs512 zfs513 zfs514 zfs515 zfs516 zfs517 zfs518 zfs519 zfs520 zfs521 zfs522 zfs523 zfs524 zfs525 zfs526 zfs527 zfs528 zfs529 zfs530 zfs531 zfs532 zfs533 zfs534 zfs535 zfs536 zfs537 zfs538 zfs539 zfs540 zfs541 zfs542 zfs543 zfs544 zfs545 zfs546 zfs547 zfs548 zfs549 zfs550 zfs551 zfs552 zfs553 zfs554 zfs555 zfs556 zfs557 zfs558 zfs559 zfs560 zfs561 zfs562 zfs563 zfs564 zfs565 zfs566 zfs567 zfs568 zfs569 zfs570 zfs571 zfs572 zfs573 zfs574 zfs575 zfs576 zfs577 zfs578 zfs579 zfs580 zfs581 zfs582 zfs583 zfs584 zfs585 zfs586 zfs587 zfs588 zfs589 zfs590 zfs591 zfs592 zfs593 zfs594 zfs595 zfs596 zfs597 zfs598 zfs599 zfs600 zfs601 zfs602 zfs603 zfs604 zfs605 zfs606 zfs607 zfs608 zfs609 zfs610 zfs611 zfs612 zfs613 zfs614 zfs615 zfs616 zfs617 zfs618 zfs619 zfs620 zfs621 zfs622 zfs623 zfs624 zfs625 zfs626 zfs627 zfs628 zfs629 zfs630 zfs631 zfs632 zfs633 zfs634 zfs635 zfs636 zfs637 zfs638 zfs639 zfs640 zfs641 zfs642 zfs643 zfs644 zfs645 zfs646 zfs647 zfs648 zfs649 zfs650 zfs651 zfs652 zfs653 zfs654 zfs655 zfs656 zfs657 zfs658 zfs659 zfs660 zfs661 zfs662 zfs663 zfs664 zfs665 zfs666 zfs667 zfs668 zfs669 zfs670 zfs671 zfs672 zfs673 zfs674 zfs675 zfs676 zfs677 zfs678 zfs679 zfs680 zfs681 zfs682 zfs683 zfs684 zfs685 zfs686 zfs687 zfs688 zfs689 zfs690 zfs691 zfs692 zfs693 zfs694 zfs695 zfs696 zfs697 zfs698 zfs699 zfs700 zfs701 zfs702 zfs703 zfs704 zfs705 zfs706 zfs707 zfs708 zfs709 zfs710 zfs711 zfs712 zfs713 zfs714 zfs715 zfs716 zfs717 zfs718 zfs719 zfs720 zfs721 zfs722 zfs723 zfs724 zfs725 zfs726 zfs727 zfs728 zfs729 zfs730 zfs731 zfs732 zfs733 zfs734 zfs735 zfs736 zfs737 zfs738 zfs739 zfs740 zfs741 zfs742 zfs743 zfs744 zfs745 zfs746 zfs747 zfs748 zfs749 zfs750 zfs751 zfs752 zfs753 zfs754 zfs755 zfs756 zfs757 zfs758 zfs759 zfs760 zfs761 zfs762 zfs763 zfs764 zfs765 zfs766 zfs767 zfs768 zfs769 zfs770 zfs771 zfs772 zfs773 zfs774 zfs775 zfs776 zfs777 zfs778 zfs779 zfs780 zfs781 zfs782 zfs783 zfs784 zfs785 zfs786 zfs787 zfs788 zfs789 zfs790 zfs791 zfs792 zfs793 zfs794 zfs795 zfs796 zfs797 zfs798 zfs799 zfs800 zfs801 zfs802 zfs803 zfs804 zfs805 zfs806 zfs807 zfs808 zfs809 zfs810 zfs811 zfs812 zfs813 zfs814 zfs815 zfs816 zfs817 zfs818 zfs819 zfs820 zfs821 zfs822 zfs823 zfs824 zfs825 zfs826 zfs827 zfs828 zfs829 zfs830 zfs831 zfs832 zfs833 zfs834 zfs835 zfs836 zfs837 zfs838 zfs839 zfs840 zfs841 zfs842 zfs843 zfs844 zfs845 zfs846 zfs847 zfs848 zfs849 zfs850 zfs851 zfs852 zfs853 zfs854 zfs855 zfs856 zfs857 zfs858 zfs859 zfs860 zfs861 zfs862 zfs863 zfs864 zfs865 zfs866 zfs867 zfs868 zfs869 zfs870 zfs871 zfs872 zfs873 zfs874 zfs875 zfs876 zfs877 zfs878 zfs879 zfs880 zfs881 zfs882 zfs883 zfs884 zfs885 zfs886 zfs887 zfs888 zfs889 zfs890 zfs891 zfs892 zfs893 zfs894 zfs895 zfs896 zfs897 zfs898 zfs899 zfs900 zfs901 zfs902 zfs903 zfs904 zfs905 zfs906 zfs907 zfs908 zfs909 zfs910 zfs911 zfs912 zfs913 zfs914 zfs915 zfs916 zfs917 zfs918 zfs919 zfs920 zfs921 zfs922 zfs923 zfs924 zfs925 zfs926 zfs927 zfs928 zfs929 zfs930 zfs931 zfs932 zfs933 zfs934 zfs935 zfs936 zfs937 zfs938 zfs939 zfs940 zfs941 zfs942 zfs943 zfs944 zfs945 zfs946 zfs947 zfs948 zfs949 zfs950 zfs951 zfs952 zfs953 zfs954 zfs955 zfs956 zfs957 zfs958 zfs959 zfs960 zfs961 zfs962 zfs963 zfs964 zfs965 zfs966 zfs967 zfs968 zfs969 zfs970 zfs971 zfs972 zfs973 zfs974 zfs975 zfs976 zfs977 zfs978 zfs979 zfs980 zfs981 zfs982 zfs983 zfs984 zfs985 zfs986 zfs987 zfs988 zfs989 zfs990 zfs991 zfs992 zfs993 zfs994 zfs995 zfs996 zfs997 zfs998 zfs999 zfs1000 zfs1001 zfs1002 zfs1003 zfs1004 zfs1005 zfs1006 zfs1007 zfs1008 zfs1009 zfs1010 zfs1011 zfs1012 zfs1013 zfs1014 zfs1015 zfs1016 zfs1017 zfs1018 zfs1019 zfs1020 zfs1021 zfs1022 zfs1023 zfs1024 zfs1025 zfs1026 zfs1027 zfs1028 zfs1029 zfs1030 zfs1031 zfs1032 zfs1033 zfs1034 zfs1035 zfs1036 zfs1037 zfs1038 zfs1039 zfs1040 zfs1041 zfs1042 zfs1043 zfs1044 zfs1045 zfs1046 zfs1047 zfs1048 zfs1049 zfs1050 zfs1051 zfs1052 zfs1053 zfs1054 zfs1055 zfs1056 zfs1057 zfs1058 zfs1059 zfs1060 zfs1061 zfs1062 zfs1063 zfs1064 zfs1065 zfs1066 zfs1067 zfs1068 zfs1069 zfs1070 zfs1071 zfs1072 zfs1073 zfs1074 zfs1075 zfs1076 zfs1077 zfs1078 zfs1079 zfs1080 zfs1081 zfs1082 zfs1083 zfs1084 zfs1085 zfs1086 zfs1087 zfs1088 zfs1089 zfs1090 zfs1091 zfs1092 zfs1093 zfs1094 zfs1095 zfs1096 zfs1097 zfs1098 zfs1099 zfs1100 zfs1101 zfs1102 zfs1103 zfs1104 zfs1105 zfs1106 zfs1107 zfs1108 zfs1109 zfs1110 zfs1111 zfs1112 zfs1113 zfs1114 zfs1115 zfs1116 zfs1117 zfs1118 zfs1119 zfs1120 zfs1121 zfs1122 zfs1123 zfs1124 zfs1125 zfs1126 zfs1127 zfs1128 zfs1129 zfs1130 zfs1131 zfs1132 zfs1133 zfs1134 zfs1135 zfs1136 zfs1137 zfs1138 zfs1139 zfs1140 zfs1141 zfs1142 zfs1143 zfs1144 zfs1145 zfs1146 zfs1147 zfs1148 zfs1149 zfs1150 zfs1151 zfs1152 zfs1153 zfs1154 zfs1155 zfs1156 zfs1157 zfs1158 zfs1159 zfs1160 zfs1161 zfs1162 zfs1163 zfs1164 zfs1165 zfs1166 zfs1167 zfs1168 zfs1169 zfs1170 zfs1171 zfs1172 zfs1173 zfs1174 zfs1175 zfs1176 zfs1177 zfs1178 zfs1179 zfs1180 zfs1181 zfs1182 zfs1183 zfs1184 zfs1185 zfs1186 zfs1187 zfs1188 zfs1189 zfs1190 zfs1191 zfs1192 zfs1193 zfs1194 zfs1195 zfs1196 zfs1197 zfs1198 zfs1199 zfs1200 zfs1201 zfs1202 zfs1203 zfs1204 zfs1205 zfs1206 zfs1207 zfs1208 zfs1209 zfs1210 zfs1211 zfs1212 zfs1213 zfs1214 zfs1215 zfs1216 zfs1217 zfs1218 zfs1219 zfs1220 zfs1221 zfs1222 zfs1223 zfs1224 zfs1225 zfs1226 zfs1227 zfs1228 zfs1229 zfs1230 zfs1231 zfs1232 zfs1233 zfs1234 zfs1235 zfs1236 zfs1237 zfs1238 zfs1239 zfs1240 zfs1241 zfs1242 zfs1243 zfs1244 zfs1245 zfs1246 zfs1247 zfs1248 zfs1249 zfs1250 zfs1251 zfs1252 zfs1253 zfs1254 zfs1255 zfs1256 zfs1257 zfs1258 zfs1259 zfs1260 zfs1261 zfs1262 zfs1263 zfs1264 zfs1265 zfs1266 zfs1267 zfs1268 zfs1269 zfs1270 zfs1271 zfs1272 zfs1273 zfs1274 zfs1275 zfs1276 zfs1277 zfs1278 zfs1279 zfs1280 zfs1281 zfs1282 zfs1283 zfs1284 zfs1285 zfs1286 zfs1287 zfs1288 zfs1289 zfs1290 zfs1291 zfs1292 zfs1293 zfs1294 zfs1295 zfs1296 zfs1297 zfs1298 zfs1299 zfs1300 zfs1301 zfs1302 zfs1303 zfs1304 zfs1305 zfs1306 zfs1307 zfs1308 zfs1309 zfs1310 zfs1311 zfs1312 zfs1313 zfs1314 zfs1315 zfs1316 zfs1317 zfs1318 zfs1319 zfs1320 zfs1321 zfs1322 zfs1323 zfs1324 zfs1325 zfs1326 zfs1327 zfs1328 zfs1329 zfs1330 zfs1331 zfs1332 zfs1333 zfs1334 zfs1335 zfs1336 zfs1337 zfs1338 zfs1339 zfs1340 zfs1341 zfs1342 zfs1343 zfs1344 zfs1345 zfs1346 zfs1347 zfs1348 zfs1349 zfs1350 zfs1351 zfs1352 zfs1353 zfs1354 zfs1355 zfs1356 zfs1357 zfs1358 zfs1359 zfs1360 zfs1361 zfs1362 zfs1363 zfs1364 zfs1365 zfs1366 zfs1367 zfs1368 zfs1369 zfs1370 zfs1371 zfs1372 zfs1373 zfs1374 zfs1375 zfs1376 zfs1377 zfs1378 zfs1379 zfs1380 zfs1381 zfs1382 zfs1383 zfs1384 zfs1385 zfs1386 zfs1387 zfs1388 zfs1389 zfs1390 zfs1391 zfs1392 zfs1393 zfs1394 zfs1395 zfs1396 zfs1397 zfs1398 zfs1399 zfs1400 zfs1401 zfs1402 zfs1403 zfs1404 zfs1405 zfs1406 zfs1407 zfs1408 zfs1409 zfs1410 zfs1411 zfs1412 zfs1413 zfs1414 zfs1415 zfs1416 zfs1417 zfs1418 zfs1419 zfs1420 zfs1421 zfs1422 zfs1423 zfs1424 zfs1425 zfs1426 zfs1427 zfs1428 zfs1429 zfs1430 zfs1431 zfs1432 zfs1433 zfs1434 zfs1435 zfs1436 zfs1437 zfs1438 zfs1439 zfs1440 zfs1441 zfs1442 zfs1443 zfs1444 zfs1445 zfs1446 zfs1447 zfs1448 zfs1449 zfs1450 zfs1451 zfs1452 zfs1453 zfs1454 zfs1455 zfs1456 zfs1457 zfs1458 zfs1459 zfs1460 zfs1461 zfs1462 zfs1463 zfs1464 zfs1465 zfs1466 zfs1467 zfs1468 zfs1469 zfs1470 zfs1471 zfs1472 zfs1473 zfs1474 zfs1475 zfs1476 zfs1477 zfs1478 zfs1479 zfs1480 zfs1481 zfs1482 zfs1483 zfs1484 zfs1485 zfs1486 zfs1487 zfs1488 zfs1489 zfs1490 zfs1491 zfs1492 zfs1493 zfs1494 zfs1495 zfs1496 zfs1497 zfs1498 zfs1499 zfs1500 zfs1501 zfs1502 zfs1503 zfs1504 zfs1505 zfs1506 zfs1507 zfs1508 zfs1509 zfs1510 zfs1511 zfs1512 zfs1513 zfs1514 zfs1515 zfs1516 zfs1517 zfs1518 zfs1519 zfs1520 zfs1521 zfs1522 zfs1523 zfs1524 zfs1525 zfs1526 zfs1527 zfs1528 zfs1529 zfs1530 zfs1531 zfs1532 zfs1533 zfs1534 zfs1535 zfs1536 zfs1537 zfs1538 zfs1539 zfs1540 zfs1541 zfs1542 zfs1543 zfs1544 zfs1545 zfs1546 zfs1547 zfs1548 zfs1549 zfs1550 zfs1551 zfs1552 zfs1553 zfs1554 zfs1555 zfs1556 zfs1557 zfs1558 zfs1559 zfs1560 zfs1561 zfs1562 zfs1563 zfs1564 zfs1565 zfs1566 zfs1567 zfs1568 zfs1569 zfs1570 zfs1571 zfs1572 zfs1573 zfs1574 zfs1575 zfs1576 zfs1577 zfs1578 zfs1579 zfs1580 zfs1581 zfs1582 zfs1583 zfs1584 zfs1585 zfs1586 zfs1587 zfs1588 zfs1589 zfs1590 zfs1591 zfs1592 zfs1593 zfs1594 zfs1595 zfs1596 zfs1597 zfs1598 zfs1599 zfs1600 zfs1601 zfs1602 zfs1603 zfs1604 zfs1605 zfs1606 zfs1607 zfs1608 zfs1609 zfs1610 zfs1611 zfs1612 zfs1613 zfs1614 zfs1615 zfs1616 zfs1617 zfs1618 zfs1619 zfs1620 zfs1621 zfs1622 zfs1623 zfs1624 zfs1625 zfs1626 zfs1627 zfs1628 zfs1629 zfs1630 zfs1631 zfs1632 zfs1633 zfs1634 zfs1635 zfs1636 zfs1637 zfs1638 zfs1639 zfs1640 zfs1641 zfs1642 zfs1643 zfs1644 zfs1645 zfs1646 zfs1647 zfs1648 zfs1649 zfs1650 zfs1651 zfs1652 zfs1653 zfs1654 zfs1655 zfs1656 zfs1657 zfs1658 zfs1659 zfs1660 zfs1661 zfs1662 zfs1663 zfs1664 zfs1665 zfs1666 zfs1667 zfs1668 zfs1669 zfs1670 zfs1671 zfs1672 zfs1673 zfs1674 zfs1675 zfs1676 zfs1677 zfs1678 zfs1679 zfs1680 zfs1681 zfs1682 zfs1683 zfs1684 zfs1685 zfs1686 zfs1687 zfs1688 zfs1689 zfs1690 zfs1691 zfs1692 zfs1693 zfs1694 zfs1695 zfs1696 zfs1697 zfs1698 zfs1699 zfs1700 zfs1701 zfs1702 zfs1703 zfs1704 zfs1705 zfs1706 zfs1707 zfs1708 zfs1709 zfs1710 zfs1711 zfs1712 zfs1713 zfs1714 zfs1715 zfs1716 zfs1717 zfs1718 zfs1719 zfs1720 zfs1721 zfs1722 zfs1723 zfs1724 zfs1725 zfs1726 zfs1727 zfs1728 zfs1729 zfs1730 zfs1731 zfs1732 zfs1733 zfs1734 zfs1735 zfs1736 zfs1737 zfs1738 zfs1739 zfs1740 zfs1741 zfs1742 zfs1743 zfs1744 zfs1745 zfs1746 zfs1747 zfs1748 zfs1749 zfs1750 zfs1751 zfs1752 zfs1753 zfs1754 zfs1755 zfs1756 zfs1757 zfs1758 zfs1759 zfs1760 zfs1761 zfs1762 zfs1763 zfs1764 zfs1765 zfs1766 zfs1767 zfs1768 zfs1769 zfs1770 zfs1771 zfs1772 zfs1773 zfs1774 zfs1775 zfs1776 zfs1777 zfs1778 zfs1779 zfs1780 zfs1781 zfs1782 zfs1783 zfs1784 zfs1785 zfs1786 zfs1787 zfs1788 zfs1789 zfs1790 zfs1791 zfs1792 zfs1793 zfs1794 zfs1795 zfs1796 zfs1797 zfs1798 zfs1799 zfs1800 zfs1801 zfs1802 zfs1803 zfs1804 zfs1805 zfs1806 zfs1807 zfs1808 zfs1809 zfs1810 zfs1811 zfs1812 zfs1813 zfs1814 zfs1815 zfs1816 zfs1817 zfs1818 zfs1819 zfs1820 zfs1821 zfs1822 zfs1823 zfs1824 zfs1825 zfs1826 zfs1827 zfs1828 zfs1829 zfs1830 zfs1831 zfs1832 zfs1833 zfs1834 zfs1835 zfs1836 zfs1837 zfs1838 zfs1839 zfs1840 zfs1841 zfs1842 zfs1843 zfs1844 zfs1845 zfs1846 zfs1847 zfs1848 zfs1849 zfs1850 zfs1851 zfs1852 zfs1853 zfs1854 zfs1855 zfs1856 zfs1857 zfs1858 zfs1859 zfs1860 zfs1861 zfs1862 zfs1863 zfs1864 zfs1865 zfs1866 zfs1867 zfs1868 zfs1869 zfs1870 zfs1871 zfs1872 zfs1873 zfs1874 zfs1875 zfs1876 zfs1877 zfs1878 zfs1879 zfs1880 zfs1881 zfs1882 zfs1883 zfs1884 zfs1885 zfs1886 zfs1887 zfs1888 zfs1889 zfs1890 zfs1891 zfs1892 zfs1893 zfs1894 zfs1895 zfs1896 zfs1897 zfs1898 zfs1899 zfs1900 zfs1901 zfs1902 zfs1903 zfs1904 zfs1905 zfs1906 zfs1907 zfs1908 zfs1909 zfs1910 zfs1911 zfs1912 zfs1913 zfs1914 zfs1915 zfs1916 zfs1917 zfs1918 zfs19	

Name	Expression	Matches
	<code>^(In)?[Ll]oop[Bb]ack[0-9._]*\$</code>	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.
	<code>^NULL[0-9.]*\$</code>	Strings starting with "NULL" optionally followed by any number of digits or dots.
	<code>^[Ll]o[0-9.]*\$</code>	Strings starting with "Lo" or "lo" and optionally followed by any number of digits or dots.
	<code>^[Ss]ystem\$</code>	"System" or "system"
	<code>^Nu[0-9.]*\$</code>	Strings starting with "Nu" optionally followed by any number of digits or dots.
Storage devices for SNMP discovery	<code>^(Physical memory\\ Virtual memory\\ Memory buffers\\ Cached memory\\ Swap space)\$</code>	"Physical memory" or "Virtual memory" or "Memory buffers" or "Cached memory" or "Swap space"
Windows service names for discovery	<code>^(MMCSS\\ gupdate\\ SysmonLog\\ clr_optimization_v4.0.30319\\ MMCSS50727_32\\ clr_optimization_v2.0.50727_32\\ clr_optimization_v4.0.30319_32\$</code>	"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.
Windows service startup states for discovery	<code>^(automatic\\ automatic delayed)\$</code>	"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$`

Test string

TESTDATABASE

Test expressions

Result	Expression type	Expression	Result
	Result is FALSE	<code>^TESTDATABASE</code>	FALSE
	Combined result		FALSE

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 (?) to match the characters "error":

`(?i)error`

Test string

Sometexthere1345Error1357

Test expressions

Result	Expression type	Expression	Result
	Result is TRUE	(?i)error	TRUE
	Combined result		TRUE

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 characters

Test string

Some text here for your consideration
1235kfd345
match eveRything after this line
Continuation

Test expressions

Result	Expression type	Expression	Result
	Result is TRUE	(?<=match (?i)everything(?-i) after this line\n)(?sx).*# we add s modifier to allow . match newline characters	TRUE
	Combined result		TRUE

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 [pcre syntax man page](#). For more information about PCRE syntax please refer to [PCRE HTML documentation](#).

More complex example

A custom regular expression may consist of multiple subexpressions, and it can be tested in the **Test** tab by providing a test string.

Expressions

Test

Test string

lo

Test expressions

Result	Expression type	Expression	Result
Result is FALSE		^Software Loopback Interface	TRUE
Result is FALSE		^(In)?[Ll]oop[Bb]ack[0-9._]*\$	TRUE
Result is FALSE		^NULL[0-9.]*\$	TRUE
Result is FALSE		^[Ll]o[0-9.]*\$	FALSE
Result is FALSE		^[Ss]ystem\$	TRUE
Result is FALSE		^Nu[0-9.]*\$	TRUE
	Combined result		FALSE

Update

Clone

Delete

Cancel

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.

Regular expression support by location

Location	Regular expression	Global regular expression	Comments
Agent items			
	eventlog[log[log.count[logrt[	Yes	regexp, severity, source, eventid parameters regexp parameter
	logrt.count[proc.cpu.util[proc.mem[proc.num[sensor[	Yes/No	regexp parameter supports both, file_regexp parameter supports non-global expressions only
	system.hw.macaddr[system.sw.packages[	No	cmdline parameter device and sensor parameters on Linux 2.4 interface parameter package parameter

Location	Regular expression	Global regular expression	Comments
	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[]		regexp parameter
	vfs.file.regmatch[]		
	web.page.regexp[]		
SNMP traps	snmptrap[]	Yes	regexp parameter
Item value preprocessing	Yes	No	pattern parameter
Trigger functions	count() ~~if~~	Yes	pattern parameter if operator parameter is <i>regexp</i> or <i>iregexp</i> pattern parameter
	logeventid()		
	logsource() iregexp() regexp()		
Low-level discovery	Yes	Yes	<i>Filter</i> field
Action conditions	Yes	No	In <i>matches</i> , does not match options for <i>Host</i> name and <i>Host</i> <i>metadata</i> auto- registration conditions
Web monitoring	Yes	No	<i>Variables</i> with a regex : prefix <i>Required</i> <i>string</i> field
Macro functions	regsub() ~~if~~	No	pattern parameter
	iregsub()		
Icon mapping	Yes	Yes	<i>Expression</i> field

13. Problem acknowledgement

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, navigate from the problem list to the problem update screen 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 more coordinated way.

Acknowledgement 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 screen, which allows to acknowledge 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 acknowledgement status of problems in:
 - *Monitoring → Dashboard (Problems and Problems by severity widgets)*
 - *Monitoring → Problems*
 - *Monitoring → Problems → Event details*
 - *Monitoring → Screens (Host group issues, Host issues, Problems by severity elements)*

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 screen.

- You can click on an unresolved problem cell in:
 - *Monitoring → Dashboard (Trigger overview widget)*
 - *Monitoring → Overview*
 - *Monitoring → Screens (Trigger overview element)*

The popup menu contains an *Acknowledge* option that will take you to the problem update screen.

Updating problems The problem update screen allows to:

- comment on the problem
- view comments and actions so far
- change problem severity
- acknowledge problem
- manually close problem

Update problem

Message

Resolved.

History

Time	User	User action	Message
2018-06-20 07:46:43	Admin (Zabbix Administrator)		Started working on it.

Scope

☒ Only selected problem

☐ Selected and all other problems of related triggers 1 event

Change severity

☒

Not classified

Information

Warning

Average

High

Disaster

Acknowledge

☒

Close problem

☐

* At least one update operation or message must exist.

Update

Cancel

All mandatory input fields are marked with a red asterisk.

Parameter	Description
Message	Enter text to comment on the problem.
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.
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 acknowledgement/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.

Parameter	Description
<i>Change severity</i>	Mark the checkbox and click on the severity button to update problem severity. Since Zabbix 4.2.6, 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 <i>Update</i>. If read-write permissions exist for none of the selected triggers, the checkbox is disabled. Before Zabbix 4.2.6, changing severity in this way was possible if read-write permissions existed for each selected problem. If at least one problem was not read-writable, the checkbox was disabled.
<i>Acknowledge</i>	Mark the checkbox to acknowledge the problem. This option is disabled if all selected problems are already acknowledged. If at least one is unacknowledged, the option is available and in this case it will be possible to acknowledge only the unacknowledged problem(s). It is not possible to add another acknowledgement for an already acknowledged problem (it is possible to add another comment though).
<i>Close problem</i>	Mark the checkbox to manually close the problem. Since Zabbix 4.2.6, the checkbox for closing a problem is available if the <i>Allow manual close</i> 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 <i>Update</i>. If no problem is manually closeable, the checkbox is disabled. Already closed problems will not be closed repeatedly. Before Zabbix 4.2.6, closing a problem in this way was possible if the <i>Allow manual close</i> option was checked in trigger configuration for each selected problem. If at least one problem was not manually closeable, the checkbox was disabled.

Display Based on acknowledgement 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 (acknowledgement, etc.) it is possible to configure update operations - send message or execute remote commands.

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 XML into Zabbix.
- integrate with third-party tools - the universal XML format makes 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)
- **templates**
- **hosts**
- **network maps**

- images
- **screens**
- value maps

Export format

Data can be exported using the Zabbix web frontend or **Zabbix API**. Supported export formats are:

- XML - in the frontend
- XML or JSON - in Zabbix API

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 imported XML 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 both XML and JSON, the import file must have a correct file extension: .xml for XML and .json for JSON.
- See **compatibility information** about supported XML versions.

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
  <version>4.0</version>
  <date>2016-10-04T06:20:11Z</date>
</zabbix_export>
```

XML base format

```
<?xml version="1.0" encoding="UTF-8"?>
```

Default header for XML documents.

```
<zabbix_export>
```

Root element for Zabbix XML export.

```
<version>4.0</version>
```

Export version.

```
<date>2016-10-04T06:20:11Z</date>
```

Date when export was created in ISO 8601 long format.

Other tags are dependent on exported objects.

1 Host groups

In the frontend host groups can be **exported** only with host or template export. When a host or template is exported all groups it belongs to are exported with it automatically.

API allows to export host groups independently from hosts or templates.

```

<groups>
  <group>
    <name>Zabbix servers</name>
  </group>
</groups>

```

groups/group

Parameter	Type	Description	Details
name	<i>string</i>	Group name.	

2 Templates

Overview

Templates are **exported** with many related objects and object relations.

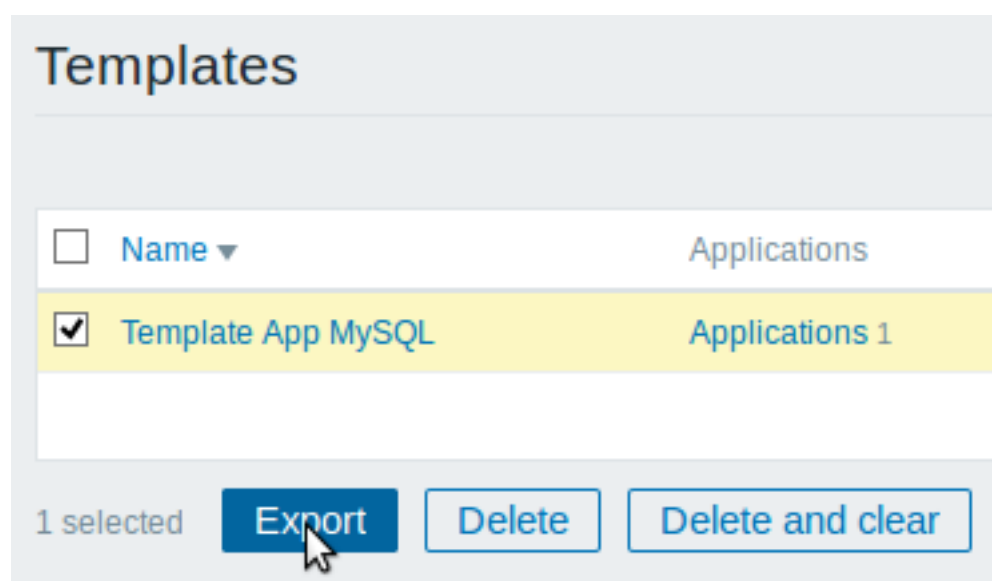
Template export contains:

- linked host groups
- template data
- linkage to other templates
- linkage to host groups
- directly linked applications
- directly linked items
- directly linked triggers
- directly linked graphs
- directly linked screens
- 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



Selected templates are exported to a local XML file with default name *zabbix_export_templates.xml*.

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*

* Import file

Browse... No file selected.

Rules	Update existing	Create new	Delete missing
Groups		☒	
Hosts	☐	☐	
Templates	☒	☒	
Template screens	☒	☒	☐
Template linkage		☒	
Applications		☒	☐
Items	☒	☒	☐
Discovery rules	☒	☒	☐
Triggers	☒	☒	☐
Graphs	☒	☒	☐
Web scenarios	☒	☒	☐
Screens	☐	☐	
Maps	☐	☐	
Images	☐	☐	
Value mappings	☐	☒	

Import

Cancel

When a template is imported and updated, it can only be linked to additional templates and never be unlinked from any.

A success or failure message of the import will be displayed in the frontend.

Import rules:

Rule	Description
<i>Update existing</i>	Existing elements will be updated with data taken from the import file. Otherwise they will not be updated.
<i>Create new</i>	The import will add new elements using data from the import file. Otherwise it will not add them.
<i>Delete missing</i>	The import will remove existing elements not present in the import file. Otherwise it will not remove them.

Export format

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
```

```

<version>4.0</version>
<date>2018-04-03T06:25:22Z</date>
<groups>
  <group>
    <name>Templates/Databases</name>
  </group>
</groups>
<templates>
  <template>
    <template>Template DB MySQL</template>
    <name>Template DB MySQL</name>
    <description/>
    <groups>
      <group>
        <name>Templates/Databases</name>
      </group>
    </groups>
    <applications>
      <application>
        <name>MySQL</name>
      </application>
    </applications>
    <items>
      <item>
        <name>MySQL status</name>
        <type>0</type>
        <snmp_community/>
        <snmp_oid/>
        <key>mysql.ping</key>
        <delay>1m</delay>
        <history>1w</history>
        <trends>365d</trends>
        <status>0</status>
        <value_type>3</value_type>
        <allowed_hosts/>
        <units/>
        <snmpv3_contextname/>
        <snmpv3_securityname/>
        <snmpv3_securitylevel>0</snmpv3_securitylevel>
        <snmpv3_authprotocol>0</snmpv3_authprotocol>
        <snmpv3_authpassphrase/>
        <snmpv3_privprotocol>0</snmpv3_privprotocol>
        <snmpv3_privpassphrase/>
        <params/>
        <ipmi_sensor/>
        <authtype>0</authtype>
        <username/>
        <password/>
        <publickey/>
        <privatekey/>
        <port/>
        <description>It requires user parameter mysql.ping, which is defined in userparameter_
0 - MySQL server is down
1 - MySQL server is up</description>
        <inventory_link>0</inventory_link>
        <applications>
          <application>
            <name>MySQL</name>
          </application>
        </applications>
        <valuemap>

```

```

        <name>Service state</name>
    </valuemap>
    <logtimefmt/>
    <preprocessing/>
    <jmx_endpoint/>
    <timeout>3s</timeout>
    <url/>
    <query_fields/>
    <posts/>
    <status_codes>200</status_codes>
    <follow_redirects>1</follow_redirects>
    <post_type>0</post_type>
    <http_proxy/>
    <headers/>
    <retrieve_mode>0</retrieve_mode>
    <request_method>1</request_method>
    <output_format>0</output_format>
    <allow_traps>0</allow_traps>
    <ssl_cert_file/>
    <ssl_key_file/>
    <ssl_key_password/>
    <verify_peer>0</verify_peer>
    <verify_host>0</verify_host>
    <master_item/>
</item>
<item>
    <name>MySQL begin operations per second</name>
    <type>0</type>
    <snmp_community/>
    <snmp_oid/>
    <key>mysql.status[Com_begin]</key>
    <delay>1m</delay>
    <history>1w</history>
    <trends>365d</trends>
    <status>0</status>
    <value_type>0</value_type>
    <allowed_hosts/>
    <units>qps</units>
    <snmpv3_contextname/>
    <snmpv3_securityname/>
    <snmpv3_securitylevel>0</snmpv3_securitylevel>
    <snmpv3_authprotocol>0</snmpv3_authprotocol>
    <snmpv3_authpassphrase/>
    <snmpv3_privprotocol>0</snmpv3_privprotocol>
    <snmpv3_privpassphrase/>
    <params/>
    <ipmi_sensor/>
    <authtype>0</authtype>
    <username/>
    <password/>
    <publickey/>
    <privatekey/>
    <port/>
    <description>It requires user parameter mysql.status[*], which is defined in userparam
    <inventory_link>0</inventory_link>
    <applications>
        <application>
            <name>MySQL</name>
        </application>
    </applications>
    <valuemap/>
    <logtimefmt/>

```

```

    <preprocessing>
      <step>
        <type>10</type>
        <params/>
        <error_handler>0</error_handler>
        <error_handler_params/>
      </step>
    </preprocessing>
  </jmx_endpoint/>
  <timeout>3s</timeout>
  <url/>
  <query_fields/>
  <posts/>
  <status_codes>200</status_codes>
  <follow_redirects>1</follow_redirects>
  <post_type>0</post_type>
  <http_proxy/>
  <headers/>
  <retrieve_mode>0</retrieve_mode>
  <request_method>1</request_method>
  <output_format>0</output_format>
  <allow_traps>0</allow_traps>
  <ssl_cert_file/>
  <ssl_key_file/>
  <ssl_key_password/>
  <verify_peer>0</verify_peer>
  <verify_host>0</verify_host>
  <master_item/>
</item>
<item>
  <name>MySQL queries per second</name>
  <type>0</type>
  <snmp_community/>
  <snmp_oid/>
  <key>mysql.status[Questions]</key>
  <delay>1m</delay>
  <history>1w</history>
  <trends>365d</trends>
  <status>0</status>
  <value_type>0</value_type>
  <allowed_hosts/>
  <units>qps</units>
  <snmpv3_contextname/>
  <snmpv3_securityname/>
  <snmpv3_securitylevel>0</snmpv3_securitylevel>
  <snmpv3_authprotocol>0</snmpv3_authprotocol>
  <snmpv3_authpassphrase/>
  <snmpv3_privprotocol>0</snmpv3_privprotocol>
  <snmpv3_privpassphrase/>
  <params/>
  <ipmi_sensor/>
  <authtype>0</authtype>
  <username/>
  <password/>
  <publickey/>
  <privatekey/>
  <port/>
  <description>It requires user parameter mysql.status[*], which is defined in userparam
  <inventory_link>0</inventory_link>
  <applications>
    <application>
      <name>MySQL</name>

```



```

        </application>
    </applications>
    <valuemap/>
    <logtimefmt/>
    <preprocessing>
        <step>
            <type>10</type>
            <params/>
            <error_handler>0</error_handler>
            <error_handler_params/>
        </step>
    </preprocessing>
    <jmx_endpoint/>
    <timeout>3s</timeout>
    <url/>
    <query_fields/>
    <posts/>
    <status_codes>200</status_codes>
    <follow_redirects>1</follow_redirects>
    <post_type>0</post_type>
    <http_proxy/>
    <headers/>
    <retrieve_mode>0</retrieve_mode>
    <request_method>1</request_method>
    <output_format>0</output_format>
    <allow_traps>0</allow_traps>
    <ssl_cert_file/>
    <ssl_key_file/>
    <ssl_key_password/>
    <verify_peer>0</verify_peer>
    <verify_host>0</verify_host>
    <master_item/>
</item>
</items>
<discovery_rules/>
<httptests/>
<macros/>
<templates/>
<screens>
    <screen>
        <name>MySQL performance</name>
        <hsize>2</hsize>
        <vsize>1</vsize>
        <screen_items>
            <screen_item>
                <resourcetype>0</resourcetype>
                <width>500</width>
                <height>200</height>
                <x>0</x>
                <y>0</y>
                <colspan>1</colspan>
                <rowspan>1</rowspan>
                <elements>0</elements>
                <valign>1</valign>
                <halign>0</halign>
                <style>0</style>
                <url/>
                <dynamic>0</dynamic>
                <sort_triggers>0</sort_triggers>
                <resource>
                    <name>MySQL operations</name>
                    <host>Template DB MySQL</host>

```

```

        </resource>
        <max_columns>3</max_columns>
        <application/>
    </screen_item>
</screen_items>
</screen>
</screens>
</template>
</templates>
<triggers>
    <trigger>
        <expression>{Template DB MySQL:mysql.ping.last(0)}=0</expression>
        <recovery_mode>0</recovery_mode>
        <recovery_expression/>
        <name>MySQL is down</name>
        <correlation_mode>0</correlation_mode>
        <correlation_tag/>
        <url/>
        <status>0</status>
        <priority>2</priority>
        <description/>
        <type>0</type>
        <manual_close>0</manual_close>
        <dependencies/>
        <tags/>
    </trigger>
</triggers>
<graphs>
    <graph>
        <name>MySQL operations</name>
        <width>900</width>
        <height>200</height>
        <yaxismin>0.0000</yaxismin>
        <yaxismax>100.0000</yaxismax>
        <show_work_period>1</show_work_period>
        <show_triggers>1</show_triggers>
        <type>0</type>
        <show_legend>1</show_legend>
        <show_3d>0</show_3d>
        <percent_left>0.0000</percent_left>
        <percent_right>0.0000</percent_right>
        <ymin_type_1>0</ymin_type_1>
        <ymin_type_1>0</ymin_type_1>
        <ymin_item_1>0</ymin_item_1>
        <ymin_item_1>0</ymin_item_1>
        <graph_items>
            <graph_item>
                <sortorder>0</sortorder>
                <drawtype>0</drawtype>
                <color>C8C800</color>
                <yaxisside>0</yaxisside>
                <calc_fnc>2</calc_fnc>
                <type>0</type>
                <item>
                    <host>Template DB MySQL</host>
                    <key>mysql.status[Com_begin]</key>
                </item>
            </graph_item>
        </graph_items>
    </graph>
</graphs>
<value_maps>

```

```

    <value_map>
      <name>Service state</name>
      <mappings>
        <mapping>
          <value>0</value>
          <newvalue>Down</newvalue>
        </mapping>
        <mapping>
          <value>1</value>
          <newvalue>Up</newvalue>
        </mapping>
      </mappings>
    </value_map>
  </value_maps>
</zabbix_export>

```

Element tags

Element tag values are explained in the table below.

Template tags

Element	Element property	Type	Range	Description
templates				Root element for templates.
template				Individual template.
	template	string		Unique template name.
	name	string		Visible template name.
	description	text		Template description.
groups				Root element for host groups.
group				Individual host group.
	name	string		Unique group name.
applications				Root element for template applications.
application				Individual template application.
	name			Application name.
macros				Root element for template user macros.
macro				Individual template user macro.
	name			User macro name.
	value			User macro value.
templates				Root element for linked templates.
template				Individual template.
	name	string		Template name.

Template item tags

Element	Element property	Type	Range	Description
items				Root element for items.
item				Individual item.
	name	string		Item name.

Element	Element property	Type	Range	Description
	type	integer	0 - Zabbix agent 1 - SNMPv1 agent 2 - Zabbix trapper 3 - simple check 4 - SNMPv2 agent 5 - internal 6 - SNMPv3 agent 7 - Zabbix agent (active) 8 - aggregate 9 - HTTP test (web monitoring scenario step) 10 - external 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 item	Item type.
	snmp_community	string		SNMP community name if 'type' is 1,4.
	snmp_oid	string		SNMP object ID.
	key	string		Item key.
	delay	string		Update interval of the item. Seconds, time unit with suffix, custom intervals, user macros or LLD macros.
	history	string		A time unit of how long the history data should be stored. Time unit with suffix, user macro or LLD macro.
	trends	string		A time unit of how long the trends data should be stored. Time unit with suffix, user macro or LLD macro.
	status	integer	0 - enabled 1 - disabled	Item status.

Element	Element property	Type	Range	Description
	value_type	integer	0 - float 1 - character 2 - log 3 - unsigned integer 4 - text	Received value type.
	allowed_hosts	string		List of IP addresses (comma delimited) of hosts allowed sending data for the item if 'type' is 2 or 19.
	units	string		Units of returned values (bps, B).
	snmpv3_contextname	string		SNMPv3 context name.
	snmpv3_securityname	string		SNMPv3 security name.
	snmpv3_securitylevel	integer	0 - noAuthNoPriv 1 - authNoPriv 2 - authPriv	SNMPv3 security level.
	snmpv3_authprotocol	integer	0 - MD5 1 - SHA	SNMPv3 authentication protocol.
	snmpv3_authpassphrase	string		SNMPv3 authentication passphrase.
	snmpv3_privprotocol	integer	0 - DES 1 - AES	SNMPv3 privacy protocol.
	snmpv3_privpassphrase	string		SNMPv3 privacy passphrase.
	params	text		Name of the "Executed script" if 'type' is 13,14 "SQL query" field if 'type' is 11 "Formula" field if 'type' is 15.
	ipmi_sensor	string		IPMI sensor ID if 'type' is 12.
	authtype	integer	Authentication type for SSH agent items: 0 - password 1 - key Authentication type for HTTP agent items: 0 - none 1 - basic 2 - NTLM	Authentication type if 'type' is 13 or 19.
	username	string		User name if 'type' is 11,13,14,19.
	password	string		Password if 'type' is 11,13,14,19.
	publickey	string		Name of the public key file if 'type' is 13.
	privatekey	string		Name of the private key file if 'type' is 13.
	port	string		Custom port for the item.
	description	text		Item description.

Element	Element property	Type	Range	Description
	inventory_link	integer	0 - no link <i>number</i> - number of field in the 'host_inventory' table	Use item value to populate this inventory field.
	logtimefmt	string		Format of the time in log entries. Used only by log items.
	jmx_endpoint	string		JMX endpoint if 'type' is 16.
	url	string		URL string if 'type' is 19.
	allow_traps	integer	0 - Do not allow trapping. 1 - Allow trapping.	Property allows to send data to item if 'type' is 19.
	follow_redirects	integer	0 - Do not follow redirects. 1 - Follow redirects.	Follow HTTP redirects if 'type' is 19.
	headers	object		Object with HTTP(S) request headers if 'type' is 19.
	http_proxy	string		HTTP(S) proxy connection string if 'type' is 19.
	output_format	integer	0 - Store as is. 1 - Convert to JSON.	How to process response if 'type' is 19.
	post_type	integer	0 - Raw data. 2 - JSON data. 3 - XML data.	Type of request body if 'type' is 19.
	posts	text		Request body if 'type' is 19.
	query_fields	array		Array of objects for request query fields if 'type' is 19.
	request_method	integer	0 - GET 1 - POST 2 - PUT 3 - HEAD	Request method if 'type' is 19.
	retrieve_mode	integer	0 - Body. 1 - Headers. 2 - Both body and headers will be stored.	What part of response should be stored if 'type' is 19.
	ssl_cert_file	string		Public SSL Key file path if 'type' is 19.
	ssl_key_file	string		Private SSL Key file path if 'type' is 19.
	ssl_key_password	string		Password for SSL Key file if 'type' is 19.
	status_codes	string		Ranges of required HTTP status codes separated by commas if 'type' is 19.
	timeout	string		Item data polling request timeout if 'type' is 19.

Element	Element property	Type	Range	Description
	verify_host	integer	0 - Do not validate. 1 - Validate.	Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate if 'type' is 19.
	verify_peer	integer	0 - Do not validate. 1 - Validate.	Validate is host certificate authentic if 'type' is 19.
value map	name	string		Value map.
applications				Name of the value map to use for the item.
application	name			Root element for applications.
preprocessing step				Individual application. Application name. Item value preprocessing. Individual item value preprocessing step.

Element	Element property	Type	Range	Description
	type	integer	1 - custom multiplier 2 - right trim 3 - left trim 4 - trim from both sides 5 - regular expression matching 6 - boolean to decimal 7 - octal to decimal 8 - hexadecimal 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 - 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	Type of the item value preprocessing step.

Element	Element property	Type	Range	Description
master_item	params	string		Parameters of the item value preprocessing step.
	error_handler	integer	0 - error message is set by Zabbix server 1 - discard value 2 - set custom value 3 - set custom error message	Action type used in case of preprocessing step failure.
	error_handler_params	string		Error handler parameters used with 'error_handler'.
	key	string		Individual item master item data. Dependent item master item key value.

Template low-level discovery rule tags

Element	Element property	Type	Range	Description
discovery_rules				Root element for low-level discovery rules.
discovery_rule				Individual low-level discovery rule.
	<i>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.</i>			
	lifetime	string		Time period after which items that are no longer discovered will be deleted. Seconds, time unit with suffix or user macro. Individual filter.
filter	evaltype	integer	0 - And/or logic 1 - And logic 2 - Or logic 3 - custom 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.
condition	macro	string		Individual filter condition. Low-level discovery macro name.
	value	string		Filter value: regular expression or global regular expression.
	operator formulaid	integer character		Filter condition ID. Used in the custom calculation formula.

Element	Element property	Type	Range	Description
lld_macro_paths				Root element for LLD macro paths.
lld_macro_path	macro	string		Individual LLD macro path.
	path	string		Low-level discovery macro name.
preprocessing				Selector for value which will be assigned to corresponding macro.
step				LLD rule value preprocessing.
	type	integer		Individual LLD rule value preprocessing step.
			5 - regular expression matching 12 - JSONPath 15 - does not match regular expression 16 - check for error in JSON 20 - discard unchanged with heartbeat	Type of the item value preprocessing step.
	params	string		Parameters of the LLD rule value preprocessing step.
	error_handler	integer		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 used with 'error_handler'.
item_prototypes				Root element for item_prototypes.
item_prototype				Individual item_prototype.
	For most of the element tag values, see element tag values for a regular item. Only the tags that are specific to item_prototypes, are described below.			
application_prototypes				Root element for application prototypes.
application_prototype				Individual application prototype.
	name			Application prototype name.
master_item				Application prototype name. Individual item prototype master item/item prototype data.

Element	Element property	Type	Range	Description
	key	string		Dependent item prototype master item/item prototype key value.

Template trigger tags

Element	Element property	Type	Range	Description
triggers				Root element for triggers.
trigger				Individual trigger.
	expression	string		Trigger expression.
	recovery_mode	integer	0 - expression 1 - recovery expression 2 - none	Basis for generating OK events.
	recovery_expression	string		Trigger recovery expression.
	name	string		Trigger name.
	correlation_mode	integer	0 - no event correlation 1 - event correlation by tag	Correlation mode.
	correlation_tag	string		The tag name to be used for event correlation.
	url	string		Trigger URL.
	status	integer	0 - enabled 1 - disabled	Trigger status.
	priority	integer	0 - not classified 1 - information 2 - warning 3 - average 4 - high 5 - disaster	Trigger severity.
	description	text		Trigger description.
	type	integer	0 - single problem event 1 - multiple problem events	Event generation type.
	manual_close	integer	0 - not allowed 1 - allowed	Manual closing of problem events.
dependencies				Root element for dependencies.
dependency				Individual dependency.
	name	string		Dependency trigger name.
	expression	string		Dependency trigger expression.
	recovery_expression	string		Dependency trigger recovery expression.
tags				Root element for event tags.
tag				Individual event tag.
	tag	string		Tag name.
	value	string		Tag value.

Template graph tags

Element	Element property	Type	Range	Description
graphs				Root element for graphs.
graph				Individual graph.
	name	string		Graph name.
	width	integer		Graph width, in pixels. Used for preview and for pie/exploded graphs.
	height	integer		Graph height, in pixels. Used for preview and for pie/exploded graphs.
	yaxismin	double		Value of Y axis minimum if 'ymin_type_1' is 1.
	yaxismax	double		Value of Y axis maximum if 'ymax_type_1' is 1.
	show_work_period	integer	0 - no 1 - yes	Highlight non-working hours if 'type' is 0,1.
	show_triggers	integer	0 - no 1 - yes	Display simple trigger values as a line if 'type' is 0,1.
	type	integer	0 - normal 1 - stacked 2 - pie 3 - exploded 4 - 3D pie 5 - 3D exploded	Graph type.
	show_legend	integer	0 - no 1 - yes	Display graph legend.
	show_3d	integer	0 - 2D 1 - 3D	Enable 3D style if 'type' is 2,3.
	percent_left	double		Show the percentile line for left axis if 'type' is 0.
	percent_right	double		Show the percentile line for right axis if 'type' is 0.
	ymin_type_1	integer	0 - calculated 1 - fixed 2 - last value of the selected item	Minimum value of Y axis if 'type' is 0,1.
	ymax_type_1	integer	0 - calculated 1 - fixed 2 - last value of the selected item	Maximum value of Y axis if 'type' is 0,1.
	ymin_item_1	string	null or item details	Item details if 'ymin_type_1' is 2.
	ymax_item_1	string	null or item details	Item details if 'ymax_type_1' is 2.
graph_items				Root element for graph items.
graph_item				Individual graph item.
	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	Type	Range	Description
	drawtype	integer	0 - single line 1 - filled region 2 - bold line 3 - dotted line 4 - dashed line	Draw style if graph 'type' is 0.
	color	string		Element colour (6 symbols, hex).
	yaxiside	integer	0 - left axis 1 - right axis	Y axis position (left or right) the element belongs to if graph 'type' is 0,1.
	calc_fnc	integer	1 - minimum 2 - average 4 - maximum 7 - all (minimum, average and maximum, if graph 'type' is 0) 9 - last (if graph 'type' is not 0,1)	Data to draw if more than one value exists for an item.
	type	integer	1 - value of the item is represented proportionally on the pie 2 - value of the item represents the whole pie (graph sum)	Draw type for pie/exploded graphs.
item				Individual item.
	host key	string string		Item host. Item key.

Template web scenario tags

Element	Element property	Type	Range	Description
httptests				Root element for web scenarios.
httpstest				Individual web scenario.
	name	string		Web scenario name.
	delay	string		Frequency of executing the web scenario. Seconds, time unit with suffix or user macro.
	attempts	integer	1-10	The number of attempts for executing web scenario steps.

Element	Element property	Type	Range	Description
	agent	string		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]@]pro
	variables	text		List of scenario-level variables (macros) that may be used in scenario steps.
	headers	text		HTTP headers that will be sent when performing a request.
	status	integer	0 - enabled 1 - disabled	Web scenario status.
	authentication	integer	0 - none 1 - basic 2 - NTLM	Authentication method.
	http_user	string		Authentication user name.
	http_password	string		Authentication password for specified user name.
	verify_peer	integer	0 - no 1 - yes	Verify the SSL certificate of the web server.
	verify_host	integer	0 - no 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.
	ssl_key_file	string		Name of the SSL private key file used for client authentication.
	ssl_key_password	string		SSL private key file password.
	steps			Root element for web scenario steps.
	step			Individual web scenario step.
	name	string		Web scenario step name.
	url	string		URL for monitoring.
	posts	text		List of 'Post' variables.
	variables	text		List 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
	headers	text		HTTP headers that will be sent when performing a request.
	follow_redirects	integer	0 - no 1 - yes	Follow HTTP redirects.

Element	Element property	Type	Range	Description
	retrieve_mode	integer	0 - content 1 - headers only	HTTP response retrieve mode.
	timeout	string		Timeout of step execution. Seconds, time unit with suffix or user macro.
	required	string		Required string. Ignored if empty.
	status_codes	string		A comma delimited list of accepted status codes. Ignored if empty. For example: 200-201,210-299

3 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 applications
- 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

Hosts

☐	Name ▼	Applications	Items	Triggers	Graphs	Discovery	V
☒	Zabbix server	Applications 12	Items 79	Triggers 46	Graphs 12	Discovery 3	V
☒	Zabbix host	Applications 10	Items 43	Triggers 21	Graphs 10	Discovery 2	V

2 selected

Enable

Disable

Export

Mass update

Delete

Selected hosts are exported to a local XML file with default name `zbx_export_hosts.xml`.

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*

* Import file zbx_export_hosts.xml

Rules	Update existing	Create new	Delete missing
Groups		☒	
Hosts	☒	☒	
Templates	☐	☐	
Template screens	☐	☐	☐
Template linkage		☒	
Applications		☒	☐
Items	☒	☒	☐
Discovery rules	☒	☒	☐
Triggers	☒	☒	☐
Graphs	☒	☒	☐
Web scenarios	☒	☒	☐
Screens	☐	☐	
Maps	☐	☐	
Images	☐	☐	
Value mappings	☐	☒	

When a host is imported and updated, it can only be linked to additional templates and never be unlinked from any.

A success or failure message of the import will be displayed in the frontend.

Import rules:

Rule	Description
<i>Update existing</i>	Existing elements will be updated with data taken from the import file. Otherwise they will not be updated.
<i>Create new</i>	The import will add new elements using data from the import file. Otherwise it will not add them.
<i>Delete missing</i>	The import will remove existing elements not present in the import file. Otherwise it will not remove them.

Export format

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
  <version>4.0</version>
  <date>2018-04-03T08:11:42Z</date>
  <groups>
    <group>
      <name>Zabbix servers</name>
    </group>
  </groups>
```

```

<hosts>
  <host>
    <host>Export host</host>
    <name>Export host</name>
    <description/>
    <proxy/>
    <status>0</status>
    <ipmi_authtype>-1</ipmi_authtype>
    <ipmi_privilege>2</ipmi_privilege>
    <ipmi_username/>
    <ipmi_password/>
    <tls_connect>1</tls_connect>
    <tls_accept>1</tls_accept>
    <tls_issuer/>
    <tls_subject/>
    <tls_psk_identity/>
    <tls_psk/>
    <templates/>
    <groups>
      <group>
        <name>Zabbix servers</name>
      </group>
    </groups>
    <interfaces>
      <interface>
        <default>1</default>
        <type>1</type>
        <useip>1</useip>
        <ip>127.0.0.1</ip>
        <dns/>
        <port>10050</port>
        <bulk>1</bulk>
        <interface_ref>if1</interface_ref>
      </interface>
    </interfaces>
    <applications>
      <application>
        <name>Application</name>
      </application>
    </applications>
    <items>
      <item>
        <name>Item</name>
        <type>0</type>
        <snmp_community/>
        <snmp_oid/>
        <key>item.key</key>
        <delay>30s</delay>
        <history>90d</history>
        <trends>365d</trends>
        <status>0</status>
        <value_type>3</value_type>
        <allowed_hosts/>
        <units/>
        <snmpv3_contextname/>
        <snmpv3_securityname/>
        <snmpv3_securitylevel>0</snmpv3_securitylevel>
        <snmpv3_authprotocol>0</snmpv3_authprotocol>
        <snmpv3_authpassphrase/>
        <snmpv3_privprotocol>0</snmpv3_privprotocol>
        <snmpv3_privpassphrase/>
        <params/>

```

```

        <ipmi_sensor/>
        <authtype>0</authtype>
        <username/>
        <password/>
        <publickey/>
        <privatekey/>
        <port/>
        <description/>
        <inventory_link>0</inventory_link>
        <applications>
            <application>
                <name>Application</name>
            </application>
        </applications>
        <valuemap>
            <name>Host status</name>
        </valuemap>
        <logtimefmt/>
        <preprocessing/>
        <jmx_endpoint/>
        <timeout>3s</timeout>
        <url/>
        <query_fields/>
        <posts/>
        <status_codes>200</status_codes>
        <follow_redirects>1</follow_redirects>
        <post_type>0</post_type>
        <http_proxy/>
        <headers/>
        <retrieve_mode>0</retrieve_mode>
        <request_method>1</request_method>
        <output_format>0</output_format>
        <allow_traps>0</allow_traps>
        <ssl_cert_file/>
        <ssl_key_file/>
        <ssl_key_password/>
        <verify_peer>0</verify_peer>
        <verify_host>0</verify_host>
        <master_item/>
        <interface_ref>if1</interface_ref>
    </item>
</items>
<discovery_rules/>
<httptests/>
<macros/>
<inventory/>
</host>
</hosts>
<triggers>
    <trigger>
        <expression>{Export host:item.key.last()}<>0</expression>
        <recovery_mode>0</recovery_mode>
        <recovery_expression/>
        <name>Item value not 0</name>
        <correlation_mode>0</correlation_mode>
        <correlation_tag/>
        <url/>
        <status>0</status>
        <priority>2</priority>
        <description/>
        <type>0</type>
        <manual_close>0</manual_close>
    </trigger>

```

```

        <dependencies/>
        <tags/>
    </trigger>
</triggers>
<graphs>
    <graph>
        <name>Example graph</name>
        <width>900</width>
        <height>200</height>
        <yaxismin>0.0000</yaxismin>
        <yaxismax>100.0000</yaxismax>
        <show_work_period>1</show_work_period>
        <show_triggers>1</show_triggers>
        <type>0</type>
        <show_legend>1</show_legend>
        <show_3d>0</show_3d>
        <percent_left>0.0000</percent_left>
        <percent_right>0.0000</percent_right>
        <ymin_type_1>0</ymin_type_1>
        <ymin_type_1>0</ymin_type_1>
        <ymin_item_1>0</ymin_item_1>
        <ymin_item_1>0</ymin_item_1>
        <graph_items>
            <graph_item>
                <sortorder>0</sortorder>
                <drawtype>0</drawtype>
                <color>1A7C11</color>
                <yaxisside>0</yaxisside>
                <calc_fnc>2</calc_fnc>
                <type>0</type>
                <item>
                    <host>Export host</host>
                    <key>item.key</key>
                </item>
            </graph_item>
        </graph_items>
    </graph>
</graphs>
<value_maps>
    <value_map>
        <name>Host status</name>
        <mappings>
            <mapping>
                <value>0</value>
                <newvalue>Up</newvalue>
            </mapping>
            <mapping>
                <value>2</value>
                <newvalue>Unreachable</newvalue>
            </mapping>
        </mappings>
    </value_map>
</value_maps>
</zabbix_export>

```

Element tags

Element tag values are explained in the table below.

Host tags

Element	Element property	Type	Range	Description
groups				Root element for groups.

Element	Element property	Type	Range	Description
group				Individual group.
	name	string		Unique group name.
hosts				Root element for hosts.
host				Individual host.
	host	string		Unique host name.
	name	string		Visible host name.
	description	text		Host description.
	status	integer	0 - monitored 1 - not monitored	Host status.
	ipmi_authtype	integer	-1 - default 0 - none 1 - MD2 2 - MD5 4 - straight 5 - OEM	IPMI session authentication type.
	ipmi_privilege	integer	1 - callback 2 - user 3 - operator 4 - admin 5 - OEM	IPMI session privilege level.
	ipmi_username	string		Username for IPMI checks.
	ipmi_password	string		Password for IPMI checks.
	tls_connect	integer	1 - unencrypted 2 - TLS with PSK 4 - TLS with certificate	Type of outgoing connection.
	tls_accept	integer	1 - unencrypted 2 - TLS with PSK 3 - unencrypted and TLS with PSK 4 - TLS with certificate 5 - unencrypted and TLS with certificate 6 - TLS with PSK or certificate 7 - unencrypted and TLS with PSK or certificate	Type of incoming connection.
	tls_issuer	string		Allowed agent/proxy certificate issuer.
	tls_subject	string		Allowed agent/proxy certificate subject.
	tls_psk_identity	string		PSK identity string.
	tls_psk	string		PSK value string.
proxy				Proxy.
	name	string		Name of the proxy (if any) that monitors the host.

Element	Element property	Type	Range	Description
templates				Root element for linked templates.
template				Individual template.
interfaces	name	string		Template name.
interface				Root element for host interfaces.
	default	integer	0 - secondary 1 - primary (default)	Individual interface. Interface status. Only one primary interface of one type can be on a host.
	type	integer	0 - unknown 1 - Zabbix agent 2 - SNMP 3 - IPMI 4 - JMX	Interface type.
	useip	integer	0 - use DNS name 1 - use IP address	Interface to use for connecting to the host.
	ip	string		IP address, can be either IPv4 or IPv6.
	dns	string		DNS name.
	port	string		Port number.
	bulk	integer	0 - disable 1 - enable	Use bulk requests for SNMP.
	interface_ref	string		Interface reference name to be used in items.
applications				Root element for applications.
application				Individual application.
macros	name			Application name.
macro				Root element for macros.
	name			Individual macro.
	value			User macro name. User macro value.

Host item tags

Element	Element property	Type	Range	Description
items				Root element for items.
item				Individual item.
	name	string		Item name.

Element	Element property	Type	Range	Description
	type	integer	0 - Zabbix agent 1 - SNMPv1 agent 2 - Zabbix trapper 3 - simple check 4 - SNMPv2 agent 5 - internal 6 - SNMPv3 agent 7 - Zabbix agent (active) 8 - aggregate 9 - HTTP test (web monitoring scenario step) 10 - external 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 item	Item type.
	snmp_community	string		SNMP community name if 'type' is 1,4.
	snmp_oid	string		SNMP object ID.
	key	string		Item key.
	delay	string		Update interval of the item. Seconds, time unit with suffix, custom intervals or user macros.
	history	string		A time unit of how long the history data should be stored. Time unit with suffix or user macro.
	trends	string		A time unit of how long the trends data should be stored. Time unit with suffix or user macro.
	status	integer	0 - enabled 1 - disabled	Item status.

Element	Element property	Type	Range	Description
	value_type	integer	0 - float 1 - character 2 - log 3 - unsigned integer 4 - text	Received value type.
	allowed_hosts	string		List of IP addresses (comma delimited) of hosts allowed sending data for the item if 'type' is 2 or 19.
	units	string		Units of returned values (bps, B).
	snmpv3_contextname	string		SNMPv3 context name.
	snmpv3_securityname	string		SNMPv3 security name.
	snmpv3_securitylevel	integer	0 - noAuthNoPriv 1 - authNoPriv 2 - authPriv	SNMPv3 security level.
	snmpv3_authprotocol	integer	0 - MD5 1 - SHA	SNMPv3 authentication protocol.
	snmpv3_authpassphrase	string		SNMPv3 authentication passphrase.
	snmpv3_privprotocol	integer	0 - DES 1 - AES	SNMPv3 privacy protocol.
	snmpv3_privpassphrase	string		SNMPv3 privacy passphrase.
	params	text		Name of the "Executed script" if 'type' is 13,14 "SQL query" field if 'type' is 11 "Formula" field if 'type' is 15.
	ipmi_sensor	string		IPMI sensor ID if 'type' is 12.
	authtype	integer	Authentication type for SSH agent items: 0 - password 1 - key Authentication type for HTTP agent items: 0 - none 1 - basic 2 - NTLM	Authentication type if 'type' is 13 or 19.
	username	string		User name if 'type' is 11,13,14,19.
	password	string		Password if 'type' is 11,13,14,19.
	publickey	string		Name of the public key file if 'type' is 13.
	privatekey	string		Name of the private key file if 'type' is 13.
	port	string		Custom port for the item.
	description	text		Item description.

Element	Element property	Type	Range	Description
	inventory_link	integer	0 - no link <i>number</i> - number of field in the 'host_inventory' table	Use item value to populate this inventory field.
	logtimefmt	string		Format of the time in log entries. Used only by log items.
	interface_ref	string		Reference to host interface.
	jmx_endpoint	string		JMX endpoint if 'type' is 16.
	url	string		URL string if 'type' is 19.
	allow_traps	integer	0 - Do not allow trapping. 1 - Allow trapping.	Property to allow to send data to item if 'type' is 19.
	follow_redirects	integer	0 - Do not follow redirects. 1 - Follow redirects.	Follow HTTP redirects if 'type' is 19.
	headers	object		Object with HTTP(S) request headers if 'type' is 19.
	http_proxy	string		HTTP(S) proxy connection string if 'type' is 19.
	output_format	integer	0 - Store as is. 1 - Convert to JSON.	How to process response if 'type' is 19.
	post_type	integer	0 - Raw data. 2 - JSON data. 3 - XML data.	Type of request body if 'type' is 19.
	posts	text		Request body if 'type' is 19.
	query_fields	array		Array of objects for request query fields if 'type' is 19.
	request_method	integer	0 - GET 1 - POST 2 - PUT 3 - HEAD	Request method if 'type' is 19.
	retrieve_mode	integer	0 - Body. 1 - Headers. 2 - Both body and headers will be stored.	What part of response should be stored if 'type' is 19.
	ssl_cert_file	string		Public SSL Key file path if 'type' is 19.
	ssl_key_file	string		Private SSL Key file path if 'type' is 19.
	ssl_key_password	string		Password for SSL Key file if 'type' is 19.
	status_codes	string		Ranges of required HTTP status codes separated by commas if 'type' is 19.
	timeout	string		Item data polling request timeout if 'type' is 19.

Element	Element property	Type	Range	Description
	verify_host	integer	0 - Do not validate. 1 - Validate.	Validate host name in URL is in Common Name field or a Subject Alternate Name field of host certificate if 'type' is 19.
	verify_peer	integer	0 - Do not validate. 1 - Validate.	Validate is host certificate authentic if 'type' is 19.
value map	name	string		Value map.
applications				Name of the value map to use for the item.
application	name			Root element for applications.
preprocessing step				Individual application. Application name. Item value preprocessing. Individual item value preprocessing step.

Element	Element property	Type	Range	Description
	type	integer	1 - custom multiplier 2 - right trim 3 - left trim 4 - trim from both sides 5 - regular expression matching 6 - boolean to decimal 7 - octal to decimal 8 - hexadecimal 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 - 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	Type of the item value preprocessing step.

Element	Element property	Type	Range	Description
master_item	params	string		Parameters of the item value preprocessing step.
	error_handler	integer	0 - error message is set by Zabbix server 1 - discard value 2 - set custom value 3 - set custom error message	Action type used in case of preprocessing step failure.
	error_handler_params	string		Error handler parameters used with 'error_handler'.
	key	string		Individual item master item data. Dependent item master item key value.

Host low-level discovery rule tags

Element	Element property	Type	Range	Description
discovery_rules				Root element for low-level discovery rules.
discovery_rule				Individual low-level discovery rule.
	<i>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.</i>			
	lifetime	string		Time period after which items that are no longer discovered will be deleted. Seconds, time unit with suffix or user macro. Individual filter.
filter	evaltype	integer	0 - And/or logic 1 - And logic 2 - Or logic 3 - custom 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.
condition	macro	string		Individual filter condition. Low-level discovery macro name.
	value	string		Filter value: regular expression or global regular expression.
	operator formulaid	integer character		Filter condition ID. Used in the custom calculation formula.

Element	Element property	Type	Range	Description
lld_macro_paths				Root element for LLD macro paths.
lld_macro_path	macro	string		Individual LLD macro path.
	path	string		Low-level discovery macro name.
preprocessing				Selector for value which will be assigned to corresponding macro.
step				LLD rule value preprocessing.
	type	integer	5 - regular expression matching 12 - JSONPath 15 - does not match regular expression 16 - check for error in JSON 20 - discard unchanged with heartbeat	Individual LLD rule value preprocessing step.
	params	string		Type of the item value preprocessing step.
	error_handler	integer	0 - error message is set by Zabbix server 1 - discard value 2 - set custom value 3 - set custom error message	Parameters of the LLD rule value preprocessing step.
	error_handler_params	string		Action type used in case of preprocessing step failure.
item_prototypes				Error handler parameters used with 'error_handler'.
item_prototype				Root element for item_prototypes.
	For most of the element tag values, see element tag values for a regular item. Only the tags that are specific to item_prototypes, are described below.			Individual item_prototype.
application_prototypes				Root element for application prototypes.
application_prototype				Individual application prototype.
	name			Application prototype name.
master_item				Application prototype name.
				Individual item prototype master item/item prototype data.

Element	Element property	Type	Range	Description
	key	string		Dependent item prototype master item/item prototype key value.

Host trigger tags

Element	Element property	Type	Range	Description
triggers				Root element for triggers.
trigger				Individual trigger.
	expression	string		Trigger expression.
	recovery_mode	integer	0 - expression 1 - recovery expression 2 - none	Basis for generating OK events.
	recovery_expression	string		Trigger recovery expression.
	name	string		Trigger name.
	correlation_mode	integer	0 - no event correlation 1 - event correlation by tag	Correlation mode.
	correlation_tag	string		The tag name to be used for event correlation.
	url	string		Trigger URL.
	status	integer	0 - enabled 1 - disabled	Trigger status.
	priority	integer	0 - not classified 1 - information 2 - warning 3 - average 4 - high 5 - disaster	Trigger severity.
	description	text		Trigger description.
	type	integer	0 - single problem event 1 - multiple problem events	Event generation type.
	manual_close	integer	0 - not allowed 1 - allowed	Manual closing of problem events.
dependencies				Root element for dependencies.
dependency				Individual dependency.
	name	string		Dependency trigger name.
	expression	string		Dependency trigger expression.
	recovery_expression	string		Dependency trigger recovery expression.
tags				Root element for event tags.
tag				Individual event tag.
	tag	string		Tag name.
	value	string		Tag value.

Host graph tags

Element	Element property	Type	Range	Description
graphs				Root element for graphs.
graph				Individual graph.
	name	string		Graph name.
	width	integer		Graph width, in pixels. Used for preview and for pie/exploded graphs.
	height	integer		Graph height, in pixels. Used for preview and for pie/exploded graphs.
	yaxismin	double		Value of Y axis minimum if 'ymin_type_1' is 1.
	yaxismax	double		Value of Y axis maximum if 'ymax_type_1' is 1.
	show_work_period	integer	0 - no 1 - yes	Highlight non-working hours if 'type' is 0,1.
	show_triggers	integer	0 - no 1 - yes	Display simple trigger values as a line if 'type' is 0,1.
	type	integer	0 - normal 1 - stacked 2 - pie 3 - exploded 4 - 3D pie 5 - 3D exploded	Graph type.
	show_legend	integer	0 - no 1 - yes	Display graph legend.
	show_3d	integer	0 - 2D 1 - 3D	Enable 3D style if 'type' is 2,3.
	percent_left	double		Show the percentile line for left axis if 'type' is 0.
	percent_right	double		Show the percentile line for right axis if 'type' is 0.
	ymin_type_1	integer	0 - calculated 1 - fixed 2 - last value of the selected item	Minimum value of Y axis if 'type' is 0,1.
	ymax_type_1	integer	0 - calculated 1 - fixed 2 - last value of the selected item	Maximum value of Y axis if 'type' is 0,1.
	ymin_item_1	string	null or item details	Item details if 'ymin_type_1' is 2.
	ymax_item_1	string	null or item details	Item details if 'ymax_type_1' is 2.
graph_items				Root element for graph items.
graph_item				Individual graph item.
	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	Type	Range	Description
	drawtype	integer	0 - single line 1 - filled region 2 - bold line 3 - dotted line 4 - dashed line	Draw style if graph 'type' is 0.
	color	string		Element colour (6 symbols, hex).
	yaxiside	integer	0 - left axis 1 - right axis	Y axis position (left or right) the element belongs to if graph 'type' is 0,1.
	calc_fnc	integer	1 - minimum 2 - average 4 - maximum 7 - all (minimum, average and maximum, if graph 'type' is 0) 9 - last (if graph 'type' is not 0,1)	Data to draw if more than one value exists for an item.
	type	integer	1 - value of the item is represented proportionally on the pie 2 - value of the item represents the whole pie (graph sum)	Draw type for pie/exploded graphs.
item				Individual item.
	host key	string string		Item host. Item key.

Host web scenario tags

Element	Element property	Type	Range	Description
httptests				Root element for web scenarios.
httpstest				Individual web scenario.
	name	string		Web scenario name.
	delay	string		Frequency of executing the web scenario. Seconds, time unit with suffix or user macro.
	attempts	integer	1-10	The number of attempts for executing web scenario steps.

Element	Element property	Type	Range	Description
	agent	string		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]@]pro
	variables	text		List of scenario-level variables (macros) that may be used in scenario steps.
	headers	text		HTTP headers that will be sent when performing a request.
	status	integer	0 - enabled 1 - disabled	Web scenario status.
	authentication	integer	0 - none 1 - basic 2 - NTLM	Authentication method.
	http_user	string		Authentication user name.
	http_password	string		Authentication password for specified user name.
	verify_peer	integer	0 - no 1 - yes	Verify the SSL certificate of the web server.
	verify_host	integer	0 - no 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.
	ssl_key_file	string		Name of the SSL private key file used for client authentication.
	ssl_key_password	string		SSL private key file password.
	steps			Root element for web scenario steps.
	step			Individual web scenario step.
	name	string		Web scenario step name.
	url	string		URL for monitoring.
	posts	text		List of 'Post' variables.
	variables	text		List 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
	headers	text		HTTP headers that will be sent when performing a request.
	follow_redirects	integer	0 - no 1 - yes	Follow HTTP redirects.

Element	Element property	Type	Range	Description
	retrieve_mode	integer	0 - content 1 - headers only	HTTP response retrieve mode.
	timeout	string		Timeout of step execution. Seconds, time unit with suffix or user macro.
	required	string		Required string. Ignored if empty.
	status_codes	string		A comma delimited list of accepted status codes. Ignored if empty. For example: 200-201,210-299

4 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

Not exported are host groups, hosts, triggers, other maps or any other elements that may be related to the exported map. 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

☐ Name ▲	Width	Height
☒ Network	590	400
☐ Offices	700	550
☐ User map	800	600

1 selected **Export** **Delete**

Selected maps are exported to a local XML file with default name *zabbix_export_maps.xml*.

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*

* Import file

Browse...zbx_export_maps.xml

Rules	Update existing	Create new	Delete missing
Groups		☐	
Hosts	☐	☐	
Templates	☐	☐	
Template screens	☐	☐	☐
Template linkage		☐	
Applications		☐	☐
Items	☐	☐	☐
Discovery rules	☐	☐	☐
Triggers	☐	☐	☐
Graphs	☐	☐	☐
Web scenarios			
Screens	☐	☐	
Maps	☒	☒	
Images	☐	☐	
Value mappings	☐	☐	

Import

Cancel

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
<i>Update existing</i>	Existing maps will be updated with data taken from the import file. Otherwise they will not be updated.
<i>Create new</i>	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 Zabbix Super Admin users.

Warning:

If replacing an existing image, it will affect all maps that are using this image.

Export format

Exporting a small network map with three elements, their images and some links between them. Note that images are truncated to save space.

```

<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
  <version>4.0</version>
  <date>2016-10-05T08:16:20Z</date>
  <images>
    <image>
      <name>Server_(64)</name>
      <imagetype>1</imagetype>
      <encodedImage>iVBOR...SuQmCC</encodedImage>
    </image>
    <image>
      <name>Workstation_(64)</name>
      <imagetype>1</imagetype>
      <encodedImage>iVBOR...SuQmCC</encodedImage>
    </image>
    <image>
      <name>Zabbix_server_3D_(96)</name>
      <imagetype>1</imagetype>
      <encodedImage>iVBOR...ggg==</encodedImage>
    </image>
  </images>
  <maps>
    <map>
      <name>Network</name>
      <width>590</width>
      <height>400</height>
      <label_type>0</label_type>
      <label_location>0</label_location>
      <highlight>1</highlight>
      <expandproblem>0</expandproblem>
      <markelements>1</markelements>
      <show_unack>0</show_unack>
      <severity_min>2</severity_min>
      <show_suppressed>0</show_suppressed>
      <grid_size>40</grid_size>
      <grid_show>1</grid_show>
      <grid_align>1</grid_align>
      <label_format>0</label_format>
      <label_type_host>2</label_type_host>
      <label_type_hostgroup>2</label_type_hostgroup>
      <label_type_trigger>2</label_type_trigger>
      <label_type_map>2</label_type_map>
      <label_type_image>2</label_type_image>
      <label_string_host/>
      <label_string_hostgroup/>
      <label_string_trigger/>
      <label_string_map/>
      <label_string_image/>
      <expand_macros>0</expand_macros>
      <background/>
      <iconmap/>
      <urls/>
      <selements>
        <selement>
          <elementtype>0</elementtype>
          <label>Host 1</label>
          <label_location>-1</label_location>
          <x>476</x>
          <y>28</y>
          <elementsubtype>0</elementsubtype>
          <areatype>0</areatype>
          <width>200</width>

```

```

    <height>200</height>
    <viewtype>0</viewtype>
    <use_iconmap>0</use_iconmap>
    <selementid>8</selementid>
    <elements>
        <element>
            <host>Discovered host</host>
        </element>
    </elements>
    <icon_off>
        <name>Server_(64)</name>
    </icon_off>
    <icon_on/>
    <icon_disabled/>
    <icon_maintenance/>
    <application/>
    <urls/>
</selement>
<selement>
    <elementtype>0</elementtype>
    <label>Zabbix server</label>
    <label_location>-1</label_location>
    <x>252</x>
    <y>50</y>
    <elementsubtype>0</elementsubtype>
    <areatype>0</areatype>
    <width>200</width>
    <height>200</height>
    <viewtype>0</viewtype>
    <use_iconmap>0</use_iconmap>
    <selementid>6</selementid>
    <elements>
        <element>
            <host>Zabbix server</host>
        </element>
    </elements>
    <icon_off>
        <name>Zabbix_server_3D_(96)</name>
    </icon_off>
    <icon_on/>
    <icon_disabled/>
    <icon_maintenance/>
    <application/>
    <urls/>
</selement>
<selement>
    <elementtype>0</elementtype>
    <label>New host</label>
    <label_location>-1</label_location>
    <x>308</x>
    <y>230</y>
    <elementsubtype>0</elementsubtype>
    <areatype>0</areatype>
    <width>200</width>
    <height>200</height>
    <viewtype>0</viewtype>
    <use_iconmap>0</use_iconmap>
    <selementid>7</selementid>
    <elements>
        <element>
            <host>Zabbix host</host>
        </element>
    </elements>

```

```

        </elements>
        <icon_off>
            <name>Workstation_(64)</name>
        </icon_off>
        <icon_on/>
        <icon_disabled/>
        <icon_maintenance/>
        <application/>
        <urls/>
    </selement>
</selements>
<links>
    <link>
        <drawtype>0</drawtype>
        <color>008800</color>
        <label/>
        <selementid1>6</selementid1>
        <selementid2>8</selementid2>
        <linktriggers/>
    </link>
    <link>
        <drawtype>2</drawtype>
        <color>00CC00</color>
        <label>100MBps</label>
        <selementid1>7</selementid1>
        <selementid2>6</selementid2>
        <linktriggers>
            <linktrigger>
                <drawtype>0</drawtype>
                <color>DD0000</color>
                <trigger>
                    <description>Zabbix agent on {HOST.NAME} is unreachable for 5 minutes</des
                    <expression>{Zabbix host:agent.ping.nodata(5m)}=1</expression>
                    <recovery_expression/>
                </trigger>
            </linktrigger>
        </linktriggers>
    </link>
</links>
</map>
</maps>
</zabbix_export>

```

Element tags

Element tag values are explained in the table below.

Element	Element property	Type	Range	Description
images				Root element for images.
image	name	string		Individual image. Unique image name.
	imagetype	integer	1 - image 2 - background	Image type.
	encodedImage			Base64 encoded image.
maps				Root element for maps.
map	name	string		Individual map. Unique map name.
	width	integer		Map width, in pixels.
	height	integer		Map height, in pixels.

Element	Element property	Type	Range	Description
	label_type	integer	0 - label 1 - host IP address 2 - element name 3 - status only 4 - nothing	Map element label type.
	label_location	integer	0 - bottom 1 - left 2 - right 3 - top	Map element label location by default.
	highlight	integer	0 - no 1 - yes	Enable icon highlighting for active triggers and host statuses.
	expandproblem	integer	0 - no 1 - yes	Display problem trigger for elements with a single problem.
	markelements	integer	0 - no 1 - yes	Highlight map elements that have recently changed their status.
	show_unack	integer	0 - count of all problems 1 - count of unacknowledged problems 2 - count of acknowledged and unacknowledged problems separately	Problem display.
	severity_min	integer	0 - not classified 1 - information 2 - warning 3 - average 4 - high 5 - disaster	Minimum trigger severity to show on the map by default.
	show_suppressed	integer	0 - no 1 - yes	Display problems which would otherwise be suppressed (not shown) because of host maintenance.
	grid_size	integer	20, 40, 50, 75 or 100	Cell size of a map grid in pixels, if "grid_show=1"
	grid_show	integer	0 - yes 1 - no	Display a grid in map configuration.
	grid_align	integer	0 - yes 1 - no	Automatically align icons in map configuration.
	label_format	integer	0 - no 1 - yes	Use advanced label configuration.

Element	Element property	Type	Range	Description
	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).

Element	Element property	Type	Range	Description
urls				
url	name	string		Individual URL.
	url	string		Link name.
	elementtype	integer	0 - host 1 - map 2 - trigger 3 - host group 4 - image	Link URL. Map item type the link belongs to.
selements				
selement	elementtype	integer	0 - host 1 - map 2 - trigger 3 - host group 4 - image	Individual map element. Map element type.
	label	string		Icon label.
	label_location	integer	-1 - use map default 0 - bottom 1 - left 2 - right 3 - top	
	x	integer		Location on the X axis.
	y	integer		Location on the Y axis.
	elementsubtype	integer	0 - single host group 1 - all host groups	Element subtype, if "elementtype=3"
	areatype	integer	0 - same as whole map 1 - custom size	Area size, if "elementsubtype=1"
	width	integer		Width of area, if "areatype=1"
	height	integer		Height of area, if "areatype=1"
	viewtype	integer	0 - place evenly in the area	Area placement algorithm, if "elementsubtype=1"
	use_iconmap	integer	0 - no 1 - yes	Use icon mapping for this element. Relevant only if iconmapping is activated on map level.
	selementid	id		Unique element record ID.
	application	string		Application name filter. If an application name is given, only problems of triggers that belong to the given application will be displayed on the map.
elements				
element	host			Individual Zabbix entity that is represented on the map (map, hostgroup, host, etc).
icon_off				Image to use when element is in 'OK' status.
icon_on				Image to use when element is in 'Problem' status.

Element	Element property	Type	Range	Description
icon_disabled				Image to use when element is disabled.
icon_maintenance				Image to use when element is in maintenance.
links	name	string		Unique image name.
link				Individual link 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				Individual link status indicator.
linktrigger				Link style when trigger is in the 'Problem' state.
	drawtype	integer	0 - line 2 - bold line 3 - dotted line 4 - dashed line	
	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.

5 Screens

Overview

Screen **export** contains the screen structure - all screen settings and all screen elements along with their configuration.

Anything included in the screen itself (like a host, host group or any other data) is not exported. Thus, if at least one of the elements the screen refers to is missing, importing it will fail.

Exporting

To export screens, do the following:

- Go to: *Monitoring* → *Screens*
- Mark the checkboxes of the screens to export
- Click on *Export* below the list

☐ Name ▲	Dimension (cols x rows)
☐ Servers	2 x 3
☒ Zabbix server	2 x 3
☐ Zabbix server2	3 x 3
1 selected	
Export Delete	

Selected screens are exported to a local XML file with default name *zabbix_export_screens.xml*.

Importing

To import screens, do the following:

- Go to: *Monitoring* → *Screens*
- Click on *Import* to the right
- Select the import file
- Mark the required options in import rules
- Click on *Import*

★ Import file

Browse...

zbx_export_screens.xml

Rules	Update existing	Create new	Delete missing
Groups		☐	
Hosts	☐	☐	
Templates	☐	☐	
Template screens	☐	☐	☐
Template linkage		☐	
Applications		☐	☐
Items	☐	☐	☐
Discovery rules	☐	☐	☐
Triggers	☐	☐	☐
Graphs	☐	☐	☐
Web scenarios			
Screens	☒	☒	
Maps	☐	☐	
Images	☐	☐	
Value mappings	☐	☐	

Import

Cancel

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
<i>Update existing</i>	Existing screens will be updated with data taken from the import file. Otherwise they will not be updated.
<i>Create new</i>	The import will add new screens using data from the import file. Otherwise it will not add them.

Export format

Exporting a small screen with two graphs taking up the first row of the screen.

```
<?xml version="1.0" encoding="UTF-8"?>
<zabbix_export>
  <version>4.0</version>
  <date>2016-10-07T08:02:40Z</date>
  <screens>
    <screen>
      <name>Zabbix server</name>
      <hsize>2</hsize>
```

```

<vsize>3</vsize>
<screen_items>
  <screen_item>
    <resourcetype>0</resourcetype>
    <width>300</width>
    <height>80</height>
    <x>0</x>
    <y>0</y>
    <colspan>1</colspan>
    <rowspan>1</rowspan>
    <elements>0</elements>
    <valign>0</valign>
    <halign>0</halign>
    <style>0</style>
    <url/>
    <dynamic>1</dynamic>
    <sort_triggers>0</sort_triggers>
    <resource>
      <name>CPU load</name>
      <host>Zabbix host</host>
    </resource>
    <max_columns>3</max_columns>
    <application/>
  </screen_item>
  <screen_item>
    <resourcetype>0</resourcetype>
    <width>300</width>
    <height>80</height>
    <x>1</x>
    <y>0</y>
    <colspan>1</colspan>
    <rowspan>1</rowspan>
    <elements>0</elements>
    <valign>0</valign>
    <halign>0</halign>
    <style>0</style>
    <url/>
    <dynamic>1</dynamic>
    <sort_triggers>0</sort_triggers>
    <resource>
      <name>CPU utilization</name>
      <host>Zabbix host</host>
    </resource>
    <max_columns>3</max_columns>
    <application/>
  </screen_item>
</screen_items>
</screen>
</screens>
</zabbix_export>

```

Element tags

Element tag values are explained in the table below.

Element	Element property	Type	Range	Description
screens				
screen	name	string		Unique screen name.
	hsize	integer		Horizontal size, number of columns.
	vsize	integer		Vertical size, number of rows.

Element	Element property	Type	Range	Description
screen_items screen_item	resourcetype	integer	0 - graph 1 - simple graph 2 - map 3 - plain text 4 - host info 5 - trigger info 6 - server info 7 - clock 8 - screen 9 - trigger overview 10 - data overview 11 - URL 12 - history of actions 13 - history of events 14 - host group issues 15 - problems by severity 16 - host issues 19 - simple graph prototype 20 - graph prototype	Resource type.
	width	integer		Width of the screen item (in pixels) if 'resourcetype' is 0, 1, 7, 11, 19 or 20.
	height	integer		Height of the screen item (in pixels) if 'resourcetype' is 0, 1, 7, 11, 19 or 20.
	x	integer		X-coordinates of the screen item on the screen, from left to right. '0' means start from first column.
	y	integer		Y-coordinates of the screen item on the screen, from top to bottom. '0' means start from first row.
	colspan	integer		Number of columns the screen item will span across.
	rowspan	integer		Number of rows the screen item will span across.
	elements	integer		Number of lines to display on the screen item if 'resourcetype' is 3, 12, 13, 14 or 16.

Element	Element property	Type	Range	Description
resource	application	string		Filter by application name if 'resourcetype' is 9 or 10.
	name	string		Resource name.
	host	string		Resource host.

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
<i>Service Discovered</i>	The service is 'up' after it was 'down' or when discovered for the first time.
<i>Service Up</i>	The service is 'up', consecutively.
<i>Service Lost</i>	The service is 'down' after it was 'up'.
<i>Service Down</i>	The service is 'down', consecutively.
<i>Host Discovered</i>	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.
<i>Host Up</i>	At least one service of a host is 'up', consecutively.
<i>Host Lost</i>	All services of a host are 'down' after at least one was 'up'.
<i>Host Down</i>	All services of a host are 'down', consecutively.

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.

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.
- 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

* Name Local network

Discovery by proxy No proxy

* IP range 192.168.1.1-254

* Update interval 1h

* Checks

SNMPv2 agent "iso.3.6.1.2.1.1.1.0"

[Edit](#) [Remove](#)

[New](#)

Check type SNMPv2 agent

* Port range 161

* SNMP community public

* SNMP OID iso.3.6.1.2.1.1.1.0

[Update](#) [Cancel](#)

Device uniqueness criteria

☐ IP address

☒ SNMPv2 agent "iso.3.6.1.2.1.1.1.0"

Host name

☐ DNS name

☐ IP address

☒ SNMPv2 agent "iso.3.6.1.2.1.1.1.0"

Visible name

☒ Host name

☐ DNS name

☐ IP address

☐ SNMPv2 agent "iso.3.6.1.2.1.1.1.0"

Enabled ☒

Add

Cancel

Parameter	Description
<i>Name</i>	Unique name of the rule. For example, "Local network".
<i>Discovery by proxy</i>	What performs discovery: no proxy - Zabbix server is doing discovery <proxy name> - this proxy performs discovery
<i>IP range</i>	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.
<i>Update interval</i>	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).
<i>Checks</i>	Zabbix will use this list of checks for discovery. 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
<i>Device uniqueness criteria</i>	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.
<i>Host name</i>	<discovery check> - either Zabbix agent or SNMP agent check. 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 .
<i>Visible name</i>	This option is supported since 4.2.0. 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 .
<i>Enabled</i>	This option is supported since 4.2.0. 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 *Template OS Linux* for Linux hosts
- use *Template OS Windows* for Windows hosts

Step 1

Defining a network discovery rule for our IP range.

* Name	Local network		
Discovery by proxy	No proxy		
* IP range	192.168.1.1-254		
* Update interval	10m		
* Checks	Zabbix agent "system.uname" Edit Remove New		
Device uniqueness criteria	☒ IP address ☐ Zabbix agent "system.uname"		
Host name	☐ DNS name ☐ IP address ☒ Zabbix agent "system.uname"		
Visible name	☒ Host name ☐ DNS name ☐ IP address ☐ Zabbix agent "system.uname"		
Enabled	☒		

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 Template OS Windows, Linux servers to Template OS Linux.

The rule will be executed every 10 minutes.

When this rule is added, Zabbix will automatically start the discovery and generating discovery-based events for further processing.

Step 2

Defining an **action** for adding the discovered Linux servers to the respective group/template.

Action

Operations

* Name

Add discovered Linux servers

Type of calculation

And/Or

A and B and C and D

Conditions

Label	Name
A	Received value like <i>Linux</i>
B	Discovery status = <i>Up</i>
C	Service type = <i>Zabbix agent</i>
D	Uptime/Downtime $\geq$ 3600

New condition

Uptime/Downtime

$\geq$

600

Add

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

Action	Operations
Default subject	Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IP/
Default 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}
Operations	Details Add to host groups: Linux servers Link to templates: Template OS Linux New

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 "Template OS Linux" template. Zabbix will automatically start monitoring the host using items and triggers from "Template OS Linux".

Step 3

Defining an action for adding the discovered Windows servers to the respective group/template.

Action	Operations										
* Name	Add discovered Windows servers										
Type of calculation	And/Or ▼ A and B and C and D										
Conditions	<table border="1"> <thead> <tr> <th>Label</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>A</td> <td>Received value like <i>Windows</i></td> </tr> <tr> <td>B</td> <td>Discovery status = <i>Up</i></td> </tr> <tr> <td>C</td> <td>Service type = <i>Zabbix agent</i></td> </tr> <tr> <td>D</td> <td>Uptime/Downtime >= 3600</td> </tr> </tbody> </table>	Label	Name	A	Received value like <i>Windows</i>	B	Discovery status = <i>Up</i>	C	Service type = <i>Zabbix agent</i>	D	Uptime/Downtime >= 3600
Label	Name										
A	Received value like <i>Windows</i>										
B	Discovery status = <i>Up</i>										
C	Service type = <i>Zabbix agent</i>										
D	Uptime/Downtime >= 3600										
New condition	<table border="1"> <tr> <td>Uptime/Downtime ▼</td> <td>>= ▼</td> <td>600</td> </tr> </table> Add	Uptime/Downtime ▼	>= ▼	600							
Uptime/Downtime ▼	>= ▼	600									

Action	Operations
Default subject	Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPA
Default 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}
Operations	Details Add to host groups: Windows servers Link to templates: Template OS Windows New

Step 4

Defining an action for removing lost servers.

Action	Operations								
* Name	Remove lost servers								
Type of calculation	And/Or v A and B and C								
Conditions	<table border="1"> <thead> <tr> <th>Label</th> <th>Name</th> </tr> </thead> <tbody> <tr> <td>A</td> <td>Uptime/Downtime >= 86400</td> </tr> <tr> <td>B</td> <td>Discovery status = Down</td> </tr> <tr> <td>C</td> <td>Service type = Zabbix agent</td> </tr> </tbody> </table>	Label	Name	A	Uptime/Downtime >= 86400	B	Discovery status = Down	C	Service type = Zabbix agent
Label	Name								
A	Uptime/Downtime >= 86400								
B	Discovery status = Down								
C	Service type = Zabbix agent								
New condition	<table border="1"> <tr> <td>Service type v </td> <td>= v </td> <td>FTP v </td> </tr> </table> Add	Service type v	= v	FTP v					
Service type v	= v	FTP v							

Action	Operations						
Default subject	Discovery: {DISCOVERY.DEVICE.STATUS} {DISCOVERY.DEVICE.IPA						
Default 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}						
Operations	<table> <tr> <th>Details</th><th>Action</th></tr> <tr> <td>Remove host</td><td>Edit Remove</td></tr> <tr> <td>New</td><td></td></tr> </table>	Details	Action	Remove host	Edit Remove	New	
Details	Action						
Remove host	Edit Remove						
New							

A server will be removed if "Zabbix agent" service is 'down' for more than 24 hours (86400 seconds).

2 Active agent auto-registration

Overview

It is possible to allow active Zabbix agent auto-registration, after which the server can start monitoring them. This way new hosts can be added for monitoring without configuring them manually on the server.

Auto registration 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 auto-registration 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 auto-registered 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.

Auto-registration 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 auto-registration 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.

Restart the agent after making any changes to the configuration file.

Action for active agent auto-registration

When server receives an auto-registration request from an agent it calls an **action**. An action of event source "Auto registration" must be configured for agent auto-registration.

Note:

Setting up **network discovery** is not required to have active agents auto-register.

In the Zabbix frontend, go to *Configuration* → *Actions*, select *Auto registration* 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 auto-registering 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.

Using host metadata

When agent is sending an auto-registration 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.

<note:important>An auto-registration 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 auto-registered 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 "Template OS Linux" and "Template OS Windows" 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 auto-registration, 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: Template OS 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: Template OS 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-OPXGGSTYNHO 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: Auto registration action Linux
- Conditions:
 - * Type of calculation: AND
 - * Condition (A): Host metadata contains //Linux//
 - * Condition (B): Host metadata contains //21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
- * Operations:
 - * Send message to users: Admin via all media
 - * Add to host groups: Linux servers
 - * Link to templates: Template OS 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: Auto registration action Linux
- Conditions:
 - * Type of calculation: AND
 - * Condition (A): Host metadata contains Linux
 - * Condition (B): Host metadata contains 21df83bf21bf0be663090bb8d4128558ab9b95fba66a6dbf834f8b91ae5e08ae
- * Operations:
 - * Send message to users: Admin via all media
 - * Add to host groups: Linux servers
 - * Link to templates: Template OS Linux
 - * Link to templates: Template DB MySQL

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*
- Click on *Discovery* in the row of an appropriate template

Templates							
☐	Name ▲	Applications	Items	Triggers	Graphs	Screens	Discovery
☐	Template OS Linux	Applications 10	Items 32	Triggers 15	Graphs 5	Screens 1	Discovery 2

- 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 four 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

The **Discovery rule** tab contains the item key to use for discovery (as well as some general discovery rule attributes):

The screenshot shows the 'Discovery rule' configuration form. It has four tabs: 'Discovery rule' (selected), 'Preprocessing', 'LLD macros', and 'Filters'. The form contains the following fields and sections:

- Name:** Mounted filesystem discovery (marked with a red asterisk).
- Type:** Zabbix agent (dropdown menu).
- Key:** vfs.fs.discovery (marked with a red asterisk).
- Host interface:** 192.168.6.87 : 10050 (dropdown menu, marked with a red asterisk).
- Update interval:** 1h (marked with a red asterisk).
- Custom intervals:** A table with columns 'Type', 'Interval', and 'Period'. It shows 'Flexible' and 'Scheduling' types with an interval of '50s' and a period of '1-7,00:00-2'. There is an 'Add' link below the table.
- Keep lost resources period:** 30d (marked with a red asterisk).
- Description:** Discovery of file systems of different types as defined in global regular expression "File systems for discovery".
- Enabled:** A checkbox that is checked.
- Buttons:** 'Add' and 'Cancel' buttons at the bottom.

All mandatory input fields are marked with a red asterisk.

Parameter	Description
<i>Name</i>	Name of discovery rule.
<i>Type</i>	The type of check to perform discovery; should be <i>Zabbix agent</i> or <i>Zabbix agent (active)</i> for file system discovery. 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 (<i>Dependent item</i>) and specify the master item in the 'Master item' field. The master item must exist.

Parameter	Description
<i>Key</i>	An item with "vfs.fs.discovery" key is built into Zabbix agent on many platforms (see supported item key list for details), and will return a JSON with the list of file systems present on the computer and their types.
<i>Update interval</i>	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. <i>Note:</i> If set to '0', the item will not be polled. However, if a flexible interval also exists with a non-zero value, the item will be polled during the flexible interval duration. <i>Note</i> that for an existing discovery rule the discovery can be performed immediately by pushing the <i>Check now</i> button.
<i>Custom intervals</i>	You can create custom rules for checking the item: Flexible - create an exception to the <i>Update interval</i> (interval with different frequency) Scheduling - create a custom polling schedule. For detailed information see Custom intervals . Scheduling is supported since Zabbix 3.0.0.
<i>Keep lost resources period</i>	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. <i>Note:</i> 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.
<i>Description</i>	Enter a description.
<i>Enabled</i>	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.

Discovery rule
Preprocessing
LLD macros
Filters

Preprocessing steps	Name	Parameters	Custom on fail
1:	Regular expression	pattern output	☐
2:	JSONPath	\$.pools	☐

Add
Add
Cancel

Type	Transformation	Description
Text		

Type	Transformation	Description
	<i>Regular expression</i>	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 <i>Custom on fail</i> 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.
Structured data	<i>JSONPath</i>	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"}] 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']. If you mark the <i>Custom on fail</i> 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	<i>JavaScript</i>	Enter JavaScript code in the block that appears when clicking in the parameter field or on <i>Open</i>. Note that available JavaScript length depends on the database used.
Validation	<i>Does not match regular expression</i>	Specify a regular expression that a value must not match. E.g. Error: (.*?)\. If you mark the <i>Custom on fail</i> 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.
	<i>Check for error in JSON</i>	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.
Throttling		

Type	Transformation	Description
	<i>Discard unchanged with heartbeat</i>	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. <i>Note:</i> Changing item prototypes does not reset throttling. Throttling is reset only when preprocessing steps are changed.
Prometheus	<i>Prometheus to JSON</i>	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;
- You may be using an item that returns a JSON 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.

Discovery rule
Preprocessing
LLD macros
Filters

LLD macros

LLD macro	JSONPath
<code>{#NAME}</code>	<code>\$.name</code>
<code>{#FSNAME}</code>	<code>\$.fs.name</code>
<code>{#FSMOUNT}</code>	<code>\$.fs.mountpoint</code>
<code>{#MACRO}</code>	<code>\$.path.to.node</code>

Add

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
<i>LLD macro</i>	Name of the low-level discovery macro, using the following syntax: <code>{#MACRO}</code> .

Parameter	Description
<i>JSONPath</i>	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 defined JSON path 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

The **Filters** tab contains discovery rule filter definitions allowing to filter discovery values:

Discovery rule

Preprocessing

LLD macros

Filters

Type of calculation

And/Or

A and B

Filters

	Label	Macro		Regular expression
A		{#FSTYPE}	matches	@File systems for discovery
B		{#MACRO}	does not match	regular expression

Add

Add

Cancel

Parameter	Description
<i>Type of calculation</i>	The following options for calculating filters are available: And - all filters must be passed; Or - enough if one filter is passed; And/Or - uses <i>And</i> with different macro names and <i>Or</i> 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.
<i>Filters</i>	A filter can be used to generate real items, triggers, and graphs only for certain file systems. It expects 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: <pre>for f in ext2 nfs reiserfs smbfs; do echo \$f \ grep -E '^ext'</pre> macro on Windows is supported since Zabbix 3.0.0. Defining several filters is supported since Zabbix 2.4.0. Note that if some macro from the filter is missing in the response, the found entity will be ignored. Filter drop-down offers two values to specify whether a macro matches a regular expression or does not match.

Warning:

A mistake or typo in the regular expression used in LLD rule may cause deleting thousands of configuration elements, historical values and events for many hosts. For example, an incorrect "File systems for discovery" regular expression may cause deleting thousands of items, triggers, historical values and events.

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.

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 . <i>Note that when performing discovery immediately, configuration cache is not updated, thus the result will not reflect very recent changes to discovery rule configuration.</i>
Delete	Delete the discovery rule.
Cancel	Cancel the editing of discovery rule properties.

Item prototypes

Once a rule is created, go to the items for that rule and press "Create prototype" to create an item prototype. Note how macro {#FSNAME} is used where a file system name is required. When the discovery rule is processed, this macro will be substituted with the discovered file system.

* Name Free disk space on {#FSNAME} (percentage)

Type Zabbix agent

* Key vfs.fs.size[{#FSNAME},pfree]

Type of information Numeric (float)

Units %

* Update interval 1m

Custom intervals

Type	Interval	Period
☒ Flexible ☐ Scheduling	50s	1-7,00:00
Add		

* History storage period 1w

* Trend storage period 365d

Show value As is [show value mappings](#)

New application

Applications

-None-
CPU
Filesystems
General
Memory
Network interfaces
OS
Performance
Processes
Security

New application prototype Application_{#FSNAME}

Application prototypes

-None-

Description

Create enabled ☒

Add

Cancel

Low-level discovery **macros** and user **macros** may be used 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
<i>New application prototype</i>	You may define a new application prototype. In application prototypes you can use low-level discovery macros that, after discovery, will be substituted with real values to create applications that are specific for the discovered entity. See also application discovery notes for more specific information.
<i>Application prototypes</i>	Select from the existing application prototypes.
<i>Create enabled</i>	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.

We can create several item prototypes for each file system metric we are interested in:

Item prototypes

All templates / Template OS Linux

Discovery list / Mounted filesystem discovery

Item prototypes

☐ NAME	KEY	INTERVAL
☐ Free disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},free]	1m
☐ Free disk space on {#FSNAME} (percentage)	vfs.fs.size[{#FSNAME},pfree]	1m
☐ Free inodes on {#FSNAME} (percentage)	vfs.fs.inode[{#FSNAME},pfree]	1m
☐ Total disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},total]	1h
☐ Used disk space on {#FSNAME}	vfs.fs.size[{#FSNAME},used]	1m

Mass update option is available if you want to update properties of several item prototypes at once.

Trigger prototypes

We create trigger prototypes in a similar way as item prototypes:

Trigger prototype
Dependencies

* Name

Free disk space is less than 20% on volume {#FSNAME}

Severity

Not classified

Information

Warning

Average

High

Low

* Expression

{Template OS Linux:vfs.fs.size[{#FSNAME},pfree].last(0)}<20

[Expression constructor](#)

OK event generation

Expression

Recovery expression

None

PROBLEM event generation mode

Single

Multiple

OK event closes

All problems

All problems if tag values match

Tags

tag

value

[Add](#)

Allow manual close

☐

URL

Description

Create enabled

☒

Attributes that are specific for trigger prototypes:

Parameter	Description
<i>Create enabled</i>	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.

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.

Trigger prototypes		
All templates / Template OS Linux Discovery list / Mounted filesystem discovery Item prototypes 5		
☐	SEVERITY	NAME ▲
☐	Warning	Free disk space is less than 20% on volume {#FSNAME}
☐	Warning	Free inodes is less than 20% on volume {#FSNAME}

Graph prototypes

We can create graph prototypes, too:

Graph prototype

Preview

* Name

Disk space usage {#FSNAME}

* Width

600

* Height

340

Graph type

Pie

Show legend

☒

3D view

☒

* Items

Name	Type
1: Template OS Linux: Total disk space on {#FSNAME}	Graph
2: Template OS Linux: Free disk space on {#FSNAME}	Simple

[Add](#) [Add prototype](#)

Graph prototypes		
All templates / Template OS Linux Discovery list / Mounted filesystem discovery Item prototypes 5		
☐	NAME ▲	WIDTH
☐	Disk space usage {#FSNAME}	600

Finally, we have created a discovery rule that looks like shown below. It has five item prototypes, two trigger prototypes, and one graph prototype.

Discovery rules

All templates / Template OS Linux Applications 10 Items 32 Triggers 15 Graphs 5 Screens 1

☐	NAME ▲	ITEMS	TRIGGERS	GRAPHS	H
☐	Mounted filesystem discovery	Item prototypes 5	Trigger prototypes 2	Graph prototypes 1	H

Note: For configuring host prototypes, see the section about **host prototype** configuration in virtual machine monitoring.

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.

Items

All hosts / Remote proxy: New host Enabled ZBX SNMP JMX IPMI Applications 11 Items 41

☐	Wizard	Name	Triggers	Key
☐	...	Mounted filesystem discovery : Free disk space on / (percentage)	Triggers 1	vfs.fs.size[/,pfr
☐	...	Mounted filesystem discovery : Used disk space on /		vfs.fs.size[/,use
☐	...	Mounted filesystem discovery : Free disk space on /		vfs.fs.size[/,fre
☐	...	Mounted filesystem discovery : Free inodes on / (percentage)	Triggers 1	vfs.fs.inode[/,p

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.

1m	7d	1y	Zabbix agent	Enabled	
The item is not discovered anymore and will be deleted in 29d 23h 44m (on 2015-08-31 at 23:27).					

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.

Triggers

Group all

All hosts / Remote proxy: New host
Enabled
ZBX
SNMP
JMX
IPMI
Applications 11
Items 41
T

☐	Severity	Name ▲
☐	Warning	Mounted filesystem discovery: Free disk space is less than 20% on volume /
☐	Warning	Mounted filesystem discovery: Free inodes is less than 20% on volume /

Graphs

Group all

All hosts / Remote proxy: New host
Enabled
ZBX
SNMP
JMX
IPMI
Applications 11
Items 41
T

☐	Name ▲
☐	Template OS Linux: CPU jumps
☐	Template OS Linux: CPU load
☐	Template OS Linux: CPU utilization
☐	Mounted filesystem discovery: Disk space usage /

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](#).

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 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, for example, 2048 bytes on a Zabbix proxy run with IBM DB2 database.

Multiple LLD rules for 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.

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](#).

Using LLD macros in user macro contexts

User macros **with context** can be used to accomplish more flexible thresholds in trigger expressions. Different thresholds may be defined on user macro level and then used in trigger constants depending on the discovered context. Discovered context appears when the **low-level discovery macros** used in the macros are resolved to real values.

To illustrate we can use data from the example above and assume that the following file systems will be discovered: /, /home, /tmp, /usr, /var.

We may define a free-disk-space trigger prototype for a host, where the threshold is expressed by a user macro with context:

```
{host:vfs.fs.size[{#FSNAME},pfree].last()}<{$LOW_SPACE_LIMIT:"{#FSNAME}"}
```

Then add user macros:

- {\$LOW_SPACE_LIMIT} **10**
- {\$LOW_SPACE_LIMIT:/home} **20**
- {\$LOW_SPACE_LIMIT:/tmp} **50**

Now, once the file systems are discovered, events will be generated if /, /usr and /var filesystems have less than **10%** of free disk space, the /home filesystem - less than **20%** of free disk space or the /tmp filesystem - less than **50%** of free disk space.

1 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]"

2 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>]"

3 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

Templates						
☐ Name ▼	Applications	Items	Triggers	Graphs	Screens	Discovery
☐ Template Net Network Generic Device SNMPv2	Applications 3	Items 11	Triggers 5	Graphs	Screens	Discovery 2

- 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

Discovery rule
Preprocessing
LLD macros
Filters

* Name

Network Interfaces

Type

SNMPv2 agent

* Key

ifDescr

* SNMP OID

discovery[{#IFDESCR},IF:MIB::ifDescr]

* SNMP community

{SNMP_COMMUNITY}

Port

* Update interval

1h

Custom intervals

Type	Interval	Period
Flexible	Scheduling	50s
		1-7,00:00-

Add

* Keep lost resources period

30d

Description

You may also consider using IF:MIB::ifType or IF:MIB::ifAlias for discovery depending on your filtering needs.

{SNMP_COMMUNITY} is a global macro.

Enabled

☒

Add

Cancel

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:

Item prototype

Preprocessing

*

Name

Incoming traffic on interface {#IFDESCR}

Type

SNMPv2 agent

*

Key

ifInOctets[{#IFDESCR}]

*

SNMP OID

IF-MIB::ifInOctets.{#IFDESCR}

*

SNMP community

{\$SNMP_COMMUNITY}

Port

Type of information

Numeric (unsigned)

Units

bps

*

Update interval

1m

Custom intervals

Type	Interval	Period
Flexible Scheduling	50s	1-7,00:00-2

Add

*

History storage period

1w

*

Trend storage period

365d

Show value

As is

show value mappings

New application

Again, creating as many item prototypes as needed:

Item prototypes

[All templates / Template SNMP Interfaces](#)

[Discovery list / Network interfaces](#)

[Item prototypes 8](#)

☐ NAME ▲	KEY	INTERVAL	HI
☐ Admin status of interface {#IFDESCR}	ifAdminStatus[{#IFDESCR}]	1m	7d
☐ Alias of interface {#IFDESCR}	ifAlias[{#IFDESCR}]	1h	7d
☐ Description of interface {#IFDESCR}	ifDescr[{#IFDESCR}]	1h	7d
☐ Inbound errors on interface {#IFDESCR}	ifInErrors[{#IFDESCR}]	1m	7d
☐ Incoming traffic on interface {#IFDESCR}	ifInOctets[{#IFDESCR}]	1m	7d
☐ Operational status of interface {#IFDESCR}	ifOperStatus[{#IFDESCR}]	1m	7d
☐ Outbound errors on interface {#IFDESCR}	ifOutErrors[{#IFDESCR}]	1m	7d
☐ Outgoing traffic on interface {#IFDESCR}	ifOutOctets[{#IFDESCR}]	1m	7d

Trigger prototypes

The following screenshot illustrates how we can use these macros in trigger prototypes:

The following screenshot illustrates how we can use these macros in graph prototypes:

Graph prototype

Preview

* Name

Traffic on interface {#IFDESCR}

* Width

900

* Height

200

Graph type

Normal

Show legend

☒

Show working time

☒

Show triggers

☒

Percentile line (left)

☐

Percentile line (right)

☐

Y axis MIN value

Calculated

Y axis MAX value

Calculated

* Items

	Name	Function	Draw st
⋮	1: Template SNMP Interfaces: Incoming traffic on interface {#IFDESCR}	avg	Gradie
⋮	2: Template SNMP Interfaces: Outgoing traffic on interface {#IFDESCR}	avg	Gradie

[Add](#) [Add prototype](#)

Graph prototypes

All templates / Template SNMP Interfaces

Discovery list / Network interfaces

Item prototypes 8

T

☐ NAME ▲	WIDTH
☐ Traffic on interface {#SNMPVALUE}	900

A summary of our discovery rule:

Discovery rules

[All templates](#) / [Template SNMP Interfaces](#) [Applications 1](#) [Items 1](#) [Triggers](#) [Graphs](#) [Screens](#)

☐ NAME ▲	ITEMS	TRIGGERS	GRAPHS	HO
☐ Network interfaces	Item prototypes 8	Trigger prototypes 1	Graph prototypes 1	Ho

Discovered entities

When server runs, it will create real items, triggers and graphs based on the values the SNMP discovery rule returns. In the host configuration they are prefixed with an orange link to a discovery rule they come from.

Items

[All hosts](#) / [Switch1](#) [Enabled](#) [ZBX](#) [SNMP](#) [JMX](#) [IPMI](#) [Applications 1](#) [Items 241](#) [Triggers 30](#) [Gr](#)

Filter ▼

☐ Wizard	Name	Triggers	Key ▲
☐	Network interfaces : Admin status of interface 1		ifAdminStatus[1]
☐	Network interfaces : Admin status of interface 2		ifAdminStatus[2]
☐	Network interfaces : Admin status of interface 3		ifAdminStatus[3]
☐	Network interfaces : Admin status of interface 4		ifAdminStatus[4]

Triggers

[All hosts](#) / [Switch1](#) [Enabled](#) [ZBX](#) [SNMP](#) [JMX](#) [IPMI](#) [Applications 1](#) [Items 241](#) [Triggers 30](#) [Gr](#)

Filter ▼

☐ Severity	Name ▲	Exp
☐ Information	Network interfaces : Operational status was changed on {HOST.NAME} interface 1	{pr
☐ Information	Network interfaces : Operational status was changed on {HOST.NAME} interface 2	{pr
☐ Information	Network interfaces : Operational status was changed on {HOST.NAME} interface 3	{pr
☐ Information	Network interfaces : Operational status was changed on {HOST.NAME} interface 4	{pr

Graphs

Group all

[All hosts](#) / [Switch1](#)
Enabled
ZBX
SNMP
JMX
IPMI
Applications 1
Items 241
Triggers 30
Gr

☐ Name ▲
☐ Network interfaces : Traffic on interface 1
☐ Network interfaces : Traffic on interface 2
☐ Network interfaces : Traffic on interface 3
☐ Network interfaces : Traffic on interface 4

4 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 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.

The item key to use is

```
jmx.discovery[<discovery mode>,<object name>]
```

where

- discovery mode - one of the following: *attributes* (retrieve JMX MBean attributes, default) or *beans* (retrieve JMX MBeans)
- object name - object name pattern identifying the MBean names to be retrieved (empty by default, retrieving all registered beans).

You may consult ObjectName [documentation](#) for the options of specifying object name pattern.

If no parameters are passed, all MBean attributes from JMX are requested.

Attention:

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.

This item key is supported since Zabbix Java gateway 3.4.

Item key examples:

```
jmx.discovery #Retrieve all JMX MBean attributes
jmx.discovery[beans] #Retrieve all JMX MBeans
jmx.discovery[attributes,"*:type=GarbageCollector,name=*"] #Retrieve all garbage collector attributes
jmx.discovery[beans,"*:type=GarbageCollector,name=*"] #Retrieve all garbage collectors
```

This item returns a JSON object. 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. (<i>Zabbix reserved name</i>)
{#JMXOBJ}	Object name. (<i>Zabbix reserved name</i>)

Macro	Description
{#JMX<key property>}	MBean properties (like {#JMXTYPE}, {#JMXNAME}). Some important notes to pay attention to when defining MBean attribute name that is created from MBean property name by the following algorithm: * attribute name case is changed to uppercase; * attribute name case is ignored (no LLD macro is generated) in case it consists of not supported characters. Supported characters can be described by the following regular expression: "A-Z0-9_\."; * if an attribute name is called "obj" or "domain" it will be replaced with the values of 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

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

LLD rule collecting 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:

Discovery rule

Preprocessing

LLD macros

Filters

* Name

JMX garbage collectors

Type

JMX agent

* Key

jmx.discovery[beans,"*:type=GarbageCollector,name=*"]

* Host interface

127.0.0.1 : 12345

Key used:

```
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:

Item prototypes	
All hosts / JMX Enabled ZBX SNMP JMX IPMI Discovery list / JMX garbage collectors Item prototypes 3 Trigger prot	
☐ Name ▲	Key
☐ GC {#JMXNAME} CollectionCount	jmx[{#JMXOBJ},CollectionCount]
☐ GC {#JMXNAME} CollectionTime	jmx[{#JMXOBJ},CollectionTime]
☐ GC {#JMXNAME} Valid	jmx[{#JMXOBJ},Valid]

Keys used:

```
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):

		Filter ▼
☐ Wizard	Name ▲	Triggers Key
☐	JMX garbage collectors: GC PS MarkSweep CollectionCount	jmx["java.lang:type=GarbageCollector,name=PS MarkSweep",CollectionCount]
☐	JMX garbage collectors: GC PS MarkSweep CollectionTime	jmx["java.lang:type=GarbageCollector,name=PS MarkSweep",CollectionTime]
☐	... JMX garbage collectors: GC PS MarkSweep Valid	jmx["java.lang:type=GarbageCollector,name=PS MarkSweep",Valid]
☐	JMX garbage collectors: GC PS Scavenge CollectionCount	jmx["java.lang:type=GarbageCollector,name=PS Scavenge",CollectionCount]
☐	JMX garbage collectors: GC PS Scavenge CollectionTime	jmx["java.lang:type=GarbageCollector,name=PS Scavenge",CollectionTime]
☐	... JMX garbage collectors: GC PS Scavenge Valid	jmx["java.lang:type=GarbageCollector,name=PS Scavenge",Valid]

LLD rule collecting Mbean attributes

This rule will return 9 objects with the following fields: MBean1Attribute1, MBean2Attribute1, Mbean3Attribute1,MBean1Attribute2,MBean2Attribute1, 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:

Discovery rule	Preprocessing	LLD macros	Filters
* Name	JMX garbage collectors		
Type	JMX agent		
* Key	<input *:type='GarbageCollector,name=*"]"/' type="text" value="jmx.discovery[attributes,\"/>		
* Host interface	127.0.0.1 : 12345		

Key used:

```
jmx.discovery[attributes,\"*:type=GarbageCollector,name=*"]
```

All the garbage collectors with a single item attribute will be discovered.

Item prototypes	
All hosts / JMX Enabled ZBX SNMP JMX IPMI Discovery list / JMX garbage collectors Item prototypes 1	
☐ Name ▲	Key
☐ {#JMXOBJ} {#JMXATTR}	jmx[{#JMXOBJ},{#JMXATTR}]

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.

5 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, the only difference being that a

```
db.odbc.discovery[<description>,dsn]
```

key should be used instead of "db.odbc.select[<description>,dsn]".

Discovery using SQL queries is supported since Zabbix server/proxy 3.0.

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:

Discovery rule
Preprocessing
LLD macros
Filters

* Name

Proxy discovery

Type

Database monitor

* Key

db.odbc.discovery[proxies,{SDSN}]

User name

Password

* SQL query

SELECT h1.host, COUNT(h2.host) AS count FROM hosts h1 LEFT JOIN hosts h2 ON h1.hostid = h2.proxy_hostid WHERE h1.status IN (5, 6) GROUP BY h1.host;

* Update interval

30s

Custom intervals

Type	Interval	Period
Flexible	Scheduling	50s
1-7,00		

Add

* Keep lost resources period

30d

Description

Enabled

☒

Add

Cancel

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_hostid
+-----+-----+
| 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 I
```

```
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:

Item prototype

Preprocessing

*

Name

Last acces time of proxy {#HOST}

Type

Zabbix internal

*

Key

zabbix[proxy,{#HOST},lastaccess]

Type of information

Numeric (unsigned)

Units

unixtime

*

Update interval

60s

Custom intervals

Type	Interval	Period
Flexible	Scheduling	50s
1-7,00:00-24:00		

Add

*

History storage period

90d

*

Trend storage period

365d

Show value

As is

show value mappings

Once discovery is performed, an item will be created for each proxy:

☐	Wizard	Name	Triggers	Key ▲
☐		Proxy discovery: Last access time of proxy Japan1		zabbix[proxy,Japan1,lastacce
☐		Proxy discovery: Last access time of proxy Japan2		zabbix[proxy,Japan2,lastacce
☐		Proxy discovery: Last access time of proxy Latvia		zabbix[proxy,Latvia,lastaccess

6 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.

Macro	Description
{#SERVICE.DISPLAYNAME}	Displayed service name.
{#SERVICE.DESCRPTION}	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 (<i>Running, Paused, Start pending, Pause pending, Continue pending, Stop pending, Stopped</i> or <i>Unknown</i>).
{#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 (<i>Automatic, Automatic delayed, Manual, Disabled, Unknown</i>).
{#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 <i>Automatic (trigger start)</i> , <i>Automatic delayed (trigger start)</i> and <i>Manual (trigger start)</i> .

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.

7 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`:

Discovery rule	Preprocessing	LLD macros	Filters						
<table border="1"> <thead> <tr> <th>Preprocessing steps</th> <th>Name</th> <th>Parameters</th> </tr> </thead> <tbody> <tr> <td>1:</td> <td>Prometheus to JSON</td> <td>wmi_logical_disk_free_bytes{volume=~".*"} Add</td> </tr> </tbody> </table>				Preprocessing steps	Name	Parameters	1:	Prometheus to JSON	wmi_logical_disk_free_bytes{volume=~".*"} Add
Preprocessing steps	Name	Parameters							
1:	Prometheus to JSON	wmi_logical_disk_free_bytes{volume=~".*"} Add							

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:

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".

Notes on low-level discovery

Application discovery

Application prototypes support LLD macros.

One application prototype can be used by several item prototypes of the same discovery rule.

If created application prototype is not used by any item prototype it gets removed from 'Application prototypes' list automatically.

Like other discovered entities applications follow the lifetime defined in discovery rule ('keep lost resources period' setting) - they are removed after not being discovered for the specified number of days.

If an application is not discovered anymore all discovered items are automatically removed from it, even if the application itself is not yet removed because of the 'lost resources period' setting.

Application prototypes defined by one discovery rule can't discover the same application. In this situation only the first prototype discovery will succeed, the rest will report appropriate LLD error. Only application prototypes defined in different discovery rules can result in discovering the same application.

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
Centralised configuration	Yes
Generates notifications	No

Note:
[1] Automatic DB creation feature only works with SQLite. Other databases require a manual setup.

1 Proxies

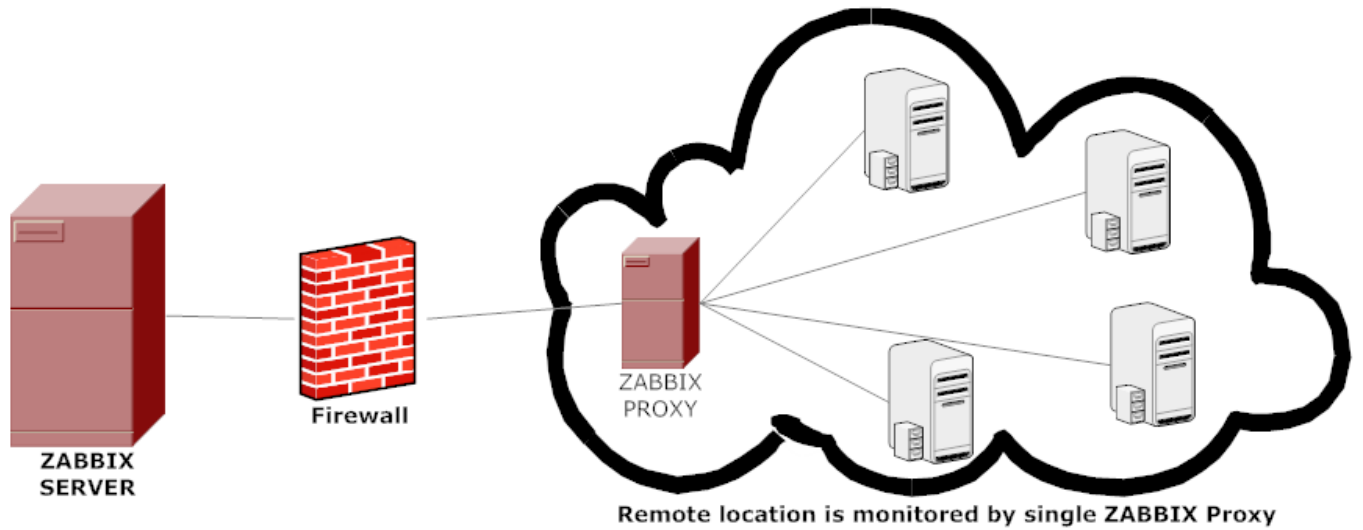
Overview

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	
<i>Zabbix agent checks</i>	Yes
<i>Zabbix agent checks (active)</i>	Yes ¹
<i>Simple checks</i>	Yes
<i>Trapper items</i>	Yes
<i>SNMP checks</i>	Yes
<i>SNMP traps</i>	Yes
<i>IPMI checks</i>	Yes
<i>JMX checks</i>	Yes
<i>Log file monitoring</i>	Yes
<i>Internal checks</i>	Yes
<i>SSH checks</i>	Yes
<i>Telnet checks</i>	Yes
<i>External checks</i>	Yes
<i>Dependent items</i>	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	<i>No</i>
Processing events	<i>No</i>
Event correlation	<i>No</i>
Sending alerts	<i>No</i>

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.

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*

Proxy
Encryption

* Proxy name

Remote proxy

Proxy mode

Active Passive

Proxy address

127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com

Description

Add

Cancel

Parameter	Description
<i>Proxy name</i>	Enter the proxy name. It must be the same name as in the <i>Hostname</i> parameter in the proxy configuration file.
<i>Proxy mode</i>	Select the proxy mode. Active - the proxy will connect to the Zabbix server and request configuration data Passive - Zabbix server connects to the proxy <i>Note</i> 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 <i>Proxy address</i> field.
<i>Proxy address</i>	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 <i>Proxy mode</i> field. Macros are not supported.
<i>Interface</i>	This option is supported since Zabbix 4.0.0. Enter interface details for the passive proxy. This field is only available if a passive proxy is selected in the <i>Proxy mode</i> field.
	<i>IP address</i> IP address of the passive proxy (optional).
	<i>DNS name</i> DNS name of the passive proxy (optional).
	<i>Connect to</i> 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
<i>Description</i>	<i>Port</i> TCP/UDP port number of the passive proxy (10051 by default). Enter the proxy description.

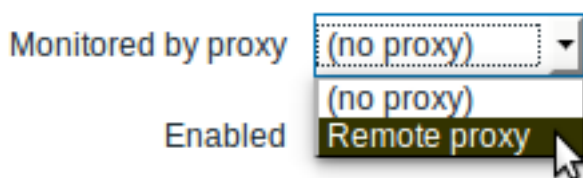
The **Encryption** tab allows you to require encrypted connections with the proxy.

Parameter	Description
<i>Connections to proxy</i>	How the server connects to the passive proxy: no encryption (default), using PSK (pre-shared key) or certificate.
<i>Connections from proxy</i>	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
<i>Issuer</i>	Allowed issuer of certificate. Certificate is first validated with CA (certificate authority). If it is valid, signed by the CA, then the <i>Issuer</i> 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.
<i>Subject</i>	Allowed subject of certificate. Certificate is first validated with CA. If it is valid, signed by the CA, then the <i>Subject</i> field can be used to allow only one value of <i>Subject</i> string. If this field is empty then any valid certificate signed by the configured CA is accepted.
<i>PSK identity</i>	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.
<i>PSK</i>	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

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 server, Zabbix proxy, Zabbix agent, `zabbix_sender` and `zabbix_get` utilities using Transport Layer Security (TLS) protocol v.1.2. Encryption is supported starting with Zabbix 3.0. Certificate-based and pre-shared key-based encryption is supported.

Encryption is optional and configurable for individual components (e.g. 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 web server running Zabbix frontend and user web browser,
 - * between Zabbix frontend and Zabbix server,
 - * between Zabbix server (proxy) and Zabbix database.
- * Currently each encrypted connection opens with a full TLS handshake, no session caching and tickets are
- * Adding encryption increases time of checks and actions, depending on network latency.\\ For example, if
- * Encryption is not supported by `[/manual/discovery/network_discovery|network discovery]`. Zabbix agent c

Compiling Zabbix with encryption support To support encryption Zabbix must be compiled and linked with one of four crypto libraries:

- *GnuTLS* (from version 3.1.18)
- *OpenSSL* (versions 1.0.1, 1.0.2, 1.1.0). *OpenSSL* 1.1.1 is supported from Zabbix versions 3.0.23, 3.4.15, 4.0.1.
- *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.
- *MBED TLS* (formerly *PolarSSL*)(version 1.3.9 and later 1.3.x). *MBED TLS* 2.x is not currently supported, it is not a drop-in replacement for 1.3 branch, Zabbix will not compile with *MBED TLS* 2.x.

The library is selected by specifying an option to "configure" script:

- `--with-gnutls [=DIR]`
- `--with-openssl [=DIR]` (also used for *LibreSSL*)
- `--with-mbedtls [=DIR]`

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 --with-libxml2 --with-openssl
```

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 *MBED TLS* libraries in Zabbix components using PSKs. *GnuTLS* and *MBED TLS* libraries support PSK ciphersuites with [Perfect Forward Secrecy](#). *OpenSSL* library (versions 1.0.1, 1.0.2c) does 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 for connections between Zabbix components:

- `TLSCConnect`
- `TLSAccept`

`TLSCConnect` specifies what encryption to use for outgoing connections and can take one of 3 values (unencrypted, PSK, certificate). `TLSCConnect` is used in configuration files for Zabbix proxy (in active mode, specifies only connections to server) and Zabbix agentd (for active checks). In Zabbix frontend the `TLSCConnect` equivalent is *Connections to host* field in *Configuration→Hosts→<some host>→Encryption* tab and *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` specifies what types of connections are allowed for incoming connections. Connection types are: unencrypted, PSK, certificate. One or more values can be specified. `TLSAccept` is used in configuration files for Zabbix proxy (in passive mode, specifies only connections from server) and Zabbix agentd (for passive checks). In Zabbix frontend the `TLSAccept` equivalent is *Connections from host* field in *Configuration→Hosts→<some host>→Encryption* tab and *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 encryption type, e.g. from unencrypted to certificate-based with minimum downtime and rollback possibility.

To achieve this you can set `TLSAccept=unencrypted,cert` in agentd configuration file and restart Zabbix agent.

Then you can test connection with `zabbix_get` to the agent using certificate. If it works, you can reconfigure encryption for that agent in Zabbix frontend in *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 monitoring by proxy) then connections to that agent will be encrypted.

If everything works as expected you can set `TLSAccept=cert` in 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 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.

For overview, encryption configuration for each host is displayed in Zabbix frontend *Configuration→Hosts* on the right side, in column *AGENT ENCRYPTION*. Configuration display examples:

Example	Connections TO host	Allowed connections FROM host	Rejected connections FROM host
NONE	Unencrypted	Unencrypted	Encrypted certificate and PSK-based
	Encrypted, certificate-based	Encrypted certificate-based	Unencrypted and PSK-based
	Encrypted, PSK-based	Encrypted PSK-based	Unencrypted and certificate-based
	Encrypted, PSK-based	Unencrypted and PSK-based encrypted	Certificate-based
	Encrypted, certificate-based	Unencrypted, PSK or certificate-based encrypted	-

Attention:

Default is unencrypted connections. Encryption must be configured for each host and proxy individually.

zabbix_get and zabbix_sender with encryption See man-pages [zabbix_get](#) and [zabbix_sender](#) for using them with encryption.

Ciphersuites Ciphersuites are configured internally during Zabbix startup and depend on crypto library, currently they are not user-configurable.

Configured ciphersuites by library type in order from higher to lower priority:

Library	Certificate ciphersuites	PSK ciphersuites
<i>mbed TLS (PolarSSL) 1.3.9</i>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256 TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA256 TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA TLS-RSA-WITH-AES-128-GCM-SHA256 TLS-RSA-WITH-AES-128-CBC-SHA256 TLS-RSA-WITH-AES-128-CBC-SHA	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256 TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA TLS-PSK-WITH-AES-128-GCM-SHA256 TLS-PSK-WITH-AES-128-CBC-SHA256 TLS-PSK-WITH-AES-128-CBC-SHA
<i>GnuTLS 3.1.18</i>	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
<i>OpenSSL 1.0.2c</i>	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-SHA256 AES128-SHA	PSK-AES128-CBC-SHA
<i>OpenSSL 1.1.0</i>	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

Cipher suites using certificates:

	TLS server		
TLS client	<i>mbed TLS (PolarSSL)</i>	<i>GnuTLS</i>	<i>OpenSSL 1.0.2</i>

<i>MBED TLS (PolarSSL)</i>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256
<i>GnuTLS</i>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256
<i>OpenSSL 1.0.2</i>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256

Cipher suites using PSK:

TLS client	TLS server		
<i>MBED TLS (PolarSSL)</i>	<i>MBED TLS (PolarSSL)</i>	<i>GnuTLS</i>	<i>OpenSSL 1.0.2</i>
<i>MBED TLS (PolarSSL)</i>	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-PSK-WITH-AES-128-CBC-SHA
<i>GnuTLS</i>	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-PSK-WITH-AES-128-CBC-SHA
<i>OpenSSL 1.0.2</i>	TLS-PSK-WITH-AES-128-CBC-SHA	TLS-PSK-WITH-AES-128-CBC-SHA	TLS-PSK-WITH-AES-128-CBC-SHA

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
<i>TLSCAFile</i>	*	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.
<i>TLSCRLFile</i>		Full pathname of a file containing Certificate Revocation Lists. See notes in Certificate Revocation Lists (CRL) .
<i>TLSCertFile</i>	*	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).
<i>TLSKeyFile</i>	*	Full pathname of a file containing private key. Set access rights to this file - it must be readable only by Zabbix user.
<i>TLSServerCertIssuer</i>		Allowed server certificate issuer.
<i>TLSServerCertSubject</i>		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+MRMwEQYKCZImiZPyLQG

....

9wEzdN8uTrqoyU78gi12npLj08LegRKjb5hFTVm0

-----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/ MRMwEQYKCZImiZPyLQG

...

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-----
MIIECDCCAvCgAwIBAgIBATANBgkqhkiG9w0BAQUFADCgTETMBEGCgmSJomT8ixk
...
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+MRMwEQYKCZImiZPyLQG
...
dyCeWnvL7u5sd6ffo8iRny0QzbHKmQt/wUtcVIvWXdMIFJMOHw==
-----END CERTIFICATE-----

```

Here the first is Zabbix server certificate, followed by intermediate CA certificate.

3. Put Zabbix server private key into file, for example, /home/zabbix/zabbix_server.key:

```

-----BEGIN PRIVATE KEY-----
MIIEWAIBADANBgkqhkiG9w0BAQEFAASCBAKowggSMAgEAAoIBAQc9tIXIJoVnNXDl
...
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

The screenshot shows the 'Encryption' tab for a proxy configuration. Under 'Connections to proxy', the 'Certificate' button is selected. Under 'Connections from proxy', the 'Certificate' checkbox is checked. The 'Issuer' field contains 'CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com' and the 'Subject' field contains 'CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com'. At the bottom are buttons for 'Update', 'Clone', 'Delete', and 'Cancel'.

For passive proxy

The screenshot shows the 'Encryption' tab for a proxy configuration. Under 'Connections to proxy', the 'Certificate' button is selected. Under 'Connections from proxy', the 'No encryption' checkbox is checked. The 'Issuer' field contains 'CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com' and the 'Subject' field contains 'CN=Zabbix proxy,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com'. At the bottom are buttons for 'Update', 'Clone', 'Delete', and 'Cancel'.

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.

The screenshot shows the 'Encryption' tab in the Zabbix frontend. At the top, there are tabs for 'Host', 'Templates', 'IPMI', 'Macros', 'Host inventory', and 'Encryption'. The 'Encryption' tab is selected. Below the tabs, there are three sections: 'Connections to host', 'Connections from host', and 'Issuer/Subject'. The 'Connections to host' section has three buttons: 'No encryption', 'PSK', and 'Certificate'. The 'Connections from host' section has three checkboxes: 'No encryption', 'PSK', and 'Certificate'. The 'Issuer' field is filled with 'CN=Signing CA,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com'. The 'Subject' field is filled with 'CN=www01,OU=Development group,O=Zabbix SIA,DC=zabbix,DC=com'. At the bottom, there are five buttons: 'Update', 'Clone', 'Full clone', 'Delete', and 'Cancel'.

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:

- escape characters '"' (U+0022), '+' U+002B, ',' U+002C, ';' U+003B, '<' U+003C, '>' U+003E, '\\ ' U+005C
 - escape characters space (' ' U+0020) or number sign ('#' U+0023) at the beginning of string.
 - escape character space (' ' U+0020) at the end of string.
- Match fails if a null character (U+0000) is encountered ([[http://tools.ietf.org/html/rfc4514|RFC 4514]])
- Requirements of ([[http://tools.ietf.org/html/rfc4517| RFC 4517 Lightweight Directory Access Protocol (LD

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
...
treZeUPjb7LSmZ3K2hpbZN7So0ZcAoHQ3GWd9npuctg=
-----END X509 CRL-----
-----BEGIN X509 CRL-----
MIIB+TCB4gIBATANBgkqhkiG9w0BAQUFADB/MRMwEQYKCZImiZPyLQGGRYDY29t
...
CAEebS2CND3ShBedZ8YSi15906JvaDP611R51Ns=
-----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`.

Limitations on using CRL extensions

- **Authority Key Identifier** extension.
CRLs for CAs with identical names may not work in case of *mbedTLS* (*PolarSSL*), even with "Authority Key Identifier" extension.

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
<i>Zabbix</i>	128 UTF-8 characters	128-bit (16-byte PSK, entered as 32 hexadecimal digits)	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
<i>GnuTLS</i>	128 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
<i>mbed TLS</i> (<i>PolarSSL</i>)	128 UTF-8 characters	-	256-bit (default limit) (32-byte PSK, entered as 64 hexadecimal digits)
<i>OpenSSL 1.0.x, 1.1.0</i>	127 bytes (may include UTF-8 characters)	-	2048-bit (256-byte PSK, entered as 512 hexadecimal digits)
<i>OpenSSL 1.1.1</i>	127 bytes (may include UTF-8 characters)	-	512-bit (64-byte PSK, entered as 128 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 a user 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:

It is a user 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 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:

Host Templates IPMI Macros Host inventory **Encryption**

Connections to host **No encryption** **PSK** Certificate

Connections from host ☐ No encryption ☒ PSK ☐ Certificate

* PSK identity PSK 001

* PSK 1f87b595725ac58dd977beef14b97461a7c1045b9a1c963065002c547319495

Update Clone Full clone Delete Cancel

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 "e560cb0d918d26d31b4f642181f5f570ad89a390931102e5391d08327ba434e9" 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, mbed TLS (PolarSSL)) 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 *mbed TLS (PolarSSL)* 1.3.11)

Get value from agent failed: `ssl_handshake(): SSL - The connection indicated an EOF`

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 *mbed TLS (PolarSSL)*):

failed to accept an incoming connection: from 127.0.0.1: `ssl_handshake():\`
`SSL - The server has no ciphersuites in common with the client`

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 peer
```

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 successfully'
...send value error: TCP successful, cannot establish TLS to [[192.168.1.2]:10051]: SSL_connect() I/O error: [0x00000000] The operation completed successfully
```

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_ERROR_SSL
```

In accepting-side log:

```
...failed to accept an incoming connection: from 192.168.1.2: TLS handshake set result code to 1: file ssl.c:1040
```

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 *MBED TLS (PolarSSL)* and *OpenSSL* peers:

```
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 failed:
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:padding is not correct
block type is not 01 file rsa_eay.c line 705: error:04067072: rsa routines:RSA_EAY_PUBLIC_DECRYPT:padding is not correct
```

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.
```

MBED TLS (PolarSSL), 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]:
invalid peer certificate: revoked
```

- after expiration:

cannot connect to proxy "proxy-openssl-1.0.1e": TCP successful, cannot establish TLS to [[127.0.0.1]:20004
invalid peer certificate: revoked, CRL expired

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.

PSK longer than 32 bytes is passed to mbed TLS (PolarSSL)

In any Zabbix log:

ssl_set_psk(): SSL - Bad input parameters to function

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.

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:

Trying to access two Zabbix frontend installations on the same host, on different ports, simultaneously will fail. Logging into the second one will terminate the session on the first one - unless the default frontend session name is adjusted for the second frontend in frontend [definitions](#) (see ZBX_SESSION_NAME).

1 Frontend sections

1 Monitoring

Overview

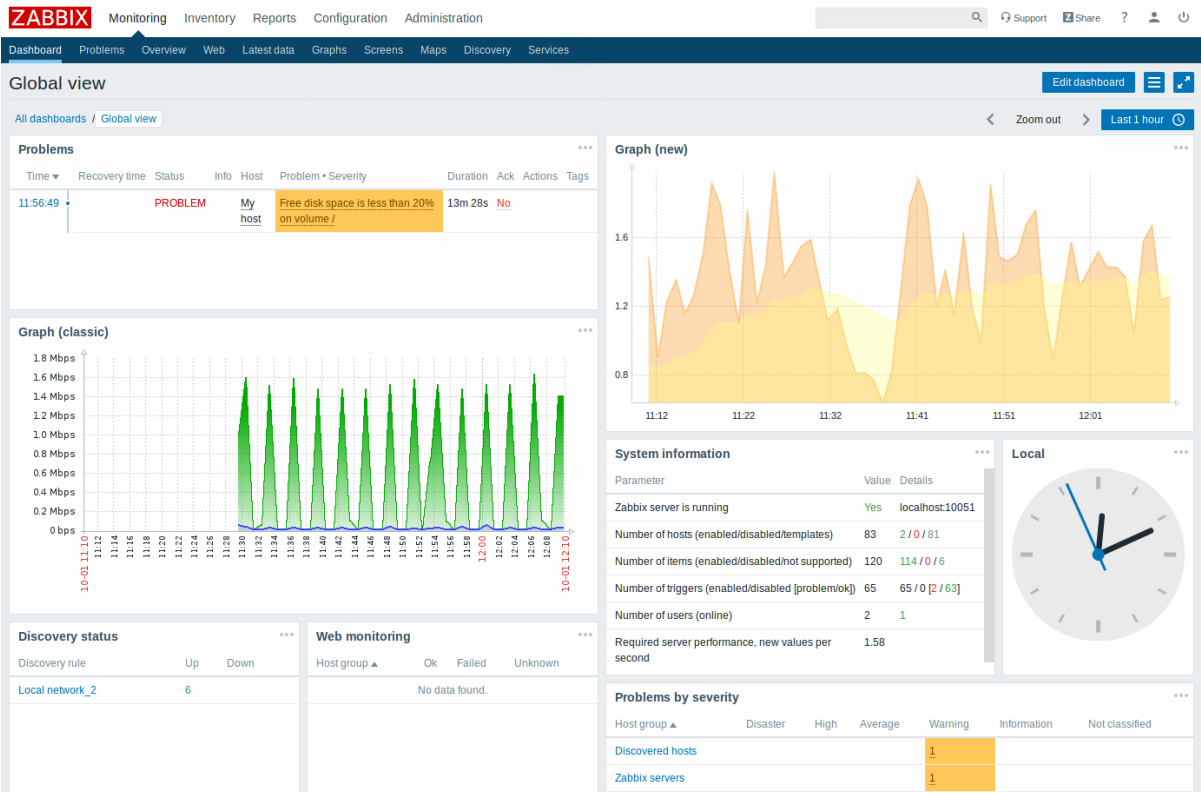
The Monitoring menu is all about displaying data. Whatever information Zabbix is configured to gather, visualize and act upon, it will be displayed in the various sections of the Monitoring menu.

1 Dashboard

Overview

The *Monitoring* → *Dashboard* section is designed to display summaries of all the important information.

A dashboard 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.



Widgets are added and edited in the dashboard editing mode. Widgets are viewed in the dashboard viewing mode.

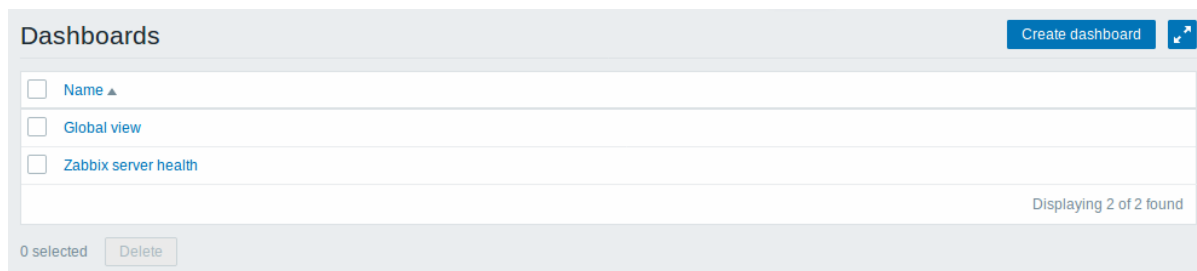
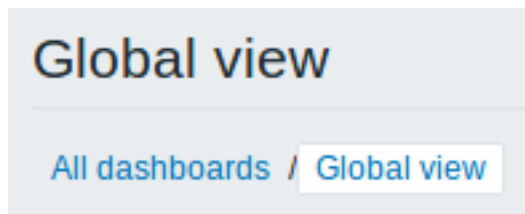
While in a single dashboard you can group widgets from various sources for a quick overview, it is also possible to create several dashboards containing different sets of overviews and switch between them.

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 to expand and collapse the time period selector.

Note that when the dashboard is displayed in kiosk mode (accessible from the fullscreen mode) and widgets only are displayed, it is possible to zoom out the graph period by double clicking in the graph.

Viewing dashboards

To access all configured dashboards, click on the *All dashboards* link just below the section title.

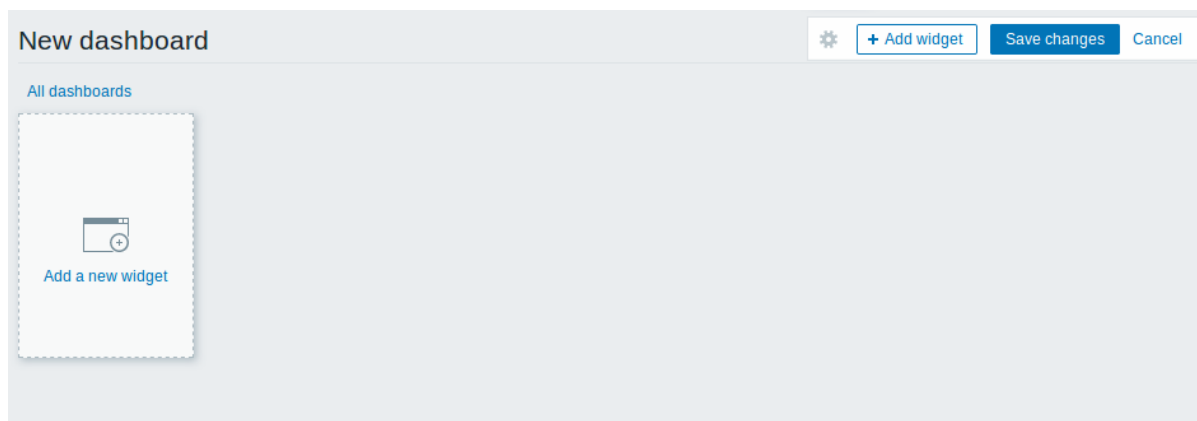


To view a single dashboard, click on its name in the list of all dashboards.

To delete one or several dashboards, mark the checkboxes of the respective dashboards and click on *Delete* below the list.

Creating a dashboard

When viewing all dashboards, you can click on the *Create dashboard* button to create a new dashboard:



Initially the dashboard is empty. To populate the dashboard, you can add widgets.

Click on the *Save changes* button to save the dashboard. If you click on *Cancel*, the dashboard will not be created.

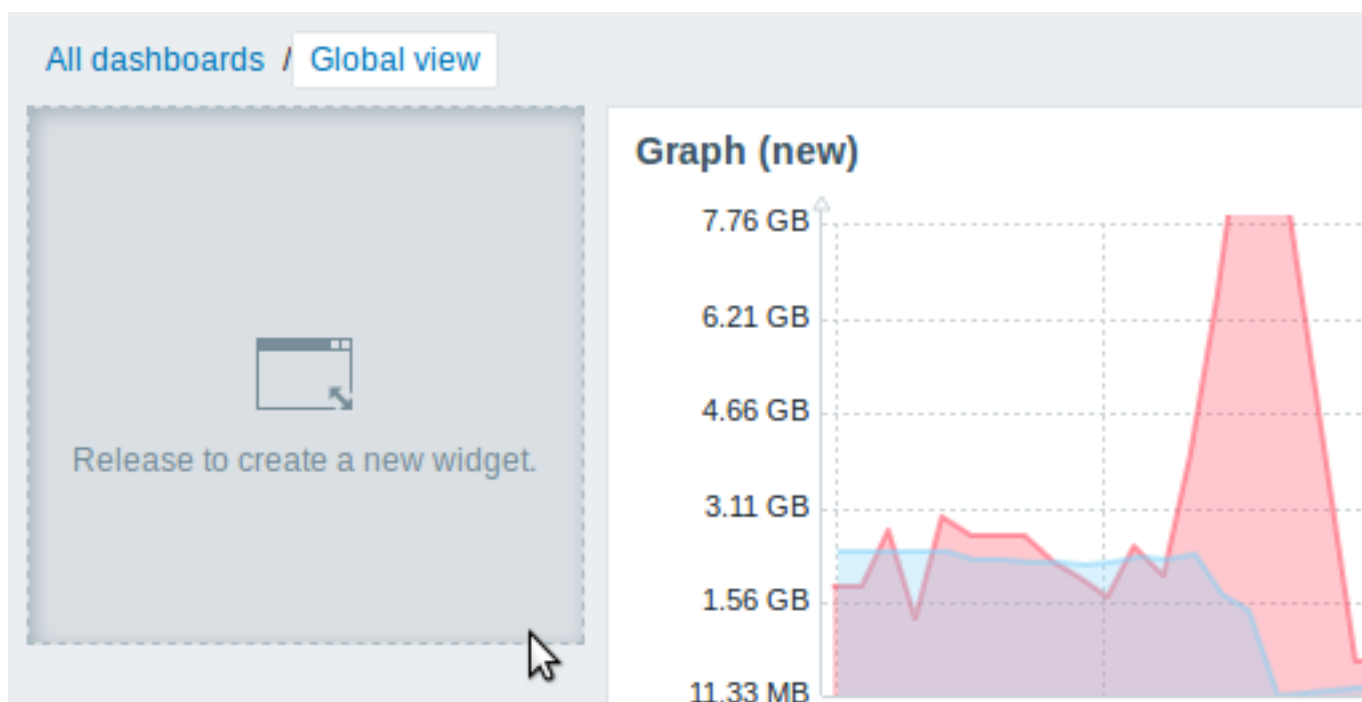
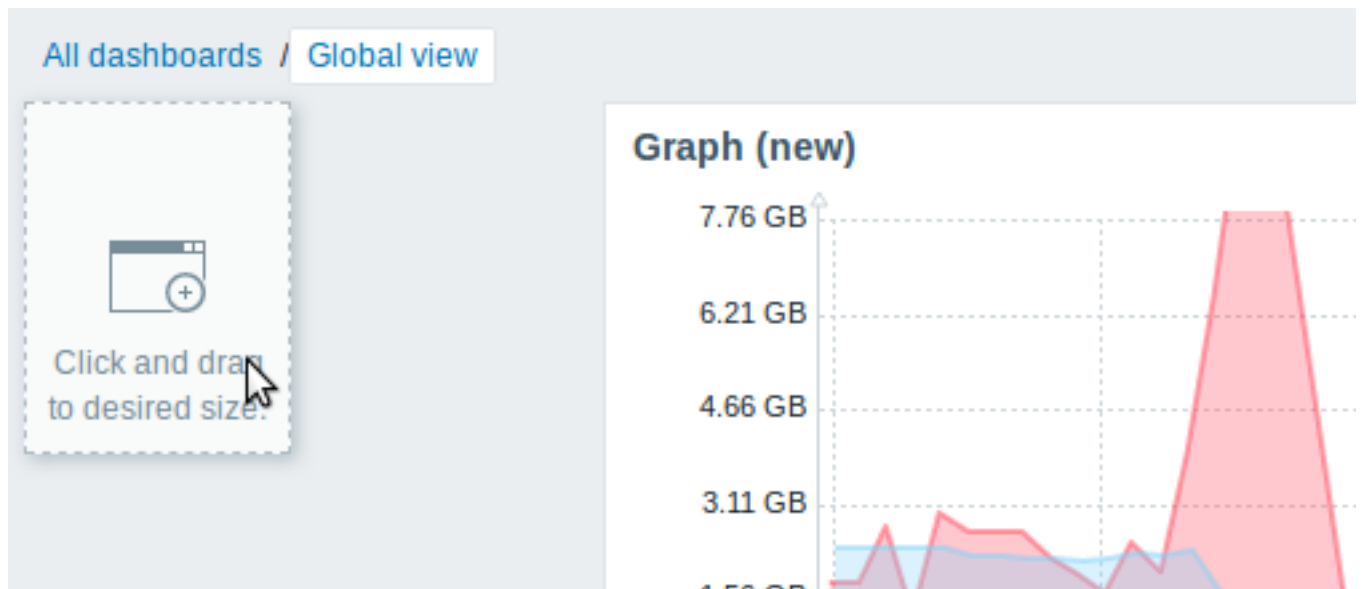
Adding widgets

To add a widget to a dashboard:

- Click on the *Add widget* button in dashboard editing mode. The widget will be created in its default size and placed after the existing widgets (if any);

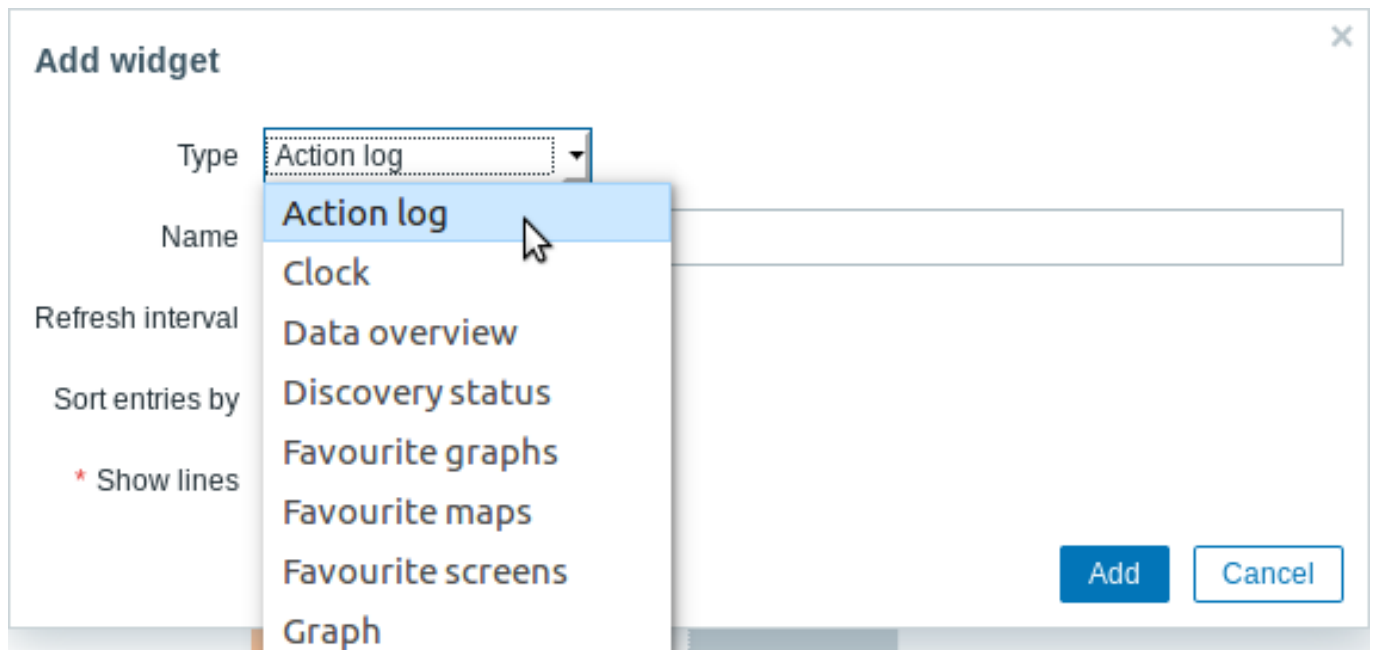
Or

- 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 take up all the available space if its default size is bigger. Alternatively you may click and drag the placeholder to the desired widget size, then release.



Then, in the widget configuration form:

- Select the *Type* of widget
- Enter widget parameters
- Click on *Add*



The following widgets can be added to a dashboard:

- Action log
- Clock
- Data overview
- Discovery status
- Favourite graphs
- Favourite maps
- Favourite screens
- Graph
- Graph (classic)
- Problem hosts
- Map
- Map navigation tree
- Plain text
- Problems
- System information
- Problems by severity
- 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 within the widget to:

-  - edit a widget;
-  - remove a widget

Click on *Save changes* for the dashboard to make any changes to the widgets permanent.

Dynamic widgets

When **configuring** some of the widgets:

- Graphs (simple and custom)
- 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 two new dropdowns have appeared atop the dashboard for selecting the host group/host:

Thus you have a widget, which can display content that is based on the data from the host that is selected in the dropdown. 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.

Viewing and editing a dashboard

When viewing a single dashboard, the following options are available:

Edit dashboard	Switch to the dashboard editing mode.
Sharing	Open the action menu. Edit sharing preferences for the dashboard. 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 to other users and user groups. For details on configuring sharing, see the map configuration section.
Create new	Create a new dashboard. First you are prompted to enter general properties of the new dashboard - owner and name. Then, the new dashboard opens in editing mode and you can add widgets.
Clone	Create a new dashboard by copying properties of the existing one. First you are prompted to enter general properties of the new dashboard - owner and name. Then, the new dashboard opens in editing mode with all the widgets of the original dashboard.
Delete	Delete the dashboard.
	Display dashboard in fullscreen mode. Fullscreen mode can also be accessed with the following URL parameters: /zabbix.php?action=dashboard.view&fullsc To exit: /zabbix.php?action=dashboard.view&fullsc



Display dashboard in kiosk mode. In this mode only widgets are displayed. The kiosk mode button appears when the fullscreen mode is activated. To exit kiosk mode, move the mouse



cursor until the exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).

Kiosk mode can also be accessed with the following URL parameters:

`/zabbix.php?action=dashboard.view&kiosk=`

To exit to normal mode:

`/zabbix.php?action=dashboard.view&fullsc`

Editing mode is opened:

- when a new dashboard is being created
- when you click the *Edit dashboard* button for an existing dashboard

In the dashboard editing mode the following options are available:

	Edit general dashboard properties - name and owner.
	Add a new widget.
	Save dashboard changes.
	Cancel dashboard changes.

Permissions to dashboards

Permissions to dashboards for regular and Zabbix Admin users 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 custom scripts, inventory, latest data, problems, graphs and screens for the host.

The screenshot shows the Zabbix 'Problems' table. The table has columns: Time, Info, Host, and Problem • Severity. Two problem entries are visible:

Time	Info	Host	Problem • Severity
2018-06-12 13:10:07		Zabbix server	Free disk space is less than 20% on volume /
2017-10-25 11:30:05		New host	Free disk space is less than 20% on

A context menu is open for the 'New host' entry, showing the following options:

- SCRIPTS
 - Detect operating system
 - MyScripts
 - Ping
 - Traceroute
- GO TO
 - Host inventory
 - Latest data
 - Problems
 - Graphs
 - Host screens

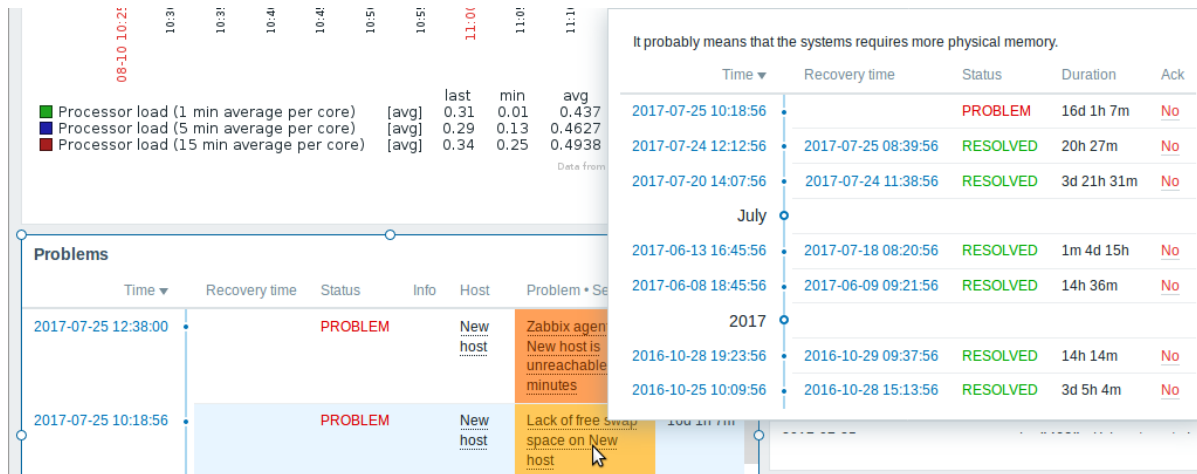
Below the table, there is a 'Data overview' section and an 'Items' link.

The host menu is accessible by clicking on a host in several other frontend sections:

- Monitoring → [Problems](#)
- Monitoring → [Problems](#) → Event details
- Monitoring → [Overview](#) (on *Hosts: left*)
- Monitoring → [Latest data](#)
- Monitoring → [Screens](#) (in *Host issues* and *Host group issues* widgets)
- 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 name in the *Problem-Severity* column of the *Problems* widget. The popup disappears once you remove the mouse from the problem name.
- click on the problem name in the *Problem-Severity* column of the *Problems* widget. The popup disappears only if you click on the problem name again.

Attention:

Resolved values of {ITEM.VALUE} and {ITEM.LASTVALUE} macros in trigger descriptions are truncated to 20 characters. To see the entire values you may use **macro functions** with these macros, e.g. {{ITEM.VALUE}.regsub("(.*)", \1)}, {{ITEM.LASTVALUE}.regsub("(.*)", \1)} as a workaround.

1 Dashboard widgets

Overview

This section lists available **dashboard** widgets and provides details for widget configuration.

The following parameters are common for every single widget:

<i>Name</i>	Enter a widget name.
<i>Refresh interval</i>	Configure default refresh interval. Default refresh intervals for widgets range from <i>No refresh</i> to <i>15 minutes</i> depending on the type of widget. For example:// <i>No refresh</i> for URL widget, <i>1 minute</i> for action log widget, <i>15 minutes</i> for clock widget.

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 a priority over widget's setting and once it's set it's always preserved when widget's setting is modified.

Action log

In the action log widget you can display details of action operations (notifications, remote commands). It replicates information from *Administration* → *Audit*.

To configure, select *Action log* as type:

Add widget

Type

Name

Refresh interval

Sort entries by

* Show lines

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.

Clock

In the clock widget you may display local, server or specified host time.

To configure, select *Clock* as type:

Add widget

Type

Name

Refresh interval

Time type

You may set the following specific options:

Time type

Select local, server or specified host time.

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.

Data overview

In the data overview widget you can display the latest data for a group of hosts. It replicates information from *Monitoring → Overview* (when *Data* is selected as Type there).

To configure, select *Data overview* as type:

Add widget

Type

Data overview

Name

Data overview

Refresh interval

Default (1 minute)

Host groups

Discovered hosts

type here to search

Select

Application

CPU

Show suppressed problems

☒

Hosts location

Left

Top

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.
Application	Enter application name.
Show suppressed problems	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.
Hosts location	Select host location - left or top.

Discovery status

This widget displays a status summary of the active network discovery rules.

Favourite graphs

This widget contains shortcuts to the most needed graphs, sorted alphabetically.

The list of shortcuts is populated when you **view** a graph and then click on its  *Add to favourites* button.

Favourite maps

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 favourites* button.

Favourite screens

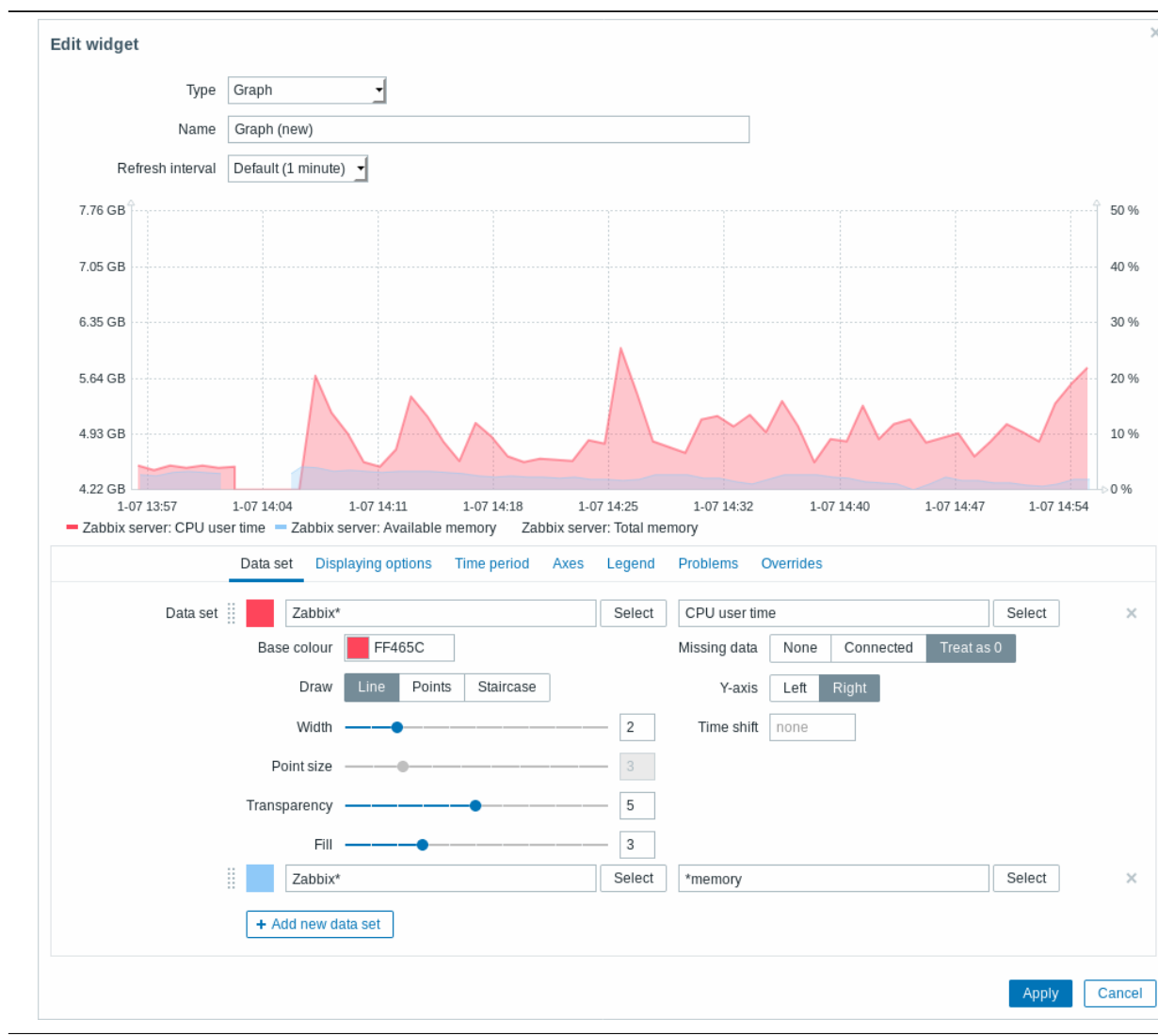
This widget contains shortcuts to the most needed screens and slide shows, sorted alphabetically.

The list of shortcuts is populated when you **view** a screen/slide show and then click on its  *Add to favourites* button.

Graph

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. The graph widget supported before Zabbix 4.0 can still be used and is renamed to **Graph (classic)**.

To configure, select *Graph* as type:



The **Data set** tab allows to add data sets and define their visual representation:

	Select hosts and items to display on the graph. Alternatively you may enter host and item patterns. Wildcard patterns may be used (for example, * will return results that match zero or more characters). Up to 50 items may be displayed in the graph. Multiple entities (hosts, host patterns, items, item patterns) are separated with a comma. Therefore hosts or items with a comma in the name are not supported. Host pattern and item pattern fields are mandatory. <i>Note</i> that item names containing deprecated positional macros (\$1-\$9) are not resolved in this field. Therefore it will not be possible to add any of the items named like CPU \$2 time individually to the graph using its resolved name (like CPU user time); while using its unresolved name CPU \$2 time will add all corresponding items to the graph (e.g. CPU user time, CPU system time, CPU idle time, etc.).
Base colour	Adjust base colour, either from the colour picker or manually. Base colour is used to calculate different colours for each item of the data set. Base colour input field is mandatory.
Draw	Choose the draw type of the metric. Possible draw types are <i>Line</i> (set by default), <i>Points</i> and <i>Staircase</i>. <i>Note</i> that if there's only one data point in line/staircase graph it is drawn as point regardless of draw type. The point size is calculated from line width, but it cannot be smaller than 3 pixels, even if line width is less.
Width	Set the line width. This option is available when <i>Line</i> or <i>Staircase</i> draw type is selected.
Point size	Set the point size. This option is available when <i>Points</i> draw type is selected.
Transparency	Set the transparency level.
Fill	Set the fill level. This option is available when <i>Line</i> or <i>Staircase</i> 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
Y-axis	Not applicable for <i>Points</i> draw type. Select the side of the graph where Y-axis will be displayed.

<i>Time shift</i>	Specify time shift if required. You may use time suffixes in this field. Negative values are allowed.

Existing data sets are displayed in a list. You may:

- + Add new data set

 - click on this button to add a new data set
- click on the colour icon to expand/collapse data set details
- click on the move icon and drag a data set to a new place in the list

The **Displaying options** tab allows to define history data selection:

Data set

Displaying options

Time period

Axes

Legend

Problems

Overrides

History data selection

Auto

History

Trends

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

The **Time period** tab allows to set a custom time period:

Data set

Displaying options

Time period

Axes

Legend

Problems

Overrides

Set custom time period ☒

From

now-1h

To

now

Set custom time period

Mark this checkbox to set 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:

Data set	Displaying options	Time period	Axes	Legend	Problems	Overrides
Left Y ☒ Show Min calculated Max calculated Units Auto value Right Y ☒ Show Min 10 Max 100 Units Auto value X-Axis ☒ Show						

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
Min

Unmark this checkbox to hide X axis (marked by default). 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 *Auto* option is chosen axis values are displayed using units of the first item of corresponding axis. *Static* option allows you to assign corresponding axis' custom name. If *Static* option is chosen and *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:

Data set	Displaying options	Time period	Axes	Legend	Problems	Overrides
Show legend ☒ Number of rows <input type="range" value="2"/> 2						

Show legend

Unmark this checkbox to hide the legend on the graph (marked by default).

Number of rows

Set the number of rows to be displayed on the graph.

The **Problems** tab allows to customize the problem display:

Data set
Displaying options
Time period
Axes
Legend
Problems
Overrides

Show problems ☒

Selected items only ☒

Problem hosts Select

Severity ☒ Not classified ☒ Information ☒ Warning ☒ Average ☒ High ☒ Disaster

Problem

Tags And/Or Or

Contains Equals Remove

Add

Show problems

Selected items only

Problem hosts

Severity

Problem

Tags

Mark this checkbox to enable problem displaying on the graph (unmarked, i.e. disabled by default).

Mark this checkbox to include problems for the selected items only to be displayed on the graph.

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).

Multiple hosts are separated with a comma. Therefore hosts with a comma in the visible name are not supported.

Mark the problem severities to be displayed on the graph.

Specify the problem's name to be displayed on the graph.

Specify tag name and value to limit the number of problems displayed on the graph. To add more tag names and values, click on *Add*.

There are two calculation types for several conditions:

And/Or - all conditions must be met, conditions having same tag name will be grouped by Or condition

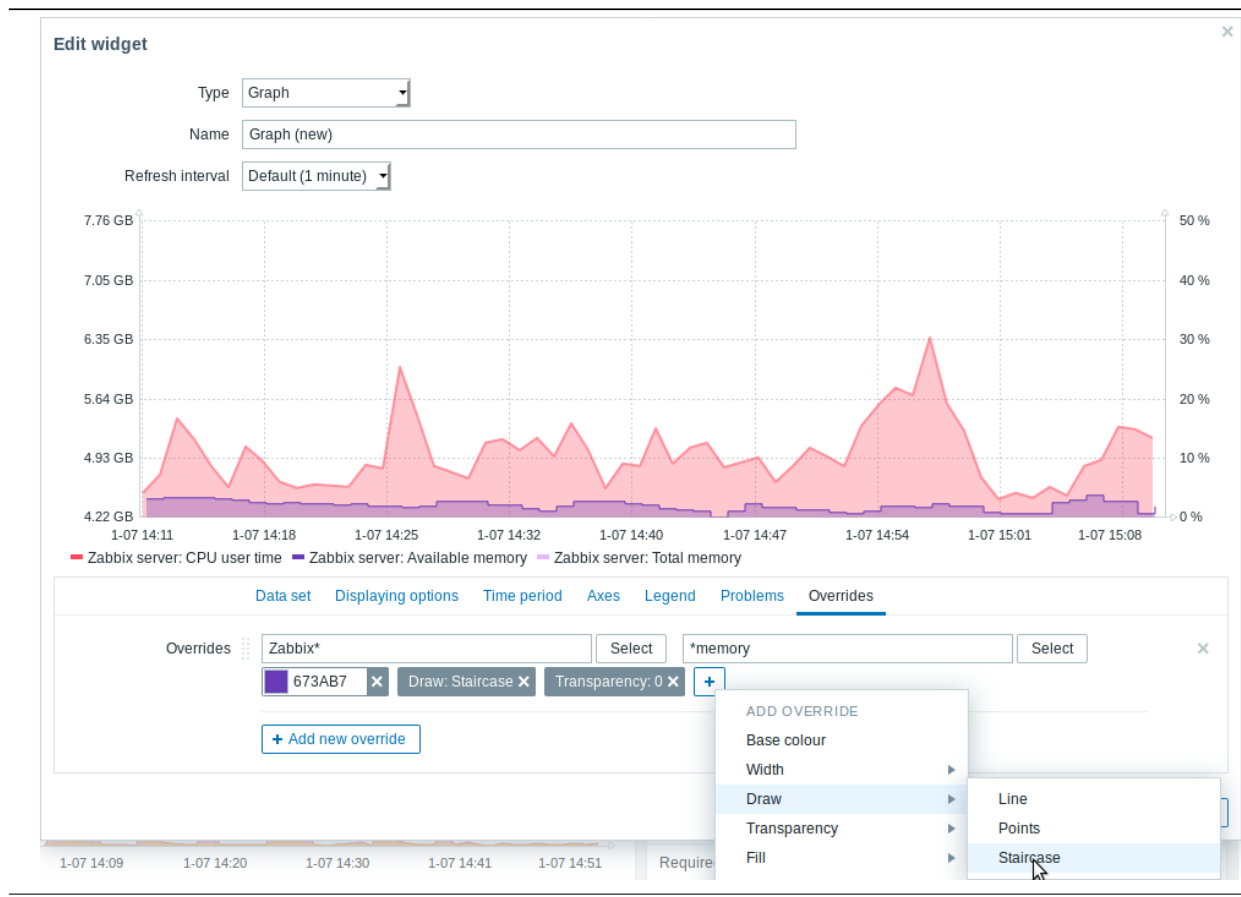
Or - enough if one condition is met

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)

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 colour 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). Multiple entities (hosts, host patterns, items, item patterns) are separated with a comma. Therefore hosts or items with a comma in the name are not supported. 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.

Graph (classic)

In the classic graph widget you can display a single custom graph or simple graph.

To configure, select *Graph* as type:

×

Edit widget

Type

Graph

Name

CPU

Refresh interval

Default (1 minute)

Source

GraphSimple graph

* Graph

My host: CPU load

Select

Dynamic item

☒

Apply

Cancel

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 <i>Source</i> .
Item	Select the item to display in a simple graph. This option is available if 'Simple graph' is selected as <i>Source</i> .
Dynamic item	Set graph to display different data depending on the selected host.

Map

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).

To configure, select *Map* as type:

×

Edit widget

Type

Map

Name

Local network

Refresh interval

Default (15 minutes)

Source type

MapMap navigation tree

* Map

Local network

Select

Apply

Cancel

You may set the following specific options:

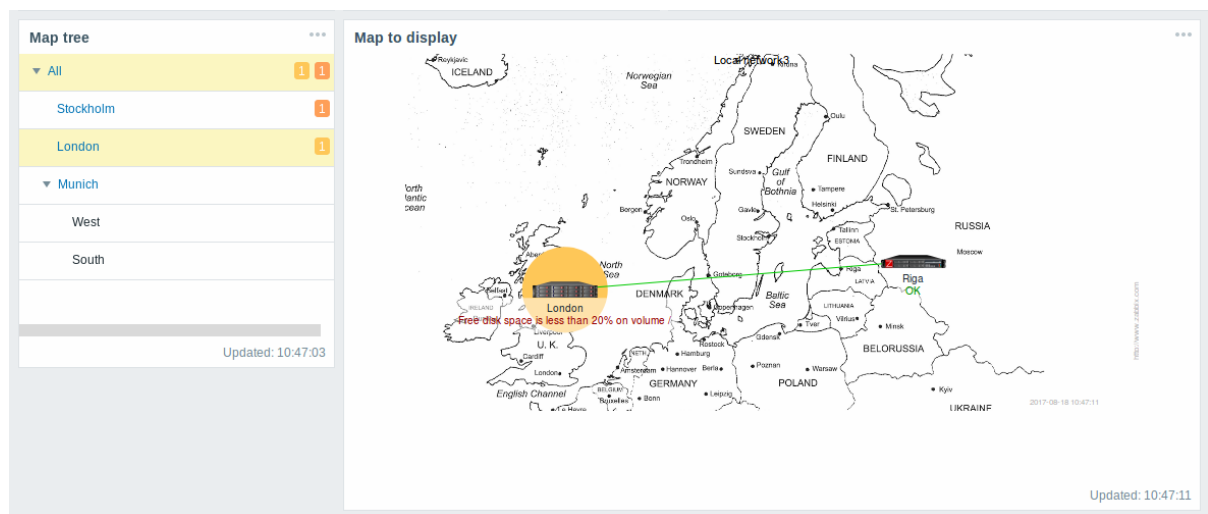
Source type	Select to display: Map - network map Map navigation tree - one of the maps in the selected map navigation tree
Map	Select the map to display. This option is available if 'Map' is selected as <i>Source type</i> .
Filter	Select the map navigation tree to display the maps of. This option is available if 'Map navigation tree' is selected as <i>Source type</i> .

See also: [known issue with IE11](#)

Map navigation tree

This widget allows to build a hierarchy of existing maps while also displaying problem statistics with each included map and map group.

It becomes even more powerful if you link the *Map* widget to the navigation tree. In this case, clicking on a map name in the navigation tree displays the map in full in the *Map* widget.



Statistics with the top level map in the hierarchy display a sum of problems of all submaps and its own problems.

To configure the navigation tree widget, select *Map navigation tree* as type:

Add widget

Type
Map navigation tree

Name
Map tree

Refresh interval
Default (15 minutes)

Show unavailable maps
☐

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 greyed out icon. Note that if this checkbox is marked, available submaps are displayed even if the parent level map is unavailable. If unmarked, available submaps to an unavailable parent map will not be displayed at all. Problem count is calculated based on available maps and available map elements.

Plain text

In the plain text widget you can display latest item data in plain text.

To configure, select *Plain text* as type:

Add widget

Type

Plain text

Name

Text item

Refresh interval

Default (1 minute)

* Items

Zabbix server: Available memory

Zabbix server: CPU idle time

type here to search

Select

Items location

Left

Top

* Show lines

25

Show text as HTML

☐

Dynamic items

☐

Add

Cancel

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.

Problem hosts

In the host information widget you can display high-level information about host availability.

To configure, select *Problem hosts* as type:

Add widget

Type

Problem hosts

Name

Problem hosts

Refresh interval

Default (1 minute)

Host groups

type here to search

Select

Exclude host groups

type here to search

Select

Hosts

type here to search

Select

Problem

Severity

☐ Not classified
☐ Information
☐ Warning
☐ Average
☐ High
☐ Disaster

Show suppressed problems

☐

Hide groups without problems

☐

Problem display

All

Separated

Unacknowledged only

Add

Cancel

You may set the following specific options:

Parameter	Description
<i>Host groups</i>	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.
<i>Exclude host groups</i>	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 <i>show</i> Group A and <i>exclude</i> Group B at the same time, only data from host 001 will be displayed in the Dashboard.
<i>Hosts</i>	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.
<i>Problem</i>	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.
<i>Severity</i>	Mark the problem severities to be displayed in the widget.

Parameter	Description
<i>Show suppressed problems</i>	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.
<i>Hide groups without problems</i>	Mark the <i>Hide groups without problems</i> option to hide data from host groups without problems in the widget.
<i>Problem display</i>	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.

Problems

In this widget you can display current problems. The information in this widget is similar to *Monitoring → Problems*.

To configure, select *Problems* as type:

Add widget

Type
Problems

Name
Problems

Refresh interval
Default (1 minute)

Show
Recent problems
Problems
History

Host groups
Discovered hosts
type here to search
Select

Exclude host groups
type here to search
Select

Hosts
type here to search
Select

Problem

Severity
☐ Not classified
☐ Information
☐ Warning
☐ Average
☐ High
☐ Disaster

Tags
And/Or
Or
tag
Contains
Equals
value
Remove
Add

Show tags
None
1
2
3

Tag name
Full
Shortened
None

Tag display priority
comma-separated list

Show suppressed problems
☐

Show unacknowledged only
☐

Show latest values
☐

Sort entries by
Time (descending)

Show timeline
☒

* Show lines
25

Add
Cancel

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, acknowledgement 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

Parameter	Description
<i>Host groups</i>	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.
<i>Exclude host groups</i>	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 <i>show</i> Group A and <i>exclude</i> Group B at the same time, only problems from host 001 will be displayed in the widget.
<i>Hosts</i>	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.
<i>Problem</i>	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.
<i>Severity</i>	Mark the problem severities to be displayed in the widget.
<i>Tags</i>	Specify event tag name and value to limit the number of problems displayed. To add more event tag names and values, click on <i>Add</i>. There are two calculation types for several conditions: And/Or - all conditions must be met, conditions having same tag name will be grouped by Or condition Or - enough if one condition is met 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) When filtered, the tags specified here will be displayed first with the problem, unless overridden by the <i>Tag display priority</i> (see below) list.
<i>Show tags</i>	Select the number of displayed tags: None - no <i>Tags</i> column in <i>Monitoring</i> → <i>Problems</i> 1 - <i>Tags</i> column contains one tag 2 - <i>Tags</i> column contains two tags 3 - <i>Tags</i> column contains three tags To see all tags for the problem roll your mouse over the three dots icon.
<i>Tag name</i>	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
<i>Tag display priority</i>	Enter tag display priority for a problem, as a comma-separated list of tags (for example: <i>Services,Applications,Application</i>). Tag names only should be used, no values. The tags of this list will always be displayed first, overriding the natural ordering by alphabet.
<i>Show suppressed problems</i>	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.
<i>Show unacknowledged only</i>	Mark the checkbox to display unacknowledged problems only.
<i>Show latest values</i>	Mark the checkbox to display latest value(s) of the items in the underlying trigger expression of the problem.

Parameter	Description
<i>Sort entries by</i>	Sort entries by: Time (descending or ascending) Severity (descending or ascending) Problem name (descending or ascending) Host (descending or ascending).
<i>Show timeline</i>	Mark the checkbox to display a visual timeline.
<i>Show lines</i>	Specify the number of problem lines to display.

Problems by severity

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.

To configure, select *Problems by severity* as type:

Add widget

Type

Problems by severity

Name

Problems by severity

Refresh interval

Default (1 minute)

Host groups

Discovered hosts

type here to search

Select

Exclude host groups

type here to search

Select

Hosts

type here to search

Select

Problem

Severity

☐ Not classified
☐ Information
☐ Warning
☐ Average
☐ High
☐ Disaster

Show suppressed problems

☐

Hide groups without problems

☐

Show latest values

☐

Problem display

All

Separated

Unacknowledged only

Show timeline

☒

Add

Cancel

You may set the following specific options:

Parameter	Description
<i>Host groups</i>	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.
<i>Exclude host groups</i>	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 <i>show</i> Group A and <i>exclude</i> Group B at the same time, only data from host 001 will be displayed in the Dashboard.
<i>Hosts</i>	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.
<i>Problem</i>	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.
<i>Severity</i>	Mark the problem severities to be displayed in the widget.
<i>Show suppressed problems</i>	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.
<i>Hide groups without problems</i>	Mark the <i>Hide groups without problems</i> option to hide data from host groups without problems in the widget.
<i>Show latest values</i>	Mark the checkbox to display latest value(s) of the items in the underlying trigger expression of the problem.
<i>Problem display</i>	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.
<i>Show timeline</i>	Mark the checkbox to display a visual timeline.

System information

In the System information widget you can display high-level Zabbix and Zabbix server information.

To configure, select *System information* as type:

Add widget

Type

System information

Name

System information

Refresh interval

Default (15 minutes)

Add

Cancel

Trigger overview

In the trigger overview widget you can display the trigger states for a group of hosts. It replicates information from *Monitoring → Overview* (when *Triggers* is selected as Type there).

To configure, select *Trigger overview* as type:

Add widget

Type

Trigger overview

Name

Trigger overview

Refresh interval

Default (1 minute)

Show

Recent problems

Problems

Any

Host groups

type here to search

Select

Application

Show suppressed problems

☐

Hosts location

Left

Top

Add

Cancel

You may set the following specific options:

Show

Filter by problem status:

Recent problems - unresolved and recently resolved problems are displayed (default)

Problems - unresolved problems are displayed

Any - history of all events is displayed

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.

Application

Enter the application name.

Show suppressed problems

Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.

Hosts location

Select host location - left or top.

In the URL widget you can display a URL content from an external resource.

To configure, select *URL* as type:

Add widget

Type

URL

Name

URL

Refresh interval

Default (No refresh)

* URL

http://

Dynamic item

☐

You may set the following specific options:

URL	Enter the URL to display. The URL must start with <code>http://</code> . {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.

Web monitoring

This widget displays a status summary of the active web monitoring scenarios.

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.

2 Problems

Overview

In *Monitoring* → *Problems* you can see what problems you currently have. Problems are those triggers that are in the “Problem” state.

Problems

Export to CSV

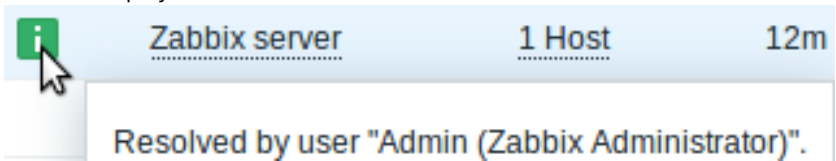
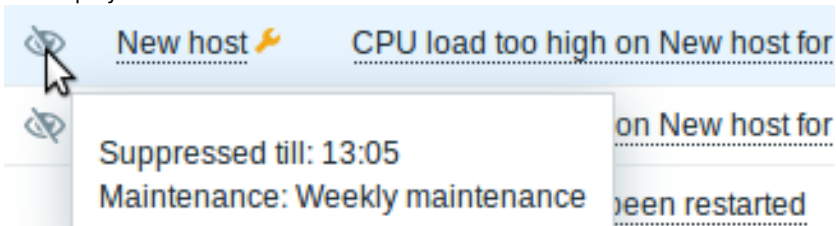
Filter

Time	Severity	Recovery time	Status	Info	Host	Problem	Duration	Ack	Actions	Tags
2017-10-25 11:30:05	Average		PROBLEM	New host	Free disk space is less than 20% on volume /	8m 13d 3h	Yes			
14:55:56	Information		PROBLEM	New host	New host has just been restarted	4m 7s	No			

Displaying 2 of 2 found

0 selected

Mass update

Column	Description
<i>Time</i>	Problem start time is displayed.
<i>Severity</i>	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 <i>Update problem screen</i>. Colour of the problem severity is used as cell background during problem time.
<i>Recovery time</i>	Problem resolution time is displayed.
<i>Status</i>	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 <i>Administration</i> → <i>General</i> → <i>Trigger displaying options</i>.
<i>Info</i>	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 <i>Show suppressed problems</i> option in the filter). Rolling a mouse over the icon will display more details: 
<i>Host</i>	Problem host is displayed.
<i>Problem</i>	Problem name is displayed. Problem name is based on the name of the underlying problem trigger. Clicking on the problem name brings up the <i>event menu</i>.
<i>Latest values</i>	Latest value(s) of the items in the underlying trigger expression of the problem are displayed. This column is only displayed if the <i>Show latest values</i> option is selected in the filter.
<i>Duration</i>	Problem duration is displayed. See also: <i>negative problem duration</i>
<i>Ack</i>	The acknowledgement 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 <i>problem update</i> 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 for the explanation on icons used in the popup for actions taken.
Tags	Event tags are displayed (if any).

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 `item.nodata()` 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 `item.nodata()` 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

Buttons to the right offer the following options:

Export to CSV

Export content from the selected page to a CSV file.



Display page in fullscreen mode.



Display page in kiosk mode. In this mode only page content is displayed.
The kiosk mode button appears when the fullscreen mode is activated.



To exit kiosk mode, move the mouse cursor until the exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).

Using filter

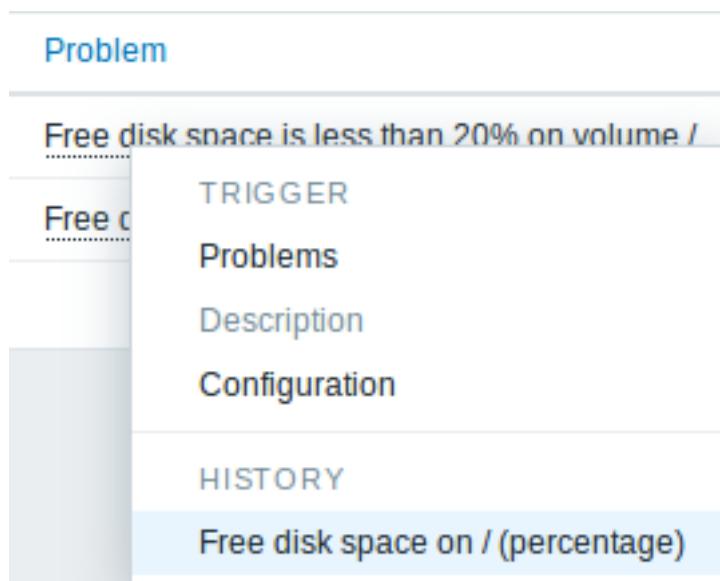
You can use the filter to display only the problems you are interested in. The filter is located above the table.

Parameter	Description
<i>Show</i>	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
<i>Host groups</i>	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
<i>Hosts</i>	Filter by one or more hosts.
<i>Application</i>	Filter by application name.
<i>Triggers</i>	Filter by one or more triggers.
<i>Problem</i>	Filter by problem name.
<i>Minimum severity</i>	Filter by minimum trigger (problem) severity.
<i>Age less than</i>	Filter by how old the problem is.
<i>Host inventory</i>	Filter by inventory type and value.
<i>Tags</i>	Filter by event tag name and value. Several conditions can be set. There are two calculation types for conditions: And/Or - all conditions must be met, conditions having same tag name will be grouped by Or condition Or - enough if one condition is met 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) When filtered, the tags specified here will be displayed first with the problem, unless overridden by the <i>Tag display priority</i> (see below) list.

Parameter	Description
<i>Show tags</i>	Select the number of displayed tags: None - no <i>Tags</i> column in <i>Monitoring</i> → <i>Problems</i> 1 - <i>Tags</i> column contains one tag 2 - <i>Tags</i> column contains two tags 3 - <i>Tags</i> column contains three tags To see all tags for the problem roll your mouse over the three dots icon.
<i>Tag name</i>	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
<i>Tag display priority</i>	Enter tag display priority for a problem, as a comma-separated list of tags (for example: <i>Services,Applications,Application</i>). Tag names only should be used, no values. The tags of this list will always be displayed first, overriding the natural ordering by alphabet.
<i>Show suppressed problems</i>	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.
<i>Compact view</i>	Mark the checkbox to enable compact view.
<i>Show details</i>	Mark the checkbox to display underlying trigger expressions of the problems. Disabled if <i>Compact view</i> checkbox is marked.
<i>Show latest values</i>	Mark the checkbox to display latest value(s) of the items in the underlying trigger expression of the problem.
<i>Show unacknowledged only</i>	Mark the checkbox to display unacknowledged problems only.
<i>Show timeline</i>	Mark the checkbox to display the visual timeline and grouping. Disabled if <i>Compact view</i> checkbox is marked.
<i>Highlight whole row</i>	Mark the checkbox to highlight the full line for unresolved problems. The problem severity colour is used for the highlighting. Enabled only if the <i>Compact view</i> checkbox is marked in the standard blue and dark themes. <i>Highlight whole row</i> is not available in the high-contrast themes.

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 description (if available) and the trigger configuration. Access to simple graph/item history of the underlying item(s) is also available.

Attention:

Resolved values of {ITEM.VALUE} and {ITEM.LASTVALUE} macros in trigger descriptions are truncated to 20 characters. To see the entire values you may use **macro functions** with these macros, e.g. {{ITEM.VALUE}.regsub("(.*)", \1)}, {{ITEM.LASTVALUE}.regsub("(.*)", \1)} as a workaround.

Viewing details

The times for problem start and recovery in *Monitoring* → *Problems* are links. Clicking on them opens more details of the event.

Event details

Trigger details

Host	My host
Trigger	Processor load is too high on My host
Severity	Warning
Problem expression	{My host.system.cpu.load[percpu.avg1].avg(5m)}>1.5
Recovery expression	
Event generation	Normal
Allow manual close	No
Enabled	Yes

Event details

Event	Processor load is too high on My host
Severity	Information
Time	2019-04-10 15:38:10
Acknowledged	Yes
Tags	App: MySQL Service: JIRA

Actions

Step	Time	User/Recipient	Action	Message/Command	Status	Info
	2019-04-10 15:40:43	Admin (Zabbix Administrator)	✓	OK		
1	2019-04-10 15:38:13	Admin (Zabbix Administrator) mndvolk@inbox.lv	✉	Problem: Processor load is too high on My host Problem started at 15:38:10 on 2019.04.10 Problem name: Processor load is too high on My host Host: My host Severity: Warning Original problem ID: 127125	Sent	
	2019-04-10 15:38:10		📅			

Event list [previous 20]

Time	Recovery time	Status	Age	Duration	Ack	Actions
2019-04-10 15:38:10		PROBLEM	2m 41s	2m 41s	Yes	🗨️ ⬇️ ⚙️
2019-04-10 15:10:10	2019-04-10 15:13:10	RESOLVED	30m 41s	3m	No	
2019-04-10 09:23:10	2019-04-10 09:24:10	RESOLVED	6h 17m 41s	1m	No	
2019-04-10 09:05:10	2019-04-10 09:14:10	RESOLVED	6h 35m 41s	9m	No	
2019-04-09 17:07:10	2019-04-09 17:13:10	RESOLVED	22h 33m 41s	6m	No	
2019-04-09 12:40:10	2019-04-09 12:42:10	RESOLVED	1d 3h	2m	No	
2019-04-09 12:27:10	2019-04-09 12:32:10	RESOLVED	1d 3h 13m	5m	No	
2019-04-09 12:23:10	2019-04-09 12:26:10	RESOLVED	1d 3h 17m	3m	No	
2019-04-09 11:50:10	2019-04-09 11:54:10	RESOLVED	1d 3h 50m	4m	No	
2019-04-09 11:16:10	2019-04-09 11:18:10	RESOLVED	1d 4h 24m	2m	No	

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
- 🗨️ - 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)
- 🔧 - remote command has been executed
- 📅 - problem event has recovered
- ✓ - problem has been closed manually

3 Overview

Overview

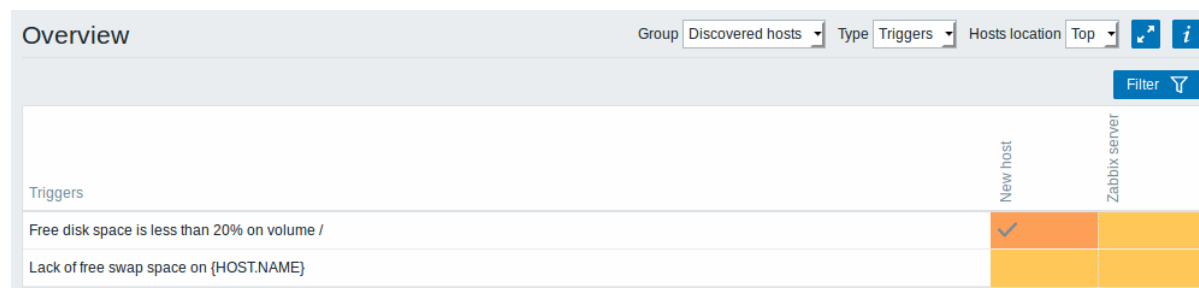
The *Monitoring* → *Overview* section offers an overview of trigger states or a comparison of data for various hosts at once.

The following display options are available:

- select all hosts or specific host groups in the *Group* dropdown
- choose what type of information to display (triggers or data) in the *Type* dropdown
- select horizontal or vertical display of information in the *Hosts location* dropdown

Overview of triggers

In the next screenshot Triggers are selected in the *Type* dropdown. As a result, trigger states of two local hosts are displayed as coloured blocks (the colour of problem triggers depends on the problem severity colour, which can be adjusted in the [problem update](#) screen):



The screenshot shows the Zabbix Overview page with the 'Group' dropdown set to 'Discovered hosts' and the 'Type' dropdown set to 'Triggers'. The 'Hosts location' dropdown is set to 'Top'. A table displays triggers for two hosts: 'New host' and 'Zabbix server'. The triggers are 'Free disk space is less than 20% on volume /' and 'Lack of free swap space on {HOST.NAME}'. The 'New host' column shows a green checkmark for the first trigger and a yellow square for the second. The 'Zabbix server' column shows a yellow square for both triggers.

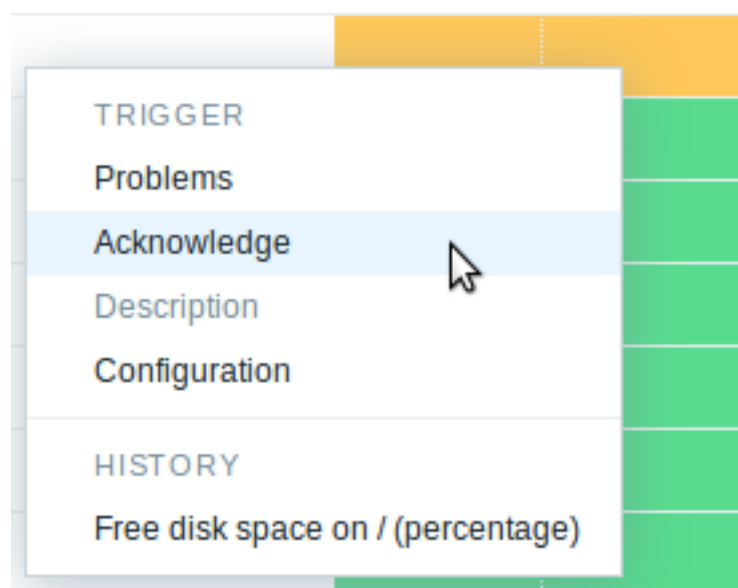
Triggers	New host	Zabbix server
Free disk space is less than 20% on volume /	✓	
Lack of free swap space on {HOST.NAME}		

Note that recent trigger changes (within the last 2 minutes) will be displayed as blinking blocks.

Blue 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. (Before 4.2.8 it was enough for the last problem to be acknowledged.)

Clicking on a trigger block provides context-dependent links to problem events of the trigger, the problem acknowledgement screen, trigger configuration, trigger URL or a simple graph/latest values list.



Buttons

Buttons to the right offer the following options:



Display page in fullscreen mode.



Display page in kiosk mode. In this mode only page content displayed.

The kiosk mode button appears when the fullscreen mode is activated.



To exit kiosk mode, move the mouse cursor until the exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).



Additional information on the page content is displayed if you roll the mouse over this button.

Using filter

You can use the filter to display only the problems you are interested in. The filter is located above the table.

Filter

Show

Recent problems

Problems

Any

Acknowledge status

Any

Minimum severity

Not classified

Age less than

☐ 14 days

Name

Application

Select

Host inventory

Type

Add

Remove

Show suppressed problems

☐

Apply

Reset

Parameter	Description
Show	Filter by problem status: Recent problems - unresolved and recently resolved problems are displayed (default) Problems - unresolved problems are displayed Any - history of all events is displayed
Acknowledge status	Filter by acknowledgement status: Any - acknowledged and unacknowledged problems are displayed (default) With unacknowledged events - problems with unacknowledged events are displayed With last event unacknowledged - problems with last event unacknowledged are displayed
Minimum severity	Filter by minimum problem severity.
Age (less than)	Mark the checkbox to filter by problem age.
Name	Filter by problem name.
Application	Filter by application.
Host inventory	Filter by inventory type and value.
Show suppressed problems	Mark the checkbox to display problems which would otherwise be suppressed (not shown) because of host maintenance.

Overview of data

In the next screenshot Data is selected in the *Type* dropdown. As a result, performance item data of two local hosts are displayed.

Overview

Group

all

Type

Data

Hosts location

Top

Filter

Application

Filesystems

Select

Show suppressed problems

☒

Apply

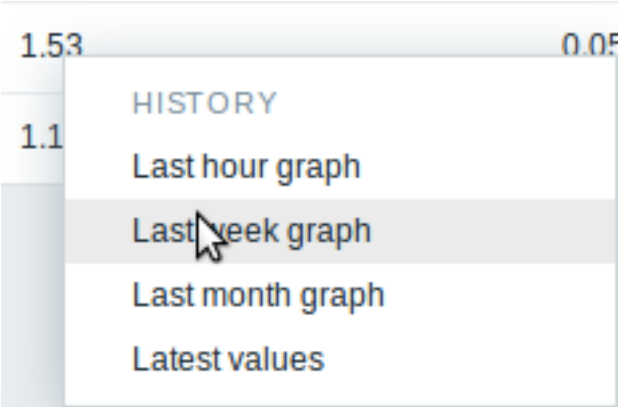
Reset

Items	New host	Zabbix server
Free disk space on /	55.27 GB	55.27 GB
Free disk space on / (percentage)	26.71 %	26.71 %
Free disk space on /boot	365.45 MB	365.45 MB
Free disk space on /boot (percentage)	81.54 %	81.54 %

The colour of problem items is based on the problem severity colour, which can be adjusted in the **problem update** screen.

Only values that fall within the last 24 hours are displayed by default. This limit has been introduced with the aim of improving initial loading times for large pages of latest data. It is also possible to change this limitation by changing the value of ZBX_HISTORY_PERIOD **constant** in *include/defines.inc.php*.

Clicking on a piece of data offers links to some predefined graphs or latest values.



4 Web

Overview

In the *Monitoring* → *Web* section current information about **web scenarios** is displayed.

Web monitoring				
		Group	Discovered hosts	Host
				Zabbix server
Host	Name ▲	Number of steps	Last check	Status
Zabbix server	Zabbix frontend	5	2018-07-25 11:11:20	OK
Displaying 1 of 1 found				

Note: The name of a disabled host is displayed in red (in both the host dropdown and the list). Data of disabled hosts is accessible starting with Zabbix 2.2.0.

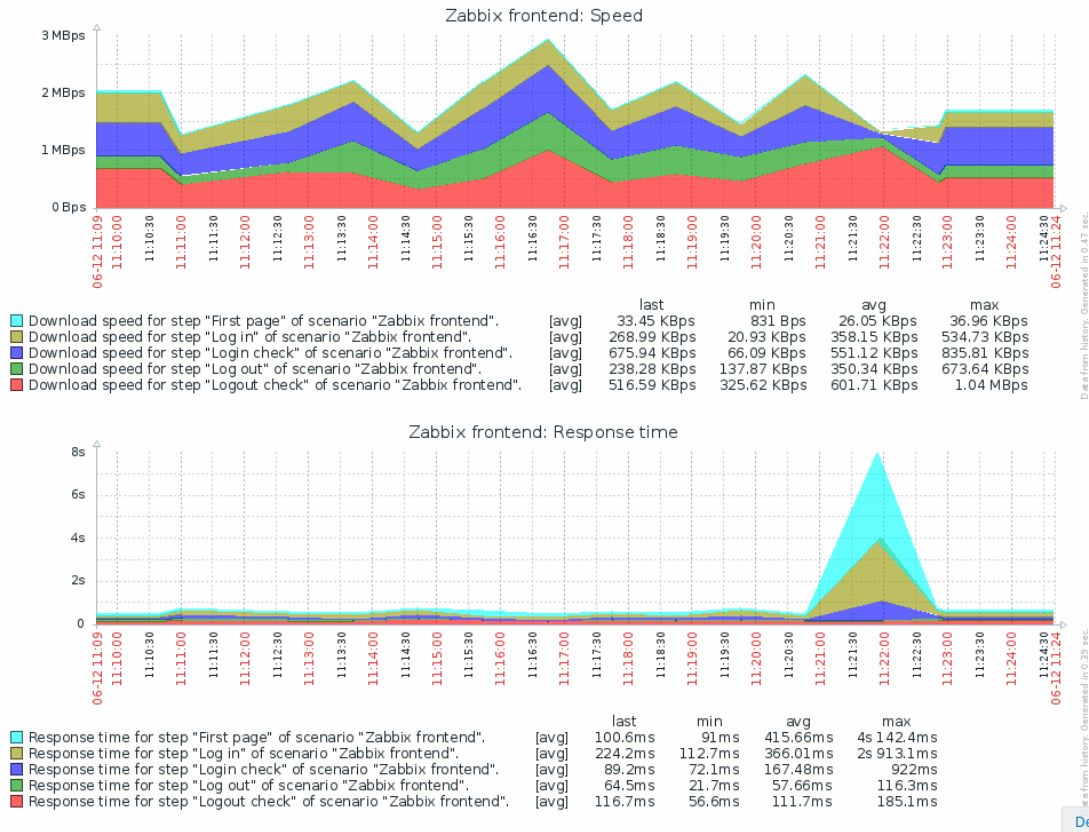
Only values that fall within the last 24 hours are displayed by default. This limit has been introduced with the aim of improving initial loading times for large pages of web monitoring. It is also possible to change this limitation by changing the value of ZBX_HISTORY_PERIOD **constant** in *include/defines.inc.php*.

The scenario name is link to more detailed statistics about it:

Details of web scenario: Zabbix frontend

Step	Speed	Response time	Response code	Status
First page	33.45 KBps	100.6ms	200	OK
Log in	268.99 KBps	224.2ms	200	OK
Login check	675.94 KBps	89.2ms	200	OK
Log out	238.28 KBps	64.5ms	200	OK
Logout check	516.59 KBps	116.7ms	200	Error: required pattern "Username" was not found on http://192.168.3.31/zabbix/index.php
TOTAL		595.2ms		Error: required pattern "Username" was not found on http://192.168.3.31/zabbix/index.php

Zoom out Last 15 minutes



Buttons

Buttons to the right offer the following options:



Display page in fullscreen mode.



Display page in kiosk mode. In this mode only page content displayed.

The kiosk mode button appears when the fullscreen mode is activated.

To exit kiosk mode, move the mouse cursor until the exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).



5 Latest data

Overview

The section in *Monitoring* → *Latest data* can be used to view latest values gathered by items as well as to access various graphs for the items.

When you open this page for the first time, nothing is displayed.

Latest data

Filter

Host groups

type here to search

Select

Name

Hosts

type here to search

Select

Show items without data

☐

Application

Select

Show details

☐

Apply

Reset

☐ Host	Name	Last check	Last value	Change
Specify some filter condition to see the values.				

To access data, you need to make selections in the filter such as host group, host, application or item name.

Filter

Host groups

type here to search

Select

Name

Hosts

My host x Zabbix server x

Select

Show items without data

☒

Application

Zabbix frontend

Select

Show details

☐

Apply

Reset

☐ Host	Name	Last check	Last value	Change
☒ My host	Zabbix frontend (18 Items)			
☐	Download speed for scenario "Zabbix frontend".	2018-06-12 12:22:30	559.77 KBps	+268.14 KBps Graph
☐	Download speed for step "First page" of scenario "Zabbix fronte ...	2018-06-12 12:22:30	28.64 KBps	-8.21 KBps Graph
☐	Download speed for step "Log in" of scenario "Zabbix frontend".	2018-06-12 12:22:30	602.82 KBps	+214.78 KBps Graph
☐	Download speed for step "Login check" of scenario "Zabbix fron...	2018-06-12 12:22:30	1.02 MBps	+535.32 KBps Graph
☐	Download speed for step "Log out" of scenario "Zabbix frontend".	2018-06-12 12:22:30	265.56 KBps	+159.42 KBps Graph
☐	Download speed for step "Logout check" of scenario "Zabbix fro...	2018-06-12 12:22:30	854.24 KBps	+439.38 KBps Graph
☐	Failed step of scenario "Zabbix frontend".	2018-06-12 12:22:30	0	Graph
☐	Last error message of scenario "Zabbix frontend".	2018-06-12 12:02:16	required pattern "Us...	History
☐	Response code for step "First page" of scenario "Zabbix frontend".	2018-06-12 12:22:30	200	Graph
☐	Response code for step "Log in" of scenario "Zabbix frontend".	2018-06-12 12:22:30	200	Graph
☐	Response code for step "Login check" of scenario "Zabbix front...	2018-06-12 12:22:30	200	Graph
☐	Response code for step "Log out" of scenario "Zabbix frontend".	2018-06-12 12:22:30	200	Graph
☐	Response code for step "Logout check" of scenario "Zabbix fron...	2018-06-12 12:22:30	200	Graph
☐	Response time for step "First page" of scenario "Zabbix frontend".	2018-06-12 12:22:30	117.5ms	+20ms Graph
☐	Response time for step "Log in" of scenario "Zabbix frontend".	2018-06-12 12:22:30	100ms	-50ms Graph
☐	Response time for step "Login check" of scenario "Zabbix fronte...	2018-06-12 12:22:30	57.5ms	-60ms Graph
☐	Response time for step "Log out" of scenario "Zabbix frontend".	2018-06-12 12:22:30	57.9ms	-90ms Graph
☐	Response time for step "Logout check" of scenario "Zabbix front...	2018-06-12 12:22:30	70.6ms	-70ms Graph
☒ Zabbix server	Zabbix frontend (18 Items)			

In the list displayed, click on  before a host and the relevant application to reveal latest values of that host and application.

You can expand all hosts and all applications, thus revealing all items by clicking on  in the header row.

Note: The name of a disabled host is displayed in red. Data of disabled hosts, including graphs and item value lists, is accessible in *Latest data* since Zabbix 2.2.0.

Items are displayed with their name, last check time, last value, change amount and a link to a simple graph/history of item values.

Only values that fall within the last 24 hours are displayed by default. This limit has been introduced with the aim of improving initial loading times for large pages of latest data. It is also possible to change this limitation by changing the value of `ZBX_HISTORY_PERIOD` **constant** in `include/defines.inc.php`.

Attention:

For items with update frequency of 1 day or more the change amount will never be displayed (with the default setting). Also in this case the last value will not be displayed at all if it was received more than 24 hours ago.

Buttons

Buttons to the right offer the following options:



Display page in fullscreen mode.



Display page in kiosk mode. In this mode only page content displayed.

The kiosk mode button appears when the fullscreen mode is activated.



To exit kiosk mode, move the mouse cursor until the exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).

Using filter

You can use the filter to display only the items you are interested in. The *Filter* link is located above the table to the right. You can use it to filter items by host group, host, application, a string in the item name; you can also select to display items that have no data gathered.

Specifying a parent host group implicitly selects all nested host groups.

Show details allows to extend displayable information on the items. Such details as refresh interval, history and trends settings, item type and item errors (fine/unsupported) are displayed. A link to item configuration is also available.

Latest data

Filter

Host groupsSelect

Hosts

My host

Zabbix server

type here to search

Select

ApplicationZabbix frontendSelect

Name

Show items without data☒

Show details☐

ApplyReset

☐ Host	Name	Last check	Last value	Change	
☐ My host	Zabbix frontend (1 item)				
☒	Download speed for scenario "Zabbix frontend".	2018-06-12 12:38:38	516.67 KBps	-3.12 KBps	Graph
☐ Zabbix server	Zabbix frontend (1 item)				
☒	Download speed for scenario "Zabbix frontend".	2018-06-12 12:38:56	638.86 KBps	+265.55 KBps	Graph

2 selectedDisplay stacked graphDisplay graph

By default, items without data are shown but details are not displayed.

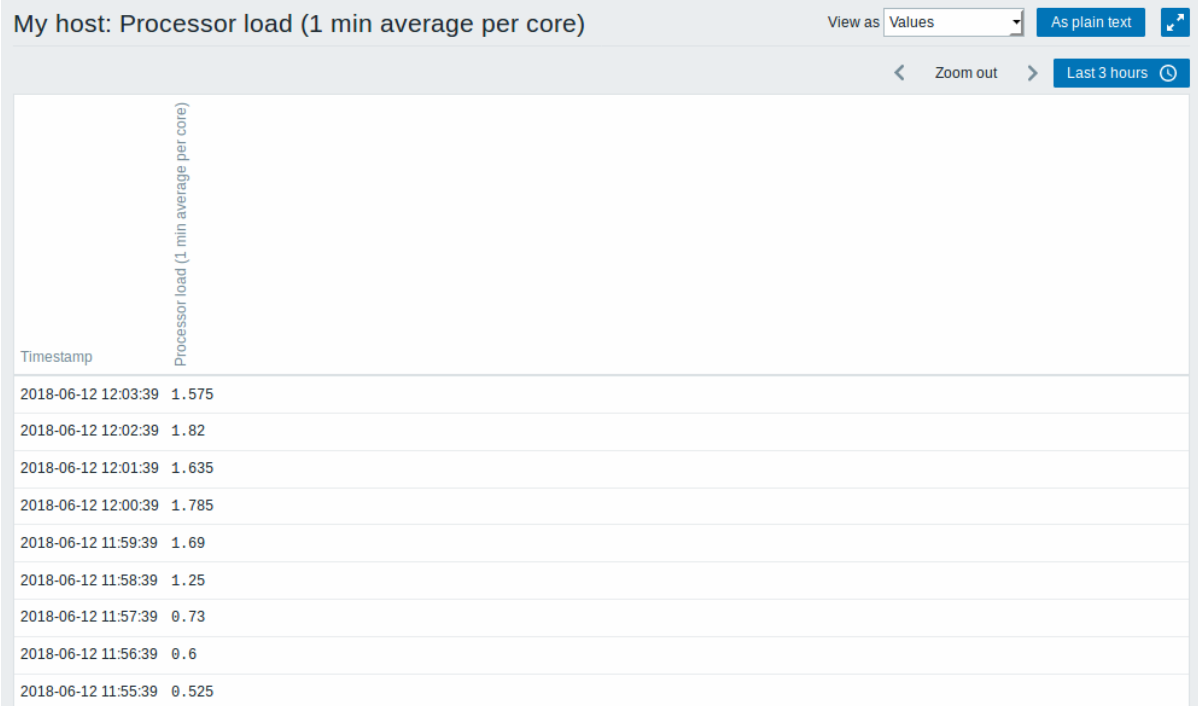
Ad-hoc graphs for comparing items

You may use the checkbox in the second column to select several items and then compare their data in a simple or stacked **ad-hoc graph**. To do that, select items of interest, then click on the required graph button below the table.

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.

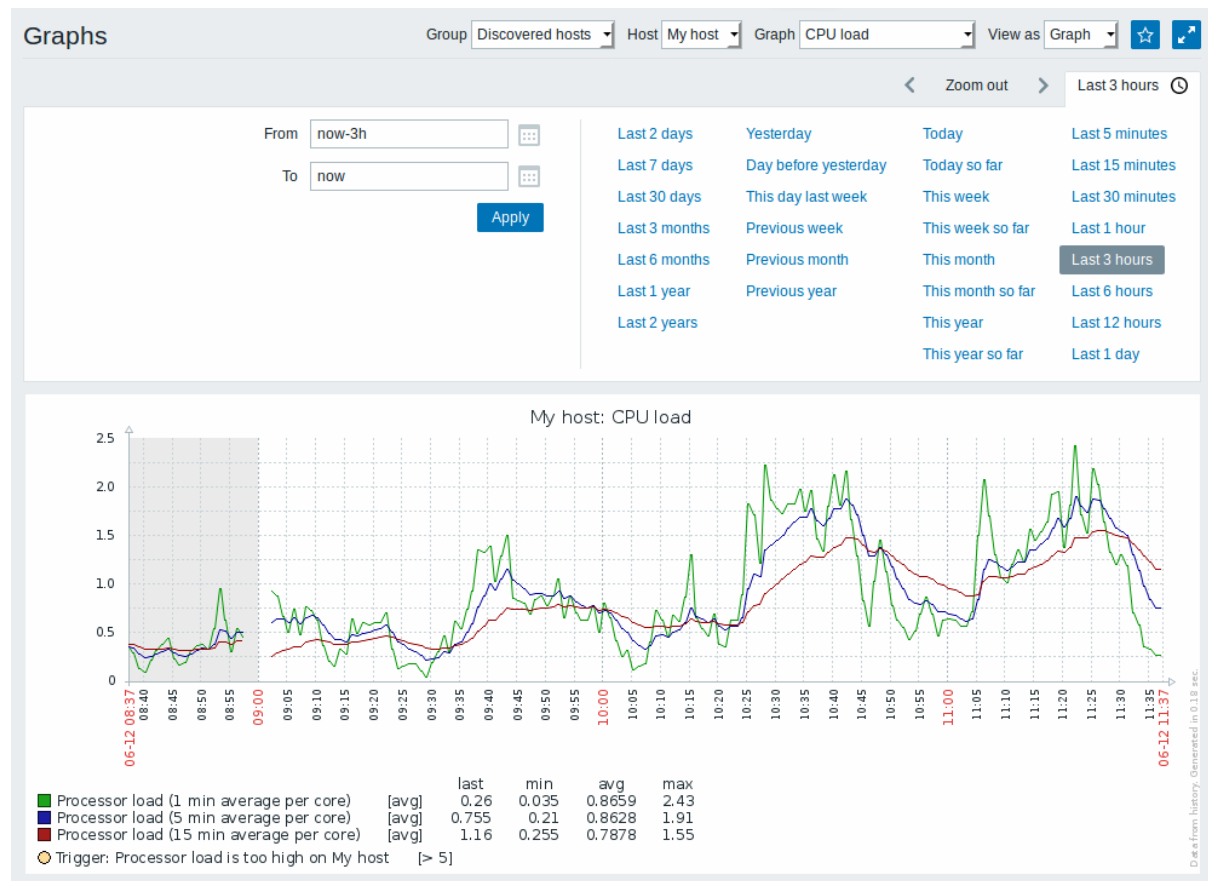
Note:

The total amount of values displayed is defined by the value of *Limit for search and filter results* parameter, set in [Administration](#) → [General](#).

6 Graphs

Overview

In the [Monitoring](#) → [Graphs](#) section any [custom graph](#) that has been configured can be displayed.



To display a graph, select the host group, host and then the graph from the dropdowns to the right.

Note: In the host dropdown, a disabled host is highlighted in red. Graphs for disabled hosts are accessible starting with Zabbix 2.2.0.

Time period selector

Take note of the time period selector above the graph. It allows to select often required periods with one mouse click.

See also: [Time period selector](#)

Buttons

Buttons to the right offer the following options:



Add graph to the favourites widget in the **Dashboard**.



The graph is in the favourites widget in the **Dashboard**. Click to remove graph from the favourites widget.



Display page in fullscreen mode.



Display page in kiosk mode. In this mode only page content displayed.
The kiosk mode button appears when the fullscreen mode is activated.



To exit kiosk mode, move the mouse cursor until the exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).

7 Screens

Overview

In the *Monitoring* → *Screens* section you can configure, manage and view Zabbix global **screens** and **slide shows**.

When you open this section, you will either see the last screen/slide show you accessed or a listing of all entities you have access to. Screen/slide show listing can be filtered by name.

Since Zabbix 3.0 all screens/slide shows can be either public or private. The public ones are available to all users, while private ones are accessible only to their owner and the users the entity is shared with.

Use the dropdown in the title bar to switch between screens and slide shows.

Screen listing

Screens

Screens

Create screen

Import

Filter

Name

Apply

Reset

☐	Name ▲	Dimension (cols x rows)	Actions
☐	Offices	3 x 2	Properties Constructor
☐	Zabbix server	3 x 2	Properties Constructor

Displaying 2 of 2 found

Displayed data:

Column	Description
<i>Name</i>	Name of the screen. Click on the name to view the screen.

Column	Description
<i>Dimensions</i>	The number of columns and rows of the screen.
<i>Actions</i>	Two actions are available: Properties - edit general screen properties (name and dimensions) Constructor - access the grid of screen elements for editing

To **create** a new screen, click on the *Create screen* button in the top right-hand corner. To import a screen from an XML file, click on the *Import* button in the top right-hand corner. The user who imports the screen will be set as its owner.

Mass editing options

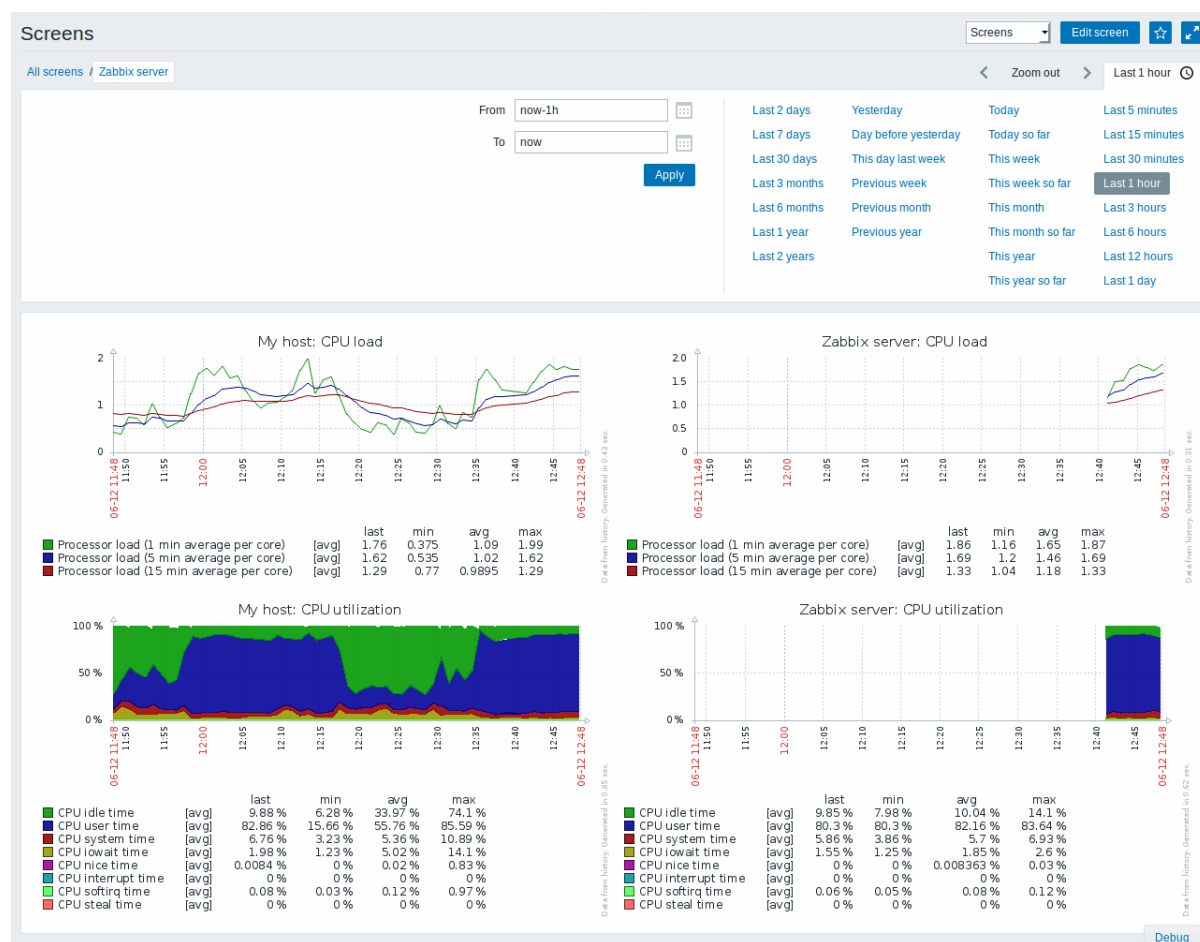
Buttons below the list offer some mass-editing options:

- *Export* - export the screens to an XML file
- *Delete* - delete the screens

To use these options, mark the checkboxes before the respective screens, then click on the required button.

Viewing screens

To view a screen, click on its name in the list of all screens.



Time period selector

Take note of the time period selector above the screen. It allows to select often required periods with one mouse click, affecting the data displayed in graphs etc.

See also: [Time period selector](#)

Buttons

Buttons to the right offer the following options:

[Edit screen](#)

Go to the screen constructor to edit the screen.



Add screen to the favourites widget in the **Dashboard**.

The screen is in the favourites widget in the **Dashboard**.
Click to remove screen from the favourites widget.

Display page in fullscreen mode.

Display page in kiosk mode. In this mode only page content displayed.
The kiosk mode button appears when the fullscreen mode is activated.



To exit kiosk mode, move the mouse cursor until the exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).

Slide show listing

Use the dropdown in the title bar to switch from screens to slide shows.

Slide shows

Slide shows

Create slide show

Filter

Name

Apply

Reset

☐	Name	Delay	Number of slides	Actions
☐	Zabbix	30s	2	Properties

Displaying 1 of 1 found

Displayed data:

Column	Description
Name	Name of the slide show. Click on the name to view the slide show.
Delay	The default duration of showing one slide is displayed.
Number of slides	The number of slides in the slide show is displayed.
Actions	One action is available: Properties - edit slide show properties

To **create** a new slide show, click on the *Create slide show* button in the top right-hand corner.

Mass editing options

A button below the list offers one mass-editing option:

- *Delete* - delete the slide shows

To use this option, mark the checkboxes before the respective slide shows and click on *Delete*.

Viewing slide shows

To view a slide show, click on its name in the list of all slide shows.

Buttons

Buttons to the right offer the following options:

Edit slide show

Go to the slide show properties.



Add slide show to the favourites widget in the **Dashboard**.

The slide show is in the favourites widget in the **Dashboard**.
Click to remove slide show from the favourites widget.

Slow down or speed up a slide show.

Display page in fullscreen mode.

Display page in kiosk mode. In this mode only page content displayed.
The kiosk mode button appears when the fullscreen mode is activated.

To exit kiosk mode, move the mouse cursor until the  exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).

Referencing a screen

Screens can be referenced by both `elementid` and `screenname` GET parameters. For example,

`http://zabbix/zabbix/screens.php?screenname=Zabbix%20server`

will open the screen with that name (Zabbix server).

If both `elementid` (screen ID) and `screenname` (screen name) are specified, `screenname` has higher priority.

8 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. Map listing can be filtered by name.

Since Zabbix 3.0 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

Maps

Create mapImport

Filter

Name

ApplyReset

☐ Name ▲	Width	Height	Actions
☐ Local network	680	600	PropertiesConstructor
☐ Local network_2	600	450	PropertiesConstructor
☐ Offices	750	600	PropertiesConstructor

Displaying 3 of 3 found

Displayed data:

Column	Description
<i>Name</i>	Name of the map. Click on the name to view the map.
<i>Width</i>	Map width is displayed.

Column	Description
<i>Height</i>	Map height is displayed.
<i>Actions</i>	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 an XML file, click on the *Import* button in the top right-hand corner. The user who imports the map will be set as its owner.

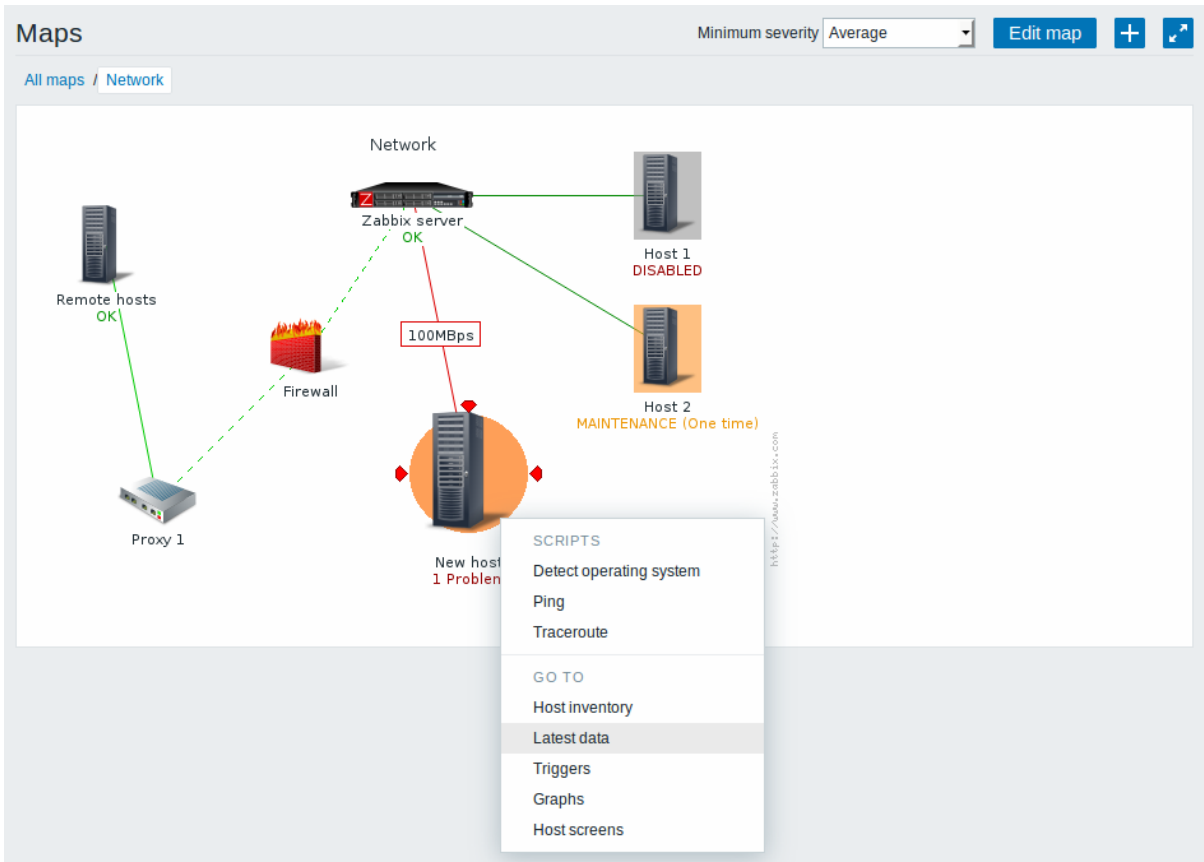
Two buttons below the list offer some mass-editing options:

- *Export* - export the maps to an XML file
- *Delete* - delete the maps

To use these options, mark the checkboxes before the respective maps, then click on the required button.

Viewing maps

To view a map, click on its name in the list of all maps.



You can use the dropdown 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 map configuration. If the map contains a submap, navigating to the submap will retain the higher-level map severity.

Icon highlighting

If a map element is in problem status, it is highlighted with a round circle. The fill colour of the circle corresponds to the severity colour 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.
- a disabled (not-monitored) host is highlighted with a grey, 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:

	Go to map constructor to edit the map content.
	Add map to the favourites widget in the Dashboard .
	The map is in the favourites widget in the Dashboard . Click to remove map from the favourites widget.
	Display page in fullscreen mode.
	Display page in kiosk mode. In this mode only page content displayed. The kiosk mode button appears when the fullscreen mode is activated.
	To exit kiosk mode, move the mouse cursor until the  exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).

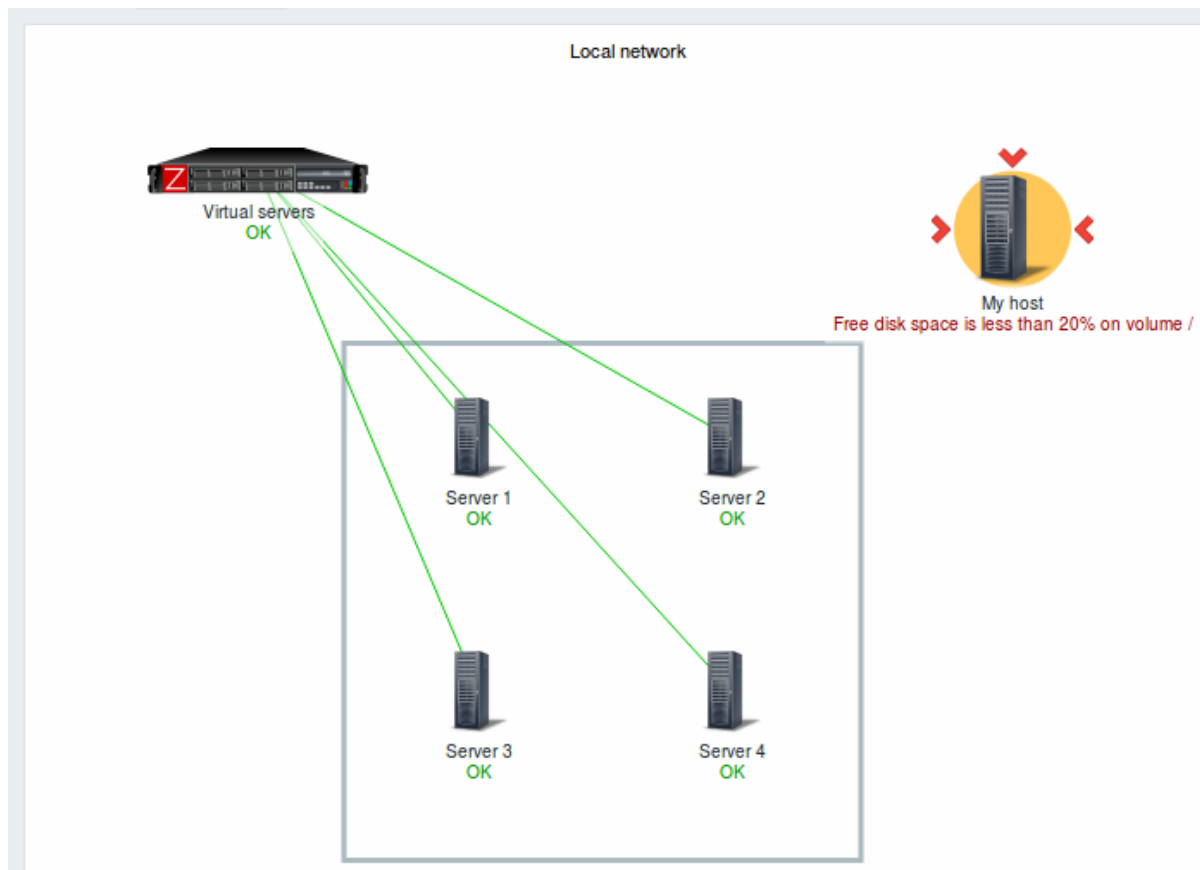
Readable summary in maps

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.

9 Discovery

Overview

In the *Monitoring* → *Discovery* section results of **network discovery** are shown. Discovered devices are sorted by the discovery rule.

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.

Status of discovery

Discovery rule

type here to search

Select

Apply

Reset

Filter

Discovered device

Monitored host

Uptime/Downtime

Local network (14 devices)

192.168.3.114 (radix-ilo.zabbix.lan)

Integrated Lights-Out 4 2.61 Jul 27 2018

1d 2h 47m

192.168.3.72 (winxp.zabbix.lan)

Linux zeus 4.8.6.5-smp_2 SMP Sun Nov 13 14_58_11 CDT 2016 i686

7 days, 20:37:53

7d 20h 37m

192.168.3.70 (win2008r3.zabbix.lan)

Hardware_x86 Family 6 Model 23 Stepping 6 AT_AT COMPATIBLE - Software_Windows Version 6.0_Build 6001 Multiprocessor Free_

2 days, 02:23:47

2d 2h 23m

SNMPv2 agent: iso.3.6.1.2.1.1.0

Note that in Zabbix 4.2.0 there is a dropdown instead of the filter for discovery rule selecting. The filter is available since Zabbix 4.2.1.

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

Buttons to the right offer the following options:



Display page in fullscreen mode.



Display page in kiosk mode. In this mode only page content displayed.
The kiosk mode button appears when the fullscreen mode is activated.

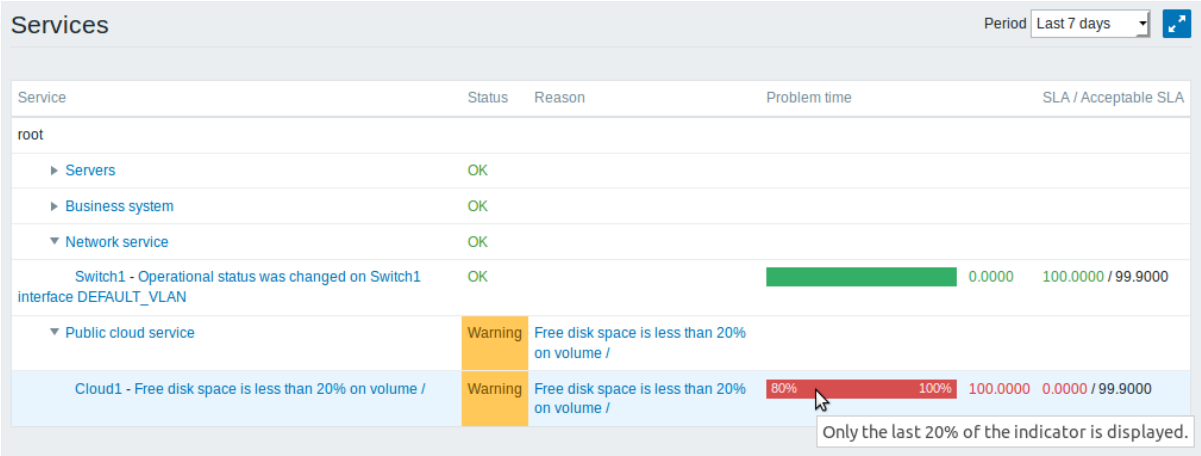


To exit kiosk mode, move the mouse cursor until the exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).

10 Services

Overview

In the *Monitoring* → *Services* section the status of IT infrastructure or business *services* is displayed.



A list of the existing services is displayed along with data of their status and SLA. From the dropdown in the upper right corner you can select a desired period for display.

Displayed data:

Parameter	Description
Service	Service name.
Status	Status of service: OK - no problems (trigger colour and severity) - indicates a problem and its severity
Reason	Indicates the reason of problem (if any).
Problem time	Displays SLA bar. Green/red ratio indicates the proportion of availability/problems. The bar displays the last 20% of SLA (from 80% to 100%). The bar contains a link to a graph of availability data.

Parameter	Description
SLA/Acceptable SLA	Displays current SLA/expected SLA value. If current value is below the acceptable level, it is displayed in red.

You can also click on the service name to access the *Service availability report*.

Service availability report: Switch1						Period	Weekly	Year	2017	
From	Till	Ok	Problems	Downtime	SLA	Acceptable SLA				
2017-02-20 00:00	2017-02-21 12:13	1d 12h 13m			100.0000	99.9000				
2017-02-13 00:00	2017-02-20 00:00	7d 0h 0m			100.0000	99.9000				
2017-02-06 00:00	2017-02-13 00:00	7d 0h 0m			100.0000	99.9000				

Here you can assess service availability data over a longer period of time on daily/weekly/monthly/yearly basis.

Buttons

Buttons to the right offer the following options:



Display page in fullscreen mode.



Display page in kiosk mode. In this mode only page content displayed.

The kiosk mode button appears when the fullscreen mode is activated.



To exit kiosk mode, move the mouse cursor until the exit button appears and click on it. Note that you will be taken back to normal mode (not fullscreen mode).

2 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 a host group (or all groups) and the inventory field by which to display data. The number of hosts corresponding to each entry of the chosen field will be displayed.

Host inventory overview		Group	all	Grouping by	Type
TYPE	HOST COUNT ▼				
Zabbix server	1				
Workstation	1				
Switch	1				

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.

Host inventory

Group all

Filter

Field Type

contains

Zab

Apply

Reset

Host	Group	Name	Type	OS	Serial number A	Tag	MAC address A
Zabbix server	Discovered hosts, Zabbix servers	martins-HP-Pro-3010-Small-Form-Factor-PC	Zabbix server	Linux martins-HP-Pro-3010-Small-Form-Factor-PC 4.4.0-135-generic			

Displaying 1 of 1 found

2 Hosts

Overview

In the *Inventory* → *Hosts* section **inventory data** of hosts are displayed.

Select a group from the dropdown in the upper right corner to display the inventory data of hosts in that group. You can also filter the hosts by any inventory field to display only the hosts you are interested in.

Host inventory

Group all

Filter

Field Type

contains

Zab

Apply

Reset

Host	Group	Name	Type	OS	Serial number A	Tag	MAC address A
Zabbix server	Discovered hosts, Zabbix servers	martins-HP-Pro-3010-Small-Form-Factor-PC	Zabbix server	Linux martins-HP-Pro-3010-Small-Form-Factor-PC 4.4.0-135-generic			

Displaying 1 of 1 found

To display all host inventories, select "all" in the group dropdown, 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:

Host inventory

Overview Details

Host name

[Zabbix server 1](#)

Visible name

Zabbix server

Agent interfaces

IP address	DNS name	Connect to	Port	Default
192.168.3.220		IP DNS	10050	☒

SNMP interfaces

IP address	DNS name	Connect to	Port	Default
127.0.0.1		IP DNS	161	☐

OS

Linux linux-qvvt 3.11.10-21-default #1 SMP Mon Jul 21 15:28:46 U

Description

Added on 2015-07-28.

Monitoring

[Web](#)
[Latest data](#)
[Triggers](#)
[Problems](#)
[Graphs](#)
[Screens](#)

Configuration

[Host](#)
[Applications 13](#)
[Items 81](#)
[Triggers 47](#)
[Graphs 12](#)
[Discovery 3](#)
[Web 1](#)

Cancel

The **Details** tab contains all available inventory details for the host:

Overview

Details

Type

Zabbix server

Name

linux-qvvt

OS

Linux linux-qvvt 3.11.10-21-default #1 SMP Mon Jul 21 15:28:46 U

OS (Full details)

Linux version 3.11.10-21-default (geeko@buildhost) (gcc version 4.8.1 20130909 [gcc-4_8-branch revision 202388] (SUSE Linux)) #1 SMP Mon Jul 21 15:28:46 UTC 2014 (9a9565d)

MAC address A

[enp0s3] 08:00:27:62:c4:53, [enp0s3] 08:00:27:62:c4:53

Location

Head Office

Site city

Riga

Cancel

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.

3 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 system data is displayed.

System information		
Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Number of hosts (enabled/disabled/templates)	92	12 / 0 / 80
Number of items (enabled/disabled/not supported)	349	330 / 0 / 19
Number of triggers (enabled/disabled [problem/ok])	211	208 / 3 [16 / 192]
Number of users (online)	8	1
Required server performance, new values per second	7.32	

This report is also displayed as a widget in the **Dashboard**.

Displayed data

Parameter	Value	Details
<i>Zabbix server is running</i>	Status of Zabbix server: Yes - server is running No - server is not running <i>Note:</i> 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 <code>zabbix_server.conf</code> file > 0).	Location and port of Zabbix server.
<i>Number of hosts</i>	Total number of hosts configured is displayed. Templates are counted as a type of host too.	Number of monitored hosts/not monitored hosts/templates.
<i>Number of items</i>	Total number of items is displayed.	Number of monitored/disabled/unsupported items. Items on disabled hosts are counted as disabled.
<i>Number of triggers</i>	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.
<i>Number of users</i>	Total number of users configured is displayed.	Number of users online.
<i>Required server performance, new values per second</i>	The expected number of new values processed by Zabbix server per second is displayed.	<i>Required server performance</i> is an estimate and can be useful as a guideline. For precise numbers of values processed, use the <code>zabbix[wcache,values,all]</code> 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.

2 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.

Availability report
Mode By host

Zoom out
Last 30 days
Filter

Host group all
Host New host
Apply
Reset

Host	Name	Problems	Ok	Graph
New host	/etc/passwd has been changed on New host	0.4167%	99.5833%	Show
New host	Configured max number of opened files is too low on New host		100.0000%	Show
New host	Configured max number of processes is too low on New host		100.0000%	Show
New host	CPU load too high on New host for 3 minutes	2.9527%	97.0473%	Show
New host	Disk I/O is overloaded on New host	7.4999%	92.5001%	Show
New host	Free disk space is less than 20% on volume /	100.0000%		Show
New host	Free inodes is less than 20% on volume /		100.0000%	Show
New host	Host information was changed on New host	0.2778%	99.7222%	Show
New host	Host name of zabbix_agentd was changed on New host		100.0000%	Show

From the dropdown in the upper right corner you can choose the selection mode - whether to display triggers by hosts or by triggers belonging to a template.

Availability report
Mode By trigger template

Zoom out
Last 30 days
Filter

Template group all
Template Template OS Linux
Template trigger Disk I/O is overloaded on {HOST.NAME}
Host group all
Apply
Reset

Host	Name	Problems	Ok	Graph
New host	Disk I/O is overloaded on New host	7.4999%	92.5001%	Show
Zabbix server	Disk I/O is overloaded on Zabbix server	7.6502%	92.3498%	Show

Displaying 2 of 2 found

The name of the trigger is a link to the latest events of that trigger.

Using filter

You may use the filter to display narrow down the selection. Specifying a parent host group implicitly selects all nested host groups.

The filter is located below the *Availability report* bar. It can be opened and collapsed by clicking on the *Filter* tab on the left.

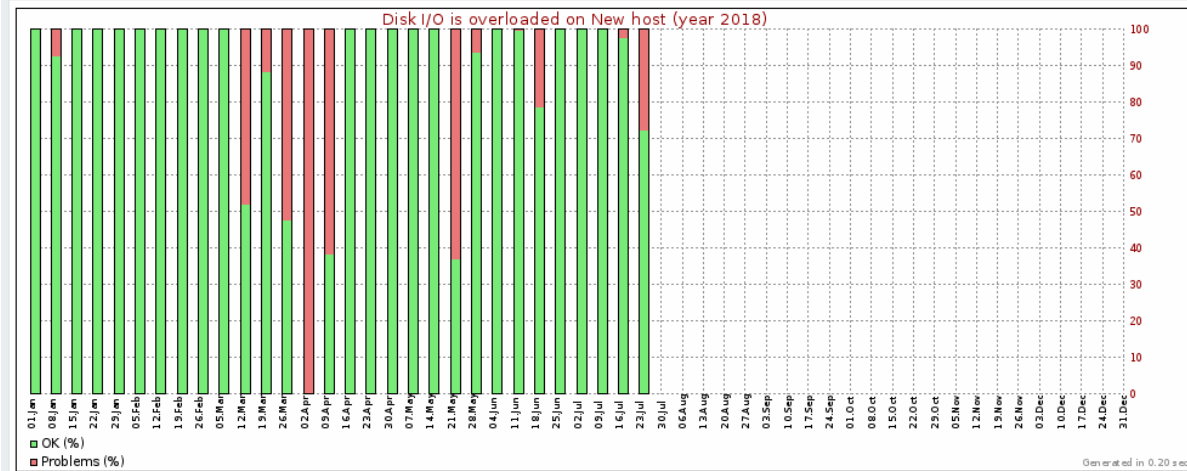
Time period selector

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.

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.

Availability report

New host Disk I/O is overloaded on New host



The green part of a bar stands for OK time and red for problem time.

3 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.

100 busiest triggers

Zoom out

Last 30 days

Filter

Host groups

type here to search

Select

Hosts

type here to search

Select

Severity

☒ Not classified

☒ Warning

☒ High

☒ Information

☒ Average

☒ Disaster

Apply

Reset

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 screens 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.

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.

4 Audit

Overview

In the *Reports* → *Audit* section users can view records of changes made in the frontend.

Audit log

In this screen the audit log of various changes made in the frontend can be seen.

Audit log

< Zoom out > This month Filter

User Select

Action Update

Resource All

Apply Reset

Time	User	IP	Resource	Action	ID	Description	Details
2018-06-12 12:48:16	Admin	192.168.3.31	Screen	Updated	16	Zabbix server	Cell changed coordinates "1,1" resource type "0"
2018-06-12 12:47:41	Admin	192.168.3.31	Screen	Updated	16	Zabbix server	Cell changed screen itemid "90" resource type "0"
2018-06-12 12:47:19	Admin	192.168.3.31	Screen	Updated	16	Zabbix server	Cell changed screen itemid "44" resource type "0"
2018-06-12 12:45:35	Admin	192.168.3.31	Screen	Updated	16	Zabbix server	Cell changed screen itemid "44" resource type "0"
2018-06-12 12:45:14	Admin	192.168.3.31	Screen	Updated	16	Zabbix server	Cell changed coordinates "0,1" resource type "2"
2018-06-12 12:44:56	Admin	192.168.3.31	Screen	Updated	16	Zabbix server	Cell changed coordinates "0,0" resource type "0"
2018-06-12 12:44:36	Admin	192.168.3.31	Screen	Updated	16	Zabbix server	Column added
2018-06-12 12:36:51	Admin	192.168.3.31	Host	Updated	10152	Zabbix server	hosts.status: 1 => 0

Displayed data:

Column	Description
Time	Timestamp of the audit record.
User	User of the activity.
IP	IP that was used in the activity.
Resource	Affected resource is displayed.
Action	Activity type is displayed - <i>Login, Logout, Added, Updated, Deleted, Enabled or Disabled</i> .
ID	ID of the affected resource is displayed.
Description	Description of the resource is displayed.
Details	Detailed information on the performed activity is displayed.

Using filter

You may use the filter to narrow down the records by user, activity type and affected resource.

The filter is located below the *Audit 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.

5 Action log

Overview

In this screen details of operations (notifications, remote commands) executed within an action are displayed.

Action log

< Zoom out
> This month
Filter

Recipient Select

Apply Reset

Time	Action	Type	Recipient	Message	Status	Info
2018-06-12 11:23:41	Auto registration	Email	Admin (Zabbix Administrator) Martins.Valkovskis@zabbix.com	Subject: Auto registration: My host Message: Host name: My host Host IP: 192.168.3.31 Agent port: 10050 Remote proxy For testing.	Sent	
2018-06-12 11:23:41	Auto registration		user (New user)	Subject: Auto registration: My host Message: Host name: My host Host IP: 192.168.3.31 Agent port: 10050 Remote proxy For testing.	Failed	

Debug

Displayed data:

Column	Description
<i>Time</i>	Timestamp of the operation.
<i>Action</i>	Name of the action causing operations is displayed. Action name is displayed since Zabbix 2.4.0.
<i>Type</i>	Operation type is displayed - <i>Email</i> or <i>Command</i> .
<i>Recipient(s)</i>	User alias, name and surname (in parenthesis) and e-mail address of the notification recipient is displayed.
<i>Message</i>	User alias, name and surname are displayed since Zabbix 2.4.0. The content of the message/remote command is displayed. A remote command is separated from the target host with a colon symbol: <code><host>:<command></code> . If the remote command is executed on Zabbix server, then the information has the following format: <code>Zabbix server:<command></code>
<i>Status</i>	Operation status is displayed: <i>In progress</i> - 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. <i>Sent</i> - notification has been sent <i>Executed</i> - command has been executed <i>Not sent</i> - action has not been completed.
<i>Info</i>	Error information (if any) regarding the action execution is displayed.

Using filter

You may use the filter to narrow down the records by the recipient of e-mail.

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.

6 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.

Notifications				
		Media type	all	Period Daily Year 2016
DAY	ADMIN (ZABBIX ADMINISTRATOR)	GUEST	USER (NEW USER)	
2016-01-01				
2016-01-02	6 (6/0/0/0)			
2016-01-03	2 (2/0/0/0)			
2016-01-04	10 (10/0/0/0)			
2016-01-05	24 (24/0/0/0)			
2016-01-06	10 (10/0/0/0)			
2016-01-07	6 (6/0/0/0)			
2016-01-08	4 (4/0/0/0)			

Each column displays totals per one system user.

4 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 visualisation and others.

1 Host groups

Overview

In the *Configuration* → *Host groups* section users can configure and maintain host groups. A host group can contain both templates and hosts.

A listing of existing host groups with their details is displayed. You can search and filter host groups by name.

Host groups				
				Create host group
				Filter
Name Apply Reset				
☐ Name	Hosts	Templates	Members	Info
☐ Zabbix servers	Hosts 1	Templates	Zabbix server	
☐ Windows servers	Hosts	Templates		
☐ Virtual machines	Hosts	Templates		
☐ Templates/Network Devices	Hosts	Templates 1	Template Net Alcatel Timetra TIMOS SNMPv2	
☐ Templates/Modules	Hosts	Templates 12	Template Module EtherLike-MIB SNMPv1, Template Module EtherLike-MIB SNMPv2, Template Module Generic SNMPv1, Template Module Generic SNMPv2, Template Module HOST-RESOURCES-MIB SNMPv1, Template Module HOST-RESOURCES-MIB SNMPv2, Template Module ICMP Ping, Template Module Interfaces Simple SNMPv1, Template Module Interfaces Simple SNMPv2, Template Module Interfaces SNMPv1, Template Module Interfaces SNMPv2, Template Module Interfaces Windows SNMPv2	
☐ Templates/Applications	Hosts	Templates 1	Template App Zabbix Server	
☐ Templates	Hosts	Templates 38	ggg, Template App HTTP Service, Template App HTTPS Service, Template App IMAP Service, Template App LDAP Service, Template App MySQL, Template App NNTP Service, Template App NTP Service, Template App POP Service, Template App SMTP Service, Template App SSH Service, Template App Telnet Service, Template App Zabbix Agent, Template App Zabbix Proxy, Template ICMP Ping, Template IPMI Intel SR1530, Template IPMI Intel SR1630, Template JMX Generic, Template JMX Tomcat, Template OS AIX, Template OS FreeBSD, Template OS HP-UX, Template OS Linux, Template OS Linux 222, Template OS Mac OS X, Template OS OpenBSD, Template OS Solaris, Template OS Windows, Template SNMP Device, Template SNMP Disks, Template SNMP Generic, Template SNMP Interfaces, Template SNMP OS Linux, Template SNMP OS Windows, Template SNMP Processors, Template Virt VMware, Template Virt VMware Guest, Template Virt VMware Hypervisor	
☐ Linux servers	Hosts 4	Templates	Server 1, Server 2, Server 3, Server 4	

Displayed data:

Column	Description
<i>Name</i>	Name of the host group. Clicking on the group name opens the host group configuration form .
<i>Hosts</i>	Number of hosts in the group (displayed in grey). Clicking on "Hosts" will, in the whole listing of hosts, filter out those that belong to the group.
<i>Templates</i>	Number of templates in the group (displayed in grey). Clicking on "Templates" will, in the whole listing of templates, filter out those that belong to the group.
<i>Members</i>	Names of group members. Template names are displayed in grey, monitored host names in blue and non-monitored host names in red. Clicking on a name will open the template/host configuration form.
<i>Info</i>	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.

2 Templates

Overview

In the *Configuration* → *Templates* section users can configure and maintain templates.

A listing of existing templates with their details is displayed.

Templates										
Group all										Create template Import
Filter										
☐ Name ▼	Applications	Items	Triggers	Graphs	Screens	Discovery	Web	Linked templates	Linked to	Tags
☐ Template VM VMware Hypervisor	Applications 6	Items 21	Triggers	Graphs	Screens	Discovery 1	Web			
☐ Template VM VMware Guest	Applications 8	Items 19	Triggers	Graphs	Screens	Discovery 3	Web			
☐ Template VM VMware	Applications 3	Items 3	Triggers	Graphs	Screens	Discovery 3	Web			
☐ Template Server Supermicro Aten SNMPv2	Applications 4	Items 11	Triggers 5	Graphs	Screens	Discovery 2	Web	Template Module Generic SNMPv2		
☐ Template Server Intel SR1630 IPMI	Applications 3	Items 11	Triggers 21	Graphs 2	Screens	Discovery	Web			
☐ Template Server Intel SR1530 IPMI	Applications 3	Items 8	Triggers 11	Graphs 2	Screens	Discovery	Web			
☐ Template Server IBM IMM SNMPv2	Applications 7	Items 14	Triggers 9	Graphs	Screens	Discovery 6	Web	Template Module Generic SNMPv2		
☐ Template Server IBM IMM SNMPv1	Applications 7	Items 14	Triggers 9	Graphs	Screens	Discovery 6	Web	Template Module Generic SNMPv1		
☐ Template Server HP iLO SNMPv2	Applications 9	Items 15	Triggers 8	Graphs	Screens	Discovery 12	Web	Template Module Generic SNMPv2		

From the dropdown to the right in the title bar you can choose whether to display all templates or only those belonging to a group.

Displayed data:

Column	Description
<i>Templates</i>	Name of the template. Clicking on the template name opens the template configuration form .
<i>Entities (Applications, Items, Triggers, Graphs, Screens, Discovery, Web)</i>	Number of the respective entities in the template (displayed in grey). Clicking on the entity name will, in the whole listing of that entity, filter out those that belong to the template.
<i>Linked templates</i>	Templates that are linked to the template, in a nested setup where the template will inherit all entities of the linked templates.
<i>Linked to</i>	The hosts and templates that the template is linked to.
<i>Tags</i>	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 an XML file, click on the *Import* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Export* - export the template to an XML 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.

Filter

As the list may contain very many templates, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list of templates. If you click on it, a filter becomes available where you can filter templates by name and directly linked templates.

Filter

Name

Linked templates

type here to search

Select

Tags

And/OrOr

tag

ContainsEquals

value

Remove

Add

Apply

Reset

Filtering is possible only by template-level tags (not inherited ones).

3 Hosts

Overview

In the *Configuration* → *Hosts* section users can configure and maintain hosts.

A listing of existing hosts with their details is displayed.

From the dropdown to the right in the *Hosts* bar you can choose whether to display all hosts or only those belonging to one particular group.

Hosts

Groupall

Create host

Import

Filter

☐	Name	Applications	Items	Triggers	Graphs	Discovery	Web	Interface	Templates	Status	Availability	Agent encryption	Info	Tags
☐	Remote proxy: My host	Applications 10	Items 43	Triggers 22	Graphs 10	Discovery 2	Web 192.168.6.87: 10050		Template OS Linux (Template App Zabbix Agent)	Enabled	ZBXSNMPJMXIPMI	NONE		
☐	Zabbix server	Applications 12	Items 83	Triggers 49	Graphs 15	Discovery 2	Web 1127.0.0.1: 10050		Template App Zabbix Server, Template OS Linux (Template App Zabbix Agent)	Enabled	ZBXSNMPJMXIPMI	NONE		

Displaying 2 of 2 found

0 selected

Enable

Disable

Export

Mass update

Delete

Displayed data:

Column	Description
Name	Name of the host. Clicking on the host name opens the host configuration form .
Elements (Applications, Items, Triggers, Graphs, Discovery, Web)	Clicking on the element name will display items, triggers etc. of the host. The number of the respective elements is displayed in gray.
Interface	The main interface of the host is displayed.
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 - <i>Enabled</i> or <i>Disabled</i> . By clicking on the status you can change it. An orange wrench icon before the host status indicates that this host is in maintenance (since Zabbix 4.2.3). Maintenance details are displayed when the mouse pointer is positioned over the icon.

Column	Description
<i>Availability</i>	Availability of the host is displayed. Four icons each represent a supported interface (Zabbix agent, SNMP, IPMI, JMX). The current status of the interface is displayed by the respective colour: Green - available Red - not available (upon mouseover, details of why the interface cannot be reached are displayed) Gray - unknown or not configured Note that active Zabbix agent items do not affect host availability.
<i>Agent encryption</i>	Encryption status for connections to the host is displayed: None - no encryption PSK - using pre-shared key Cert - using certificate
<i>Info</i>	Error information (if any) regarding the host is displayed.
<i>Tags</i>	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 an XML 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 an XML 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.

Filter

As the list may contain very many hosts, it may be needed to filter out the ones you really need.

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 name, linked templates, if they are monitored by server or by proxy, proxy name, DNS, IP or port number.

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 host status

Zabbix server sets the host availability icon to gray (unknown status) for the corresponding agent interface (Zabbix, SNMP, IMP, JMX) if:

- there are no enabled items on the interface (they were removed or disabled);
- there are only active Zabbix agent items;
- 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 host availability to unknown is done after server configuration cache synchronization. Restoring host availability (available/unavailable) on hosts monitored by proxies is done after proxy configuration cache synchronization.

See also more details about host [unreachability](#).

1 Applications

Overview

The application list for a template can be accessed from *Configuration* → *Templates* and then clicking on Applications for the respective template.

The application list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Applications for the respective host.

A list of existing applications is displayed.

The screenshot shows the 'Applications' page in Zabbix. At the top, there's a header with 'Applications' and filters for 'Group' (set to 'all') and 'Host' (set to 'Zabbix server'). A 'Create application' button is on the right. Below the header, there's a navigation bar with tabs: 'All hosts / Zabbix server', 'Enabled', 'ZBX', 'SNMP', 'JMX', 'IPMI', 'Applications 12', 'Items 71', 'Triggers 44', 'Graphs 12', 'Discovery rules 2', and 'Web scenarios 1'. The main table lists applications with checkboxes, names, and 'Items' links with counts. The applications listed are: 'APPLICATION' (header), 'Template OS Linux: CPU' (13 items), 'Template OS Linux: Filesystems' (5 items), 'Template OS Linux: General' (5 items), 'Template OS Linux: Memory' (5 items), 'Template OS Linux: Network interfaces' (2 items), 'Template OS Linux: OS' (8 items), 'Template OS Linux: Performance' (13 items), 'Template OS Linux: Processes' (2 items), 'Template OS Linux: Security' (2 items), 'Template App Zabbix Agent: Zabbix agent' (3 items), 'Zabbix frontend' (no count), and 'Template App Zabbix Server: Zabbix server' (30 items).

Displayed data:

Column	Description
<i>Application</i>	Name of the application, displayed as a blue link for directly created applications. Clicking on the application name link opens the application configuration form . If the host application belongs to a template, the template name is displayed before the application name, as a grey link. Clicking on the template link will open the application list on the template level.
<i>Items</i>	Click on Items to view the items contained in the application. The number of items is displayed in grey.
<i>Info</i>	Error information (if any) regarding the application is displayed.

To configure a new application, click on the *Create application* button in the top right-hand corner.

Mass editing options

Buttons below the list offer some mass-editing options:

- *Enable* - change application status to *Enabled*
- *Disable* - change application status to *Disabled*
- *Delete* - delete the applications

To use these options, mark the checkboxes before the respective applications, then click on the required button.

2 Items

Overview

The item list for a template can be accessed from *Configuration* → *Templates* and then clicking on Items for the respective template.

The item list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Items for the respective host.

A list of existing items is displayed.

Items Create item										
All hosts / Remote proxy: New host Enabled ZBX SNMP JMX IPMI Applications 11 Items 41 Triggers 18 Graphs 7 Discovery rules 2 Web scenarios 1 Filter										
☐	Wizard	Name	Triggers	Key	Interval	History	Trends	Type	Applications	Status
☐	...	Mounted filesystem discovery: Free disk space on / (percentage)	Triggers 1	vfs.fs.size[/,pfree]	1m	1w	365d	Zabbix agent	Filesystems	Enabled
☐	...	Mounted filesystem discovery: Used disk space on /		vfs.fs.size[/,used]	1m	1w	365d	Zabbix agent	Filesystems	Enabled
☐	...	Mounted filesystem discovery: Free disk space on /		vfs.fs.size[/,free]	1m	1w	365d	Zabbix agent	Filesystems	Enabled
☐	...	Template OS Linux: Free swap space in %	Triggers 1	system.swap.size[,pfree]	1m	1w	365d	Zabbix agent	Memory	Enabled
☐	...	Template OS Linux: Free swap space		system.swap.size[,free]	1m	1w	365d	Zabbix agent	Memory	Enabled
☐	...	Mounted filesystem discovery: Total disk space on /		vfs.fs.size[/,total]	1h	1w	365d	Zabbix agent	Filesystems	Enabled
☐	...	Template OS Linux: Total swap space		system.swap.size[,total]	1h	1w	365d	Zabbix agent	Memory	Enabled
Displaying 7 of 7 found										
0 selected Enable Disable Check now Clear history Copy Mass update Delete										

Displayed data:

Column	Description
<i>Wizard</i>	The wizard icon is a link to a wizard for creating a trigger based on the item.
<i>Host</i>	Host of the item.
<i>Name</i>	This column is displayed only if multiple hosts are selected in the filter. 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 grey 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.
<i>Triggers</i>	Moving the mouse over Triggers will display an info box displaying the triggers associated with the item.
<i>Key</i>	The number of the triggers is displayed in grey.
<i>Interval</i>	Item key is displayed.
<i>History</i>	Frequency of the check is displayed.
<i>Trends</i>	Note that passive items can also be checked immediately by pushing the Check now button.
<i>Type</i>	How many days item data history will be kept is displayed.
<i>Applications</i>	How many days item trends history will be kept is displayed.
<i>Status</i>	Item type is displayed (Zabbix agent, SNMP agent, simple check, etc).
<i>Info</i>	Item applications are displayed.
	Item status is displayed - <i>Enabled</i> , <i>Disabled</i> or <i>Not supported</i> . By clicking on the status you can change it - from Enabled to Disabled (and back); from Not supported to Disabled (and back).
	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.

To configure a new item, click on the *Create item* button in the top right-hand 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*
- *Check 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

As the list may contain very many items, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list. If you click on it, a filter becomes available where you can filter items by several properties.

Filter

Host groups
Select

Hosts

Select

Application
Select

Name

Key

Type
all

Update interval

Type of information
all

History

Trends

State
all

Status
all

Triggers
all

Template
all

Discovery
all

Apply

Reset

Subfilter affects only filtered data

APPLICATIONS

CPU 13
Filesystems 5
General 5
Memory 5
Network interfaces 3
OS 8
Performance 13
Processes 2
Security 2
Zabbix agent 3

TYPES

Zabbix agent 40
Zabbix agent (active) 3

TYPE OF INFORMATION

Character 4
Numeric (float) 14
Numeric (unsigned) 22
Text 3

STATUS

Disabled 1
Enabled 42

STATE

Normal 42
Not supported 1

TEMPLATE

Inherited items 32
Not inherited items 11

WITH TRIGGERS

Without triggers 25
With triggers 18

DISCOVERY

Discovered 7
Regular 36

HISTORY

7d 40
3m 3

INTERVAL

30s 2
1m 28
10m 2
1h 10
1d 1

Parameter	Description
<i>Host groups</i>	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
<i>Hosts</i>	Filter by one or more hosts.
<i>Application</i>	Filter by application.
<i>Name</i>	Filter by item name.
<i>Key</i>	Filter by item key.
<i>Type</i>	Filter by item type (Zabbix agent, SNMP agent, etc.).
<i>Update interval</i>	Filter by item update interval.
<i>Type of information</i>	Filter by type of information (Numeric unsigned, float, etc.).
<i>History</i>	Filter by how long item history is kept.
<i>Trends</i>	Filter by how long item trends are kept.
<i>State</i>	Filter by item state - <i>Normal</i> or <i>Not supported</i> .
<i>Status</i>	Filter by item status - <i>Enabled</i> or <i>Disabled</i> .
<i>Triggers</i>	Filter items with (or without) triggers.
<i>Template</i>	Filter items inherited (or not inherited) from a template.
<i>Discovery</i>	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. If you click on a group it gets highlighted and only the items with this parameter value remain in the list.

3 Triggers

Overview

The trigger list for a template can be accessed from *Configuration* → *Templates* and then clicking on Triggers for the respective template.

The trigger list for a host can be accessed from *Configuration* → *Hosts* and then clicking on Triggers for the respective host.

Triggers							Create trigger
All hosts / Remote proxy: My host Enabled ZBX SNMP JMX IPMI Applications 10 Items 43 Triggers 23 Graphs 10 Discovery rules 2 Web scenarios							Filter
☐	Severity	Value	Name ▲	Expression	Status	Info	Tags
☐	Warning	OK	Template OS Linux: /etc/passwd has been changed on {HOST.NAME}	{My host:vfs.file.cksum[/etc/passwd].diff(0)}>0	Enabled		
☐	Information	OK	Template OS Linux: Configured max number of opened files is too low on {HOST.NAME}	{My host:kernel.maxfiles.last(0)}<1024	Enabled		
☐	Information	OK	Template OS Linux: Configured max number of processes is too low on {HOST.NAME}	{My host:kernel.maxproc.last(0)}<256	Enabled		
☐	Warning	OK	Template OS Linux: Disk I/O is overloaded on {HOST.NAME}	{My host:system.cpu.util[,iowait].avg(5m)}>20	Enabled		
☐	Warning	PROBLEM	Mounted filesystem discovery: Free disk space is less than 20% on volume /	{My host:vfs.fs.size[/,pfree].last(0)}<20	Enabled		
☐	Warning	OK	Mounted filesystem discovery: Free inodes is less than 20% on volume /	{My host:vfs.fs.inode[/,pfree].last(0)}<20	Enabled		
☐	Information	OK	Template OS Linux: Host information was changed on {HOST.NAME}	{My host:system.uname.diff(0)}>0	Enabled		
☐	Information	OK	Template App Zabbix Agent: Host name of zabbix_agentd was changed on {HOST.NAME}	{My host:agent.hostname.diff(0)}>0	Enabled		

Displayed data:

Column	Description
<i>Severity</i>	Severity of the trigger is displayed by both name and cell background colour.
<i>Value</i>	Trigger value is displayed: OK - trigger is in OK state PROBLEM - trigger is in problem state
<i>Host</i>	Host of the trigger. This column is displayed only if multiple hosts are selected in the filter.
<i>Name</i>	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 grey 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.
<i>Expression</i>	Trigger expression is displayed. The host-item part of the expression is displayed as a link, leading to the item configuration form.
<i>Status</i>	Trigger status is displayed - <i>Enabled</i> , <i>Disabled</i> or <i>Unknown</i> . By clicking on the status you can change it - from Enabled to Disabled (and back); from Unknown to Disabled (and back).
<i>Info</i>	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.
<i>Tags</i>	If trigger contains tags, tag name and value are displayed in this column.

To configure a new trigger, click on the *Create trigger* button in the top right-hand 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. The filter is located above the table.

Parameter	Description
<i>Host groups</i>	Filter by one or more host groups. Specifying a parent host group implicitly selects all nested host groups.
<i>Hosts</i>	Filter by one or more hosts. If host groups are already selected above, host selection is limited to those groups.
<i>Name</i>	Filter by trigger name.
<i>Severity</i>	Select to filter by one or several trigger severities.
<i>State</i>	Filter by trigger state.
<i>Status</i>	Filter by trigger status.
<i>Value</i>	Filter by trigger value.
<i>Tags</i>	Filter by trigger tag name and tag value. Several conditions can be set. There are two calculation types for conditions: And/Or - all conditions must be met, conditions having same tag name will be grouped by Or condition Or - enough if one condition is met 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) When filtered, the tags specified here will be displayed first with the trigger. Macros and macro functions are supported both in tag name and tag value fields.
<i>Inherited</i>	Filter triggers inherited (or not inherited) from a template.
<i>Discovered</i>	Filter triggers discovered (or not discovered) by low-level discovery.
<i>With dependencies</i>	Filter triggers with (or without) dependencies.

4 Graphs

Overview

The custom graph list for a template can be accessed from *Configuration* → *Templates* and then clicking on Graphs for the respective template.

The custom graph list for a host can be accessed from *Configuration* → *Hosts* and then clicking on *Graphs* for the respective host.

A list of existing graphs is displayed.

Graphs

Group: Discovered hosts Host: New host Create graph

All hosts / Remote proxy: New host Enabled ZBX SNMP JMX IPMI Applications 11 Items 41 Triggers 18 Graphs 7 Discovery rules 2 Web scenarios 1

☐ Name ▲	Width	Height	Graph type
☐ Template OS Linux: CPU jumps	900	200	Normal
☐ Template OS Linux: CPU load	900	200	Normal
☐ Template OS Linux: CPU utilization	900	200	Stacked
☐ Mounted filesystem discovery: Disk space usage /	600	340	Pie
☐ Template OS Linux: Memory usage	900	200	Normal
☐ Network interface discovery: Network traffic on eth0	900	200	Normal
☐ Template OS Linux: Swap usage	600	340	Pie

Displaying 7 of 7 found

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 grey 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.

To configure a new graph, click on the *Create graph* button in the top right-hand 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.

5 Discovery rules

Overview

The list of low-level discovery rules for a template can be accessed from *Configuration* → *Templates* and then clicking on *Discovery* for the respective template.

The list of low-level discovery rules for a host can be accessed from *Configuration* → *Hosts* and then clicking on *Discovery* for the respective host.

A list of existing low-level discovery rules is displayed.

Discovery rules

Create discovery rule

All hosts / Remote proxy: My host Enabled ZBX SNMP JMX IPMI Applications 11 Items 40 Triggers 17 Graphs 7 Discovery rules 2 Web scenarios 1

☐ Name ▲	Items	Triggers	Graphs	Hosts	Key	Interval	Type	Status	Info
☐ Template OS Linux: Mounted filesystem discovery	Item prototypes 5	Trigger prototypes 2	Graph prototypes 1	Host prototypes	vfs.fs.discovery	1h	Zabbix agent	Enabled	
☐ Template OS Linux: Network interface discovery	Item prototypes 2	Trigger prototypes	Graph prototypes 1	Host prototypes	net.if.discovery	1h	Zabbix agent	Enabled	

Displaying 2 of 2 found

0 selected Enable Disable Check now Delete

Displayed data:

Column	Description
<i>Name</i>	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 grey link. Clicking on the template link will open the rule list on the template level.
<i>Items</i>	A link to the list of item prototypes is displayed. The number of existing item prototypes is displayed in grey.
<i>Triggers</i>	A link to the list of trigger prototypes is displayed. The number of existing trigger prototypes is displayed in grey.
<i>Graphs</i>	A link to the list of graph prototypes displayed. The number of existing graph prototypes is displayed in grey.
<i>Hosts</i>	A link to the list of host prototypes displayed. The number of existing host prototypes is displayed in grey.
<i>Key</i>	The item key used for discovery is displayed.
<i>Interval</i>	The frequency of performing discovery is displayed. <i>Note</i> that discovery can also be performed immediately by pushing the <i>Check now</i> button below the list.
<i>Type</i>	The item type used for discovery is displayed (Zabbix agent, SNMP agent, etc).
<i>Status</i>	Discovery rule status is displayed - <i>Enabled</i> , <i>Disabled</i> or <i>Not supported</i> . By clicking on the status you can change it - from Enabled to Disabled (and back); from Not supported to Disabled (and back).
<i>Info</i>	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.

To configure a new low-level 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 low-level discovery rule status to *Enabled*
- *Disable* - change the low-level discovery rule status to *Disabled*
- *Check 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.

6 Web scenarios

Overview

The web scenario list for a template can be accessed from *Configuration* → *Templates* and then clicking on *Web* for the respective template.

The web scenario list for a host can be accessed from *Configuration* → *Hosts* and then clicking on *Web* for the respective host.

A list of existing web scenarios is displayed. From the dropdown to the right in the *Scenarios* bar you can choose whether to display all web scenarios or only those belonging to one particular group and host. Additionally you can choose to hide disabled scenarios (or show them again) by clicking on the respective link.

Web monitoring Group: Discovered hosts Host: New host [Create web scenario](#)

All hosts / Remote proxy: New host Enabled ZBX SNMP JMX IPMI Applications 11 Items 41 Triggers 18 Graphs 7 Discovery rules 2 Web scenarios 1 Filter

Status: all Enabled Disabled

[Apply](#) [Reset](#)

☐ Name ▲	Number of steps	Interval	Attempts	Authentication	HTTP proxy	Application	Status	Info
☐ Zabbix frontend	5	1m	1	None	No	Zabbix frontend	Enabled	

Displaying 1 of 1 found

Displayed data:

Column	Description
<i>Name</i>	Name of the web scenario. Clicking on the web scenario name opens the web scenario configuration form .
<i>Number of steps</i>	The number of steps contained in the scenario.
<i>Update interval</i>	How often the scenario is performed.
<i>Attempts</i>	How many attempts for executing web scenario steps are performed.
<i>Authentication</i>	Authentication method is displayed - Basic, NTLM on None.
<i>HTTP proxy</i>	Displays HTTP proxy or 'No' if not used.
<i>Application</i>	Web scenario application is displayed.
<i>Status</i>	Web scenario status is displayed - <i>Enabled</i> or <i>Disabled</i> . By clicking on the status you can change it.
<i>Info</i>	If everything is fine, no icon is displayed in this column. If there are errors, a red square icon with a cross is displayed. Move the mouse over the icon and you will see a tooltip with the error description.

To configure a new web scenario, click on the *Create web scenario* button in the top right-hand 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.

4 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.

From the dropdown to the right in the *Maintenance periods* bar you can choose whether to display all maintenance periods or only those belonging to one particular group.

Maintenance periods Group: Discovered hosts [Create maintenance period](#)

[Filter](#)

☐ Name ▲	Type	Active since	Active till	State	Description
☐ Weekly maintenance	With data collection	2018-06-29 00:00	2019-01-01 00:00	Active	We break and fix things at this time.

Displaying 1 of 1 found

Displayed data:

Column	Description
<i>Name</i>	Name of the maintenance period. Clicking on the maintenance period name opens the maintenance period configuration form .
<i>Type</i>	The type of maintenance is displayed: <i>With data collection</i> or <i>No data collection</i>
<i>Active since</i>	The date and time when executing maintenance periods becomes active.
<i>Active till</i>	The date and time when executing maintenance periods stops being active.
<i>State</i>	The state of the maintenance period: Approaching - will become active soon Active - is active Expired - is not active any more
<i>Description</i>	Description of the maintenance period is displayed.

Name, *Type*, *Active since* and *Active till* are sortable columns that can be sorted in ascending/descending order. To sort, click on the column name.

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*.

Filter

As the list may contain a number of maintenance periods, it may be needed to filter out the ones you really need.

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 name and state.

5 Actions

Overview

In the *Configuration* → *Actions* section users can configure and maintain actions.

A listing of existing actions with their details is displayed. The actions displayed are actions assigned to the selected event source (triggers, discovery, auto-registration).

To view actions assigned to a different event source, change the source from the dropdown to the right in the *Actions* bar.

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

Displayed data:

Column	Description
<i>Name</i>	Name of the action. Clicking on the action name opens the action configuration form .
<i>Conditions</i>	Action conditions are displayed.
<i>Operations</i>	Action operations are displayed. Since Zabbix 2.2, the operation list also displays the media type (e-mail, SMS, Jabber, etc) used for notification as well as the name and surname (in parentheses after the alias) of a notification recipient. Action operation can both be a notification or a remote command depending on the selected type of operation.
<i>Status</i>	Action status is displayed - <i>Enabled</i> or <i>Disabled</i> . 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.

Filter

As the list may contain a number of actions, it may be needed to filter out the ones you really need.

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.

Filter 

Name

Status Any Enabled Disabled

Apply

Reset

6 Event correlation

Overview

In the *Configuration* → *Event correlation* section users can configure and maintain global correlation rules for Zabbix events.

Event correlation

Create correlation

Filter

☐ Name ▲	Conditions	Operations	Status
☐ Close old event	New event tag <i>State</i> = <i>Up</i> New event tag <i>Application</i> = <i>ABC</i> Old event tag <i>Application</i> = <i>ABC</i> Old event tag <i>Application</i> = new event tag <i>Application</i>	Close old events	Enabled

Displaying 1 of 1 found

Displayed data:

Column	Description
<i>Name</i>	Name of the correlation rule. Clicking on the correlation rule name opens the rule configuration form .
<i>Conditions</i>	Correlation rule conditions are displayed.
<i>Operations</i>	Correlation rule operations are displayed.

Column	Description
<i>Status</i>	Correlation rule status is displayed - <i>Enabled</i> or <i>Disabled</i> . 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.

Filter

As the list may contain a number of correlation rules, it may be needed to filter out the ones you really need.

The *Filter* link is available above the list of correlation rules. If you click on it, a filter becomes available where you can filter correlation rules by name and status.

7 Discovery

Overview

In the *Configuration* → *Discovery* section users can configure and maintain discovery rules.

A listing of existing discovery rules with their details is displayed.

Discovery rules					Create discovery rule
					Filter
☐ Name	IP range	Interval	Checks	Status	
☐ Network discovery	192.168.3.1-254	1h	HTTP, HTTPS, ICMP ping, Zabbix agent	Enabled	
Displaying 1 of 1 found					

Displayed data:

Column	Description
<i>Name</i>	Name of the discovery rule. Clicking on the discovery rule name opens the discovery rule configuration form .
<i>IP range</i>	The range of IP addresses to use for network scanning is displayed.
<i>Interval</i>	The frequency of performing discovery displayed.
<i>Checks</i>	The types of checks used for discovery are displayed.
<i>Status</i>	Action status is displayed - <i>Enabled</i> or <i>Disabled</i> . 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.

Filter

As the list may contain a number of discovery rules, it may be needed to filter out the ones you really need.

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.

Filter 

Name

Status

AnyEnabledDisabled

Apply

Reset

8 Services

Overview

In the *Configuration* → *Services* section users can configure and maintain an IT services hierarchy.

When you first open this section it only contains a *root* entry.

You can use it as a starting point of building the hierarchy of monitored infrastructure. Click on *Add child* to add services and then other services below the ones you have added.

IT services			
SERVICE	ACTION		TRIGGER
root	Add child		
▼ SLA by service	Add child		Problem, if all children have problems
Server 1	Add child	Delete	Problem, if at least one child has a problem
Server 2	Add child	Delete	Problem, if at least one child has a problem
Server 3	Add child	Delete	Problem, if at least one child has a problem
Server 4	Add child	Delete	Problem, if at least one child has a problem
Server 5	Add child	Delete	Problem, if at least one child has a problem

For details on adding services, see the [Service monitoring](#) section.

5 Administration

Overview

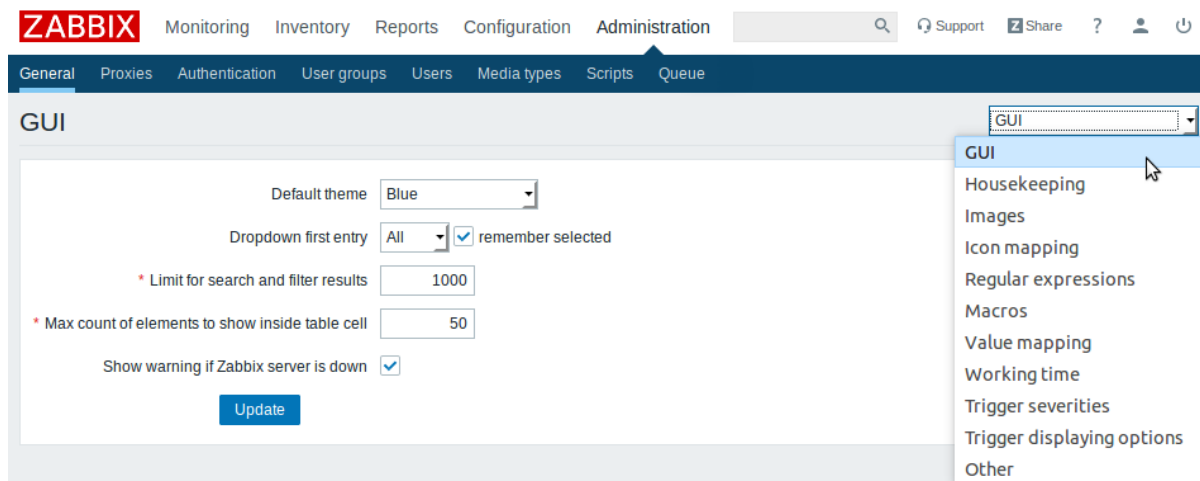
The Administration menu is for administrative functions of Zabbix. This menu is available to users of [Super Administrators](#) type only.

1 General

Overview

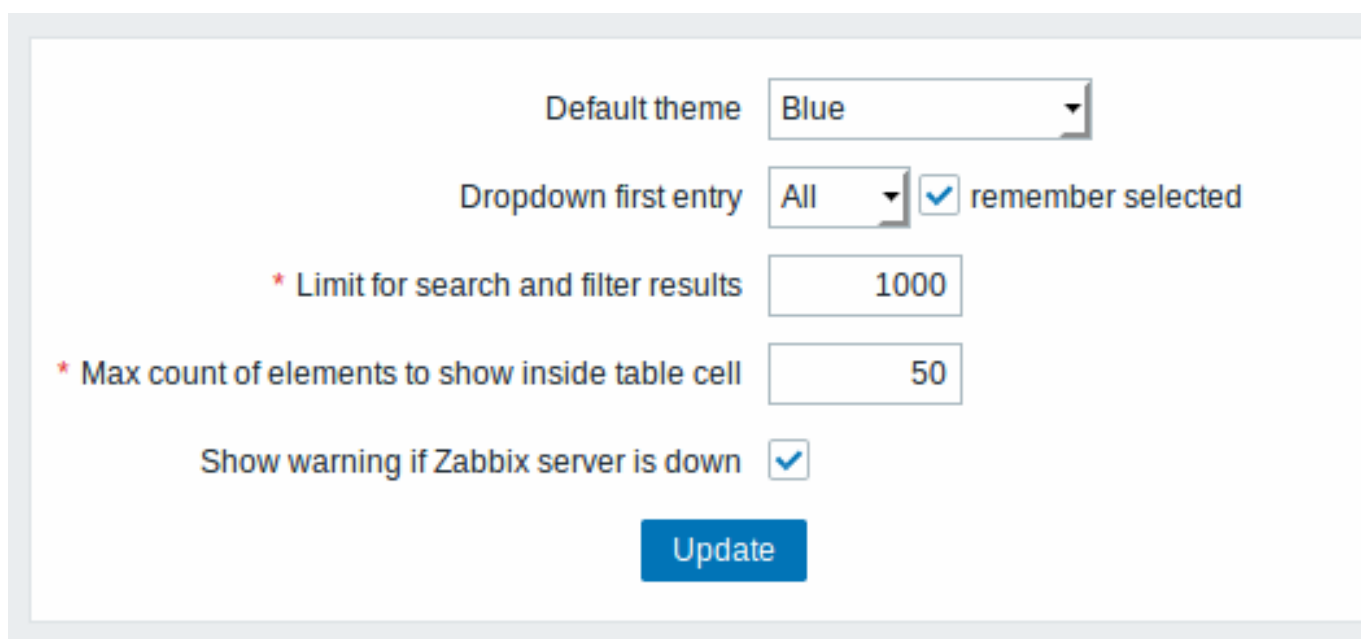
The *Administration* → *General* section contains a number of screens for setting frontend-related defaults and customizing Zabbix.

The dropdown to the right allows you to switch between different configuration screens.



1 GUI

This screen provides customization of several frontend-related defaults.



Configuration parameters:

Parameter	Description
<i>Default theme</i>	Default theme for users who have not set a specific one in their profiles.
<i>Dropdown first entry</i>	Whether first entry in element selection dropdowns should be <i>All</i> or <i>None</i> . With <i>remember selected</i> checked, the last selected element in the dropdown will be remembered (instead of the default) when navigating to another page.
<i>Limit for search and filter results</i>	Maximum amount of elements (rows) that will be displayed in a web-interface list, like, for example, in <i>Configuration</i> → <i>Hosts</i> . <i>Note:</i> 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 " <i>Displaying 1 to 50 of 50+ found</i> ". Also, if filtering is used and still there are more than 50 matches, only the first 50 will be displayed.
<i>Max count of elements to show inside table cell</i>	For entries that are displayed in a single table cell, no more than configured here will be shown.

Parameter	Description
<i>Show warning if Zabbix server is down</i>	This parameter enables a warning message to be displayed in the browser window if Zabbix server cannot be reached (may be down). The message remains visible even if the user scrolls down the page. If the mouse is moved over it, the message is temporarily hidden to reveal the contents below. This parameter is supported since Zabbix 2.0.1.

2 Housekeeper

The housekeeper is a periodical process, executed by Zabbix server. The process removes outdated information and information deleted by user.

Events and alerts

Enable internal housekeeping ☒

* Trigger data storage period

* Internal data storage period

* Network discovery data storage period

* Auto-registration data storage period

Services

Enable internal housekeeping ☒

* Data storage period

Audit

Enable internal housekeeping ☒

* Data storage period

User sessions

Enable internal housekeeping ☒

* Data storage period

History

Enable internal housekeeping ☒

Override item history period ☐

* Data storage period

Trends

Enable internal housekeeping ☒

Override item trend period ☐

* Data storage period

[Update](#)

[Reset defaults](#)

In this section housekeeping tasks can be enabled or disabled on a per-task basis separately for: events and alerts/IT services/audit/user sessions/history/trends. 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, in this case overriding the values set for individual items in *History storage period/Trend storage period* fields in **item configuration**.

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:

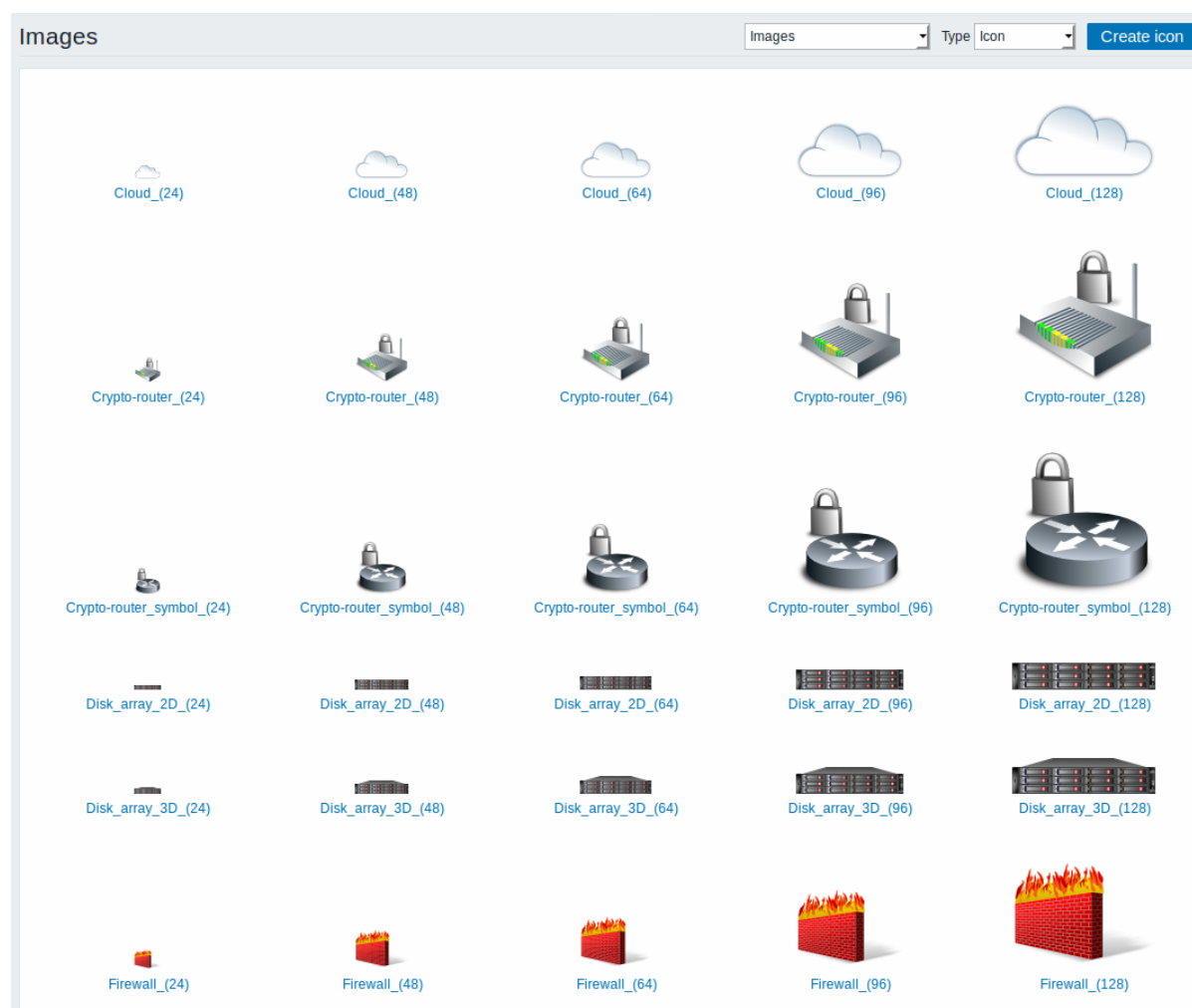
If using TimescaleDB, in order to take full advantage of TimescaleDB automatic partitioning of history and trends tables, *Override item history period* and *Override item trend period* options must be enabled. Otherwise the data kept in these tables will still be stored in partitions, however, the housekeeper will be cleaning the history and trends by deleting individual records rather than by dropping outdated partitions.

Time suffixes are supported in the period fields, e.g. 1d (one day), 1w (one week). Minimum is 1 day (1 hour for history), maximum 25 years.

Reset defaults button allows to revert any changes made.

3 Images

The Images section displays all the images available in Zabbix. Images are stored in the database.



The **Type** dropdown allows you to switch between icon and background images:

- Icons are used to display **network map** elements

- Backgrounds are used as background images of network maps

Adding image

You can add your own image by clicking on the *Create icon* or *Create background* button in the top right corner.

Image attributes:

Parameter	Description
<i>Name</i>	Unique name of an image.
<i>Upload</i>	Select the file (PNG, JPEG, GIF) from a local system to be uploaded to Zabbix. <i>Note</i> 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 value of ZBX_MAX_IMAGE_SIZE that is 1024x1024 bytes or 1 MB.

The upload of an image may fail if the image size is close to 1 MB and the `max_allowed_packet` MySQL configuration parameter is at a default of 1MB. In this case, increase the [max_allowed_packet](#) parameter.

4 Icon mapping

This section allows to create 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
<i>Name</i>	Unique name of icon map.
<i>Mappings</i>	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.

Parameter	Description
<i>Inventory field</i>	Host inventory field that will be looked into to seek a match.
<i>Expression</i>	Regular expression describing the match.
<i>Icon</i>	Icon to use if a match for the expression is found.
<i>Default</i>	Default icon to use.

5 Regular expressions

This section allows to create custom regular expressions that can be used in several places in the frontend. See [Regular expressions](#) section for details.

6 Macros

This section allows to define system-wide macros.

The screenshot shows a macro configuration interface with two rows of macro definitions. The first row has the macro '\$SNMP_COMMUNITY' mapped to the value 'public'. The second row has the macro '\$MACRO' mapped to the value 'value'. Below the rows are 'Add' and 'Update' buttons.

See [User macros](#) section for more details.

7 Value mapping

This section allows to manage value maps that are useful for human-readable representation of incoming data in Zabbix frontend.

The screenshot shows the Value mapping interface. It has a table with columns: NAME, VALUE MAP, and USED IN ITEMS. The table lists several value maps: 'Zabbix agent ping status' (1 => Up), 'Windows service 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), 'VMware VirtualMachinePowerState' (0 => poweredOff, 1 => poweredOn, 2 => suspended), 'VMware status' (0 => gray, 1 => green, 2 => yellow, 3 => red), and 'SNMP interface status (ifOperStatus)' (1 => up, 2 => down, 3 => testing, 4 => unknown, 5 => dormant, 6 => notPresent, 7 => lowerLayerDown). The 'USED IN ITEMS' column shows 'Yes' for the last three maps.

See [Value mapping](#) section for more details.

8 Working time

Working time is system-wide parameter, which defines working time. Working time is displayed as a white background in graphs, while non-working time is displayed in grey.

Use custom event status colours ☒

* Unacknowledged PROBLEM events

CC0000

☒ blinking

* Acknowledged PROBLEM events

CC0000

☒ blinking

* Unacknowledged RESOLVED events

009900

☒ blinking

* Acknowledged RESOLVED events

009900

☒ blinking

* Display OK triggers for

5m

* On status change triggers blink for

2m

Update

Reset defaults

The *Use custom event status colours* option allows to turn on the customization of colours for acknowledged/unacknowledged problems.

Also the time period for displaying OK triggers and for blinking upon trigger status change can be customized. The maximum value is 86400 seconds (24 hours). **Time suffixes** are supported in the period fields, e.g. 5m, 2h, 1d.

11 Other parameters

This section allows to configure several other frontend parameters.

* Refresh unsupported items

10m

Group for discovered hosts

Discovered hosts

Default host inventory mode

Disabled

Manual

Automatic

User group for database down message

Zabbix administrators

Log unmatched SNMP traps ☒

Update

Parameter	Description
<i>Refresh unsupported items</i>	Some items may become unsupported due to errors in user parameters or because of an item not being supported by agent. Zabbix can be configured to periodically make unsupported items active. Zabbix server will activate unsupported item every N period set here (1 day maximum). If set to 0, the automatic activation will be disabled. Note that the first attempt to reactivate the unsupported item may occur earlier than the value configured here. Time suffixes are supported, e.g. 60s, 5m, 2h, 1d. The configured value also applies to how often Zabbix proxies reactivate unsupported items.
<i>Group for discovered hosts</i>	Hosts discovered by network discovery and agent auto-registration will be automatically placed in the host group, selected here.

Parameter	Description	
<i>Default host inventory mode</i>	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/auto registration by the <i>//Set host inventory mode operation</i> . User group for database down message//	User group for sending alarm message or 'None'. Zabbix server depends on the availability of back-end 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 en-

Parameter	Description
<i>Log unmatched SNMP traps</i>	Log SNMP trap if no corresponding SNMP interfaces have been found.

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.

Proxies

Create proxy

Filter

☐ Name ▼	Mode	Encryption	Compression	Last seen (age)	Host count	Item count	Required performance (vps)	Hosts
☐ Remote proxy	Active	NONE	ON	21h 15m 15s				New host
☐ New proxy	Active	NONE	OFF	Never				

Displaying 2 of 2 found

Displayed data:

Column	Description
<i>Name</i>	Name of the proxy. Clicking on the proxy name opens the proxy configuration form .
<i>Mode</i>	Proxy mode is displayed - <i>Active</i> or <i>Passive</i> .
<i>Encryption</i>	Encryption status for connections from the proxy is displayed: None - no encryption PSK - using pre-shared key Cert - using certificate
<i>Last seen (age)</i>	The time when the proxy was last seen by the server is displayed.
<i>Host count</i>	The number of enabled hosts assigned to the proxy is displayed.
<i>Item count</i>	The number of enabled items on enabled hosts assigned to the proxy is displayed.
<i>Required performance (vps)</i>	Required proxy performance is displayed (the number of values that need to be collected per second).
<i>Hosts</i>	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:

- *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.

Filter

As the list may contain many proxies, it may be needed to filter out the ones you really need.

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.

Filter 

Name
Mode Any Active Passive

3 Authentication

Overview

In *Administration* → *Authentication* the global user authentication method to Zabbix can be specified. The available methods are internal, HTTP and LDAP authentication.

Note that the authentication method can be fine-tuned on the **user group** level.

Authentication

Authentication
HTTP settings
LDAP settings

Default authentication

Internal
LDAP

Update

By default, internal Zabbix authentication is used globally. To change:

- to HTTP - navigate to the *HTTP settings* tab and enter authentication details;
- to LDAP - select LDAP as *Default authentication* and enter authentication details in the *LDAP settings* tab.

When done, click on *Update* at the bottom of the form.

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.

Authentication
HTTP settings
LDAP settings

Enable HTTP authentication ☒

Default login form

Remove domain name

Case sensitive login ☒

Update

Configuration parameters:

Parameter	Description
<i>Enable HTTP authentication</i>	Mark the checkbox to enable HTTP authentication.
<i>Default login form</i>	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 <code>index_http.php</code> page only. If <i>Default login form</i> 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 <code>\$_SERVER</code> variable. Supported <code>\$_SERVER</code> keys are <code>PHP_AUTH_USER</code> , <code>REMOTE_USER</code> , <code>AUTH_USER</code> .
<i>Remove domain name</i>	A comma-delimited list of domain names that should be removed from the username. E.g. <code>comp,any</code> - if username is 'Admin@any', 'comp\Admin', user will be logged in as 'Admin'; if username is 'notacompany\Admin', login will be denied.
<i>Case sensitive login</i>	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 alias (e.g. Admin, admin).

Note:

In case of web server authentication all users (even with **frontend access** set to Internal) will be authenticated by the web server, not by Zabbix!

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.

While 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.

Zabbix LDAP authentication works at least with Microsoft Active Directory and OpenLDAP.

Authentication
HTTP settings
LDAP settings

Enable LDAP authentication ☒

* LDAP host

* Port
389

* Base DN

* Search attribute

Bind DN

Case sensitive login ☒

Bind password

Test authentication [must be a valid LDAP user]

* Login
Admin

* User password

Update
Test

Configuration parameters:

Parameter	Description
<i>Enable LDAP authentication</i>	Mark the checkbox to enable LDAP authentication.
<i>LDAP host</i>	Name of LDAP server. For example: ldap://ldap.zabbix.com For secure LDAP server use <i>ldaps</i> protocol. ldaps://ldap.zabbix.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.
<i>Port</i>	Port of LDAP server. Default is 389. For secure LDAP connection port number is normally 636.
<i>Base DN</i>	Not used when using full LDAP URIs. Base path to search accounts: ou=Users,ou=system (for OpenLDAP), DC=company,DC=com (for Microsoft Active Directory)
<i>Search attribute</i>	LDAP account attribute used for search: uid (for OpenLDAP), sAMAccountName (for Microsoft Active Directory)
<i>Bind DN</i>	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.

Parameter	Description
<i>Case-sensitive login</i>	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'. <i>Note</i> that with case-sensitive login disabled the login will be denied if multiple users exist in Zabbix database with similar alias (e.g. Admin, admin).
<i>Bind password</i>	LDAP password of the account for binding and searching over the LDAP server.
<i>Test authentication</i>	Header of a section for testing
<i>Login</i>	Name of a test user (which is currently logged in the 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.
<i>User password</i>	LDAP password of the test user.

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.

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.

User groups						
						Create user group
						Filter 
☐ Name ▲	#	Members	Frontend access	Debug mode	Status	
☐ Disabled	Users		System default	Disabled	Disabled	
☐ Enabled debug mode	Users 1	Admin (Zabbix Administrator)	System default	Enabled	Enabled	
☐ Guests	Users 1	guest	System default	Disabled	Enabled	
☐ Managers	Users 1	Database manager (Mr Swift)	System default	Disabled	Enabled	
☐ MySQL Administrators	Users		System default	Disabled	Enabled	
☐ No access to the frontend	Users		Disabled	Disabled	Enabled	
☐ Zabbix administrators	Users 2	Admin (Zabbix Administrator) , user (New user)	System default	Disabled	Enabled	
Displaying 7 of 7 found						

Displayed data:

Column	Description
<i>Name</i>	Name of the user group. Clicking on the user group name opens the user group configuration form .
<i>#</i>	The number of users in the group. Clicking on <i>Users</i> will display the respective users filtered out in the user list.

Column	Description
<i>Members</i>	Aliases of individual users in the user group (with name and surname in parentheses). Clicking on the alias will open the user configuration form. Users from disabled groups are displayed in red.
<i>Frontend access</i>	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.
<i>Debug mode</i>	Debug mode status is displayed - <i>Enabled</i> or <i>Disabled</i> . By clicking on the status you can change it.
<i>Status</i>	User group status is displayed - <i>Enabled</i> or <i>Disabled</i> . 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.

Filter

As the list may contain many user groups, it may be needed to filter out the ones you really need.

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.

Filter 

Name

Status Any Enabled Disabled

Apply

Reset

5 Users

Overview

In the *Administration* → *Users* section users of the system are maintained.

Users

A listing of existing users with their details is displayed.

Users										
User group All										Create user
Filter 										
☐	Alias 	Name	Surname	User type	Groups	Is online?	Login	Frontend access	Debug mode	Status
☐	Admin	Zabbix	Administrator	Zabbix Super Admin	Enabled debug mode, Zabbix administrators	Yes (2018-07-27 10:27:27)	Ok	System default	Enabled	Enabled
☐	Database manager	Mr	Swift	Zabbix User	Managers	No	Ok	System default	Disabled	Enabled
☐	guest			Zabbix User	Guests	No (2018-07-26 13:41:34)	Ok	System default	Disabled	Enabled
☐	user	New	user	Zabbix User	Zabbix administrators	No	Ok	System default	Disabled	Enabled
Displaying 4 of 4 found										

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
<i>Alias</i>	Alias of the user, used for logging into Zabbix. Clicking on the alias opens the user configuration form .
<i>Name</i>	First name of the user.
<i>Surname</i>	Second name of the user.
<i>User type</i>	User type is displayed - <i>Zabbix Super Admin</i> , <i>Zabbix Admin</i> or <i>Zabbix User</i> .
<i>Groups</i>	Groups that the user is member of are listed. Clicking on the user group name opens the user group configuration form. Disabled groups are displayed in red.
<i>Is online?</i>	The on-line status of the user is displayed - <i>Yes</i> or <i>No</i> . The time of last user activity is displayed in parentheses.
<i>Login</i>	The login status of the user is displayed - <i>Ok</i> or <i>Blocked</i> . A user can become temporarily blocked upon more than five unsuccessful login attempts. By clicking on <i>Blocked</i> you can unblock the user.
<i>Frontend access</i>	Frontend access level is displayed - <i>System default</i> , <i>Internal</i> or <i>Disabled</i> , depending on the one set for the whole user group.
<i>Debug mode</i>	Debug mode status is displayed - <i>Enabled</i> or <i>Disabled</i> , depending on the one set for the whole user group.
<i>Status</i>	User status is displayed - <i>Enabled</i> or <i>Disabled</i> , 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.

Filter

As the list may contain many users, it may be needed to filter out the ones you really need.

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 alias, name, surname and user type.

6 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.

Media types						Create media type
						Filter
☐	Name ▲	Type	Status	Used in actions	Details	Action
☐	Email	Email	Enabled	Report problems to Zabbix administrators	SMTP server: "mail.zabbix.com", SMTP helo: "zabbix.com", SMTP email: "zabbix-info@zabbix.com"	Test
☐	Jabber	Jabber	Disabled		Jabber identifier: "jabber@example.com"	Test
☐	SMS	SMS	Enabled		GSM modem: "/dev/ttyS0"	Test
						Displaying 3 of 3 found

Displayed data:

Column	Description
<i>Name</i>	Name of the media type. Clicking on the name opens the media type configuration form .
<i>Type</i>	Type of the media (e-mail, SMS, etc) is displayed.
<i>Status</i>	Media type status is displayed - <i>Enabled</i> or <i>Disabled</i> . By clicking on the status you can change it.
<i>Used in actions</i>	All actions where the media type is used directly (selected in the <i>Send only to</i> dropdown) are displayed. Clicking on the action name opens the action configuration form.
<i>Details</i>	Detailed information of the media type is displayed.
<i>Action</i>	The following action is available: Test - click to open a testing form where you can enter a recipient address and send a test message with message body and optional subject to verify that the configured media type works. See also: Testing media types

To configure a new media type, click on the *Create media type* 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*
- *Delete* - delete the media types

To use these options, mark the checkboxes before the respective media types, then click on the required button.

Filter

As the list may contain a number of media types, it may be needed to filter out the ones you really need.

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.

Testing media types

It is possible to test if a configured media type works. To do that in the media type list, click on *Test* in the last column. A testing window will open where you can enter a *Send to* recipient address and send a test message with body and optional subject by clicking on *Test*.

Test success or failure message will be displayed in the same window.

7 Scripts

Overview

In the *Administration* → *Scripts* section user-defined global scripts can be configured and maintained.

These scripts, depending on the set user permissions, then become available for execution by clicking on the host in various frontend locations (*Dashboard*, *Problems*, *Latest data*, *Maps*) and can also be run as an action operation. The scripts are executed on the Zabbix server or agent.

A listing of existing scripts with their details is displayed.

Scripts

Create script

Filter

☐ Name ▲	Type	Execute on	Commands	User group	Host group	Host access
☐ Detect operating system	Script	Server	sudo /usr/bin/nmap -O {HOST.CONN}	Zabbix administrators	All	Read
☐ MyScripts/Check disk space	Script	Agent	sleep 5 df -h	All	All	Read
☐ MyScripts/Check disk space no sleep	Script	Agent	df -h	All	All	Read
☐ MyScripts/Check disk space S	Script	Server	sleep 11 df -h	All	All	Read
☐ Ping	Script	Server	/bin/ping -c 3 {HOST.CONN}	All	All	Read
☐ Traceroute	Script	Server	/usr/bin/traceroute {HOST.CONN}	All	All	Read

Displaying 6 of 6 found

Displayed data:

Column	Description
Name	Name of the script. Clicking on the script name opens the script configuration form.
Type	Script type is displayed - <i>Script</i> or <i>IPMI</i> command.
Execute on	It is displayed whether the script will be executed on Zabbix server or agent.
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 <i>All</i> for all user groups).
Host group	The host group that the script is available for is displayed (or <i>All</i> for all host groups).
Host access	The permission level for the host group is displayed - <i>Read</i> or <i>Write</i> . 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*.

Filter

As the list may contain a number of scripts, it may be needed to filter out the ones you really need.

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.

Filter

Name

ApplyReset

Configuring a global script

* Name

Detect operating system

Type

IPMI

Script

Execute on

Zabbix agent

Zabbix server (proxy)

Zabbix server

* Commands

sudo /usr/bin/nmap -O {HOST.CONN}

Description

User group

Zabbix administrators

Host group

All

Required host permissions

Read

Write

Enable confirmation

☒

Confirmation text

Add

Cancel

Script attributes:

Parameter	Description
<i>Name</i>	Unique name of the script. Since Zabbix 2.2 the name can be prefixed with the desired path, for example, <code>Default/</code>, putting the script into the respective directory. When accessing scripts through the menu in monitoring sections, they will be organized according to the given directories. A script cannot have the same name as an existing directory (and vice versa). A script name must be unique within its directory. Unescaped script names are validated for uniqueness, i.e. "Ping" and "\Ping" cannot be added in the same folder. A single backslash escapes any symbol directly after it. For example, characters <code>'</code> and <code>\</code> can be escaped by backslash, i.e. <code>\'</code> or <code>\\</code>.
<i>Type</i>	Click the respective button to select script type - IPMI command or Script.

Parameter	Description
<i>Execute on</i>	Click the respective button to execute the script on: Zabbix agent - the script will be executed by Zabbix agent on the host Zabbix server (proxy) - the script will be executed by Zabbix server or proxy - depending on whether the host is monitored by server or proxy Zabbix server - the script will be executed by Zabbix server only The option to execute scripts on Zabbix agent is available since Zabbix 2.0 version (providing remote commands are enabled in the EnableRemoteCommands parameter in Zabbix agent configuration file).
<i>Commands</i>	Enter full path to the commands to be executed within the script. The following macros are supported in the commands: {HOST.CONN}, {HOST.IP}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}. If a macro may resolve to a value with spaces (for example, host name), don't forget to quote as needed. Since Zabbix 2.2, user macros are supported in script commands.
<i>Description</i>	Enter a description for the script.
<i>User group</i>	Select the user group that the script will be available to (or <i>All</i> for all user groups).
<i>Host group</i>	Select the host group that the script will be available for (or <i>All</i> for all host groups).
<i>Required host permissions</i>	Select the permission level for the host group - <i>Read</i> or <i>Write</i> . Only users with the required permission level will have access to executing the script.
<i>Enable confirmation</i>	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.
<i>Confirmation text</i>	Enter a custom confirmation text for the confirmation popup enabled with the checkbox above (for example, <i>Remote system will be rebooted. Are you sure?</i>). To see how the text will look like, click on <i>Test confirmation</i> next to the field. Since Zabbix 2.2, the confirmation text will expand host name macros - {HOST.HOST}, {HOST.NAME}, host connection macros - {HOST.IP}, {HOST.DNS}, {HOST.CONN} and user macros. <i>Note:</i> The macros will not be expanded when testing the confirmation message.

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 example of a script and the result window below:

```
uname
uname --non-existing-flag
/tmp/non_existing_script.sh
```

Uname

```
uname
uname --non-existing-flag
/tmp/non_existing_script.sh

Linux
uname: unrecognized option '--non-existing-flag'
Try 'uname --help' for more information.
sh: 3: /tmp/non_existing_script.sh: not found
```

Script timeout

Zabbix agent

You may encounter a situation when timeout occurs while executing a script.

See example of a script running on Zabbix agent and the result window below:

**Details ▲ Cannot execute script**

Timeout while executing a shell script. [scripts_exec.php:71 → CFrontendApiWrapper->execute CFrontendApiWrapper->callMethod() → CApiWrapper->callMethod() → CFrontendApiWrapper call_user_func_array() → CScript->execute() → CApiService::exception()] in include/classes/api

MyScripts/Check disk space

```
sleep 5
df -h
```

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 example of a script running on Zabbix server and the result window below:



Details ▲ Cannot execute script

Timeout while executing a shell script. [scripts_exec.php:71 → CFrontendApiWrapper->execute CFrontendApiWrapper->callMethod() → CApiWrapper->callMethod() → CFrontendApiWrapper call_user_func_array() → CScript->execute() → CApiService::exception() in include/classes/api

MyScripts/Check disk space S

```
sleep 11
df -h
```

It is also advised to optimize the script itself (instead of adjusting TrapperTimeout parameter to a corresponding value (in our case, > '11') by modifying the [Zabbix server configuration](#)).

8 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.

From the dropdown in the upper right corner you can select:

- queue overview by item type
- queue overview by proxy
- list of delayed items

Overview by item type

In this screen it is easy to locate if the problem is related to one or several item types.

Queue of items to be updated						
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 of items to be updated							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 of items to be updated				Details ▾
SCHEDULED CHECK	DELAYED BY	HOST	NAME	
2016-01-04 17:28:39	36s	Zabbix server 1	Zabbix busy discoverer processes, in %	
2016-01-04 17:28:40	35s	Zabbix server 1	Zabbix busy escalator processes, in %	
2016-01-04 17:28:41	34s	Zabbix server 1	Zabbix busy history syncer processes, in %	
2016-01-04 17:28:42	33s	Zabbix server 1	Zabbix busy housekeeper processes, in %	
2016-01-04 17:28:43	32s	Zabbix server 1	Zabbix busy http poller processes, in %	
2016-01-04 17:28:44	31s	Zabbix server 1	Zabbix busy icmp pinger processes, in %	

In the host column, hosts monitored by proxy are prefixed with the proxy name (since Zabbix 2.4.0).

Displayed data:

Column	Description
<i>Next check</i>	The time when the check was due is displayed.
<i>Delayed by</i>	The length of the delay is displayed.
<i>Host</i>	Host of the item is displayed.
<i>Name</i>	Name of the waiting item is displayed.

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.

2 User profile

Overview

In the user profile you can customize some Zabbix frontend features, such as the interface language, color theme, number of rows displayed in the lists etc. The changes made here will apply for the user only.

To access the user profile configuration form, click on the  user profile link in the upper right corner of Zabbix window.

Configuration

The **User** tab allows you to set various user preferences.

User
Media
Messaging

Password
Change password

Language
English (en_US)

Theme
System default

Auto-login
☒

Auto-logout
☐ 15m

* Refresh
30s

* Rows per page
50

URL (after login)

Update
Cancel

Parameter	Description
<i>Password</i>	Click on the link to display two fields for entering a new password.
<i>Language</i>	Select the interface language of your choice.
<i>Theme</i>	The php gettext extension is required for the translations to work. 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
<i>Auto-login</i>	Mark this checkbox to make Zabbix remember you and log you in automatically for 30 days. Browser cookies are used for this.
<i>Auto-logout</i>	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: * 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. In case pages refreshing data in a specific time interval (dashboards, graphs, screens, latest data, etc.) are left open session lifetime is extended, respectively disabling auto-logout feature; * If logging in with the <i>Remember me for 30 days</i> option checked.
<i>Refresh</i>	Auto-logout can accept 0, meaning that Auto-logout becomes disabled after profile settings update. 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.
<i>Rows per page</i>	Time suffixes are supported, e.g. 30s, 5m, 2h, 1d. 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.

Parameter	Description
URL (after login)	You can set a specific URL to be displayed after the login. Instead of the default <i>Monitoring → Dashboard</i> it can be, for example, the URL of <i>Monitoring → Triggers</i> .

Note:

If some language is not available for selection in the user profile it means that a locale for it is not installed on the web server. See the [link](#) at the bottom of this page to find out how to install them.

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.

Type	Send to	When active	Use if severity	Status
Email	user@company.com	1-7,00:00-24:00	N I W A H D	Enabled
Jabber	user@company.com	1-7,00:00-24:00	N I W A H D	Enabled

[Add](#)

Note:

Only **admin level** users (Admin and Super Admin) can change their own media details.

The **Messaging** tab allows you to set **global notifications**.

See also

1. [How to install additional locales to be able to select unavailable languages in the user profile](#)

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 *Host groups* or *Dashboard* pages 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**.

User
Media
Messaging

Frontend messaging ☒

Message timeout

Play sound

Trigger severity

☐ Recovery

☐ Not classified

☐ Information

☐ Warning

☐ Average

☐ High

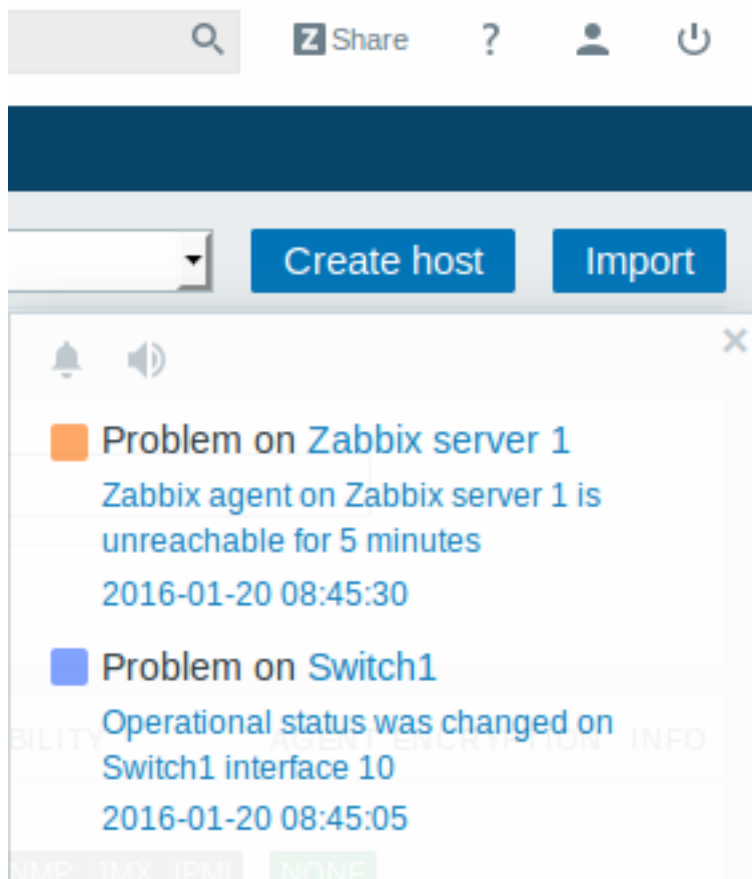
☐ Disaster

Show suppressed problems ☐

Parameter	Description
<i>Frontend messaging</i>	Mark the checkbox to enable global notifications.
<i>Message timeout</i>	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.
<i>Play sound</i>	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.
<i>Trigger severity</i>	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 <i>Recovery</i> and <i>Disaster</i> , global notifications will be displayed for the problems and the recoveries of disaster severity triggers.
<i>Show suppressed problems</i>	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 currently active alarm sound;
-  **Mute/Unmute** button switches between playing and not playing the alarm sounds.

2 Sound in browsers

Overview

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.

Default audio clips are in:

- Wave format until Zabbix 4.2.5
- MP3 format since Zabbix 4.2.6

The sounds of Zabbix frontend have been successfully tested in recent Firefox/Opera browsers on Linux and Chrome, Firefox, Internet Explorer, Opera and Safari browsers on Windows.

Attention:

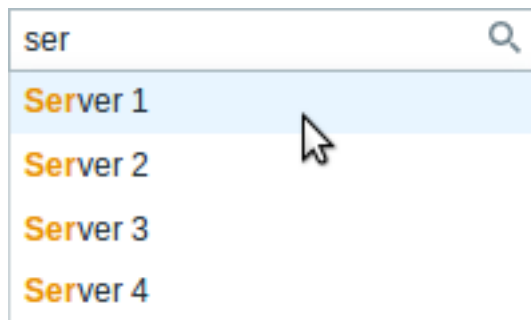
The auto play of sounds may be disabled in recent browser versions by default. In this case, you need to change this setting manually.

3 Global search

It is possible to search Zabbix frontend for hosts, host groups and templates.

The search input box is located in the upper right corner. The search can be started by pressing *Enter* or clicking on the search icon.





If there is a host that starts with the entered string, a dropdown will appear, listing all such hosts.

Starting with Zabbix 4.2.1, 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.

Properties searched

Hosts can be searched by the following properties:

- Host name
- Visible name
- IP address
- DNS name

Host groups can be searched by name. Specifying a parent host group implicitly selects all nested host groups.

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.

Search results

Search results consist of three separate blocks for hosts, host groups and templates.

Search: Zabbix server

Hosts															^
Host	IP	DNS	Latest data	Triggers	Problems	Graphs	Screens	Web	Applications	Items	Triggers	Graphs	Discovery	Web	
Zabbix server	192.168.3.31	jmsc.zabbix.lan	Latest data	Triggers	Problems	Graphs	Screens	Web	Applications 11	Items 40	Triggers 17	Graphs 7	Discovery 2	Web 1	
															Displaying 1 of 1 found
Host groups															^
Host group	Latest data		Triggers		Problems		Graphs		Web		Hosts		Templates		
Zabbix servers	Latest data		Triggers		Problems		Graphs		Web		Hosts		Templates		
															Displaying 1 of 1 found
Templates															^
Template	Applications		Items		Triggers		Graphs		Screens		Discovery		Web		
Template App Zabbix Server	Applications 1		Items 37		Triggers 32		Graphs 6		Screens 1		Discovery		Web		
															Displaying 1 of 1 found

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 [links available](#).

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 grey. **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
 - * Triggers
 - * Problems

- * Graphs
- * Host screens
- * Web scenarios
- Configuration
 - * Host properties
 - * Applications
 - * Items
 - * Triggers
 - * Graphs
 - * Discovery rules
 - * Web scenarios
- Host groups
 - Monitoring
 - * Latest data
 - * Triggers
 - * Problems
 - * Graphs
 - * Web scenarios
 - Configuration
 - * Host group properties
 - * Host group members (hosts and templates)
- Templates
 - Configuration
 - * Template properties
 - * Applications
 - * Items
 - * Triggers
 - * Graphs
 - * Template screens
 - * Discovery rules
 - * Web scenarios

4 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 the Zabbix HTML document directory on the webserver) 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');

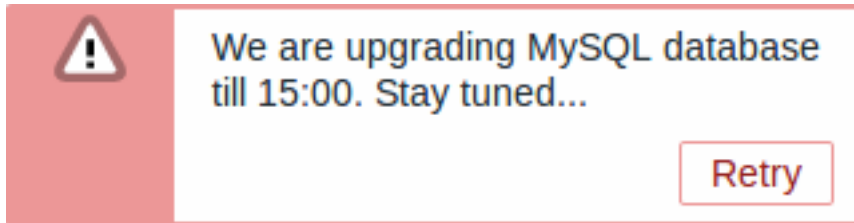
// Message shown on warning screen (optional).
$ZBX_GUI_ACCESS_MESSAGE = 'We are upgrading MySQL database till 15:00. Stay tuned...';
```

Parameter	Details
ZBX_DENY_GUI_ACCESS	Enable maintenance mode: 1 - maintenance mode is enabled, disabled otherwise

Parameter	Details
ZBX_GUI_ACCESS_IP_RANGE	Array of IP addresses, which are allowed to connect to frontend (optional). For example: <code>array('192.168.1.1', '192.168.1.2')</code>
ZBX_GUI_ACCESS_MESSAGE	A 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.

5 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:

- **sort** - sort column: clock, host, severity, name
- **sortorder** - sort order or results: DESC - descending, ASC - ascending
- **filter_set** - should be 'filter_set=1' to use "filter_*" parameters
- **filter_rst** - should be 'filter_rst=1' to reset filter elements
- **filter_show** - filter option "Show": 1 - recent problems, 2 - all, 3 - in problem state
- **filter_groupids** - filter option "Host groups": array of host groups IDs
- **filter_hostids** - filter option "Hosts": array of host IDs
- **filter_application** - filter option "Application": freeform string
- **filter_triggerids** - filter option "Triggers": array of trigger IDs
- **filter_name** - filter option "Problem": freeform string
- **filter_severity** - filter option "Minimum severity": 0 - not classified, 1 - information, 2 - warning, 3 - average, 4 - high, 5 - disaster
- **filter_age_state** - filter option "Age less than": should be 'filter_age_state=1' to enable 'filter_age'. Is used only when 'filter_show' equals 3.
- **filter_age** - filter option "Age less than": days
- **filter_inventory** - filter option "Host inventory": array of inventory fields: [field], [value]
- **filter_evaltype** - filter option "Tags", tag filtering strategy: 0 - And/Or, 2 - Or
- **filter_tags** - filter option "Tags": array of defined tags: [tag], [operator], [value]
- **filter_show_tags** - filter option "Show tags": 0 - none, 1 - one, 2 - two, 3 - three
- **filter_tag_name_format** - filter option "Tag name": 0 - full name, 1 - shortened, 2 - none
- **filter_tag_priority** - filter option "Tag display priority": comma-separated string of tag display priority
- **filter_show_suppressed** - filter option "Show suppressed problems": should be 'filter_show_suppressed=1' to show
- **filter_unacknowledged** - filter option "Show unacknowledged only": should be 'filter_unacknowledged=1' to show
- **filter_compact_view** - filter option "Compact view": should be 'filter_compact_view=1' to show
- **filter_show_timeline** - filter option "Show timeline": should be 'filter_show_timeline=1' to show
- **filter_details** - filter option "Show details": should be 'filter_details=1' to show
- **filter_highlight_row** - filter option "Highlight whole row" (use problem colour as background colour for every problem row): should be '1' to highlight; can be set only when 'filter_compact_view' is set

- from - date range start, can be 'relative' (e.g.: now-1m)
- to - date range end, can be 'relative' (e.g.: now-1m)

Fullscreen/kiosk mode

Fullscreen and kiosk modes in supported frontend pages can be activated using URL parameters. For example, in dashboards:

- /zabbix.php?action=dashboard.view&fullscreen=1 - activate fullscreen mode
- /zabbix.php?action=dashboard.view&kiosk=1 - activate kiosk mode
- /zabbix.php?action=dashboard.view&fullscreen=0 - activate normal mode

6 Definitions

Overview

While many things in the frontend can be configured using the frontend itself, some customisations 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_LOGIN_ATTEMPTS

Number of unsuccessful login attempts that is allowed to an existing system user before a login block is applied (see ZBX_LOGIN_BLOCK). By default 5 attempts. Once the set number of login attempts is tried unsuccessfully, each additional unsuccessful attempt results in a login block. Used with **internal** authentication only.

- ZBX_LOGIN_BLOCK

Number of seconds for blocking a user from accessing Zabbix frontend after a number of unsuccessful login attempts (see ZBX_LOGIN_ATTEMPTS). By default 30 seconds. Used with **internal** authentication only.

- ZBX_PERIOD_DEFAULT

Default graph period, in seconds. One hour by default.

- ZBX_MIN_PERIOD

Minimum graph period, in seconds. One minute by default.

- ZBX_MAX_PERIOD

Maximum graph period, in seconds. Two years by default since 1.6.7, one year before that.

- ZBX_HISTORY_PERIOD

The maximum period to display history data in *Latest data*, *Web*, *Overview* pages and *Data overview* screen element in seconds. By default set to 86400 seconds (24 hours). Unlimited period, if set to 0 seconds. This constant value also affects how far in the past the value is searched when {ITEM.VALUE} macro in trigger name is resolved.

- 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

- SCREEN_REFRESH_TIMEOUT (available since 2.0.4)

Used in screens and defines the timeout seconds for a screen element update. When the defined number of seconds after launching an update pass and the screen element has still not been updated, the screen element will be darkened.

Default: 30

- SCREEN_REFRESH_RESPONSIVENESS (available since 2.0.4)

Used in screens and defines the number of seconds after which query skipping will be switched off. Otherwise, if a screen element is in update status all queries on update are skipped until a response is received. With this parameter in use, another update query might be sent after N seconds without having to wait for the response to the first one.

Default: 10

- `QUEUE_DETAIL_ITEM_COUNT`

Defines retrieval limit of the total items queued. Since Zabbix 3.2.4 may be set higher than default value.

Default: 500

- `ZBX_SHOW_SQL_ERRORS` (available since 3.4.0)

Show SQL errors in the frontend, if 'true'. If changed to 'false' then SQL errors will still be displayed to all users with *Debug mode enabled*. With *Debug mode* disabled, only Zabbix Super Admin users will see SQL errors. Others will see a generic message: "SQL error. Please contact Zabbix administrator."

Default: true

- `VALIDATE_URI_SCHEMES` (available since 3.4.5)

Validate a URI against the scheme whitelist defined in `ZBX_URI_VALID_SCHEMES`.

Default: true

- `ZBX_URI_VALID_SCHEMES` (available since 3.4.2)

A comma-separated list of allowed URI schemes. Affects all places in the frontend where URIs are used, for example, in map element URLs.

Default: http,https,ftp,file,mailto,tel,ssh

- `ZBX_SHOW_TECHNICAL_ERRORS` (available since 3.4.4)

Show technical errors (PHP/SQL) to non-Zabbix Super admin users and to users that are not part of user groups with *debug mode enabled*.

Default: false

- `ZBX_SESSION_NAME` (available since 4.0.0)

String used as the name of the Zabbix frontend session cookie.

Default: `zbx_sessionid`

7 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 `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.

Note:

Since Zabbix 4.2.2, the CSS must be saved in `assets/styles`.

Step 2

Add your theme to the list of themes returned by the `Z::getThemes()` method. You can do this by overriding the `ZBase::getThemes()` method in the `Z` class. This can be done by adding the following code before the closing brace in `include/classes/core/Z.php`:

```
public static function getThemes() {
    return array_merge(parent::getThemes(), array(
        '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(), array(
        '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 colours, 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!

8 Debug mode

Overview

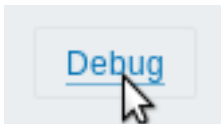
Debug mode may be used to diagnose performance problems with frontend pages.

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:

```

***** Script profiler *****
Total time: 0.249825
Total SQL time: 0.139814
SQL count: 143 (selects: 117 | executes: 26)
Peak memory usage: 6M
Memory limit: 128M

1. hostgroup.get [latest.php:124]

Parameters:          Result:
Array               Array
(
    [output] => Array    [4] => Array
        (
            [0] => groupid    [groupid] => 4
        )
)

```

Hide debug

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.

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://company.com/zabbix`, the HTTP request to call the `apiinfo.version` method may look like this:

```
POST http://company.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`.

Note:

You can use any HTTP client or a JSON-RPC testing tool to perform API requests manually, but for developing applications we suggest you use one of the [community maintained libraries](#).

Example workflow The following section will walk you through some usage examples in more detail.

Authentication Before you can access any data inside of Zabbix you'll need to log in and obtain an authentication token. This can be done using the `user.login` method. Let us suppose that you want to log in as a standard Zabbix Admin user. Then your JSON request will look like this:

```
{
  "jsonrpc": "2.0",
  "method": "user.login",
  "params": {
    "user": "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 $1",
    "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": "{Linux server:system.cpu.load[percpu,avg1].last()}>5",
    },
    {
      "description": "Too many processes on {HOST.NAME}",
      "expression": "{Linux server:proc.num[].avg(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
}
```

```
    "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.

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

Retrieve detailed service layer availability information about any service.

Service SLA calculation

Tasks

Task manager allows to check items or low-level discovery rules without config reload.

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](#) | [Maintenance API](#)

Items and applications

Define items to monitor. Create or remove applications and assign items to them.

[Item API](#) | [Application 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

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 protototype 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.

Dashboard API

Screens

Edit global and template-level screens or each screen item individually.

[Screen API](#) | [Screen item API](#) | [Template screen API](#) | [Template screen item 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](#) | [Discovery host API](#) | [Discovery 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. Configure media types and the ways users will receive alerts.

[User API](#) | [User group API](#) | [Media type API](#)

General

Change certain global configuration options.

[Icon map API](#) | [Image API](#) | [User macro API](#) | [Value map API](#)

Proxies

Manage the proxies used in your distributed monitoring setup.

Proxy 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	(<i>readonly</i>) ID of the action.
esc_period (required)	string	Default operation step duration. Must be greater than 60 seconds. Accepts seconds, time unit with suffix and user macro.
eventsource (required)	integer	(<i>constant</i>) 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.
def_longdata	string	Problem message text.
def_shortdata	string	Problem message subject.
r_longdata	string	Recovery message text.
r_shortdata	string	Recovery message subject.
ack_longdata	string	Acknowledge operation message text.
ack_shortdata	string	Acknowledge operation message subject.
status	integer	Whether the action is enabled or disabled. Possible values: 0 - (<i>default</i>) enabled; 1 - disabled.
pause_suppressed	integer	Whether to pause escalation during maintenance periods or not. Possible values: 0 - Don't pause escalation; 1 - (<i>default</i>) Pause escalation.

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	(<i>readonly</i>) ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values: 0 - send message; 1 - remote command; 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.
actionid	string	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.
esc_step_from	integer	Default: 0s. Step to start escalation from.
esc_step_to	integer	Default: 1. Step to end escalation at.
evaltype	integer	Default: 1. Operation condition evaluation method.
opcommand	object	Possible values: 0 - (<i>default</i>) AND / OR; 1 - AND; 2 - OR. Object containing the data about the command run by the operation.
opcommand_grp	array	The operation command object is described in detail below . Required for remote command operations. Host groups to run remote commands on.
opcommand_hst	array	Each object has the following properties: opcommand_grpid - (<i>string, readonly</i>) ID of the object; operationid - (<i>string</i>) ID of the operation; groupid - (<i>string</i>) ID of the host group. Required for remote command operations if opcommand_hst is not set. Host to run remote commands on.
		Each object has the following properties: opcommand_hstid - (<i>string, readonly</i>) ID of the object; operationid - (<i>string</i>) ID of the operation; hostid - (<i>string</i>) ID of the host; if set to 0 the command will be run on the current host.
		Required for remote command operations if opcommand_grp is not set.

Property	Type	Description
opconditions	array	Operation conditions used for trigger actions.
opgroup	array	The operation condition object is described in detail below. Host groups to add hosts to. Each object has the following properties: operationid - (<i>string</i>) ID of the operation; groupid - (<i>string</i>) ID of the host group.
opmessage	object	Required for "add to host group" and "remove from host group" operations. Object containing the data about the message sent by the operation. The operation message object is described in detail below.
opmessage_grp	array	Required for message operations. User groups to send messages to. Each object has the following properties: operationid - (<i>string</i>) ID of the operation; usrgrp - (<i>string</i>) 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 - (<i>string</i>) ID of the operation; userid - (<i>string</i>) ID of the user.
optemplate	array	Required for message operations if opmessage_grp is not set. Templates to link the hosts to to. Each object has the following properties: operationid - (<i>string</i>) ID of the operation; templateid - (<i>string</i>) ID of the template.
opininventory	object	Required for "link to template" and "unlink from template" operations. Inventory mode set host to. Object has the following properties: operationid - (<i>string</i>) ID of the operation; inventory_mode - (<i>string</i>) Inventory mode. Required for "Set host inventory mode" operations.

Action operation command

The operation command object contains data about the command that will be run by the operation.

Property	Type	Description
operationid	string	(<i>readonly</i>) ID of the operation.
command	string	Command to run. Required when type IN (0,1,2,3)

Property	Type	Description
type (required)	integer	Type of operation command. Possible values: 0 - custom script; 1 - IPMI; 2 - SSH; 3 - Telnet; 4 - global script.
authtype	integer	Authentication method used for SSH commands. Possible values: 0 - password; 1 - public key.
execute_on	integer	Required for SSH commands. Target on which the custom script operation command will be executed. Possible values: 0 - Zabbix agent; 1 - Zabbix server; 2 - Zabbix server (proxy).
password	string	Required for custom script commands. Password used for SSH commands with password authentication and Telnet commands.
port	string	Port number used for SSH and Telnet commands.
privatekey	string	Name of the private key file used for SSH commands with public key authentication.
publickey	string	Required for SSH commands with public key authentication. Name of the public key file used for SSH commands with public key authentication.
scriptid	string	Required for SSH commands with public key authentication. ID of the script used for global script commands.
username	string	Required for global script commands. User name used for authentication.
		Required for SSH and Telnet commands.

Action operation message

The operation message object contains data about the message that will be sent by the operation.

Property	Type	Description
operationid	string	(<i>readonly</i>) ID of the action operation.
default_msg	integer	Whether to use the default action message text and subject. Possible values: 0 - (<i>default</i>) use the data from the operation; 1 - use the data from the action.
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	(<i>readonly</i>) 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	(<i>readonly</i>) ID of the operation.
operator	integer	Condition operator. Possible values: 0 - (<i>default</i>) =.

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 actions and internal actions. It has the following properties.

Property	Type	Description
operationid	string	(<i>readonly</i>) ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values for trigger actions: 0 - send message; 1 - remote command; 11 - notify all involved. Possible values for internal actions: 0 - send message; 11 - notify all involved.
actionid	string	ID of the action that the recovery operation belongs to.
opcommand	object	Object containing the data about the command run by the recovery operation. The operation command object is described in detail above . Required for remote command operations.

Property	Type	Description
opcommand_grp	array	Host groups to run remote commands on. Each object has the following properties: opcommand_grpid - (<i>string, readonly</i>) ID of the object; operationid - (<i>string</i>) ID of the operation; groupid - (<i>string</i>) ID of the host group.
opcommand_hst	array	Required for remote command operations if opcommand_hst is not set. Host to run remote commands on. Each object has the following properties: opcommand_hstid - (<i>string, readonly</i>) ID of the object; operationid - (<i>string</i>) ID of the operation; hostid - (<i>string</i>) ID of the host; if set to 0 the command will be run on the current host.
opmessage	object	Required for remote command 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 - (<i>string</i>) ID of the operation; usrgrpid - (<i>string</i>) 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 - (<i>string</i>) ID of the operation; userid - (<i>string</i>) ID of the user. Required for message operations if opmessage_grp is not set.

Action acknowledge operation

The action acknowledge operation object defines an operation that will be performed when a problem is acknowledged. Acknowledge operations are possible for trigger actions. It has the following properties.

Property	Type	Description
operationid	string	(<i>readonly</i>) ID of the action operation.
operationtype (required)	integer	Type of operation. Possible values for trigger actions: 0 - send message; 1 - remote command; 12 - notify all involved.

Property	Type	Description
opcommand	object	Object containing the data about the command run by the recovery operation. The operation command object is described in detail above .
opcommand_grp	array	Required for remote command operations. Host groups to run remote commands on. Each object has the following properties: groupid - (<i>string</i>) ID of the host group.
opcommand_hst	array	Required for remote command operations if opcommand_hst is not set. Host to run remote commands on. Each object has the following properties: hostid - (<i>string</i>) ID of the host; if set to 0 the command will be run on the current host.
opmessage	object	Required for remote command 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	User groups to send messages to. Each object has the following properties: usrgrpId - (<i>string</i>) ID of the user group.
opmessage_usr	array	Required only for send message operations if opmessage_usr is not set. Is ignored for send acknowledge message operations. Users to send messages to. Each object has the following properties: userid - (<i>string</i>) ID of the user. Required only for send message operations if opmessage_grp is not set. Is ignored for send acknowledge 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	(<i>readonly</i>) 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	(<i>readonly</i>) 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; 15 - application; 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 auto-registration actions: 20 - proxy; 22 - host name; 24 - host metadata. Possible values for internal actions: 0 - host group; 1 - host; 13 - host template; 15 - application; 23 - event type.
value (required)	string	Value to compare with.
value2	string	Secondary value to compare with. Required for trigger actions when condition type is 26.
actionid	string	(<i>readonly</i>) 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 - <i>(default)</i> 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.

Condition	Condition name	Supported operators	Expected value
10	Discovery status	equals	Status of a discovered object. Possible values: 0 - host or service up; 1 - host or service down; 2 - host or service discovered; 3 - host or service lost.
11	Uptime or downtime duration	is greater than or equals, is less than or equals	Time indicating how long has the discovered object been in the current status in seconds.
12	Received values	equals, does not equal, is greater than or equals, is less than or equals, contains, does not contain	Value returned when performing a Zabbix agent, SNMPv1, SNMPv2 or SNMPv3 discovery check.
13	Host template	equals, does not equal	Linked template ID.
15	Application	equals, contains, does not contain	Name of the application.
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 <i>matches</i> and <i>does not match</i> in auto-registration 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.

Condition	Condition name	Supported operators	Expected value
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 <i>matches</i> and <i>does not match</i> .
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.

action.create

Description

`object action.create(object/array actions)`

This method allows to create new actions.

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.
acknowledge_operations	array	Action acknowledge 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 an action that will be run when a trigger from host "30045" that has the word "memory" in its name goes into problem state. The action must first send a message to all users in user group "7". If the event is not resolved in 4 minutes, it will run script "3" on all hosts in group "2". On trigger recovery it will notify all users who received any messages regarding the problem before. On trigger acknowledge, message with custom subject and body will be sent to all who left acknowledgements and comments via all media types.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "action.create",
  "params": {
    "name": "Trigger action",
    "eventsources": 0,
    "status": 0,
    "esc_period": "2m",
    "def_shortdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}",
    "def_longdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}\r\nLast value: {ITEM.LASTVALUE}\r\n\r\n{TRIGGER.",
    "filter": {
```

```

    "evaltype": 0,
    "conditions": [
      {
        "conditiontype": 1,
        "operator": 0,
        "value": "10084"
      },
      {
        "conditiontype": 3,
        "operator": 2,
        "value": "memory"
      }
    ]
  },
  "operations": [
    {
      "operationtype": 0,
      "esc_period": "0s",
      "esc_step_from": 1,
      "esc_step_to": 2,
      "evaltype": 0,
      "opmessage_grp": [
        {
          "usrgrp": "7"
        }
      ],
      "opmessage": {
        "default_msg": 1,
        "mediatypeid": "1"
      }
    },
    {
      "operationtype": 1,
      "esc_step_from": 3,
      "esc_step_to": 4,
      "evaltype": 0,
      "opconditions": [
        {
          "conditiontype": 14,
          "operator": 0,
          "value": "0"
        }
      ],
      "opcommand_grp": [
        {
          "groupid": "2"
        }
      ],
      "opcommand": {
        "type": 4,
        "scriptid": "3"
      }
    }
  ],
  "recovery_operations": [
    {
      "operationtype": "11",
      "opmessage": {
        "default_msg": 1
      }
    }
  ],

```

```

    "acknowledge_operations": [
      {
        "operationtype": "12",
        "opmessage": {
          "message": "Custom acknowledge operation message body",
          "subject": "Custom acknowledge operation message subject"
        }
      }
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}

```

Response:

```

{
  "jsonrpc": "2.0",
  "result": {
    "actionids": [
      "17"
    ]
  },
  "id": 1
}

```

Create a discovery action

Create an action that will link discovered hosts to template "30085".

Request:

```

{
  "jsonrpc": "2.0",
  "method": "action.create",
  "params": {
    "name": "Discovery action",
    "eventsources": 1,
    "status": 0,
    "esc_period": "0s",
    "filter": {
      "evaltype": 0,
      "conditions": [
        {
          "conditiontype": 21,
          "value": "1"
        },
        {
          "conditiontype": 10,
          "value": "2"
        }
      ]
    },
    "operations": [
      {
        "esc_step_from": 1,
        "esc_period": "0s",
        "optemplate": [
          {
            "templateid": "10091"
          }
        ],
        "operationtype": 6,
        "esc_step_to": 1
      }
    ]
  }
}

```

```

    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}

```

Response:

```

{
  "jsonrpc": "2.0",
  "result": {
    "actionids": [
      "18"
    ]
  },
  "id": 1
}

```

Using a custom expression filter

Create a trigger action that will use a custom filter condition. The action must send a message for each trigger with severity higher or equal to "Warning" for hosts "10084" and "10106". The formula IDs "A", "B" and "C" have been chosen arbitrarily.

Request:

```

{
  "jsonrpc": "2.0",
  "method": "action.create",
  "params": {
    "name": "Trigger action",
    "eventsources": 0,
    "status": 0,
    "esc_period": "2m",
    "def_shortdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}",
    "def_longdata": "{TRIGGER.NAME}: {TRIGGER.STATUS}\r\nLast value: {ITEM.LASTVALUE}\r\n\r\n{TRIGGER.",
    "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_period": "0s",
      "esc_step_from": 1,
      "esc_step_to": 2,

```

```

        "evaltype": 0,
        "opmessage_grp": [
            {
                "usrgrp": "7"
            }
        ],
        "opmessage": {
            "default_msg": 1,
            "mediatypeid": "1"
        }
    }
]
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

```

Response:

```

{
    "jsonrpc": "2.0",
    "result": {
        "actionids": [
            "18"
        ]
    },
    "id": 1
}

```

See also

- [Action filter](#)
- [Action operation](#)

Source

CAction::create() in *frontends/php/include/classes/api/services/CAction.php*.

action.delete

Description

object action.delete(array actionIds)

This method allows to delete actions.

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 *frontends/php/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.

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.
usrgrpsids	string/array	Return only actions that are configured to send messages to the given user groups.
userid	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	Returns the action filter in the <code>filter</code> property.
selectOperations	query	Return action operations in the <code>operations</code> property.
selectRecoveryOperations	query	Return action recovery operations in the <code>recoveryOperations</code> property.
selectAcknowledgeOperations	query	Return action acknowledge operations in the <code>acknowledgeOperations</code> property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>actionid</code> , <code>name</code> and <code>status</code> . These parameters being common for all get methods are described in the reference commentary .
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	

Parameter	Type	Description
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieve 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",
    "selectRecoveryOperations": "extend",
    "selectFilter": "extend",
    "filter": {
      "eventsources": 1
    }
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "actionid": "2",
      "name": "Auto discovery. Linux servers.",
      "eventsources": "1",
      "status": "1",
      "esc_period": "0s",
      "def_shortdata": "",
      "def_longdata": "",
      "r_shortdata": "",
      "r_longdata": "",
      "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": [
            {
                "operationid": "1",
                "templateid": "10001"
            }
        ]
    },
    {
        "operationid": "2",
        "actionid": "2",
        "operationtype": "4",
        "esc_period": "0s",
        "esc_step_from": "1",
        "esc_step_to": "1",
        "evaltype": "0",
        "opconditions": [],
        "opgroup": [
            {
                "operationid": "2",
                "groupid": "2"
            }
        ]
    }
],
"recoveryOperations": [
    {
        "operationid": "585",
        "actionid": "2",
        "operationtype": "11",
        "evaltype": "0",
        "opconditions": [],
        "opmessage": {
            "operationid": "585",
            "default_msg": "1",

```

```

        "subject": "{TRIGGER.STATUS}: {TRIGGER.NAME}",
        "message": "Trigger: {TRIGGER.NAME}\r\nTrigger status: {TRIGGER.STATUS}\r\nTrigger",
        "mediatypeid": "0"
    }
},
],
"acknowledgeOperations": [
    {
        "operationid": "585",
        "operationtype": "12",
        "evaltype": "0",
        "opmessage": {
            "default_msg": "1",
            "subject": "Acknowledged: {TRIGGER.NAME}",
            "message": "{USER.FULLNAME} acknowledged problem at {ACK.DATE} {ACK.TIME} with the",
            "mediatypeid": "0"
        }
    },
    {
        "operationid": "586",
        "operationtype": "0",
        "evaltype": "0",
        "opmessage": {
            "default_msg": "1",
            "subject": "Acknowledged: {TRIGGER.NAME}",
            "message": "{USER.FULLNAME} acknowledged problem at {ACK.DATE} {ACK.TIME} with the",
            "mediatypeid": "0"
        }
    },
    "opmessage_grp": [
        {
            "usrgrp": "7"
        }
    ],
    "opmessage_usr": []
},
{
    "operationid": "587",
    "operationtype": "1",
    "evaltype": "0",
    "opcommand": {
        "type": "0",
        "scriptid": "0",
        "execute_on": "0",
        "port": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "command": "notify.sh"
    },
    "opcommand_hst": [
        {
            "hostid": "0"
        }
    ],
    "opcommand_grp": []
}
],
}
],
"id": 1

```

```
}
```

See also

- [Action filter](#)
- [Action operation](#)

Source

CAction::get() in *frontends/php/include/classes/api/services/CAction.php*.

action.update

Description

`object action.update(object/array actions)`

This method allows to update existing actions.

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.
acknowledge_operations	array	Action acknowledge operations to replace existing acknowledge operations.

Return values

(object) Returns an object containing the IDs of the updated actions under the `actionids` property.

Examples

Disable action

Disable 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 *frontends/php/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.
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.

Property	Type	Description
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 acknowledgement, which generated the alert.

alert.get

Description

`integer/array alert.get(object parameters)`

The method allows to retrieve alerts according to the given parameters.

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. Refer to the event "object" property for a list of supported object types.
eventsource	integer	Default: 0 - trigger. Return only alerts generated by events of the given type. Refer to the event "source" property for a list of supported event types.
time_from	timestamp	Default: 0 - trigger events. Return only alerts that have been generated after the given time.

Parameter	Type	Description
time_till	timestamp	Return only alerts that have been generated before the given time.
selectHosts	query	Return the hosts that triggered the action operation in the <code>hosts</code> property.
selectMediatypes	query	Return the media type that was used for the message alert as an array in the <code>mediatypes</code> property.
selectUsers	query	Return the user that the message was addressed to as an array in the <code>users</code> property.
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>alertid</code> , <code>clock</code> , <code>eventid</code> , <code>mediatypeid</code> , <code>sendto</code> and <code>status</code> .
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

CAAlert::get() in *frontends/php/include/classes/api/services/CAAlert.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.

Parameters

Attention:

This method is available to unauthenticated users and must be called without the `auth` parameter in the JSON-RPC request.

(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

CAPInfo::version() in *frontends/php/include/classes/api/services/CAPInfo.php*.

Application

This class is designed to work with applications.

Object references:

- [Application](#)

Available methods:

- [application.create](#) - creating new applications
- [application.delete](#) - deleting applications
- [application.get](#) - retrieving application
- [application.massadd](#) - updating application
- [application.update](#) - adding items to applications

> Application object

The following objects are directly related to the application API.

Application

The application object has the following properties.

Property	Type	Description
applicationid	string	(<i>readonly</i>) ID of the application.
hostid (required)	string	ID of the host that the application belongs to.
name (required)	string	Cannot be updated. Name of the application
flags	integer	(<i>readonly</i>) Origin of the application. Possible values: 0 - a plain application; 4 - a discovered application.
templateids	array	(<i>readonly</i>) IDs of the parent template applications.

application.create

Description

object application.create(object/array applications)

This method allows to create new applications.

Parameters

(object/array) Applications to create.

The method accepts applications with the **standard application properties**.

Return values

(object) Returns an object containing the IDs of the created applications under the `applicationids` property. The order of the returned IDs matches the order of the passed applications.

Examples

Creating an application

Create an application to store SNMP items.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "application.create",
  "params": {
    "name": "SNMP Items",
    "hostid": "10050"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "applicationids": [
      "356"
    ]
  },
  "id": 1
}
```

Source

`CApplication::create()` in *frontends/php/include/classes/api/services/CApplication.php*.

application.delete

Description

`object application.delete(array applicationIds)`

This method allows to delete applications.

Parameters

(array) IDs of the applications to delete.

Return values

(object) Returns an object containing the IDs of the deleted applications under the `applicationids` property.

Examples

Deleting multiple applications

Delete two applications.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "application.delete",
  "params": [
    "356",
    "358"
  ],
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

```

    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}

```

Response:

```

{
  "jsonrpc": "2.0",
  "result": {
    "applicationids": [
      "356",
      "358"
    ]
  },
  "id": 1
}

```

Source

CApplication::delete() in *frontends/php/include/classes/api/services/CApplication.php*.

application.get

Description

integer/array application.get(object parameters)

The method allows to retrieve applications according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
applicationids	string/array	Return only applications with the given IDs.
groupids	string/array	Return only applications that belong to hosts from the given host groups.
hostids	string/array	Return only applications that belong to the given hosts.
inherited	boolean	If set to true return only applications inherited from a template.
itemids	string/array	Return only applications that contain the given items.
templated	boolean	If set to true return only applications that belong to templates.
templateids	string/array	Return only applications that belong to the given templates.
selectHost	query	Return the host that the application belongs to in the host property.
selectItems	query	Return the items contained in the application in the items property.
selectDiscoveryRule	query	Return the LLD rule that created the application in the discoveryRule property.
selectApplicationDiscovery	query	Return the application discovery object in the applicationDiscovery property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: applicationid and name. 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	

Parameter	Type	Description
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving applications from a host

Retrieve all applications from a host sorted by name.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "application.get",
  "params": {
    "output": "extend",
    "hostids": "10001",
    "sortfield": "name"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "applicationid": "13",
      "hostid": "10001",
      "name": "CPU",
      "templateids": []
    },
    {
      "applicationid": "5",
      "hostid": "10001",
      "name": "Filesystems",
      "templateids": []
    },
    {
      "applicationid": "21",
      "hostid": "10001",
      "name": "General",
      "templateids": []
    },
    {
      "applicationid": "15",
      "hostid": "10001",
      "name": "Memory",
      "templateids": []
    }
  ],
}
```

```
    "id": 1
}
```

See also

- [Host](#)
- [Item](#)

Source

CApplication::get() in *frontends/php/include/classes/api/services/CApplication.php*.

application.massadd

Description

object application.massadd(object parameters)

This method allows to simultaneously add multiple items to the given applications.

Parameters

(object) Parameters containing the IDs of the applications to update and the items to add to the applications.

The method accepts the following parameters.

Parameter	Type	Description
applications (required)	array/object	Applications to be updated. The applications must have the <code>applicationid</code> property defined.
items	array/object	Items to add to the given applications. The items must have the <code>itemid</code> property defined.

Return values

(object) Returns an object containing the IDs of the updated applications under the `applicationids` property.

Examples

Adding items to multiple applications

Add the given items to two applications.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "application.massadd",
  "params": {
    "applications": [
      {
        "applicationid": "247"
      },
      {
        "applicationid": "246"
      }
    ],
    "items": [
      {
        "itemid": "22800"
      },
      {
        "itemid": "22801"
      }
    ]
  },
}
```

```
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "applicationids": [
      "247",
      "246"
    ]
  },
  "id": 1
}
```

See also

- [Item](#)

Source

CApplication::massAdd() in *frontends/php/include/classes/api/services/CApplication.php*.

application.update

Description

object application.update(object/array applications)

This method allows to update existing applications.

Parameters

(object/array) [Application properties](#) to be updated.

The applicationid property must be defined for each application, 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 applications under the applicationids property.

Examples

Changing the name of an application

Change the name of the application to "Processes and performance".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "application.update",
  "params": {
    "applicationid": "13",
    "name": "Processes and performance"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "applicationids": [
      "13"
    ]
  }
}
```

```

    },
    "id": 1
}

```

Source

CApplication::update() in *frontends/php/include/classes/api/services/CApplication.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.export

Description

`string configuration.export(object parameters)`

This method allows to export configuration data as a serialized string.

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: json - JSON; xml - XML.
options (required)	object	Objects to be exported. The options object has the following parameters: 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; screens - (array) IDs of screens to export; templates - (array) IDs of templates to export; valueMaps - (array) IDs of value maps to export;

Return values

(string) Returns a serialized string containing the requested configuration data.

Examples

Exporting a host

Export the configuration of a host as an XML string.

Request:

```

{
    "jsonrpc": "2.0",
    "method": "configuration.export",
    "params": {
        "options": {
            "hosts": [

```



```

        "10161"
    ],
    },
    "format": "xml"
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

```

Response:

```

{
  "jsonrpc": "2.0",
  "result": "<?xml version='1.0' encoding='UTF-8'>\n<zabbix_export><version>4.0</version><date>2018
  "id": 1
}

```

Source

CConfiguration::export() in *frontends/php/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.

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: json - JSON; xml - XML.
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
applications	object	Rules on how to import applications. Supported parameters: createMissing - (boolean) if set to true, new applications will be created; default: false; deleteMissing - (boolean) if set to true, applications not present in the imported data will be deleted from the database; default: false.

Parameter	Type	Description
discoveryRules	object	Rules on how to import LLD rules. Supported parameters: <code>createMissing</code> - (boolean) if set to true, new LLD rules will be created; default: false; <code>updateExisting</code> - (boolean) if set to true, existing LLD rules will be updated; default: false; <code>deleteMissing</code> - (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: <code>createMissing</code> - (boolean) if set to true, new graphs will be created; default: false; <code>updateExisting</code> - (boolean) if set to true, existing graphs will be updated; default: false; <code>deleteMissing</code> - (boolean) if set to true, graphs not present in the imported data will be deleted from the database; default: false.
groups	object	Rules on how to import host groups. Supported parameters: <code>createMissing</code> - (boolean) if set to true, new host groups will be created; default: false.
hosts	object	Rules on how to import hosts. Supported parameters: <code>createMissing</code> - (boolean) if set to true, new hosts will be created; default: false; <code>updateExisting</code> - (boolean) if set to true, existing hosts will be updated; default: false.
httptests	object	Rules on how to import web scenarios. Supported parameters: <code>createMissing</code> - (boolean) if set to true, new web scenarios will be created; default: false; <code>updateExisting</code> - (boolean) if set to true, existing web scenarios will be updated; default: false; <code>deleteMissing</code> - (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: <code>createMissing</code> - (boolean) if set to true, new images will be created; default: false; <code>updateExisting</code> - (boolean) if set to true, existing images will be updated; default: false.
items	object	Rules on how to import items. Supported parameters: <code>createMissing</code> - (boolean) if set to true, new items will be created; default: false; <code>updateExisting</code> - (boolean) if set to true, existing items will be updated; default: false; <code>deleteMissing</code> - (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.
screens	object	Rules on how to import screens. Supported parameters: createMissing - (boolean) if set to true, new screens will be created; default: false; updateExisting - (boolean) if set to true, existing screens will be updated; default: false.
templateLinkage	object	Rules on how to import template links. Supported parameters: createMissing - (boolean) if set to true, new links between templates and host will be created; 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.
templateScreens	object	Rules on how to import template screens. Supported parameters: createMissing - (boolean) if set to true, new template screens will be created; default: false; updateExisting - (boolean) if set to true, existing template screens will be updated; default: false;
triggers	object	deleteMissing - (boolean) if set to true, template screens not present in the imported data will be deleted from the database; default: false. 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 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.

Return values

(boolean) Returns true if importing has been successful.

Examples

Importing hosts and items

Import the host and items contained in the XML string. If any items in XML 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": {
      "applications": {
        "createMissing": true,
        "deleteMissing": false
      },
      "valueMaps": {
        "createMissing": true,
        "updateExisting": false
      },
      "hosts": {
        "createMissing": true,
        "updateExisting": true
      },
      "items": {
        "createMissing": true,
        "updateExisting": true,
        "deleteMissing": true
      }
    },
    "source": "<?xml version=\"1.0\" encoding=\"UTF-8\"?><zabbix_export><version>4.0</version><date>2020-01-01T00:00:00</date><groups><group name=\"Zabbix\"><item name=\"Zabbix\"></item></group></zabbix_export>"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": true,
  "id": 1
}
```

Source

CConfiguration::import() in *frontends/php/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
<code>correlationid</code>	string	(<i>readonly</i>) ID of the correlation.
<code>name</code> (required)	string	Name of the correlation.
<code>description</code>	string	Description of the correlation.
<code>status</code>	integer	Whether the correlation is enabled or disabled. Possible values are: 0 - (<i>default</i>) enabled; 1 - disabled.

Correlation operation

The correlation operation object defines an operation that will be performed when a correlation is executed. It has the following properties.

Property	Type	Description
<code>type</code> (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
<code>evaltype</code> (required)	integer	Filter condition evaluation method. Possible values: 0 - and/or; 1 - and; 2 - or; 3 - custom expression.
<code>conditions</code> (required)	array	Set of filter conditions to use for filtering results.
<code>eval_formula</code>	string	(<i>readonly</i>) Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its <code>formulaid</code> . The value of <code>eval_formula</code> is equal to the value of <code>formula</code> for filters with a custom expression.
<code>formula</code>	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 <code>formulaid</code> . 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	Old 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.
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.

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 *frontends/php/include/classes/api/services/CCorrelation.php*.

correlation.delete

Description

object correlation.delete(array correlationids)

This method allows to delete correlations.

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": {
        "correlaionids": [
            "1",
            "2"
        ]
    },
    "id": 1
}

```

Source

CCorrelation::delete() in *frontends/php/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.

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	Returns the correlation filter in the <code>filter</code> property.
selectOperations	query	Return correlation operations in the <code>operations</code> property.
sortfield	string/array	Sort the result by the given properties.
		Possible values are: <code>correlationid</code> , <code>name</code> and <code>status</code> .
countOutput	boolean	These parameters being common for all <code>get</code> 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 `frontends/php/include/classes/api/services/CCorrelation.php`.

correlation.update

Description

`object correlation.update(object/array correlations)`

This method allows to update existing correlations.

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
<code>filter</code>	object	Correlation filter object to replace the current filter.
<code>operations</code>	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"
        }
      ]
    }
  },
  "id": 1
}
```

```

    "auth": "343baad4f88b4106b9b5961e77437688",
    "id": 1
}

```

Response:

```

{
  "jsonrpc": "2.0",
  "result": {
    "correlationids": [
      "1"
    ]
  },
  "id": 1
}

```

See also

- [Correlation filter](#)
- [Correlation operation](#)

Source

CCorrelation::update() in *frontends/php/include/classes/api/services/CCorrelation.php*.

Dashboard

This class is designed to work with dashboards.

Object references:

- [Dashboard](#)
- [Dashboard widget](#)
- [Dashboard widget field](#)
- [Dashboard user group](#)
- [Dashboard user](#)

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	<i>(readonly)</i> 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 - <i>(default)</i> private dashboard.		

Dashboard widget

The dashboard widget object has the following properties:

Property	Type	Description
widgetid	string	(<i>readonly</i>) ID of the dashboard widget.
type (required)	string	Type of the dashboard widget. Possible values: actionlog - Action log; clock - Clock; dataover - Data overview; discovery - Discovery status; favgraphs - Favourite graphs; favmaps - Favourite maps; favscreens - Favourite screens; graph - Graph (classic); problemhosts - Problem hosts; map - Map; navtree - Map Navigation Tree; plaintext - Plain text; problems - Problems; svggraph - Graph; systeminfo - System information; problemsbysv - Problems by severity; 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.
y	integer	Valid values range from 0 to 11. A vertical position from the top of the dashboard.
width	integer	Valid values range from 0 to 62. The widget width.
height	integer	Valid values range from 1 to 12. The widget height.
fields	array	Valid values range from 2 to 32. 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; 6 - Graph; 8 - Map;
name	string	Widget field name.
value (required)	mixed	Widget field value depending of type.

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.create

Description

object dashboard.create(object/array dashboards)

This method allows to create new dashboards.

Parameters

(object/array) Dashboards to create.

Additionally to the **standard dashboard properties**, the method accepts the following parameters.

Parameter	Type	Description
widgets	array	Dashboard widgets to be created for the dashboard.
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).

Request:

```
{
  "jsonrpc": "2.0",
  "method": "dashboard.create",
  "params": {
    "name": "My dashboard",
```

```

    "widgets": [
      {
        "type": "problems",
        "x": 0,
        "y": 0,
        "width": 6,
        "height": 5,
        "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 widget](#)
- [Dashboard widget field](#)
- [Dashboard user](#)
- [Dashboard user group](#)

Source

CDashboard::create() in *frontends/php/include/classes/api/services/CDashboard.php*.

dashboard.delete

Description

object dashboard.delete(array dashboardids)

This method allows to delete dashboards.

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 *frontends/php/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.

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.
selectWidgets	query	Return the dashboard widgets that are used in the dashboard with in widgets property.

Parameter	Type	Description
selectUsers	query	Returns users that the dashboard is shared with in <code>users</code> property.
selectUserGroups	query	Returns user groups that the dashboard is shared with in <code>userGroups</code> property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible value is: <code>dashboardid</code> . 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",
    "selectWidgets": "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",
      "users": [],
    }
  ]
}
```

```

"userGroups": [],
"widgets": [
  {
    "widgetid": "9",
    "type": "systeminfo",
    "name": "",
    "x": "6",
    "y": "8",
    "width": "6",
    "height": "5",
    "fields": []
  },
  {
    "widgetid": "8",
    "type": "problemsbysv",
    "name": "",
    "x": "6",
    "y": "4",
    "width": "6",
    "height": "4",
    "fields": []
  },
  {
    "widgetid": "7",
    "type": "problemhosts",
    "name": "",
    "x": "6",
    "y": "0",
    "width": "6",
    "height": "4",
    "fields": []
  },
  {
    "widgetid": "6",
    "type": "discovery",
    "name": "",
    "x": "3",
    "y": "9",
    "width": "3",
    "height": "4",
    "fields": []
  },
  {
    "widgetid": "5",
    "type": "web",
    "name": "",
    "x": "0",
    "y": "9",
    "width": "3",
    "height": "4",
    "fields": []
  },
  {
    "widgetid": "4",
    "type": "problems",
    "name": "",
    "x": "0",
    "y": "3",
    "width": "6",
    "height": "6",
    "fields": []
  },

```

```

        {
            "widgetid": "3",
            "type": "favmaps",
            "name": "",
            "x": "4",
            "y": "0",
            "width": "2",
            "height": "3",
            "fields": []
        },
        {
            "widgetid": "2",
            "type": "favscreens",
            "name": "",
            "x": "2",
            "y": "0",
            "width": "2",
            "height": "3",
            "fields": []
        },
        {
            "widgetid": "1",
            "type": "favgraphs",
            "name": "",
            "x": "0",
            "y": "0",
            "width": "2",
            "height": "3",
            "fields": []
        }
    ]
},
{
    "dashboardid": "2",
    "name": "My dashboard",
    "userid": "1",
    "private": "1",
    "users": [
        {
            "userid": "4",
            "permission": "3"
        }
    ],
    "userGroups": [
        {
            "usrgrpid": "7",
            "permission": "2"
        }
    ],
    "widgets": [
        {
            "widgetid": "10",
            "type": "problems",
            "name": "",
            "x": "0",
            "y": "0",
            "width": "6",
            "height": "5",
            "fields": [
                {
                    "type": "2",
                    "name": "groupids",

```

```

    "value": "4"
  }
]
}
],
"id": 1
}

```

See also

- [Dashboard widget](#)
- [Dashboard widget field](#)
- [Dashboard user](#)
- [Dashboard user group](#)

Source

CDashboard::get() in *frontends/php/include/classes/api/services/CDashboard.php*.

dashboard.update

Description

object dashboard.update(object/array dashboards)

This method allows to update existing dashboards.

Parameters

(object/array) Dashboard properties to be updated.

The dashboardid property must be defined for each dashboard, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the [standard dashboard properties](#), the method accepts the following parameters.

Parameter	Type	Description
widgets	array	Dashboard widgets to replace existing dashboard widgets. Dashboard widgets are updated by widgetid property. Widgets without widgetid property will be created.
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
}
```

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 widget](#)
- [Dashboard widget field](#)
- [Dashboard user](#)
- [Dashboard user group](#)

Source

CDashboard::update() in *frontends/php/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.

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 the discovery rule that detected the host as an array in the <code>drules</code> property.
selectDServices	query	Return the discovered services running on the host in the <code>dservices</code> property.
limitSelects	integer	Supports count. Limits the number of records returned by subselects.
sortfield	string/array	Applies to the following subselects: <code>selectDServices</code> - results will be sorted by <code>dserviceid</code> . Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>dhostid</code> and <code>druleid</code> . 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

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 *frontends/php/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
<code>dserviceid</code>	string	ID of the discovered service.
<code>dcheckid</code>	string	ID of the discovery check used to detect the service.
<code>dhostid</code>	string	ID of the discovered host running the service.
<code>dns</code>	string	DNS of the host running the service.
<code>ip</code>	string	IP address of the host running the service.
<code>lastdown</code>	timestamp	Time when the discovered service last went down.
<code>lastup</code>	timestamp	Time when the discovered service last went up.
<code>port</code>	integer	Service port number.
<code>status</code>	integer	Status of the service.
		Possible values: 0 - service up; 1 - service down.
<code>value</code>	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.

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 the discovery rule that detected the service as an array in the <code>drules</code> property.
selectDHosts	query	Return the discovered host that service belongs to as an array in the <code>dhosts</code> property.
selectHosts	query	Return the hosts with the same IP address and proxy as the service in the <code>hosts</code> property.
limitSelects	integer	Supports count. Limits the number of records returned by subselects.
sortfield	string/array	Applies to the following subselects: <code>selectHosts</code> - result will be sorted by <code>hostid</code> . Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>dserviceid</code> , <code>dhostid</code> and <code>ip</code> . These parameters being common for all <code>get</code> 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

CDServic::get() in *frontends/php/include/classes/api/services/CDServic.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.
snmp_community	string	Default: 0. SNMP community.
snmpv3_authpassphrase	string	Required for SNMPv1 and SNMPv2 agent checks. Auth passphrase used for SNMPv3 agent checks with security level set to <i>authNoPriv</i> or <i>authPriv</i> .
snmpv3_authprotocol	integer	Authentication protocol used for SNMPv3 agent checks with security level set to <i>authNoPriv</i> or <i>authPriv</i> . Possible values: 0 - (default) MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 checks.
snmpv3_privpassphrase	string	Priv passphrase used for SNMPv3 agent checks with security level set to <i>authPriv</i> .
snmpv3_privprotocol	integer	Privacy protocol used for SNMPv3 agent checks with security level set to <i>authPriv</i> . Possible values: 0 - (default) DES; 1 - AES.
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.

Property	Type	Description
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 - <i>(default)</i> 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 - <i>(default)</i> DNS; 2 - IP; 3 - discovery value of this check.
name_source	integer	Source for visible name. Possible values: 0 - <i>(default)</i> 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.

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.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: dcheckid and druleid. 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 *frontends/php/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
<code>druleid</code>	string	(<i>readonly</i>) ID of the discovery rule.
<code>iprange</code> (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.
<code>name</code> (required)	string	Name of the discovery rule.
<code>delay</code>	string	Execution interval of the discovery rule. Accepts seconds, time unit with suffix and user macro. Default: 1h.
<code>nextcheck</code>	timestamp	(<i>readonly</i>) Time when the discovery rule will be executed next.
<code>proxy_hostid</code>	string	ID of the proxy used for discovery.
<code>status</code>	integer	Whether the discovery rule is enabled. Possible values: 0 - (<i>default</i>) enabled; 1 - disabled.

`drule.create`

Description

`object drule.create(object/array $discoveryRules)`

This method allows to create new discovery rules.

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 *frontends/php/include/classes/api/services/CDRule.php*.

drule.delete

Description

`object drule.delete(array discoveryRuleIds)`

This method allows to delete discovery rules.

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 *frontends/php/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.

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.

Parameter	Type	Description
selectDChecks	query	Return discovery checks used by the discovery rule in the dchecks property.
selectDHosts	query	Supports count. Return the discovered hosts that the discovery rule created in the dhosts property.
limitSelects	integer	Supports count. Limits the number of records returned by subselects.
sortfield	string/array	Applies to the following subselects: selectDChecks - results will be sorted by dcheckid; selectDHosts - results will be sorted by dhostsid. Sort the result by the given properties.
countOutput	boolean	Possible values are: druleid and name. 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 *frontends/php/include/classes/api/services/CDRule.php*.

drule.update

Description

object drule.update(object/array discoveryRules)

This method allows to update existing discovery rules.

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 *frontends/php/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 auto-registration;
- 3 - internal event.

Property	Type	Description
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 auto-registration events: 3 - auto-registered host. Possible values for internal events: 0 - trigger; 4 - item; 5 - LLD rule.
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.
value	integer	State of the related object. Possible values for trigger 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.
severity	integer	This parameter is not used for active agent auto-registration events. 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	Problem event ID who generated OK event
correlationid	string	Correlation ID
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.

Event tag

The event tag object has the following properties.

Property	Type	Description
tag	string	Event tag name.
value	string	Event tag value.

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.
- Add message.
- Change event severity. If event already has same severity, 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 event, it should be allowed in trigger.

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 are acceptable. Possible values: 1 - close problem; 2 - acknowledge event; 4 - add message; 8 - change severity.
message	string	Text of the message. Required , if action contains 'add message' flag.
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.

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 multiple event's severity 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 *frontends/php/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.

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.
applicationids	string/array	Return only events created by objects that belong to the given applications. Applies only if object is trigger or item.
source	integer	Return only events with the given type. Refer to the event object page for a list of supported event types.
object	integer	Default: 0 - trigger events. Return only events created by objects of the given type. Refer to the event object page for a list of supported object types.
acknowledged	boolean	Default: 0 - trigger.
suppressed	boolean	If set to <code>true</code> return only acknowledged events. <code>true</code> - return only suppressed events; <code>false</code> - return events in the normal state.
severities	integer/array	Return only events with given event severities. Applies only if object is trigger.
evaltype	integer	Rules for tag searching.
tags	array of objects	Possible values: 0 - (default) And/Or; 2 - Or. 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.
eventid_from	string	Possible operator types: 0 - (default) Like; 1 - Equal. 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.

Parameter	Type	Description
problem_time_from	timestamp	Returns only events that were in the problem state starting with <code>problem_time_from</code> . Applies only if the source is trigger event and object is trigger. Mandatory if <code>problem_time_till</code> is specified.
problem_time_till	timestamp	Returns only events that were in the problem state until <code>problem_time_till</code> . Applies only if the source is trigger event and object is trigger. Mandatory if <code>problem_time_from</code> is specified.
value	integer/array	Return only events with the given values.
selectHosts	query	Return hosts containing the object that created the event in the <code>hosts</code> property. Supported only for events generated by triggers, items or LLD rules.
selectRelatedObject	query	Return the object that created the event in the <code>relatedObject</code> property. The type of object returned depends on the event type.
select_alerts	query	Return alerts generated by the event in the <code>alerts</code> property. Alerts are sorted in reverse chronological order.
select_acknowledges	query	Return event's updates in the <code>acknowledges</code> property. Event updates are sorted in reverse chronological order.
		The event update object has the following properties: <code>acknowledgeid</code> - (string) acknowledgement's ID; <code>userid</code> - (string) ID of the user that updated the event; <code>eventid</code> - (string) ID of the updated event; <code>clock</code> - (timestamp) time when the event was updated; <code>message</code> - (string) text of the message; <code>action</code> - (integer) update action that was performed see event.acknowledge; <code>old_severity</code> - (integer) event severity before this update action; <code>new_severity</code> - (integer) event severity after this update action; <code>alias</code> - (string) alias of the user that updated the event; <code>name</code> - (string) name of the user that updated the event; <code>surname</code> - (string) surname of the user that updated the event.
selectTags	query	Supports count. Return event tags in <code>tags</code> property.
selectSuppressionData	query	Return list of maintenances in the <code>suppression_data</code> property: <code>maintenanceid</code> - (string) ID of the maintenance; <code>suppress_until</code> - (integer) time until the event is suppressed.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>eventid</code> , <code>objectid</code> and <code>clock</code> . 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	

Parameter	Type	Description
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

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": [
    {
      "acknowledges": [
        {
          "acknowledgeid": "1",
          "userid": "1",
          "eventid": "9695",
          "clock": "1350640590",
          "message": "Problem resolved.\n\r----[BULK ACKNOWLEDGE]----",
          "action": "6",
          "old_severity": "0",
          "new_severity": "0",
          "alias": "Admin",
          "name": "Zabbix",
          "surname": "Administrator"
        }
      ],
      "eventid": "9695",
      "source": "0",
      "object": "0",
      "objectid": "13926",
      "clock": "1347970410",
      "value": "1",
    }
  ]
}
```

```

        "acknowledged": "1",
        "suppressed": "1",
        "ns": "413316245",
        "name": "MySQL is down",
        "severity": "5",
        "r_eventid": "0",
        "c_eventid": "0",
        "correlationid": "0",
        "userid": "0",
        "tags": [
            {
                "tag": "service",
                "value": "mysqld"
            },
            {
                "tag": "error",
                "value": ""
            }
        ],
        "suppression_data": [
            {
                "maintenanceid": "15",
                "suppress_until": "1472511600"
            }
        ]
    },
    {
        "acknowledges": [],
        "eventid": "9671",
        "source": "0",
        "object": "0",
        "objectid": "13926",
        "clock": "1347970347",
        "value": "0",
        "acknowledged": "0",
        "suppressed": "0",
        "ns": "0",
        "name": "Unavailable by ICMP ping",
        "severity": "4",
        "r_eventid": "0",
        "c_eventid": "0",
        "correlationid": "0",
        "userid": "0",
        "tags": [],
        "suppression_data": []
    }
],
    "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",
      "suppressed": "0",
      "ns": "0",
      "name": "Less than 25% free in the history cache",
      "severity": "3",
      "r_eventid": "0",
      "c_eventid": "0",
      "correlationid": "0",
      "userid": "0"
    },
    {
      "eventid": "20617",
      "source": "0",
      "object": "0",
      "objectid": "14283",
      "clock": "1350477814",
      "value": "0",
      "acknowledged": "0",
      "suppressed": "0",
      "ns": "0",
      "name": "Zabbix trapper processes more than 75% busy",
      "severity": "3",
      "r_eventid": "0",
      "c_eventid": "0",
      "correlationid": "0",
      "userid": "0"
    },
    {
      "eventid": "20618",
      "source": "0",
      "object": "0",
      "objectid": "14284",
      "clock": "1350477815",
      "value": "1",
      "acknowledged": "0",
      "suppressed": "0",
      "ns": "0",
      "name": "High ICMP ping loss",
      "severity": "3",
      "r_eventid": "0",
      "c_eventid": "0",
      "correlationid": "0",
      "userid": "0"
    }
  ],
}

```

```
}  
  "id": 1  
}
```

See also

- [Alert](#)
- [Item](#)
- [Host](#)
- [LLD rule](#)
- [Trigger](#)

Source

CEvent::get() in *frontends/php/include/classes/api/services/CEvent.php*.

Graph

This class is designed to work with items.

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	<i>(readonly)</i> 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.
flags	integer	<i>(readonly)</i> Origin of the graph. Possible values are: 0 - <i>(default)</i> a plain graph; 4 - a discovered graph.
graphtype	integer	Graph's layout type. Possible values: 0 - <i>(default)</i> normal; 1 - stacked; 2 - pie; 3 - exploded.
percent_left	float	Left percentile.
percent_right	float	Default: 0. Right percentile.
		Default: 0.

Property	Type	Description
show_3d	integer	Whether to show pie and exploded graphs in 3D. Possible values: 0 - <i>(default)</i> show in 2D; 1 - show in 3D.
show_legend	integer	Whether to show the legend on the graph. Possible values: 0 - hide; 1 - <i>(default)</i> show.
show_work_period	integer	Whether to show the working time on the graph. Possible values: 0 - hide; 1 - <i>(default)</i> show.
templateid	string	<i>(readonly)</i> 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.
ymax_type	integer	Maximum value calculation method for the Y axis. Possible values: 0 - <i>(default)</i> calculated; 1 - fixed; 2 - item.
ymin_itemid	string	ID of the item that is used as the minimum value for the Y axis.
ymin_type	integer	Minimum value calculation method for the Y axis. Possible values: 0 - <i>(default)</i> calculated; 1 - fixed; 2 - item.

graph.create

Description

`object graph.create(object/array graphs)`

This method allows to create new graphs.

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 *frontends/php/include/classes/api/services/CGraph.php*.

graph.delete

Description

object graph.delete(array graphIds)

This method allows to delete graphs.

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 *frontends/php/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.

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 in the given host 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 <code>true</code> return only graphs that belong to templates.
inherited	boolean	If set to <code>true</code> return only graphs inherited from a template.
expandName	flag	Expand macros in the graph name.
selectGroups	query	Return the host groups that the graph belongs to in the <code>groups</code> property.
selectTemplates	query	Return the templates that the graph belongs to in the <code>templates</code> property.
selectHosts	query	Return the hosts that the graph belongs to in the <code>hosts</code> property.
selectItems	query	Return the items used in the graph in the <code>items</code> property.

Parameter	Type	Description
selectGraphDiscovery	query	Return the graph discovery object in the <code>graphDiscovery</code> property. The graph discovery objects links the graph to a graph prototype from which it was created.
selectGraphItems	query	It has the following properties: <code>graphid</code> - (string) ID of the graph; <code>parent_graphid</code> - (string) ID of the graph prototype from which the graph has been created. Return the graph items used in the graph in the <code>gitems</code> property.
selectDiscoveryRule	query	Return the low-level discovery rule that created the graph in the <code>discoveryRule</code> 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: <code>host</code> - technical name of the host that the graph belongs to; <code>hostid</code> - ID of the host that the graph belongs to. Sort the result by the given properties.
sortfield	string/array	
countOutput	boolean	Possible values are: <code>graphid</code> , <code>name</code> and <code>graphtype</code> . 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

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": "0.0000",
      "yaxismax": "100.0000",
      "templateid": "439",
      "show_work_period": "1",
      "show_triggers": "1",
      "graphtype": "0",
      "show_legend": "1",
      "show_3d": "0",
      "percent_left": "0.0000",
      "percent_right": "0.0000",
      "ymin_type": "0",
      "ymax_type": "0",
      "ymin_itemid": "0",
      "ymax_itemid": "0",
      "flags": "0"
    },
    {
      "graphid": "613",
      "name": "CPU load",
      "width": "900",
      "height": "200",
      "yaxismin": "0.0000",
      "yaxismax": "100.0000",
      "templateid": "433",
      "show_work_period": "1",
      "show_triggers": "1",
      "graphtype": "0",
      "show_legend": "1",
      "show_3d": "0",
      "percent_left": "0.0000",
      "percent_right": "0.0000",
      "ymin_type": "1",
      "ymax_type": "0",
      "ymin_itemid": "0",
      "ymax_itemid": "0",
      "flags": "0"
    },
    {
      "graphid": "614",
      "name": "CPU utilization",
      "width": "900",
      "height": "200",
      "yaxismin": "0.0000",
      "yaxismax": "100.0000",
      "templateid": "387",
      "show_work_period": "1",
      "show_triggers": "0",
      "graphtype": "1",
      "show_legend": "1",

```

```

        "show_3d": "0",
        "percent_left": "0.0000",
        "percent_right": "0.0000",
        "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": "0.0000",
        "yaxismax": "0.0000",
        "templateid": "0",
        "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",
        "flags": "4"
    }
],
    "id": 1
}

```

See also

- [Discovery rule](#)
- [Graph item](#)
- [Item](#)
- [Host](#)
- [Host group](#)
- [Template](#)

Source

CGraph::get() in *frontends/php/include/classes/api/services/CGraph.php*.

graph.update

Description

object graph.update(object/array graphs)

This method allows to update existing graphs.

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 <code>gitemid</code> 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 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 `frontends/php/include/classes/api/services/CGraph.php`.

Graph item

This class is designed to work with hosts.

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
<code>gitemid</code>	string	(<i>readonly</i>) ID of the graph item.
<code>color</code> (required)	string	Graph item's draw color as a hexadecimal color code.
<code>itemid</code> (required)	string	ID of the item.
<code>calc_fnc</code>	integer	Value of the item that will be displayed. Possible values: 1 - minimum value; 2 - (<i>default</i>) average value; 4 - maximum value; 7 - all values; 9 - last value, used only by pie and exploded graphs.
<code>drawtype</code>	integer	Draw style of the graph item. Possible values: 0 - (<i>default</i>) line; 1 - filled region; 2 - bold line; 3 - dot; 4 - dashed line; 5 - gradient line.
<code>graphid</code>	string	ID of the graph that the graph item belongs to.
<code>sortorder</code>	integer	Position of the item in the graph. Default: starts with 0 and increases by one with each entry.
<code>type</code>	integer	Type of graph item. Possible values: 0 - (<i>default</i>) simple; 2 - graph sum, used only by pie and exploded graphs.
<code>yaxisside</code>	integer	Side of the graph where the graph item's Y scale will be drawn. Possible values: 0 - (<i>default</i>) 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.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
<code>graphids</code>	string/array	Return only graph items that belong to the given graphs.

Parameter	Type	Description
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 the graph that the item belongs to as an array in the <code>graphs</code> property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>gitemid</code> . 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": "Template OS Linux"
    },
  ],
}
```

```

    {
        "gititemid": "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": "Template OS Linux"
    },
    {
        "gititemid": "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": "Template OS Linux"
    }
],
"id": 1
}

```

See also

- [Graph](#)

Source

CGraphItem::get() in *frontends/php/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	<i>(readonly)</i> ID of the graph prototype.
height (required)	integer	Height of the graph prototype in pixels.
name (required)	string	Name of the graph prototype.
width (required)	integer	Width of the graph prototype in pixels.
graphtype	integer	Graph prototypes's layout type. Possible values: 0 - <i>(default)</i> normal; 1 - stacked; 2 - pie; 3 - exploded.
percent_left	float	Left percentile.
percent_right	float	Default: 0. Right percentile.
show_3d	integer	Default: 0. Whether to show discovered pie and exploded graphs in 3D.
show_legend	integer	Possible values: 0 - <i>(default)</i> show in 2D; 1 - show in 3D. Whether to show the legend on the discovered graph.
show_work_period	integer	Possible values: 0 - hide; 1 - <i>(default)</i> show. Whether to show the working time on the discovered graph.
templateid	string	Possible values: 0 - hide; 1 - <i>(default)</i> show. <i>(readonly)</i> 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.
ymax_type	integer	Maximum value calculation method for the Y axis. Possible values: 0 - <i>(default)</i> calculated; 1 - fixed; 2 - item.
ymin_itemid	string	ID of the item that is used as the minimum value for the Y axis.
ymin_type	integer	Minimum value calculation method for the Y axis. Possible values: 0 - <i>(default)</i> calculated; 1 - fixed; 2 - item.

graphprototype.create

Description

`object graphprototype.create(object/array graphPrototypes)`

This method allows to create new graph prototypes.

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 *frontends/php/include/classes/api/services/CGraphPrototype.php*.

graphprototype.delete

Description

object graphprototype.delete(array graphPrototypeIds)

This method allows to delete graph prototypes.

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 *frontends/php/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.

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.

Parameter	Type	Description
groupids	string/array	Return only graph prototypes that belong to hosts in the given host groups.
hostids	string/array	Return only graph prototypes that belong to the given hosts.
inherited	boolean	If set to <code>true</code> 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 <code>true</code> return only graph prototypes that belong to templates.
templateids	string/array	Return only graph prototypes that belong to the given templates.
selectDiscoveryRule	query	Return the LLD rule that the graph prototype belongs to in the <code>discoveryRule</code> property.
selectGraphItems	query	Return the graph items used in the graph prototype in the <code>gitems</code> property.
selectGroups	query	Return the host groups that the graph prototype belongs to in the <code>groups</code> property.
selectHosts	query	Return the hosts that the graph prototype belongs to in the <code>hosts</code> property.
selectItems	query	Return the items and item prototypes used in the graph prototype in the <code>items</code> property.
selectTemplates	query	Return the templates that the graph prototype belongs to in the <code>templates</code> 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: <code>host</code> - technical name of the host that the graph prototype belongs to; <code>hostid</code> - ID of the host that the graph prototype belongs to.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>graphid</code> , <code>name</code> and <code>graphtype</code> . These parameters being common for all <code>get</code> 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 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"
    }
  ],
  "id": 1
}
```

See also

- [Discovery rule](#)
- [Graph item](#)
- [Item](#)
- [Host](#)
- [Host group](#)
- [Template](#)

Source

CGraphPrototype::get() in *frontends/php/include/classes/api/services/CGraphPrototype.php*.

graphprototype.update

Description

object graphprototype.update(object/array graphPrototypes)

This method allows to update existing graph prototypes.

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 <code>gitemid</code> 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 *frontends/php/include/classes/api/services/CGraphPrototype.php*.

History

This class is designed to work with history data.

Object references:

- [History](#)

Available methods:

- [history.get](#) - retrieving history data.

> History object

The following objects are directly related to the `history` API.

Note:

History objects differ depending on the item's type of information. They are created by the Zabbix server and cannot be modified via the API.

Float history

The float history object has the following properties.

Property	Type	Description
clock	timestamp	Time when that value was received.
itemid	string	ID of the related item.
ns	integer	Nanoseconds when the value was received.
value	float	Received value.

Integer history

The integer history object has the following properties.

Property	Type	Description
clock	timestamp	Time when that value was received.
itemid	string	ID of the related item.
ns	integer	Nanoseconds when the value was received.
value	integer	Received value.

String history

The string history object has the following properties.

Property	Type	Description
clock	timestamp	Time when that value was received.
itemid	string	ID of the related item.
ns	integer	Nanoseconds when the value was received.
value	string	Received value.

Text history

The text history object has the following properties.

Property	Type	Description
id	string	ID of the history entry.
clock	timestamp	Time when that value was received.
itemid	string	ID of the related item.
ns	integer	Nanoseconds when the value was received.
value	text	Received value.

Log history

The log history object has the following properties.

Property	Type	Description
id	string	ID of the history entry.
clock	timestamp	Time when that value was received.
itemid	string	ID of the related item.
logeventid	integer	Windows event log entry ID.

Property	Type	Description
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.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.

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.
countOutput	boolean	Possible values are: <code>itemid</code> and <code>clock</code> . These parameters being common for all <code>get</code> 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": "0.0850",
      "ns": "563157632"
    },
    {
      "itemid": "23296",
      "clock": "1351090936",
      "value": "0.1600",
      "ns": "549216402"
    },
    {
      "itemid": "23296",
      "clock": "1351090876",
      "value": "0.1800",
      "ns": "537418114"
    },
    {
      "itemid": "23296",
      "clock": "1351090816",
      "value": "0.2100",
      "ns": "522659528"
    },
    {
      "itemid": "23296",
      "clock": "1351090756",
      "value": "0.2150",
      "ns": "507809457"
    },
    {
      "itemid": "23296",
      "clock": "1351090696",
      "value": "0.2550",
      "ns": "495509699"
    }
  ]
}
```

```

    },
    {
        "itemid": "23296",
        "clock": "1351090636",
        "value": "0.3600",
        "ns": "477708209"
    },
    {
        "itemid": "23296",
        "clock": "1351090576",
        "value": "0.3750",
        "ns": "463251343"
    },
    {
        "itemid": "23296",
        "clock": "1351090516",
        "value": "0.3150",
        "ns": "447947017"
    },
    {
        "itemid": "23296",
        "clock": "1351090456",
        "value": "0.2750",
        "ns": "435307141"
    }
],
"id": 1
}

```

Source

CHistory::get() in *frontends/php/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	(<i>readonly</i>) ID of the host.

Property	Type	Description
host	string	Technical name of the host.
(required)		
available	integer	<i>(readonly)</i> Availability of Zabbix agent. Possible values are: 0 - <i>(default)</i> unknown; 1 - available; 2 - unavailable.
description	text	Description of the host.
disable_until	timestamp	<i>(readonly)</i> The next polling time of an unavailable Zabbix agent.
error	string	<i>(readonly)</i> Error text if Zabbix agent is unavailable.
errors_from	timestamp	<i>(readonly)</i> Time when Zabbix agent became unavailable.
flags	integer	<i>(readonly)</i> Origin of the host. Possible values: 0 - a plain host; 4 - a discovered host.
inventory_mode	integer	<i>(writeonly)</i> Host inventory population mode.
ipmi_authtype	integer	Possible values are: -1 - disabled; 0 - <i>(default)</i> manual; 1 - automatic. IPMI authentication algorithm.
ipmi_available	integer	Possible values are: -1 - <i>(default)</i> default; 0 - none; 1 - MD2; 2 - MD5 4 - straight; 5 - OEM; 6 - RMCP+. <i>(readonly)</i> Availability of IPMI agent.
ipmi_disable_until	timestamp	Possible values are: 0 - <i>(default)</i> unknown; 1 - available; 2 - unavailable. <i>(readonly)</i> The next polling time of an unavailable IPMI agent.
ipmi_error	string	<i>(readonly)</i> Error text if IPMI agent is unavailable.
ipmi_errors_from	timestamp	<i>(readonly)</i> Time when IPMI agent became unavailable.
ipmi_password	string	IPMI password.
ipmi_privilege	integer	IPMI privilege level. Possible values are: 1 - callback; 2 - <i>(default)</i> user; 3 - operator; 4 - admin; 5 - OEM.
ipmi_username	string	IPMI username.
jmx_available	integer	<i>(readonly)</i> Availability of JMX agent. Possible values are: 0 - <i>(default)</i> unknown; 1 - available; 2 - unavailable.
jmx_disable_until	timestamp	<i>(readonly)</i> The next polling time of an unavailable JMX agent.

Property	Type	Description
jmx_error	string	<i>(readonly)</i> Error text if JMX agent is unavailable.
jmx_errors_from	timestamp	<i>(readonly)</i> Time when JMX agent became unavailable.
maintenance_from	timestamp	<i>(readonly)</i> Starting time of the effective maintenance.
maintenance_status	integer	<i>(readonly)</i> Effective maintenance status. Possible values are: 0 - <i>(default)</i> no maintenance; 1 - maintenance in effect.
maintenance_type	integer	<i>(readonly)</i> Effective maintenance type. Possible values are: 0 - <i>(default)</i> maintenance with data collection; 1 - maintenance without data collection.
maintenanceid	string	<i>(readonly)</i> ID of the maintenance that is currently in effect on the host.
name	string	Visible name of the host.
proxy_hostid	string	Default: host property value.
snmp_available	integer	ID of the proxy that is used to monitor the host. <i>(readonly)</i> Availability of SNMP agent. Possible values are: 0 - <i>(default)</i> unknown; 1 - available; 2 - unavailable.
snmp_disable_until	timestamp	<i>(readonly)</i> The next polling time of an unavailable SNMP agent.
snmp_error	string	<i>(readonly)</i> Error text if SNMP agent is unavailable.
snmp_errors_from	timestamp	<i>(readonly)</i> Time when SNMP agent became unavailable.
status	integer	Status and function of the host. Possible values are: 0 - <i>(default)</i> monitored host; 1 - unmonitored host.
tls_connect	integer	Connections to host.
tls_accept	integer	Connections from host. Possible values are: 1 - <i>(default)</i> No encryption; 2 - PSK; 4 - certificate.
tls_issuer	string	Possible bitmap values are: 1 - <i>(default)</i> No encryption; 2 - PSK; 4 - certificate. Certificate issuer.
tls_subject	string	Certificate subject.
tls_psk_identity	string	PSK identity. Required if either <code>tls_connect</code> or <code>tls_accept</code> 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	The preshared key, at least 32 hex digits. Required if either <code>tls_connect</code> or <code>tls_accept</code> has PSK enabled.

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
	inventory_mode	integer	(<i>readonly</i>) Host inventory population mode. Possible values are: -1 - disabled; 0 - (<i>default</i>) manual; 1 - automatic.
4	alias	string	Alias.
11	asset_tag	string	Asset tag.
28	chassis	string	Chassis.
23	contact	string	Contact person.
32	contract_number	string	Contract number.
47	date_hw_decomm	string	HW decommissioning date.
46	date_hw_expiry	string	HW maintenance expiry date.
45	date_hw_install	string	HW installation date.
44	date_hw_purchase	string	HW purchase date.
34	deployment_status	string	Deployment status.
14	hardware	string	Hardware.
15	hardware_full	string	Detailed hardware.
39	host_netmask	string	Host subnet mask.
38	host_networks	string	Host networks.
40	host_router	string	Host router.
30	hw_arch	string	HW architecture.
33	installer_name	string	Installer name.
24	location	string	Location.
25	location_lat	string	Location latitude.
26	location_lon	string	Location longitude.
12	macaddress_a	string	MAC address A.
13	macaddress_b	string	MAC address B.
29	model	string	Model.
3	name	string	Name.
27	notes	string	Notes.
41	oob_ip	string	OOB IP address.
42	oob_netmask	string	OOB host subnet mask.
43	oob_router	string	OOB router.
5	os	string	OS name.
6	os_full	string	Detailed OS name.
7	os_short	string	Short OS name.
61	poc_1_cell	string	Primary POC mobile number.
58	poc_1_email	string	Primary email.
57	poc_1_name	string	Primary POC name.
63	poc_1_notes	string	Primary POC notes.
59	poc_1_phone_a	string	Primary POC phone A.
60	poc_1_phone_b	string	Primary POC phone B.
62	poc_1_screen	string	Primary POC screen name.
68	poc_2_cell	string	Secondary POC mobile number.
65	poc_2_email	string	Secondary POC email.
64	poc_2_name	string	Secondary POC name.
70	poc_2_notes	string	Secondary POC notes.
66	poc_2_phone_a	string	Secondary POC phone A.
67	poc_2_phone_b	string	Secondary POC phone B.
69	poc_2_screen	string	Secondary POC screen name.
8	serialno_a	string	Serial number A.
9	serialno_b	string	Serial number B.
48	site_address_a	string	Site address A.
49	site_address_b	string	Site address B.
50	site_address_c	string	Site address C.
51	site_city	string	Site city.

ID	Property	Type	Description
53	site_country	string	Site country.
56	site_notes	string	Site notes.
55	site_rack	string	Site rack location.
52	site_state	string	Site state.
54	site_zip	string	Site ZIP/postal code.
16	software	string	Software.
18	software_app_a	string	Software application A.
19	software_app_b	string	Software application B.
20	software_app_c	string	Software application C.
21	software_app_d	string	Software application D.
22	software_app_e	string	Software application E.
17	software_full	string	Software details.
10	tag	string	Tag.
1	type	string	Type.
2	type_full	string	Type details.
35	url_a	string	URL A.
36	url_b	string	URL B.
37	url_c	string	URL C.
31	vendor	string	Vendor.

Host tag

The host tag object has the following properties.

Property	Type	Description
tag (required)	string	Host tag name.
value	string	Host tag value.

host.create

Description

`object host.create(object/array hosts)`

This method allows to create new hosts.

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 <code>groupid</code> property defined.
interfaces (required)	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 <code>templateid</code> 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"
      }
    ],
    "inventory_mode": 0,
    "inventory": {
      "macaddress_a": "01234",
      "macaddress_b": "56768"
    }
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "hostids": [
      "107819"
    ]
  }
}
```

```

    ]
  },
  "id": 1
}

```

See also

- [Host group](#)
- [Template](#)
- [User macro](#)
- [Host interface](#)
- [Host inventory](#)
- [Host tag](#)

Source

`CHost::create()` in *frontends/php/include/classes/api/services/CHost.php*.

host.delete

Description

`object host.delete(array hosts)`

This method allows to delete hosts.

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 *frontends/php/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.

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.
applicationids	string/array	Return only hosts that have the given applications.
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.
with_applications	flag	Overrides the with_monitored_items and with_simple_graph_items parameters. Return only hosts that have applications.
with_graphs	flag	Return only hosts that have graphs.
with_httptests	flag	Return only hosts that have web checks.
with_monitored_httptests	flag	Overrides the with_monitored_httptests parameter. Return only hosts that have enabled web checks.
with_monitored_items	flag	Return only hosts that have enabled items.
with_monitored_triggers	flag	Overrides the with_simple_graph_items parameter. Return only hosts that have enabled triggers. All of the items used in the trigger must also be enabled.
with_simple_graph_items	flag	Return only hosts that have items with numeric type of information.
with_triggers	flag	Return only hosts that have triggers.
withInventory	flag	Overrides the with_monitored_triggers parameter. Return only hosts that have inventory data.
evaltype	integer	Rules for tag searching.

Possible values:
0 - (default) And/Or;
2 - Or.

Parameter	Type	Description
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.
selectGroups	query	Return a groups property with host groups data that the host belongs to.
selectTags	query	Return a tags property with host tags.
selectApplications	query	Return an applications property with host applications.
selectDiscoveries	query	Supports count. Return a discoveries property with host low-level discovery rules.
selectDiscoveryRule	query	Supports count. 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.
selectHostDiscovery	query	Supports count. 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 - (<i>string</i>) host of the host prototype; hostid - (<i>string</i>) ID of the discovered host or host prototype; parent_hostid - (<i>string</i>) ID of the host prototype from which the host has been created; parent_itemid - (<i>string</i>) ID of the LLD rule that created the discovered host; lastcheck - (<i>timestamp</i>) time when the host was last discovered; ts_delete - (<i>timestamp</i>) time when a host that is no longer discovered will be deleted.
selectHttpTests	query	Return an httpTests property with host web scenarios.
selectInterfaces	query	Supports count. Return an interfaces property with host interfaces.
selectInventory	query	Supports count.
selectItems	query	Return an inventory property with host inventory data. Return an items property with host items.
selectMacros	query	Supports count.
selectParentTemplates	query	Return a macros property with host macros. Return a parentTemplates property with templates that the host is linked to.
selectScreens	query	Supports count. Return a screens property with host screens. Supports count.

Parameter	Type	Description
selectTriggers	query	Return a triggers property with host triggers.
filter	object	Supports count. 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.
limitSelects	integer	Allows filtering by interface properties. Limits the number of records returned by subselects.
search	object	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; selectApplications - sorted by name; selectScreens - sorted by name. 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.
searchInventory	object	Allows searching by interface properties. Works only with text fields. Return only hosts that have inventory data matching the given wildcard search.
sortfield	string/array	This parameter is affected by the same additional parameters as search. Sort the result by the given properties.
countOutput	boolean	Possible values are: hostid, host, name, status. These parameters being common for all get methods are described in detail in the reference commentary.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

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": [
    {
      "maintenances": [],
      "hostid": "10160",
      "proxy_hostid": "0",
      "host": "Zabbix server",
      "status": "0",
      "disable_until": "0",
      "error": "",
      "available": "0",
      "errors_from": "0",
      "lastaccess": "0",
      "ipmi_authtype": "-1",
      "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": "Zabbix server",
      "description": "The Zabbix monitoring server.",
      "tls_connect": "1",
      "tls_accept": "1",
      "tls_issuer": "",
      "tls_subject": "",
      "tls_psk_identity": "",
      "tls_psk": ""
    },
    {

```

```

        "maintenances": [],
        "hostid": "10167",
        "proxy_hostid": "0",
        "host": "Linux server",
        "status": "0",
        "disable_until": "0",
        "error": "",
        "available": "0",
        "errors_from": "0",
        "lastaccess": "0",
        "ipmi_authtype": "-1",
        "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 server",
        "description": "",
        "tls_connect": "1",
        "tls_accept": "1",
        "tls_issuer": "",
        "tls_subject": "",
        "tls_psk_identity": "",
        "tls_psk": ""
    }
],
    "id": 1
}

```

Retrieving host groups

Retrieve names of the groups host "Zabbix server" is member of, but no host details themselves.

Request:

```

{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "output": ["hostid"],
        "selectGroups": "extend",
        "filter": {
            "host": [
                "Zabbix server"
            ]
        }
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 2
}

```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "hostid": "10085",
      "groups": [
        {
          "groupid": "2",
          "name": "Linux servers",
          "internal": "0",
          "flags": "0"
        },
        {
          "groupid": "4",
          "name": "Zabbix servers",
          "internal": "0",
          "flags": "0"
        }
      ]
    }
  ],
  "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": "Template OS Linux",
          "templateid": "10001"
        },
        {
          "name": "Template App Zabbix Server",
          "templateid": "10047"
        }
      ]
    }
  ],
  "id": 1,
}
```

```
    "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
}

```

See also

- [Host group](#)
- [Template](#)
- [User macro](#)
- [Host interface](#)

Source

`CHost::get()` in `frontends/php/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.

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.
groups	object/array	The hosts must have the <code>hostid</code> property defined. Host groups to add to the given hosts.
interfaces	object/array	The host groups must have the <code>groupid</code> property defined. Host interfaces to be created for the given hosts.
macros	object/array	User macros to be created for the given hosts.

Parameter	Type	Description
templates	object/array	Templates to link to the given hosts. The templates must have the <code>templateid</code> 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"
      }
    ]
  },
  "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 *frontends/php/include/classes/api/services/CHost.php*.

host.massremove

Description

object `host.massremove(object parameters)`

This method allows to remove related objects from multiple hosts.

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 *frontends/php/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.

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 <code>hostid</code> property defined.
groups	object/array	Host groups to replace the current host groups the hosts belong to. The host groups must have the <code>groupid</code> 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 <code>inventory</code> parameter, use <code>inventory_mode</code> instead.
inventory_mode	integer	Host inventory population mode. Refer to the host inventory object page for a list of supported inventory modes.
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 <code>templateid</code> property defined.
templates_clear	object/array	Templates to unlink and clear from the given hosts. The templates must have the <code>templateid</code> 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 *frontends/php/include/classes/api/services/CHost.php*.

host.update

Description

`object host.update(object/array hosts)`

This method allows to update existing hosts.

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.

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 <code>groupid</code> property defined. All host groups that are not listed in the request will be unlinked.

Parameter	Type	Description
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.
templates_clear	object/array	The templates must have the <code>templateid</code> property defined. Templates to unlink and clear from the host. The templates must have the <code>templateid</code> 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"
      }
    ]
  },
  "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
}
```

See also

- [host.massadd](#)
- [host.massupdate](#)
- [host.massremove](#)
- [Host group](#)
- [Template](#)
- [User macro](#)
- [Host interface](#)
- [Host inventory](#)
- [Host tag](#)

Source

`CHost::update()` in *frontends/php/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.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	<i>(readonly)</i> ID of the host group.
name (required)	string	Name of the host group.
flags	integer	<i>(readonly)</i> Origin of the host group. Possible values: 0 - a plain host group; 4 - a discovered host group.
internal	integer	<i>(readonly)</i> Whether the group is used internally by the system. An internal group cannot be deleted. Possible values: 0 - <i>(default)</i> not internal; 1 - internal.

hostgroup.create

Description

`object hostgroup.create(object/array hostGroups)`

This method allows to create new host groups.

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 `frontends/php/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.

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 *frontends/php/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.

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 or templates 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.
monitored_hosts	flag	Return only host groups that contain monitored hosts.
real_hosts	flag	Return only host groups that contain hosts.
templated_hosts	flag	Return only host groups that contain templates.
templateids	string/array	Return only host groups that contain the given templates.
triggerids	string/array	Return only host groups that contain hosts or templates with the given triggers.
with_applications	flag	Return only host groups that contain hosts with applications.
with_graphs	flag	Return only host groups that contain hosts with graphs.
with_hosts_and_templates	flag	Return only host groups that contain hosts or templates.

Parameter	Type	Description
with_httptests	flag	Return only host groups that contain hosts with web checks.
with_items	flag	Overrides the with_monitored_httptests parameter. Return only host groups that contain hosts or templates with items.
with_monitored_httptests	flag	Overrides the with_monitored_items and with_simple_graph_items parameters. Return only host groups that contain hosts with enabled web checks.
with_monitored_items	flag	Return only host groups that contain hosts or templates with enabled items.
with_monitored_triggers	flag	Overrides the with_simple_graph_items parameter. 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.
selectDiscoveryRule	query	Overrides the with_monitored_triggers parameter. Return the LLD rule that created the host group in the discoveryRule property.
selectGroupDiscovery	query	Return the host group discovery object in the groupDiscovery property.
selectHosts	query	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.
selectTemplates	query	Return the hosts that belong to the host group in the hosts property.
limitSelects	integer	Supports count. Return the templates that belong to the host group in the templates property.
sortfield	string/array	Supports count. Limits the number of records returned by subselects. Applies to the following subselects: selectHosts - results will be sorted by host; selectTemplates - results will be sorted by host. 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 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](#)
- [Template](#)

Source

`CHostGroup::get()` in *frontends/php/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.

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 <code>groupid</code> property defined.
hosts	object/array	Hosts to add to all host groups. The hosts must have the <code>hostid</code> property defined.
templates	object/array	Templates to add to all host groups. The templates must have the <code>templateid</code> 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](#)
- [Template](#)

Source

CHostGroup::massAdd() in *frontends/php/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.

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.
templateids	string/array	Templates 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 *frontends/php/include/classes/api/services/CHostGroup.php*.

hostgroup.massupdate

Description

`object hostgroup.massupdate(object parameters)`

This method allows to simultaneously replace or remove related objects for multiple host groups.

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.
hosts	object/array	The host groups must have the <code>groupid</code> property defined. Hosts to replace the current hosts on the given host groups.
templates	object/array	The hosts must have the <code>hostid</code> property defined. Templates to replace the current templates on the given host groups. The templates must have the <code>templateid</code> 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 the host group with ID.

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](#)
- [Template](#)

Source

`CHostGroup::massUpdate()` in *frontends/php/include/classes/api/services/CHostGroup.php*.

hostgroup.update

Description

`object hostgroup.update(object/array hostGroups)`

This method allows to update existing hosts groups.

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 *frontends/php/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
<code>interfaceid</code>	string	(<i>readonly</i>) ID of the interface.
<code>dns</code> (required)	string	DNS name used by the interface. Can be empty if the connection is made via IP.
<code>hostid</code> (required)	string	ID of the host the interface belongs to.
<code>ip</code> (required)	string	IP address used by the interface. Can be empty if the connection is made via DNS.

Property	Type	Description
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.
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.
bulk	integer	Whether to use bulk SNMP requests. Possible values are: 0 - don't use bulk requests; 1 - <i>(default)</i> use bulk requests.

hostinterface.create

Description

`object hostinterface.create(object/array hostInterfaces)`

This method allows to create new host interfaces.

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",
    "dns": "",
    "ip": "127.0.0.1",
    "main": 0,
    "port": "10050",
    "type": 1,
    "useip": 1
  }
},
```

```
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "interfaceids": [
      "30062"
    ]
  },
  "id": 1
}
```

See also

- [hostinterface.massadd](#)
- [host.massadd](#)

Source

CHostInterface::create() in *frontends/php/include/classes/api/services/CHostInterface.php*.

hostinterface.delete

Description

object `hostinterface.delete(array hostInterfaceIds)`

This method allows to delete host interfaces.

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 *frontends/php/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.

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 the items that use the interface in the <code>items</code> property.
selectHosts	query	Supports <code>count</code> . Return the host that uses the interface as an array in the <code>hosts</code> property.
limitSelects	integer	Limits the number of records returned by subselects.
sortfield	string/array	Applies to the following subselects: <code>selectItems</code> . Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>interfaceid</code> , <code>dns</code> , <code>ip</code> . These parameters being common for all <code>get</code> 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	
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": "30050",
      "hostid": "30057",
      "main": "1",
      "type": "1",
      "useip": "1",
      "ip": "127.0.0.1",
      "dns": "",
      "port": "10050",
      "bulk": "1"
    },
    {
      "interfaceid": "30067",
      "hostid": "30057",
      "main": "0",
      "type": "1",
      "useip": "0",
      "ip": "",
      "dns": "localhost",
      "port": "10050",
      "bulk": "1"
    },
    {
      "interfaceid": "30068",
      "hostid": "30057",
      "main": "1",
      "type": "2",
      "useip": "1",
      "ip": "127.0.0.1",
      "dns": "",
      "port": "161",
      "bulk": "1"
    }
  ],
  "id": 1
}
```

See also

- [Host](#)
- [Item](#)

Source

CHostInterface::get() in *frontends/php/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.

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 <code>hostid</code> 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 *frontends/php/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.

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 *frontends/php/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.

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 *frontends/php/include/classes/api/services/CHostInterface.php*.

hostinterface.update

Description

`object hostinterface.update(object/array hostInterfaces)`

This method allows to update existing host interfaces.

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 *frontends/php/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	(<i>readonly</i>) ID of the host prototype.
host (required)	string	Technical name of the host prototype.
name	string	Visible name of the host prototype.
status	integer	Default: <code>host</code> property value. Status of the host prototype. Possible values are: 0 - (<i>default</i>) monitored host; 1 - unmonitored host.
templateid	string	(<i>readonly</i>) ID of the parent template host prototype.
tls_connect	integer	Connections to host. Possible values are: 1 - (<i>default</i>) No encryption; 2 - PSK; 4 - certificate.
tls_accept	integer	Connections from host. Possible bitmap values are: 1 - (<i>default</i>) No encryption; 2 - PSK; 4 - certificate.
tls_issuer	string	Certificate issuer.
tls_subject	string	Certificate subject.
tls_psk_identity	string	PSK identity. Required if either <code>tls_connect</code> or <code>tls_accept</code> 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	The preshared key, at least 32 hex digits. Required if either <code>tls_connect</code> or <code>tls_accept</code> has PSK enabled.

Host prototype inventory

The host prototype inventory object has the following properties.

Property	Type	Description
inventory_mode	integer	Host prototype inventory population mode. Possible values are: -1 - disabled; 0 - <i>(default)</i> manual; 1 - automatic.

Group link

The group link object links a host prototype with a host group and has the following properties.

Property	Type	Description
group_prototypeid	string	<i>(readonly)</i> ID of the group link.
groupid (required)	string	ID of the host group.
hostid	string	<i>(readonly)</i> ID of the host prototype
templateid	string	<i>(readonly)</i> ID of the parent template group link.

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
group_prototypeid	string	<i>(readonly)</i> ID of the group prototype.
name (required)	string	Name of the group prototype.
hostid	string	<i>(readonly)</i> ID of the host prototype
templateid	string	<i>(readonly)</i> ID of the parent template group prototype.

hostprototype.create

Description

`object hostprototype.create(object/array hostPrototypes)`

This method allows to create new host prototypes.

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 inventory	array object	Group prototypes to be created for the host prototype. Host prototype inventory properties.

Parameter	Type	Description
templates	object/array	Templates to be linked to the host prototype. The templates must have the <code>templateid</code> 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}`". Link it to host group "2".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "hostprototype.create",
  "params": {
    "host": "{#VM.NAME}",
    "ruleid": "23542",
    "groupLinks": [
      {
        "groupid": "2"
      }
    ],
    "groupPrototypes": [
      {
        "name": "{#HV.NAME}"
      }
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "hostids": [
      "10103"
    ]
  },
  "id": 1
}
```

See also

- [Group link](#)
- [Group prototype](#)
- [Host prototype inventory](#)

Source

`CHostPrototype::create()` in *frontends/php/include/classes/api/services/CHostPrototype.php*.

hostprototype.delete

Description

object `hostprototype.delete(array hostPrototypeIds)`

This method allows to delete host prototypes.

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 *frontends/php/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.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
<code>hostids</code>	string/array	Return only host prototypes with the given IDs.
<code>discoveryids</code>	string/array	Return only host prototype that belong to the given LLD rules.
<code>inherited</code>	boolean	If set to <code>true</code> return only items inherited from a template.
<code>selectDiscoveryRule</code>	query	Return the LLD rule that the host prototype belongs to in the <code>discoveryRule</code> property.
<code>selectGroupLinks</code>	query	Return the group links of the host prototype in the <code>groupLinks</code> property.

Parameter	Type	Description
selectGroupPrototypes	query	Return the group prototypes of the host prototype in the groupPrototypes property.
selectInventory	query	Return the host prototype inventory in the inventory property.
selectParentHost	query	Return the host that the host prototype belongs to in the parentHost property.
selectTemplates	query	Return the templates linked to the host prototype in the templates property.
sortfield	string/array	Supports count. Sort the result by the given properties.
countOutput	boolean	Possible values are: hostid, host, name and status. 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 and their group links and group prototypes from an LLD rule.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "hostprototype.get",
  "params": {
    "output": "extend",
    "selectGroupLinks": "extend",
    "selectGroupPrototypes": "extend",
    "discoveryids": "23554"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "hostid": "10092",
      "host": "#{HV.UUID}",

```

```

        "status": "0",
        "name": "{#HV.NAME}",
        "templateid": "0",
        "tls_connect": "1",
        "tls_accept": "1",
        "tls_issuer": "",
        "tls_subject": "",
        "tls_psk_identity": "",
        "tls_psk": "",
        "groupLinks": [
            {
                "group_prototypeid": "4",
                "hostid": "10092",
                "groupid": "7",
                "templateid": "0"
            }
        ],
        "groupPrototypes": [
            {
                "group_prototypeid": "7",
                "hostid": "10092",
                "name": "{#CLUSTER.NAME}",
                "templateid": "0"
            }
        ]
    }
],
    "id": 1
}

```

See also

- [Group link](#)
- [Group prototype](#)
- [Host prototype inventory](#)

Source

`CHostPrototype::get()` in *frontends/php/include/classes/api/services/CHostPrototype.php*.

hostprototype.update

Description

`object hostprototype.update(object/array hostPrototypes)`

This method allows to update existing host prototypes.

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
<code>groupLinks</code>	array	Group links to replace the current group links on the host prototype.
<code>groupPrototypes</code>	array	Group prototypes to replace the existing group prototypes on the host prototype.
<code>inventory</code>	object	Host prototype inventory properties.

Parameter	Type	Description
templates	object/array	Templates to replace the currently linked templates. The templates must have the <code>templateid</code> 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
}
```

See also

- [Group link](#)
- [Group prototype](#)
- [Host prototype inventory](#)

Source

`CHostPrototype::update()` in *frontends/php/include/classes/api/services/CHostPrototype.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
<code>iconmapid</code>	string	(<i>readonly</i>) ID of the icon map.
<code>default_iconid</code> (required)	string	ID of the default icon.
<code>name</code> (required)	string	Name of the icon map.

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
<code>iconmappingid</code>	string	(<i>readonly</i>) ID of the icon map.
<code>iconid</code> (required)	string	ID of the icon used by the icon mapping.
<code>expression</code> (required)	string	Expression to match the inventory field against.
<code>inventory_link</code> (required)	integer	ID of the host inventory field. Refer to the host inventory object for a list of supported inventory fields.
<code>iconmapid</code>	string	(<i>readonly</i>) ID of the icon map that the icon mapping belongs to.
<code>sortorder</code>	integer	(<i>readonly</i>) 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.

Parameters

(object/array) Icon maps to create.

Additionally to the [standard icon map properties](#), the method accepts the following parameters.

Parameter	Type	Description
<code>mappings</code> (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

ClconMap::create() in *frontends/php/include/classes/api/services/ClconMap.php*.

iconmap.delete

Description

object iconmap.delete(array iconMapIds)

This method allows to delete icon maps.

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

ClconMap::delete() in *frontends/php/include/classes/api/services/ClconMap.php*.

iconmap.get

Description

`integer/array iconmap.get(object parameters)`

The method allows to retrieve icon maps according to the given parameters.

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 used icon mappings in the mappings property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: iconmapid and name. 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

`ClconMap::get()` in *frontends/php/include/classes/api/services/ClconMap.php*.

iconmap.update

Description

`object iconmap.update(object/array iconMaps)`

This method allows to update existing icon maps.

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
<code>mappings</code>	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

`ClconMap::update()` in *frontends/php/include/classes/api/services/ClconMap.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
<code>imageid</code>	string	(<i>readonly</i>) ID of the image.
<code>name</code> (required)	string	Name of the image.
<code>imagetype</code>	integer	Type of image. Possible values: 1 - (<i>default</i>) icon; 2 - background image.

image.create

Description

`object image.create(object/array images)`

This method allows to create new images.

Parameters

(object/array) Images to create.

Additionally to the **standard image properties**, the method accepts the following parameters.

Parameter	Type	Description
<code>name</code> (required)	string	Name of the image.
<code>imagetype</code> (required)	integer	Type of image. Possible values: 1 - (<i>default</i>) icon; 2 - background image.
<code>image</code> (required)	string	Base64 encoded image. The maximum size of the encoded image is 1 MB. Maximum size can be adjusted by changing <code>ZBX_MAX_IMAGE_SIZE</code> 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": "iVBORwOKGgoAAAANSUheUgAAABgAAAAANCAYAAACzbK7QAAAABHNCsvQICAgIfAhkiAAAAAlwSFlzAAACmAAAAPgE"
  }
}
```

```
},
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "imageids": [
      "188"
    ]
  },
  "id": 1
}
```

Source

CImage::create() in *frontends/php/include/classes/api/services/CImage.php*.

image.delete

Description

object image.delete(array imageIds)

This method allows to delete images.

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 *frontends/php/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.

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 the Base64 encoded image in the <code>image</code> property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>imageid</code> and <code>name</code> . 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": "iVBORwOKGgoAAAANSUgAAABgAAAANCAYAAACzbK7QAAAABHNCSVQICAgIfAhkiAAAAAwSFlzAAACmAA
    }
  ],
  "id": 1
}

```

Source

CImage::get() in *frontends/php/include/classes/api/services/CImage.php*.

image.update

Description

object image.update(object/array images)

This method allows to update existing images.

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 *frontends/php/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
<code>itemid</code>	string	(<i>readonly</i>) ID of the item.
<code>delay</code> (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. <code>{FLEX_INTERVAL}/{FLEX_PERIOD}</code>).
<code>hostid</code> (required)	string	Optional for Zabbix trapper or dependent items. ID of the host or template that the item belongs to. For update operations this field is <i>readonly</i> .

Property	Type	Description
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, Zabbix aggregate, calculated, dependent and database monitor 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; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 7 - Zabbix agent (active); 8 - Zabbix aggregate; 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;
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}.
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 - <i>(default)</i> 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 - <i>(default)</i> password; 1 - public key. HTTP agent authentication method possible values: 0 - <i>(default)</i> none 1 - basic 2 - NTLM
description	string	Description of the item.

Property	Type	Description
error	string	<i>(readonly)</i> Error text if there are problems updating the item.
flags	integer	<i>(readonly)</i> 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 - <i>(default)</i> 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.
http_proxy	string	Default: 90d. 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.
ipmi_sensor	string	Default: 0. IPMI sensor. Used only by IPMI items.
jmx_endpoint	string	JMX agent custom connection string.
lastclock	timestamp	Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi <i>(readonly)</i> Time when the item was last updated.
lastns	integer	This property will only return a value for the period configured in ZBX_HISTORY_PERIOD . <i>(readonly)</i> Nanoseconds when the item was last updated.
lastvalue	string	This property will only return a value for the period configured in ZBX_HISTORY_PERIOD . <i>(readonly)</i> Last value of the item.
logtimefmt	string	This property will only return a value for the period configured in ZBX_HISTORY_PERIOD . 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 999 are allowed.
mtime	timestamp	Required by dependent items. Time when the monitored log file was last updated. Used only by log items.
output_format	integer	HTTP agent item field. Should response converted to JSON. 0 - <i>(default)</i> Store raw. 1 - Convert to JSON.

Property	Type	Description
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.
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.
port	string	Port monitored by the item. Used only by SNMP items.
post_type	integer	HTTP agent item field. Type of post data body stored in posts property. 0 - <i>(default)</i> 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	<i>(readonly)</i> Previous value of the item.
privatekey	string	This property will only return a value for the period configured in ZBX_HISTORY_PERIOD . 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 - <i>(default)</i> GET 1 - POST 2 - PUT 3 - HEAD
retrieve_mode	integer	HTTP agent item field. What part of response should be stored. 0 - <i>(default)</i> Body. 1 - Headers. 2 - Both body and headers will be stored.
snmp_community	string	For request_method HEAD only 1 is allowed value. SNMP community. Used only by SNMPv1 and SNMPv2 items.
snmp_oid	string	SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 items.
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 items.
snmpv3_contextname	string	Possible values: 0 - <i>(default)</i> MD5; 1 - SHA. SNMPv3 context name. Used only by SNMPv3 items.
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 items.
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 items. Possible values: 0 - <i>(default)</i> DES; 1 - AES.

Property	Type	Description
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 items. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 items.
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.
templateid	string	Example: 200,200-{\$M},{\$M},200-400 (readonly) ID of the parent template item.
timeout	string	<i>Hint:</i> Use the <code>hostid</code> property to specify the template that the item belongs to. HTTP agent item field. Item data polling request timeout. Support user macros.
trapper_hosts	string	default: 3s maximum value: 60s 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.
units	string	Default: 365d. Value units.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent items.
valuemapid	string	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. 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.

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.
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 <code>error_handler</code>. Must be empty, if <code>error_handler</code> is 0 or 1. Can be empty if, <code>error_handler</code> is 2. Cannot be empty, if <code>error_handler</code> is 3.

The following parameters and error handlers are supported for each preprocessing type.

Preprocessing type	Name	Parameter 1	Parameter 2	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

Preprocessing type	Name	Parameter 1	Parameter 2	Supported error handlers
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		
17	Check for error in XML	path ⁴		
18	Check for error using regular expression	pattern ³	output ²	
19	Discard unchanged			
20	Discard unchanged with heartbeat	seconds ^{5, 6}		
21	JavaScript	script ²		
22	Prometheus pattern	pattern ^{6, 7}	output ^{6, 8}	0, 1, 2, 3
23	Prometheus to JSON	pattern ^{6, 7}		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>.

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.

Parameters

(object/array) Items to create.

Additionally to the **standard item properties**, the method accepts the following parameters.

Parameter	Type	Description
applications	array	IDs of the applications to add the item to.
preprocessing	array	Item preprocessing options.

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 to monitor free disk space on host with ID "30074" and add it to two applications.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "item.create",
  "params": {
    "name": "Free disk space on $1",
    "key_": "vfs.fs.size[/home/joe/,free]",
    "hostid": "30074",
```



```

        "type": 0,
        "value_type": 3,
        "interfaceid": "30084",
        "applications": [
            "609",
            "610"
        ],
        "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_community": "{$SNMP_COMMUNITY}",
  "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"
      }
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 2
}

```

Response:

```

{
  "jsonrpc": "2.0",
  "result": {
    "itemids": [
      "23865"
    ]
  },
  "id": 3
}

```

Source

CItem::create() in *frontends/php/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.

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 *frontends/php/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.

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.

Parameter	Type	Description
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.
applicationids	string/array	Return only items that belong to the given applications.
webitems	flag	Include web items in the result.
inherited	boolean	If set to <code>true</code> return only items inherited from a template.
templated	boolean	If set to <code>true</code> return only items that belong to templates.
monitored	boolean	If set to <code>true</code> 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.
application	string	Return only items that belong to an application with the given name.
with_triggers	boolean	If set to <code>true</code> return only items that are used in triggers.
selectHosts	query	Returns the host that the item belongs to as an array in the <code>hosts</code> property.
selectInterfaces	query	Returns the host interface used by the item as an array in the <code>interfaces</code> property.
selectTriggers	query	Return triggers that the item is used in in the <code>triggers</code> property.
selectGraphs	query	Supports count. Return graphs that contain the item in the <code>graphs</code> property.
selectApplications	query	Supports count. Return the applications that the item belongs to in the <code>applications</code> property.
selectDiscoveryRule	query	Return the LLD rule that created the item in the <code>discoveryRule</code> property.
selectItemDiscovery	query	Return the item discovery object in the <code>itemDiscovery</code> property. The item discovery object links the item to an item prototype from which it was created.
		It has the following properties: <code>itemdiscoveryid</code> - (string) ID of the item discovery; <code>itemid</code> - (string) ID of the discovered item; <code>parent_itemid</code> - (string) ID of the item prototype from which the item has been created; <code>key_</code> - (string) key of the item prototype; <code>lastcheck</code> - (timestamp) time when the item was last discovered; <code>ts_delete</code> - (timestamp) time when an item that is no longer discovered will be deleted.

Parameter	Type	Description
selectPreprocessing	query	Return item preprocessing options in the <code>preprocessing</code> property. It has the following properties: <code>type</code> - (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. <code>params</code> - (string) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character. <code>error_handler</code> - (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. <code>error_handler_params</code> - (string) Error handler parameters.
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: <code>host</code> - 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: <code>selectGraphs</code> - results will be sorted by name; <code>selectTriggers</code> - results will be sorted by description.
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>itemid</code>, <code>name</code>, <code>key_</code>, <code>delay</code>, <code>history</code>, <code>trends</code>, <code>type</code> and <code>status</code>.

Parameter	Type	Description
countOutput	boolean	These parameters being common for all get methods are described in detail in the reference commentary page.
editable	boolean	
excludeSearch	boolean	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Finding items by key

Retrieve all items from host with ID "10084" that have the word "system" in the key and sort them by name.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "item.get",
  "params": {
    "output": "extend",
    "hostids": "10084",
    "search": {
      "key_": "system"
    },
    "sortfield": "name"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "itemid": "23298",
      "type": "0",
      "snmp_community": "",
      "snmp_oid": "",
      "hostid": "10084",
      "name": "Context switches per second",
      "key_": "system.cpu.switches",
      "delay": "1m",
      "history": "7d",
      "trends": "365d",
      "lastvalue": "2552",
      "lastclock": "1351090998",
      "prevvalue": "2641",
      "state": "0",
      "status": "0",
    }
  ]
}
```

```

"value_type": "3",
"trapper_hosts": "",
"units": "sps",
"snmpv3_securityname": "",
"snmpv3_securitylevel": "0",
"snmpv3_authpassphrase": "",
"snmpv3_privpassphrase": "",
"snmpv3_authprotocol": "0",
"snmpv3_privprotocol": "0",
"snmpv3_contextname": "",
"error": "",
"lastlogsize": "0",
"logtimefmt": "",
"templateid": "22680",
"valuemapid": "0",
"params": "",
"ipmi_sensor": "",
"authtype": "0",
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
"mtime": "0",
"lastns": "564054253",
"flags": "0",
"interfaceid": "1",
"port": "",
"description": "",
"inventory_link": "0",
"lifetime": "0s",
"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"
},
{
    "itemid": "23299",
    "type": "0",
    "snmp_community": "",
    "snmp_oid": "",
    "hostid": "10084",
    "name": "CPU $2 time",
    "key_": "system.cpu.util[,idle]",
    "delay": "1m",
    "history": "7d",

```



```

    "trends": "365d",
    "lastvalue": "86.031879",
    "lastclock": "1351090999",
    "prevvalue": "85.306944",
    "state": "0",
    "status": "0",
    "value_type": "0",
    "trapper_hosts": "",
    "units": "%",
    "snmpv3_securityname": "",
    "snmpv3_securitylevel": "0",
    "snmpv3_authpassphrase": "",
    "snmpv3_privpassphrase": "",
    "snmpv3_authprotocol": "0",
    "snmpv3_privprotocol": "0",
    "snmpv3_contextname": "",
    "error": "",
    "lastlogsize": "0",
    "logtimefmt": "",
    "templateid": "17354",
    "valuemapid": "0",
    "params": "",
    "ipmi_sensor": "",
    "authtype": "0",
    "username": "",
    "password": "",
    "publickey": "",
    "privatekey": "",
    "mtime": "0",
    "lastns": "564256864",
    "flags": "0",
    "interfaceid": "1",
    "port": "",
    "description": "The time the CPU has spent doing nothing.",
    "inventory_link": "0",
    "lifetime": "0s",
    "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"
  },
  {
    "itemid": "23300",
    "type": "0",
    "snmp_community": "",

```

```

"snmp_oid": "",
"hostid": "10084",
"name": "CPU $2 time",
"key_": "system.cpu.util[,interrupt]",
"history": "7d",
"trends": "365d",
"lastvalue": "0.008389",
"lastclock": "1351091000",
"prevvalue": "0.000000",
"state": "0",
"status": "0",
"value_type": "0",
"trapper_hosts": "",
"units": "%",
"snmpv3_securityname": "",
"snmpv3_securitylevel": "0",
"snmpv3_authpassphrase": "",
"snmpv3_privpassphrase": "",
"snmpv3_authprotocol": "0",
"snmpv3_privprotocol": "0",
"snmpv3_contextname": "",
"error": "",
"lastlogsize": "0",
"logtimefmt": "",
"templateid": "22671",
"valuemapid": "0",
"params": "",
"ipmi_sensor": "",
"authtype": "0",
"username": "",
"password": "",
"publickey": "",
"privatekey": "",
"mtime": "0",
"lastns": "564661387",
"flags": "0",
"interfaceid": "1",
"port": "",
"description": "The amount of time the CPU has been servicing hardware interrupts.",
"inventory_link": "0",
"lifetime": "0s",
"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"

```

```

    }
  ],
  "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_community": "",
      "snmp_oid": "",
      "hostid": "10116",
      "name": "Days",
      "key_": "apache.status.uptime.days",
      "delay": "",
      "history": "90d",
      "trends": "365d",
      "status": "0",
      "value_type": "3",
      "trapper_hosts": "",
      "units": "",
      "snmpv3_securityname": "",
      "snmpv3_securitylevel": "0",
      "snmpv3_authpassphrase": "",
      "snmpv3_privpassphrase": "",
      "formula": "",
      "error": "",
      "lastlogsize": "0",
      "logtimefmt": "",
      "templateid": "0",
      "valuemapid": "0",
      "params": "",
      "ipmi_sensor": "",
      "authtype": "0",
      "username": "",
      "password": "",
      "publickey": "",
      "privatekey": "",

```

```

    "mtime": "0",
    "flags": "0",
    "interfaceid": "0",
    "port": "",
    "description": "",
    "inventory_link": "0",
    "lifetime": "30d",
    "snmpv3_authprotocol": "0",
    "snmpv3_privprotocol": "0",
    "state": "0",
    "snmpv3_contextname": "",
    "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",
    "lastclock": "0",
    "lastns": "0",
    "lastvalue": "0",
    "prevvalue": "0"
  },
  {
    "itemid": "25555",
    "type": "18",
    "snmp_community": "",
    "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": "",
    "snmpv3_securityname": "",
    "snmpv3_securitylevel": "0",
    "snmpv3_authpassphrase": "",
    "snmpv3_privpassphrase": "",
    "formula": "",
    "error": "",
    "lastlogsize": "0",
    "logtimefmt": "",
    "templateid": "0",
    "valuemapid": "0",

```

```

        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "mtime": "0",
        "flags": "0",
        "interfaceid": "0",
        "port": "",
        "description": "",
        "inventory_link": "0",
        "lifetime": "30d",
        "snmpv3_authprotocol": "0",
        "snmpv3_privprotocol": "0",
        "state": "0",
        "snmpv3_contextname": "",
        "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",
        "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_community": "",
      "snmp_oid": "",
      "hostid": "10255",
      "name": "template item",
      "key_": "ti",
      "delay": "30s",
      "history": "90d",
      "trends": "365d",
      "status": "0",
      "value_type": "3",
      "trapper_hosts": "",
      "units": "",
      "snmpv3_securityname": "",
      "snmpv3_securitylevel": "0",
      "snmpv3_authpassphrase": "",
      "snmpv3_privpassphrase": "",
      "formula": "",
      "error": "",
      "lastlogsize": "0",
      "logtimefmt": "",
      "templateid": "0",
      "valuemapid": "0",
      "params": "",
      "ipmi_sensor": "",
      "authtype": "0",
      "username": "",
      "password": "",
      "publickey": "",
      "privatekey": "",
      "mtime": "0",
      "flags": "0",
      "interfaceid": "0",
      "port": "",
      "description": "",
      "inventory_link": "0",
      "lifetime": "30d",
      "snmpv3_authprotocol": "0",
      "snmpv3_privprotocol": "0",
      "state": "0",
      "snmpv3_contextname": "",
      "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",
        "lastclock": "0",
        "lastns": "0",
        "lastvalue": "0",
        "prevvalue": "0"
    }
],
    "id": 3
}

```

Retrieving items with preprocessing rules

Reatrive all items and their preprocessing rules from host with ID "10254".

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

- [Application](#)
- [Discovery rule](#)
- [Graph](#)

- [Host](#)
- [Host interface](#)
- [Trigger](#)

Source

`CItem::get()` in *frontends/php/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.

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
<code>applications</code>	array	IDs of the applications to replace the current applications.
<code>preprocessing</code>	array	Item preprocessing options to replace the current preprocessing options.

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
}

```

Source

CItem::update() in *frontends/php/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	(<i>readonly</i>) ID of the item prototype.

Property	Type	Description
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. <code>{ \$FLEX_INTERVAL } / { \$FLEX_PERIOD }</code>).
hostid (required)	string	Optional for Zabbix trapper or dependent item. ID of the host that the item prototype belongs to.
ruleid (required)	string	For update operations this field is <i>readonly</i> . ID of the LLD rule that the item belongs to.
interfaceid (required)	string	For update operations this field is <i>readonly</i> . ID of the item prototype's host interface. Used only for host item prototypes.
key_ (required)	string	Not required for Zabbix agent (active), Zabbix internal, Zabbix trapper, Zabbix aggregate, calculated, dependent and database monitor item prototypes. Item prototype key.
name (required)	string	Name of the item prototype.
type (required)	integer	Type of the item prototype. Possible values: 0 - Zabbix agent; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 7 - Zabbix agent (active); 8 - Zabbix aggregate; 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;
url (required)	string	URL string required only for HTTP agent item prototypes. Supports LLD macros, user macros, <code>{HOST.IP}</code> , <code>{HOST.CONN}</code> , <code>{HOST.DNS}</code> , <code>{HOST.HOST}</code> , <code>{HOST.NAME}</code> , <code>{ITEM.ID}</code> , <code>{ITEM.KEY}</code> .

Property	Type	Description
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 - <i>(default)</i> 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 - <i>(default)</i> password; 1 - public key. HTTP agent authentication method possible values: 0 - <i>(default)</i> none 1 - basic 2 - NTLM
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 - <i>(default)</i> 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" }
history	string	A time unit of how long the history data should be stored. Also accepts user macro and LLD macro.
http_proxy	string	Default: 90d. 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.
logtimefmt	string	Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi 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 999 are allowed.
output_format	integer	Required by Dependent items. HTTP agent item prototype field. Should response converted to JSON. 0 - <i>(default)</i> Store raw. 1 - Convert to JSON.

Property	Type	Description
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.
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent item prototypes.
port	string	Port monitored by the item prototype. Used only by SNMP items prototype.
post_type	integer	HTTP agent item prototype field. Type of post data body stored in posts property. 0 - <i>(default)</i> 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 - <i>(default)</i> GET 1 - POST 2 - PUT 3 - HEAD
retrieve_mode	integer	HTTP agent item prototype field. What part of response should be stored. 0 - <i>(default)</i> Body. 1 - Headers. 2 - Both body and headers will be stored.
snmp_community	string	For request_method HEAD only 1 is allowed value. SNMP community.
snmp_oid	string	Used only by SNMPv1 and SNMPv2 item prototypes. SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 item prototypes.
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 items. Possible values: 0 - <i>(default)</i> MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 item prototypes.
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 item prototypes.
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 items. Possible values: 0 - <i>(default)</i> DES; 1 - AES.

Property	Type	Description
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 item prototypes. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 item prototypes.
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.
status	integer	Status of the item prototype. Possible values: 0 - <i>(default)</i> 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 (readonly) ID of the parent template item prototype.
templateid	string	HTTP agent item prototype field. Item data polling request timeout. Support user macros and LLD macros.
timeout	string	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.
units	string	Default: 365d. Value units.
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent item prototypes.
valuemapid	string	Required by SSH and Telnet item prototypes. 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 - <i>(default)</i> Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent item prototype field. Validate is host certificate authentic. 0 - <i>(default)</i> Do not validate. 1 - Validate.

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.
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 <code>error_handler</code>. Must be empty, if <code>error_handler</code> is 0 or 1. Can be empty if, <code>error_handler</code> is 2. Cannot be empty, if <code>error_handler</code> is 3.

The following parameters and error handlers are supported for each preprocessing type.

Preprocessing type	Name	Parameter 1	Parameter 2	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		

Preprocessing type	Name	Parameter 1	Parameter 2	Supported error handlers
17	Check for error in XML	path ⁴	output ²	
18	Check for error using regular expression	pattern ³		
19	Discard unchanged			
20	Discard unchanged with heartbeat	seconds ^{5, 6}	output ^{6, 8}	0, 1, 2, 3
21	JavaScript	script ²		
22	Prometheus pattern	pattern ^{6, 7}		
23	Prometheus to JSON	pattern ^{6, 7}		

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>.

itemprototype.create

Description

object itemprototype.create(object/array itemPrototypes)

This method allows to create new item prototypes.

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.
applications	array	IDs of applications to be assigned to the discovered items.
applicationPrototypes	array	Names of application prototypes to be assigned to the item prototype.
preprocessing	array	Item prototype preprocessing options.

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 $1",
    "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 $1",
        "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
}

```

Source

CItemPrototype::create() in *frontends/php/include/classes/api/services/CItemPrototype.php*.

itemprototype.delete

Description

object itemprototype.delete(array itemPrototypeIds)

This method allows to delete item prototypes.

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 *frontends/php/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.

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 <code>true</code> return only item prototypes inherited from a template.
itemids	string/array	Return only item prototypes with the given IDs.
monitored	boolean	If set to <code>true</code> return only enabled item prototypes that belong to monitored hosts.
templated	boolean	If set to <code>true</code> 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.
selectApplications	query	Return applications that the item prototype belongs to in the <code>applications</code> property.
selectApplicationPrototypes	query	Return application prototypes linked to item prototype in <code>applicationPrototypes</code> property.
selectDiscoveryRule	query	Return the low-level discovery rule that the graph prototype belongs to in the <code>discoveryRule</code> property.
selectGraphs	query	Return graph prototypes that the item prototype is used in in the <code>graphs</code> property.
selectHosts	query	Supports count. Returns the host that the item prototype belongs to as an array in the <code>hosts</code> property.
selectTriggers	query	Return trigger prototypes that the item prototype is used in in the <code>triggers</code> property.

Supports count.

Parameter	Type	Description
selectPreprocessing	query	Return item preprocessing options in the <code>preprocessing</code> property. It has the following properties: <code>type</code> - (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. <code>params</code> - (string) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character. <code>error_handler</code> - (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. <code>error_handler_params</code> - (string) Error handler parameters.
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: <code>host</code> - 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: <code>selectGraphs</code> - results will be sorted by name; <code>selectTriggers</code> - results will be sorted by description.
sortfield	string/array	Sort the result by the given properties. Possible values are: <code>itemid</code>, <code>name</code>, <code>key_</code>, <code>delay</code>, <code>type</code> and <code>status</code>.

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 from an LLD rule.

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_community": "",
      "snmp_oid": "",
      "hostid": "10079",
      "name": "Incoming network traffic on $1",
      "key_": "net.if.in[en0]",
      "delay": "1m",
      "history": "1w",
      "trends": "365d",
      "status": "0",
      "value_type": "3",
      "trapper_hosts": "",
      "units": "bps",
      "snmpv3_securityname": "",
      "snmpv3_securitylevel": "0",
      "snmpv3_authpassphrase": "",
      "snmpv3_privpassphrase": "",
      "formula": "",
      "error": ""
    }
  ]
}
```

```

    "lastlogsize": "0",
    "logtimefmt": "",
    "templateid": "0",
    "valuemapid": "0",
    "params": "",
    "ipmi_sensor": "",
    "authtype": "0",
    "username": "",
    "password": "",
    "publickey": "",
    "privatekey": "",
    "mtime": "0",
    "flags": "0",
    "interfaceid": "0",
    "port": "",
    "description": "",
    "inventory_link": "0",
    "lifetime": "30d",
    "snmpv3_authprotocol": "0",
    "snmpv3_privprotocol": "0",
    "state": "0",
    "snmpv3_contextname": "",
    "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",
    "lastclock": "0",
    "lastns": "0",
    "lastvalue": "0",
    "prevvalue": "0"
},
{
    "itemid": "10010",
    "type": "0",
    "snmp_community": "",
    "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": "",
        "snmpv3_securityname": "",
        "snmpv3_securitylevel": "0",
        "snmpv3_authpassphrase": "",
        "snmpv3_privpassphrase": "",
        "formula": "",
        "error": "",
        "lastlogsize": "0",
        "logtimefmt": "",
        "templateid": "0",
        "valuemapid": "0",
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "mtime": "0",
        "flags": "0",
        "interfaceid": "0",
        "port": "",
        "description": "The processor load is calculated as system CPU load divided by number of CPU c
        "inventory_link": "0",
        "lifetime": "0",
        "snmpv3_authprotocol": "0",
        "snmpv3_privprotocol": "0",
        "state": "0",
        "snmpv3_contextname": "",
        "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",
        "lastclock": "0",
        "lastns": "0",
        "lastvalue": "0",
        "prevvalue": "0"
    }
],
    "id": 1
}

```

Finding dependent item

Find one Dependent item for item with ID "25545".

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_community": "",
      "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": "",
      "snmpv3_securityname": "",
      "snmpv3_securitylevel": "0",
      "snmpv3_authpassphrase": "",
      "snmpv3_privpassphrase": "",
      "formula": "",
      "error": "",
      "lastlogsize": "0",
      "logtimefmt": "",
      "templateid": "0",
      "valuemapid": "0",
      "params": "",
      "ipmi_sensor": "",
      "authtype": "0",
      "username": "",
      "password": "",
      "publickey": "",
      "privatekey": "",
      "mtime": "0",
      "flags": "0",
      "interfaceid": "0",
      "port": "",
      "description": "",
      "inventory_link": "0",
      "lifetime": "30d",
      "snmpv3_authprotocol": "0",
      "snmpv3_privprotocol": "0",
      "state": "0",
      "snmpv3_contextname": "",

```

```

        "evaltype": "0",
        "master_itemid": "25545",
        "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",
        "lastclock": "0",
        "lastns": "0",
        "lastvalue": "0",
        "prevvalue": "0"
    }
],
    "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_community": "",
            "snmp_oid": "",
            "hostid": "10254",
            "name": "discovered",
            "key_": "item[{"#INAME}]",
            "delay": "{"#IUPDATE}"
        }
    ]
}

```

```

    "history": "90d",
    "trends": "30d",
    "status": "0",
    "value_type": "3",
    "trapper_hosts": "",
    "units": "",
    "snmpv3_securityname": "",
    "snmpv3_securitylevel": "0",
    "snmpv3_authpassphrase": "",
    "snmpv3_privpassphrase": "",
    "formula": "",
    "error": "",
    "lastlogsize": "0",
    "logtimefmt": "",
    "templateid": "28255",
    "valuemapid": "0",
    "params": "",
    "ipmi_sensor": "",
    "authtype": "0",
    "username": "",
    "password": "",
    "publickey": "",
    "privatekey": "",
    "mtime": "0",
    "flags": "2",
    "interfaceid": "2",
    "port": "",
    "description": "",
    "inventory_link": "0",
    "lifetime": "30d",
    "snmpv3_authprotocol": "0",
    "snmpv3_privprotocol": "0",
    "state": "0",
    "snmpv3_contextname": "",
    "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"
  }
],
  "id": 17
}

```

See also

- [Application](#)

- [Host](#)
- [Graph prototype](#)
- [Trigger prototype](#)

Source

CltemPrototype::get() in *frontends/php/include/classes/api/services/CltemPrototype.php*.

itemprototype.update

Description

`object itemprototype.update(object/array itemPrototypes)`

This method allows to update existing item prototypes.

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
<code>applications</code>	array	IDs of the applications to replace the current applications.
<code>applicationPrototypes</code>	array	Names of the application prototypes to replace the current application prototypes.
<code>preprocessing</code>	array	Item prototype preprocessing options to replace the current preprocessing options.

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
}
```

Source

CItemPrototype::update() in *frontends/php/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	(<i>readonly</i>) ID of the LLD rule.

Property	Type	Description
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.
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.
key_ (required)	string	Not required for Zabbix agent (active), Zabbix internal, Zabbix trapper, dependent and database monitor LLD rules. LLD rule key.
name (required)	string	Name of the LLD rule.
type (required)	integer	Type of the LLD rule. Possible values: 0 - Zabbix agent; 1 - SNMPv1 agent; 2 - Zabbix trapper; 3 - simple check; 4 - SNMPv2 agent; 5 - Zabbix internal; 6 - SNMPv3 agent; 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;
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}.
allow_traps	integer	HTTP agent LLD rule field. Allow to populate value as in trapper item type also. 0 - (<i>default</i>) 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 - (<i>default</i>) password; 1 - public key. HTTP agent authentication method possible values: 0 - (<i>default</i>) none 1 - basic 2 - NTLM
description	string	Description of the LLD rule.
error	string	(<i>readonly</i>) Error text if there are problems updating the LLD rule.

Property	Type	Description
follow_redirects	integer	HTTP agent LLD rule field. Follow response redirects while pooling data. 0 - Do not follow redirects. 1 - <i>(default)</i> 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.
lifetime	string	Default value: service:jmx:rmi:///jndi/rmi://{HOST.CONN}:{HOST.PORT}/jmxrmi Time period after which items that are no longer discovered will be deleted. Accepts seconds, time unit with suffix and user macro.
master_itemid	integer	Default: 30d. 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.
output_format	integer	Required for Dependent item. HTTP agent LLD rule field. Should response converted to JSON. 0 - <i>(default)</i> 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.
password	string	Password for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD rules.
port	string	Port used by the LLD rule. Used only by SNMP LLD rules.
post_type	integer	HTTP agent LLD rule field. Type of post data body stored in posts property. 0 - <i>(default)</i> Raw data. 2 - JSON data. 3 - XML data.
posts	string	HTTP agent LLD rule 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 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 - <i>(default)</i> GET 1 - POST 2 - PUT 3 - HEAD

Property	Type	Description
retrieve_mode	integer	HTTP agent LLD rule field. What part of response should be stored. 0 - <i>(default)</i> Body. 1 - Headers. 2 - Both body and headers will be stored.
snmp_community	string	For request_method HEAD only 1 is allowed value. SNMP community.
snmp_oid	string	Required for SNMPv1 and SNMPv2 LLD rules. SNMP OID.
snmpv3_authpassphrase	string	SNMPv3 auth passphrase. Used only by SNMPv3 LLD rules.
snmpv3_authprotocol	integer	SNMPv3 authentication protocol. Used only by SNMPv3 LLD rules. Possible values: 0 - <i>(default)</i> MD5; 1 - SHA.
snmpv3_contextname	string	SNMPv3 context name. Used only by SNMPv3 checks.
snmpv3_privpassphrase	string	SNMPv3 priv passphrase. Used only by SNMPv3 LLD rules.
snmpv3_privprotocol	integer	SNMPv3 privacy protocol. Used only by SNMPv3 LLD rules. Possible values: 0 - <i>(default)</i> DES; 1 - AES.
snmpv3_securitylevel	integer	SNMPv3 security level. Used only by SNMPv3 LLD rules. Possible values: 0 - noAuthNoPriv; 1 - authNoPriv; 2 - authPriv.
snmpv3_securityname	string	SNMPv3 security name. Used only by SNMPv3 LLD rules.
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	<i>(readonly)</i> State of the LLD rule. Possible values: 0 - <i>(default)</i> normal; 1 - not supported.
status	integer	Status of the LLD rule. Possible values: 0 - <i>(default)</i> 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.
templateid	string	Example: 200,200-{\$M},{M},200-400 <i>(readonly)</i> ID of the parent template LLD rule.
timeout	string	HTTP agent LLD rule field. Item data polling request timeout. Support user macros. default: 3s maximum value: 60s
trapper_hosts	string	Allowed hosts. Used by trapper LLD rules or HTTP agent LLD rules.

Property	Type	Description
username	string	Username for authentication. Used by simple check, SSH, Telnet, database monitor, JMX and HTTP agent LLD rules.
verify_host	integer	Required by SSH and Telnet LLD rules. 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 - <i>(default)</i> Do not validate. 1 - Validate.
verify_peer	integer	HTTP agent LLD rule field. Validate is host certificate authentic. 0 - <i>(default)</i> Do not validate. 1 - Validate.

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	<i>(readonly)</i> Generated expression that will be used for evaluating filter conditions. The expression contains IDs that reference specific filter conditions by its <code>formulaid</code> . The value of <code>eval_formula</code> is equal to the value of <code>formula</code> 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 <code>formulaid</code> . 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.

Property	Type	Description
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 - <i>(default)</i> matches regular expression; 9 - does not match regular expression.

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; 12 - JSONPath; 15 - Does not match regular expression; 16 - Check for error in JSON; 20 - Discard unchanged with heartbeat; 23 - Prometheus 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	Supported error handlers
5	Regular expression	pattern ¹	output ²	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		
20	Discard unchanged with heartbeat	seconds ^{4, 5, 6}		
23	Prometheus to JSON	pattern ^{5, 7}		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>.

discoveryrule.copy

Description

object discoveryrule.copy(object parameters)

This method allows to copy LLD rules with all of the prototypes to the given hosts.

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 *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.create

Description

object discoveryrule.create(object/array lldRules)

This method allows to create new LLD rules.

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.

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"
        },
        {
          "macro": "{#MACRO3}",
          "value": "@regex3"
        }
      ]
    }
  },
  "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",
    "id": 35,
    "auth": "d678e0b85688ce578ff061bd29a20d3b",
  }
}

```

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
}

```

See also

- [LLD rule filter](#)
- [LLD macro paths](#)
- [LLD rule preprocessing](#)

Source

CDiscoveryRule::create() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.delete

Description

object discoveryrule.delete(array lldRuleIds)

This method allows to delete LLD rules.

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 *frontends/php/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.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
<code>itemids</code>	<code>string/array</code>	Return only LLD rules with the given IDs.
<code>hostids</code>	<code>string/array</code>	Return only LLD rules that belong to the given hosts.
<code>inherited</code>	<code>boolean</code>	If set to <code>true</code> return only LLD rules inherited from a template.
<code>interfaceids</code>	<code>string/array</code>	Return only LLD rules use the given host interfaces.
<code>monitored</code>	<code>boolean</code>	If set to <code>true</code> return only enabled LLD rules that belong to monitored hosts.
<code>templated</code>	<code>boolean</code>	If set to <code>true</code> return only LLD rules that belong to templates.
<code>templateids</code>	<code>string/array</code>	Return only LLD rules that belong to the given templates.
<code>selectFilter</code>	<code>query</code>	Returns the filter used by the LLD rule in the <code>filter</code> property.

Parameter	Type	Description
selectGraphs	query	Returns graph prototypes that belong to the LLD rule in the <code>graphs</code> property.
selectHostPrototypes	query	Supports <code>count</code> . Returns host prototypes that belong to the LLD rule in the <code>hostPrototypes</code> property.
selectHosts	query	Supports <code>count</code> . Returns the host that the LLD rule belongs to as an array in the <code>hosts</code> property.
selectItems	query	Returns item prototypes that belong to the LLD rule in the <code>items</code> property.
selectTriggers	query	Supports <code>count</code> . Returns trigger prototypes that belong to the LLD rule in the <code>triggers</code> property.
selectApplicationPrototypes	query	Supports <code>count</code> . Returns application prototypes that belong to all item prototypes that belong to this LLD rule in the <code>applicationPrototypes</code> property.
selectLLDMacroPaths	query	Returns list of LLD macros and paths to values assigned to each corresponding macro in <code>lld_macro_paths</code> property.
selectPreprocessing	query	Return LLD rule preprocessing options in the <code>preprocessing</code> property. It has the following properties: <code>type</code> - (string) The preprocessing option type: 5 - Regular expression matching; 12 - JSONPath; 15 - Does not match regular expression; 16 - Check for error in JSON; 20 - Discard unchanged with heartbeat; 23 - Prometheus to JSON. <code>params</code> - (string) Additional parameters used by preprocessing option. Multiple parameters are separated by LF (\n) character. <code>error_handler</code> - (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. <code>error_handler_params</code> - (string) Error handler parameters.
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: <code>host</code> - technical name of the host that the LLD rule belongs to.

Parameter	Type	Description
limitSelects	integer	Limits the number of records returned by subselects.
sortfield	string/array	Applies to the following subselects: selectItems; selectGraphs; selectTriggers. 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 from host "10202".

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_community": "",
      "snmp_oid": "",
      "hostid": "10202",
      "name": "Network interface discovery",
      "key_": "net.if.discovery",
      "delay": "1h",
      "state": "0",
      "status": "0",
    }
  ]
}
```

```

    "trapper_hosts": "",
    "snmpv3_securityname": "",
    "snmpv3_securitylevel": "0",
    "snmpv3_authpassphrase": "",
    "snmpv3_privpassphrase": "",
    "error": "",
    "templateid": "22444",
    "params": "",
    "ipmi_sensor": "",
    "authtype": "0",
    "username": "",
    "password": "",
    "publickey": "",
    "privatekey": "",
    "interfaceid": "119",
    "port": "",
    "description": "Discovery of network interfaces as defined in global regular expression \\"Netw
    "lifetime": "30d",
    "snmpv3_authprotocol": "0",
    "snmpv3_privprotocol": "0",
    "snmpv3_contextname": "",
    "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"
},
{
    "itemid": "27426",
    "type": "0",
    "snmp_community": "",
    "snmp_oid": "",
    "hostid": "10202",
    "name": "Mounted filesystem discovery",
    "key_": "vfs.fs.discovery",
    "delay": "1h",
    "state": "0",
    "status": "0",
    "trapper_hosts": "",
    "snmpv3_securityname": "",
    "snmpv3_securitylevel": "0",
    "snmpv3_authpassphrase": "",
    "snmpv3_privpassphrase": "",
    "error": "",
    "templateid": "22450",
    "params": "",
    "ipmi_sensor": "",
    "authtype": "0",

```

```

        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "interfaceid": "119",
        "port": "",
        "description": "Discovery of file systems of different types as defined in global regular expressions",
        "lifetime": "30d",
        "snmpv3_authprotocol": "0",
        "snmpv3_privprotocol": "0",
        "snmpv3_contextname": "",
        "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"
    }
],
    "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": "8",
                "formulaid": "B"
            },
            {
                "macro": "{#MACRO3}",
                "value": "@regex3",
                "operator": "8",
                "formulaid": "C"
            }
        ],
        "eval_formula": "A and B and C"
    }
},
    "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_community": "",
            "snmp_oid": "",
            "hostid": "10257",
            "name": "API HTTP agent",
            "key_": "api_discovery_rule",
            "delay": "5s",
            "history": "90d",
            "trends": "0",

```

```

        "status": "0",
        "value_type": "4",
        "trapper_hosts": "",
        "units": "",
        "snmpv3_securityname": "",
        "snmpv3_securitylevel": "0",
        "snmpv3_authpassphrase": "",
        "snmpv3_privpassphrase": "",
        "error": "",
        "lastlogsize": "0",
        "logtimefmt": "",
        "templateid": "0",
        "valuemapid": "0",
        "params": "",
        "ipmi_sensor": "",
        "authtype": "0",
        "username": "",
        "password": "",
        "publickey": "",
        "privatekey": "",
        "mtime": "0",
        "flags": "1",
        "interfaceid": "5",
        "port": "",
        "description": "",
        "inventory_link": "0",
        "lifetime": "30d",
        "snmpv3_authprotocol": "0",
        "snmpv3_privprotocol": "0",
        "state": "0",
        "snmpv3_contextname": "",
        "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"
    },
    ],
    ],

```



```

    "id": 39
}

```

See also

- [Graph prototype](#)
- [Host](#)
- [Item prototype](#)
- [LLD rule filter](#)
- [Trigger prototype](#)

Source

CDiscoveryRule::get() in *frontends/php/include/classes/api/services/CDiscoveryRule.php*.

discoveryrule.update

Description

object discoveryrule.update(object/array lldRules)

This method allows to update existing LLD rules.

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.

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
}

```

Source

CDiscoveryRule::update() in *frontends/php/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	(<i>readonly</i>) ID of the maintenance.
name (required)	string	Name of the maintenance.
active_since (required)	timestamp	Time when the maintenance becomes active.
active_till (required)	timestamp	Time when the maintenance stops being active.
description	string	Description of the maintenance.
maintenance_type	integer	Type of maintenance. Possible values: 0 - (<i>default</i>) with data collection; 1 - without data collection.
tags_evaltype	integer	Problem tag evaluation method. Possible values: 0 - (<i>default</i>) And/Or; 2 - Or.

Time period

The time period object is used to define periods when the maintenance must come into effect. It has the following properties.

Property	Type	Description
timeperiodid	string	(<i>readonly</i>) ID of the maintenance.
day	integer	Day of the month when the maintenance must come into effect.
dayofweek	integer	Required only for monthly time periods. 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.
every	integer	Used for weekly and monthly time periods. Required only for weekly time periods. For daily and weekly periods every defines day or week intervals at which the maintenance must come into effect. For monthly periods every defines the week of the month when the maintenance must come into effect. Possible values: 1 - first week; 2 - second week; 3 - third week; 4 - fourth week; 5 - last week.

Property	Type	Description
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.
period	integer	Required only for monthly time periods. Duration of the maintenance period in seconds.
start_date	timestamp	Default: 3600. Date when the maintenance period must come into effect. Required only for one time periods.
start_time	integer	Default: current date. Time of day when the maintenance starts in seconds.
timeperiod_type	integer	Required for daily, weekly and monthly periods. Type of time period. Possible values: 0 - <i>(default)</i> one time only; 2 - daily; 3 - weekly; 4 - monthly.

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 - <i>(default)</i> Contains.
value	string	Problem tag value.

maintenance.create

Description

`object maintenance.create(object/array maintenances)`

This method allows to create new maintenances.

Parameters

(object/array) Maintenances to create.

Additionally to the **standard maintenance properties**, the method accepts the following parameters.

Parameter	Type	Description
groupids (required)	array	IDs of the host groups that will undergo maintenance.

Parameter	Type	Description
hostids (required)	array	IDs of the hosts that will undergo maintenance.
timeperiods (required)	array	Maintenance time periods.
tags	array	Problem tags. Define what problems must be suppressed. If no tags are given, all active maintenance host problems will be suppressed.

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 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 "2" with problem tags **service:mysql** 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,
    "groupids": [
      "2"
    ],
    "timeperiods": [
      {
        "timeperiod_type": 3,
        "every": 1,
        "dayofweek": 64,
        "start_time": 64800,
        "period": 3600
      }
    ],
    "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 *frontends/php/include/classes/api/services/CMaintenance.php*.

maintenance.delete

Description

object maintenance.delete(array maintenanceIds)

This method allows to delete maintenances.

Parameters

(array) IDs of the maintenances to delete.

Return values

(object) Returns an object containing the IDs of the deleted maintenances under the `maintenanceids` property.

Examples

Deleting multiple maintenances

Delete two maintenances.

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 *frontends/php/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.

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.
selectGroups	query	Return host groups assigned to the maintenance in the groups property.
selectHosts	query	Return hosts assigned to the maintenance in the hosts property.
selectTimeperiods	query	Return the maintenance's time periods in the timeperiods property.
selectTags	query	Return the maintenance's problem tags in the tags property.
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	
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 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",
    "selectGroups": "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",
      "groups": [
        {
          "groupid": "4",
          "name": "Zabbix servers",
          "internal": "0"
        }
      ],
      "timeperiods": [
        {
          "timeperiodid": "4",
          "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 *frontends/php/include/classes/api/services/CMaintenance.php*.

maintenance.update

Description

object maintenance.update(object/array maintenances)

This method allows to update existing maintenances.

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
groupids	array	IDs of the host groups to replace the current groups.
hostids	array	IDs of the hosts to replace the current hosts.
timeperiods	array	Maintenance time periods to replace the current periods.
tags	array	Problem tags.

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 "3" with two different ones.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "maintenance.update",
  "params": {
    "maintenanceid": "3",
    "hostids": [
      "10085",
      "10084"
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "maintenanceids": [
      "3"
    ]
  },
  "id": 1
}
```

See also

- [Time period](#)

Source

CMaintenance::update() in *frontends/php/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	(<i>readonly</i>) 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 - (<i>default</i>) do not expand macros; 1 - expand macros.
expandproblem	integer	Whether the the problem trigger will be displayed for elements with a single problem. Possible values: 0 - always display the number of problems; 1 - (<i>default</i>) 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 - (<i>default</i>) enable grid aligning.

Property	Type	Description
grid_show	integer	Whether to show the grid on the map.
grid_size	integer	Possible values: 0 - do not show the grid; 1 - <i>(default)</i> show the grid. Size of the map grid in pixels. Supported values: 20, 40, 50, 75 and 100.
highlight	integer	Default: 50. Whether icon highlighting is enabled. Possible values: 0 - highlighting disabled; 1 - <i>(default)</i> highlighting enabled.
iconmapid	string	ID of the icon map used on the map.
label_format	integer	Whether to enable advanced labels.
label_location	integer	Possible values: 0 - <i>(default)</i> disable advanced labels; 1 - enable advanced labels. Location of the map element label.
label_string_host	string	Possible values: 0 - <i>(default)</i> bottom; 1 - left; 2 - right; 3 - top. Custom label for host elements.
label_string_hostgroup	string	Required for maps with custom host label type. Custom label for host group elements.
label_string_image	string	Required for maps with custom host group label type. Custom label for image elements.
label_string_map	string	Required for maps with custom image label type. Custom label for map elements.
label_string_trigger	string	Required for maps with custom map label type. Custom label for trigger elements.
label_type	integer	Required for maps with custom trigger label type. Map element label type. Possible values: 0 - label; 1 - IP address; 2 - <i>(default)</i> element name; 3 - status only; 4 - nothing.
label_type_host	integer	Label type for host elements. Possible values: 0 - label; 1 - IP address; 2 - <i>(default)</i> element name; 3 - status only; 4 - nothing; 5 - custom.

Property	Type	Description
label_type_hostgroup	integer	Label type for host group elements. Possible values: 0 - label; 2 - (<i>default</i>) element name; 3 - status only; 4 - nothing; 5 - custom.
label_type_image	integer	Label type for host group elements. Possible values: 0 - label; 2 - (<i>default</i>) element name; 4 - nothing; 5 - custom.
label_type_map	integer	Label type for map elements. Possible values: 0 - label; 2 - (<i>default</i>) element name; 3 - status only; 4 - nothing; 5 - custom.
label_type_trigger	integer	Label type for trigger elements. Possible values: 0 - label; 2 - (<i>default</i>) 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 - (<i>default</i>) do not highlight elements; 1 - highlight elements.
severity_min	integer	Minimum severity of the triggers that will be displayed on the map.
show_unack	integer	Refer to the trigger "severity" property for a list of supported trigger severities. How problems should be displayed. Possible values: 0 - (<i>default</i>) 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 - (<i>default</i>) private map.
show_suppressed	integer	Whether suppressed problems are shown. Possible values: 0 - (<i>default</i>) hide suppressed problems; 1 - show suppressed problems.

Map element

The map element object defines an object displayed on a map. It has the following properties.

Property	Type	Description
selementid	string	<i>(readonly)</i> ID of the map element.
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 - <i>(default)</i> the host group element will take up the whole map; 1 - the host group element will have a fixed size.
application	string	Name of the application to display problems from. Used only for host and host group map elements.
elementsubtype	integer	How a host group element should be displayed on a map. Possible values: 0 - <i>(default)</i> display the host group as a single element; 1 - display each host in the group separately.
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 - <i>(default)</i> 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	<i>(readonly)</i> ID of the map that the element belongs to.
urls	array	Map element URLs.

The map element URL object is [described in detail below](#).

Property	Type	Description
use_iconmap	integer	Whether icon mapping must be used for host elements.
viewtype	integer	Possible values: 0 - do not use icon mapping; 1 - <i>(default)</i> use icon mapping. Host group element placing algorithm.
width	integer	Possible values: 0 - <i>(default)</i> grid. Width of the fixed size host group element in pixels.
x	integer	Default: 200. X-coordinates of the element in pixels.
y	integer	Default: 0. 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 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	<i>(readonly)</i> ID of the map element URL.
name (required)	string	Link caption.

Property	Type	Description
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	<i>(readonly)</i> 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.
drawtype	integer	Default: 000000. Link line draw style. Possible values: 0 - <i>(default)</i> 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.
permission	integer	The map link trigger object is described in detail below . 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	<i>(readonly)</i> ID of the map link trigger.
triggerid (required)	string	ID of the trigger used as a link indicator.
color	string	Indicator color as a hexadecimal color code.
drawtype	integer	Default: DD0000. Indicator draw style. Possible values: 0 - <i>(default)</i> 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	(<i>readonly</i>) 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.
sysmapid	string	Default: 0. 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	(<i>readonly</i>) 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	(<i>readonly</i>) ID of the map user group.
usrgrpid (required)	string	User group ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

Map shapes

The map shape object defines an geometric shape (with or without text) displayed on a map. It has the following properties:

Property	Type	Description
sysmap_shapeid	string	(<i>readonly</i>) ID of the map shape element.
type (required)	integer	Type of map shape element. Possible values: 0 - rectangle; 1 - ellipse.
x	integer	Property is required when new shapes are created. X-coordinates of the shape in pixels. Default: 0.

Property	Type	Description
y	integer	Y-coordinates of the shape in pixels.
width	integer	Default: 0. Width of the shape in pixels.
height	integer	Default: 200. Height of the shape in pixels.
text	string	Default: 200. 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
font_size	integer	Default: 9. Font size in pixels.
font_color	string	Default: 11. Font color.
text_halign	integer	Default: '000000'. Horizontal alignment of text. Possible values: 0 - center; 1 - left; 2 - right.
text_valign	integer	Default: 0. Vertical alignment of text. Possible values: 0 - middle; 1 - top; 2 - bottom.
border_type	integer	Default: 0. Type of the border. Possible values: 0 - none; 1 - _____; 2 - - - - -; 3 - - - - -.
border_width	integer	Default: 0. Width of the border in pixels. Default: 0.

Property	Type	Description
border_color	string	Border color.
background_color	string	Default: '000000'. Background color (fill color).
zindex	integer	Default: (empty). Value used to order shapes (z-index).
		Default: 0.

Map lines

The map line object defines an line displayed on a map. It has the following properties:

Property	Type	Description
sysmap_shapeid	string	(<i>readonly</i>) ID of the map shape element.
x1	integer	X-coordinates of the line point 1 in pixels.
y1	integer	Default: 0. Y-coordinates of the line point 1 in pixels.
x2	integer	Default: 0. X-coordinates of the line point 2 in pixels.
y2	integer	Default: 200. Y-coordinates of the line point 2 in pixels.
line_type	integer	Default: 200. Type of the border. Possible values: 0 - none; 1 - _____; 2 - ---; 3 - - - -.
line_width	integer	Default: 0. Width of the border in pixels.
line_color	string	Default: 0. Border color.
zindex	integer	Default: '000000'. Value used to order shapes (z-index).
		Default: 0.

map.create

Description

object map.create(object/array maps)

This method allows to create new maps.

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 elements `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 *frontends/php/include/classes/api/services/CMap.php*.

map.delete

Description

object map.delete(array mapIds)

This method allows to delete maps.

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 *frontends/php/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.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
sysmapids	string/array	Return only maps with the given IDs.
userids	string/array	Return only maps that belong to the given user IDs.
expandUrls	flag	Adds global map URLs to the corresponding map elements and expands macros in all map element URLs.
selectIconMap	query	Returns the icon map used on the map in the <code>iconmap</code> property.
selectLinks	query	Returns map links between elements in the <code>links</code> property.
selectSelements	query	Returns the map elements from the map in the <code>selements</code> property.
selectUrls	query	Returns the map URLs in the <code>urls</code> property.
selectUsers	query	Returns users that the map is shared with in <code>users</code> property.
selectUserGroups	query	Returns user groups that the map is shared with in <code>userGroups</code> property.
selectShapes	query	Returns the map shapes from the map in the <code>shapes</code> property.
selectLines	query	Returns the map lines from the map in the <code>lines</code> property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>name</code> , <code>width</code> and <code>height</code> . These parameters being common for all <code>get</code> 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",
          "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",
          "viewtype": "0",
          "use_iconmap": "1",
          "application": "",
          "urls": [],
          "elements": []
        },
        {
          "selementid": "11",
          "sysmapid": "3",
          "elementtype": "4",
          "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",
        "application": "",
        "urls": [],
        "elements": []
    },
    {
        "selementid": "12",
        "sysmapid": "3",
        "elementtype": "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",
        "application": "",
        "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 *frontends/php/include/classes/api/services/CMap.php*.

map.update

Description

object map.update(object/array maps)

This method allows to update existing maps.

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 elements 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 *frontends/php/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	(<i>readonly</i>) ID of the media type.
description (required)	string	Name of the media type.
type (required)	integer	Transport used by the media type. Possible values: 0 - e-mail; 1 - script; 2 - SMS; 3 - Jabber; 100 - Ez Texting.
exec_path	string	For script media types <code>exec_path</code> contains the name of the executed script. For Ez Texting <code>exec_path</code> contains the message text limit. Possible text limit values: 0 - USA (160 characters); 1 - Canada (136 characters).
gsm_modem	string	Required for script and Ez Texting media types. Serial device name of the GSM modem.
passwd	string	Required for SMS media types. Authentication password.
smtp_email	string	Required for Jabber and Ez Texting media types. Email address from which notifications will be sent.
smtp_helo	string	Required for email media types. SMTP HELO.
smtp_server	string	Required for email media types. SMTP server.
smtp_port	integer	Required for email media types. SMTP server port to connect to.
smtp_security	integer	SMTP connection security level to use. Possible values: 0 - None; 1 - STARTTLS; 2 - SSL/TLS.

Property	Type	Description
smtp_verify_host	integer	SSL verify host for SMTP. Possible values: 0 - No; 1 - Yes.
smtp_verify_peer	integer	SSL verify peer for SMTP. Possible values: 0 - No; 1 - Yes.
smtp_authentication	integer	SMTP authentication method to use. Possible values: 0 - None; 1 - Normal password.
status	integer	Whether the media type is enabled. Possible values: 0 - <i>(default)</i> enabled; 1 - disabled.
username	string	Username or Jabber identifier.
exec_params	string	Required for Jabber and Ez Texting media types. Script parameters.
maxsessions	integer	Each parameter ends with a new line feed. The maximum number of alerts that can be processed in parallel. Possible values for SMS: 1 - <i>(default)</i>
maxattempts	integer	Possible values for other media types: 0-100 The maximum number of attempts to send an alert. Possible values: 1-10
attempt_interval	string	Default value: 3 The interval between retry attempts. Accepts seconds and time unit with suffix. Possible values: 0-60s
content_type	integer	Default value: 10s Message format. Possible values: 0 - plain text; 1 - <i>(default)</i> html.

mediatype.create

Description

object mediatype.create(object/array mediaTypes)

This method allows to create new media types.

Parameters

(object/array) Media types to create.

The method accepts media types with the **standard media type properties**.

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 a media type

Create a new e-mail media type.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "mediatype.create",
  "params": {
    "description": "E-mail",
    "type": 0,
    "smtp_server": "rootmail@company.com",
    "smtp_helo": "company.com",
    "smtp_email": "zabbix@company.com",
    "content_type": 1
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "mediatypeids": [
      "7"
    ]
  },
  "id": 1
}
```

Creating a media type with custom options

Create a new script media type with custom value for number of attempts and interval between them.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "mediatype.create",
  "params": {
    "type": 1,
    "description": "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
}
```

Source

CMediaType::create() in *frontends/php/include/classes/api/services/CMediaType.php*.

mediatype.delete

Description

object mediatype.delete(array mediaTypeIds)

This method allows to delete media types.

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 *frontends/php/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.

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.
userid	string/array	Return only media types used by the given users.
selectUsers	query	Return the users that use the media type in the <code>users</code> property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>mediatypeid</code> . 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"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "mediatypeid": "1",
      "type": "0",
      "description": "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"
},
{
    "mediatypeid": "2",
    "type": "3",
    "description": "Jabber",
    "smtp_server": "",
    "smtp_helo": "",
    "smtp_email": "",
    "exec_path": "",
    "gsm_modem": "",
    "username": "jabber@example.com",
    "passwd": "zabbix",
    "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"
},
{
    "mediatypeid": "3",
    "type": "2",
    "description": "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": "0"
    }
],
    "id": 1
}

```

See also

- [User](#)

Source

CMediaType::get() in *frontends/php/include/classes/api/services/CMediaType.php*.

mediatype.update

Description

object mediatype.update(object/array mediaTypes)

This method allows to update existing media types.

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.

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 *frontends/php/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.
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.
objectid	string	ID of the related object.
clock	timestamp	Time when the problem event was created.
ns	integer	Nanoseconds when the problem event was created.
r_eventid	string	Recovery event ID.
r_clock	timestamp	Time when the recovery event was created.
r_ns	integer	Nanoseconds when the recovery event was created.
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.

Property	Type	Description
suppressed	integer	Whether the problem is suppressed. Possible values: 0 - problem is in normal state; 1 - problem is suppressed.

Problem tag

The problem tag object has the following properties.

Property	Type	Description
tag	string	Problem tag name.
value	string	Problem tag value.

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.

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.
applicationids	string/array	Return only problems created by objects that belong to the given applications. Applies only if object is trigger or item.
source	integer	Return only problems with the given type. Refer to the problem event object page for a list of supported event types.
object	integer	Default: 0 - problem created by a trigger. Return only problems created by objects of the given type. Refer to the problem event object page for a list of supported object types.
acknowledged	boolean	Default: 0 - trigger. true - return acknowledged problems only; false - unacknowledged only.

Parameter	Type	Description
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.
tags	array of objects	Possible values: 0 - (default) And/Or; 2 - Or. 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.
recent	boolean	Possible operator types: 0 - (default) Like; 1 - Equal. true - return PROBLEM and recently RESOLVED problems (depends on Display OK triggers for N seconds) Default: false - UNRESOLVED problems only
eventid_from	string	Return only problems with IDs greater or equal to the given ID.
eventid_till	string	Return only problems with IDs less or equal to the given ID.
time_from	timestamp	Return only problems that have been created after or at the given time.
time_till	timestamp	Return only problems that have been created before or at the given time.
selectAcknowledges	query	Return problem's updates in the acknowledges property. 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) 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;
selectTags	query	Supports count. Return problem's tags. Output format: [{"tag": "<tag>", "value": "<value>"}, ...].
selectSuppressionData	query	Return list of maintenances in the suppression_data property: maintenanceid - (string) ID of the maintenance; suppress_until - (integer) time until the problem is suppressed.
sortfield	string/array	Sort the result by the given properties.
		Possible values are: eventid.

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 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",
        "suppressed": "1",
        "severity": "3",
        "acknowledges": [
            {
                "acknowledgeid": "14443",
                "userid": "1",
                "eventid": "1245463",
                "clock": "1472457281",
                "message": "problem solved",
                "action": "6",
                "old_severity": "0",
                "new_severity": "0"
            }
        ],
        "tags": [
            {
                "tag": "test tag",
                "value": "test value"
            }
        ],
        "suppression_data": [
            {
                "maintenanceid": "15",
                "suppress_until": "1472511600"
            }
        ]
    }
],
    "id": 1
}

```

See also

- [Alert](#)
- [Item](#)
- [Host](#)
- [LLD rule](#)
- [Trigger](#)

Source

CEvent::get() in *frontends/php/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	(<i>readonly</i>) 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	(<i>readonly</i>) Time when the proxy last connected to the server.
tls_connect	integer	Connections to host. Possible values are: 1 - (<i>default</i>) No encryption; 2 - PSK; 4 - certificate.
tls_accept	integer	Connections from host. Possible bitmap values are: 1 - (<i>default</i>) No encryption; 2 - PSK; 4 - certificate.
tls_issuer	string	Certificate issuer.
tls_subject	string	Certificate subject.
tls_psk_identity	string	PSK identity. Required if either <code>tls_connect</code> or <code>tls_accept</code> 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	The preshared key, at least 32 hex digits. Required if either <code>tls_connect</code> or <code>tls_accept</code> has PSK enabled.
proxy_address	string	Comma-delimited IP addresses or DNS names of active Zabbix proxy.
auto_compress	integer	(<i>readonly</i>) Indicates if communication between Zabbix server and proxy is compressed. Possible values are: 0 - No compression; 1 - Compression enabled;

Proxy interface

The proxy interface object defines the interface used to connect to a passive proxy. It has the following properties.

Property	Type	Description
interfaceid	string	(<i>readonly</i>) ID of the interface.
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.

Property	Type	Description
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.
hostid	string	(<i>readonly</i>) ID of the proxy the interface belongs to.

proxy.create

Description

object proxy.create(object/array proxies)

This method allows to create new proxies.

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.
interface	object	The hosts must have the <code>hostid</code> property defined. 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 *frontends/php/include/classes/api/services/CProxy.php*.

proxy.delete

Description

object proxy.delete(array proxies)

This method allows to delete proxies.

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 *frontends/php/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.

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 hosts monitored by the proxy in the hosts property.
selectInterface	query	Return the proxy interface used by a passive proxy in the interface property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>hostid</code> , <code>host</code> and <code>status</code> . 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

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": "",
      "tls_psk_identity": "",
      "tls_psk": "",
      "proxy_address": "",
      "auto_compress": "0",
      "proxyid": "30091",
      "interface": []
    },
    {
      "proxy_hostid": "0",
      "host": "Passive proxy",
      "status": "6",
      "lastaccess": "0",
      "description": "",
      "tls_connect": "1",
      "tls_accept": "1",

```

```

        "tls_issuer": "",
        "tls_subject": "",
        "tls_psk_identity": "",
        "tls_psk": "",
        "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 *frontends/php/include/classes/api/services/CProxy.php*.

proxy.update

Description

object proxy.update(object/array proxies)

This method allows to update existing proxies.

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.
interface	object	The hosts must have the hostid property defined. 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": [
      "10294",
      "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 *frontends/php/include/classes/api/services/CProxy.php*.

Screen

This class is designed to work with screen.

Object references:

- [Screen](#)
- [Screen user](#)
- [Screen user group](#)

Available methods:

- [screen.create](#) - creating new screen
- [screen.delete](#) - deleting screens
- [screen.get](#) - retrieving screens
- [screen.update](#) - updating screens

> Screen object

The following objects are directly related to the screen API.

Screen

The screen object has the following properties.

Property	Type	Description
screenid	string	<i>(readonly)</i> ID of the screen.
name (required)	string	Name of the screen.
hsize	integer	Width of the screen.
vsiz	integer	Default: 1 Height of the screen.
userid	string	Default: 1 Screen owner user ID.
private	integer	Type of screen sharing. Possible values: 0 - public screen; 1 - <i>(default)</i> private screen.

Screen user

List of screen permissions based on users. It has the following properties:

Property	Type	Description
screenuserid	string	<i>(readonly)</i> ID of the screen user.
userid (required)	string	User ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

Screen user group

List of screen permissions based on user groups. It has the following properties:

Property	Type	Description
screenusrgrp	string	<i>(readonly)</i> ID of the screen user group.

Property	Type	Description
usrgrpid (required)	string	User group ID.
permission (required)	integer	Type of permission level. Possible values: 2 - read only; 3 - read-write;

screen.create

Description

object screen.create(object/array screens)

This method allows to create new screens.

Parameters

(object/array) Screens to create.

Additionally to the **standard screen properties**, the method accepts the following parameters.

Parameter	Type	Description
screenitems	array	Screen items to be created for the screen.
users	array	Screen user shares to be created on the screen.
userGroups	array	Screen user group shares to be created on the screen.

Return values

(object) Returns an object containing the IDs of the created screens under the `screenids` property. The order of the returned IDs matches the order of the passed screens.

Examples

Creating a screen

Create a screen named "Graphs" with 2 rows and 3 columns and add a graph to the upper-left cell.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "screen.create",
  "params": {
    "name": "Graphs",
    "hsize": 3,
    "vsize": 2,
    "screenitems": [
      {
        "resourcetype": 0,
        "resourceid": "612",
        "rowspan": 1,
        "colspan": 1,
        "x": 0,
        "y": 0
      }
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "screenids": [
      "26"
    ]
  },
  "id": 1
}
```

Screen sharing

Create a screen with two types of sharing (user and user group).

Request:

```
{
  "jsonrpc": "2.0",
  "method": "screen.create",
  "params": {
    "name": "Screen sharing",
    "hsize": 3,
    "vsize": 2,
    "users": [
      {
        "userid": "4",
        "permission": "3"
      }
    ],
    "userGroups": [
      {
        "usrgrpid": "7",
        "permission": "2"
      }
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "screenids": [
      "83"
    ]
  },
  "id": 1
}
```

See also

- [Screen item](#)
- [Screen user](#)
- [Screen user group](#)

Source

CScreen::create() in *frontends/php/include/classes/api/services/CScreen.php*.

screen.delete

Description

object screen.delete(array screenIds)

This method allows to delete screens.

Parameters

(array) IDs of the screens to delete.

Return values

(object) Returns an object containing the IDs of the deleted screens under the `screenids` property.

Examples

Deleting multiple screens

Delete two screens.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "screen.delete",
  "params": [
    "25",
    "26"
  ],
  "auth": "3a57200802b24cda67c4e4010b50c065",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "screenids": [
      "25",
      "26"
    ]
  },
  "id": 1
}
```

Source

`CScreen::delete()` in `frontends/php/include/classes/api/services/CScreen.php`.

screen.get

Description

integer/array `screen.get(object parameters)`

The method allows to retrieve screens according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
screenids	string/array	Return only screens with the given IDs.
userids	string/array	Return only screens that belong to the given user IDs.
screenitemids	string/array	Return only screen that contain the given screen items.
selectUsers	query	Returns users that the screen is shared with in <code>users</code> property.
selectUserGroups	query	Returns user groups that the screen is shared with in <code>userGroups</code> property.
selectScreenItems	query	Return the screen items that are used in the screen.

Parameter	Type	Description
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: screenid and name. 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 screen by ID

Retrieve all data about screen "26" and its screen items.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "screen.get",
  "params": {
    "output": "extend",
    "selectScreenItems": "extend",
    "selectUsers": "extend",
    "selectUserGroups": "extend",
    "screenids": "26"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "screenitems": [
        {
          "screenitemid": "67",
          "screenid": "26",
          "resourcetype": "0",
          "resourceid": "612",
          "width": "320",
          "height": "200",
          "x": "0",
          "y": "0",
          "colspan": "0",
          "rowspan": "0",

```

```

        "elements": "25",
        "valign": "0",
        "halign": "0",
        "style": "0",
        "url": "",
        "dynamic": "0",
        "sort_triggers": "0"
    },
    ],
    "users": [
        {
            "sysmapuserid": "1",
            "userid": "2",
            "permission": "2"
        }
    ],
    "userGroups": [
        {
            "screenusrgrpid": "1",
            "usrgrpid": "7",
            "permission": "3"
        }
    ],
    "screenid": "26",
    "name": "CPU Graphs",
    "hsize": "3",
    "vsize": "2",
    "templateid": "0",
    "userid": "1",
    "private": "1"
    },
    ],
    "id": 1
}

```

See also

- [Screen item](#)
- [Screen user](#)
- [Screen user group](#)

Source

CScreen::get() in *frontends/php/include/classes/api/services/CScreen.php*.

screen.update

Description

object screen.update(object/array screens)

This method allows to update existing screens.

Parameters

(object/array) Screen properties to be updated.

The `screenid` property must be defined for each screen, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the [standard screen properties](#), the method accepts the following parameters.

Parameter	Type	Description
screenitems	array	Screen items to replace existing screen items.
users	array	Screen items are updated by coordinates, so each screen item must have the x and y properties defined.
userGroups	array	Screen user shares to replace the existing elements.
		Screen user group shares to replace the existing elements.

Return values

(object) Returns an object containing the IDs of the updated screens under the `screenids` property.

Examples

Renaming a screen

Rename a screen to "CPU Graphs".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "screen.update",
  "params": {
    "screenid": "26",
    "name": "CPU Graphs"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "screenids": [
      "26"
    ]
  },
  "id": 1
}
```

Change screen owner

Available only for admins and super admins.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "screen.update",
  "params": {
    "screenid": "83",
    "userid": "1"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 2
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "screenids": [
      "83"
    ]
  }
}
```



```
    ]  
  },  
  "id": 2  
}
```

See also

- [Screen item](#)
- [screenitem.create](#)
- [screenitem.update](#)
- [screenitem.updatebyposition](#)
- [Screen user](#)
- [Screen user group](#)

Source

CScreen::update() in *frontends/php/include/classes/api/services/CScreen.php*.

Screen item

This class is designed to work with screen items.

Object references:

- [Screen item](#)

Available methods:

- [screenitem.create](#) - creating new screen items
- [screenitem.delete](#) - deleting screen items
- [screenitem.get](#) - retrieving screen items
- [screenitem.update](#) - updating screen items
- [screenitem.updatebyposition](#) - updating screen items in a specific screen cell

> Screen item object

The following objects are directly related to the `screenitem` API.

Screen item

The screen item object defines an element displayed on a screen. It has the following properties.

Property	Type	Description
screenitemid	string	(<i>readonly</i>) ID of the screen item.

Property	Type	Description
resourcetype (required)	integer	Type of screen item. Possible values: 0 - graph; 1 - simple graph; 2 - map; 3 - plain text; 4 - hosts info; 5 - triggers info; 6 - system information; 7 - clock; 8 - screen; 9 - triggers overview 10 - data overview; 11 - URL; 12 - history of actions; 13 - history of events; 14 - latest host group issues; 15 - problems by severity; 16 - latest host issues; 19 - simple graph prototype; 20 - graph prototype.
screenid (required)	string	ID of the screen that the item belongs to.
application	string	Application or part of application name by which data in screen item can be filtered. Applies to resource types: "Data overview" and "Triggers overview".
colspan	integer	Number of columns the screen item will span across.
dynamic	integer	Default: 1. Whether the screen item is dynamic. Possible values: 0 - (<i>default</i>) not dynamic; 1 - dynamic.
elements	integer	Number of lines to display on the screen item.
halign	integer	Default: 25. Specifies how the screen item must be aligned horizontally in the cell. Possible values: 0 - (<i>default</i>) center; 1 - left; 2 - right.
height	integer	Height of the screen item in pixels.
max_columns	integer	Default: 200. Specifies the maximum amount of columns a graph prototype or simple graph prototype screen element can have. Default: 3.

Property	Type	Description
resourceid	string	ID of the object displayed on the screen item. Depending on the type of a screen item, the <code>resourceid</code> property can reference different objects.
rowspan	integer	Required for data overview, graph, map, plain text, screen, simple graph and trigger overview screen items. Unused by local and server time clocks, history of actions, history of events, hosts info, system information, problems by severity and URL screen items. Number or rows the screen item will span across.
sort_triggers	integer	Default: 1. Order in which actions or triggers must be sorted. Possible values for history of actions screen elements: 3 - time, ascending; 4 - time, descending; 5 - type, ascending; 6 - type, descending; 7 - status, ascending; 8 - status, descending; 9 - retries left, ascending; 10 - retries left, descending; 11 - recipient, ascending; 12 - recipient, descending.
style	integer	Possible values for latest host group issues and latest host issues screen items: 0 - (<i>default</i>) last change, descending; 1 - severity, descending; 2 - host, ascending. Screen item display option. Possible values for data overview and triggers overview screen items: 0 - (<i>default</i>) display hosts on the left side; 1 - display hosts on the top. Possible values for hosts info and triggers info screen elements: 0 - (<i>default</i>) horizontal layout; 1 - vertical layout. Possible values for clock screen items: 0 - (<i>default</i>) local time; 1 - server time; 2 - host time. Possible values for plain text screen items: 0 - (<i>default</i>) display values as plain text; 1 - display values as HTML.
url	string	URL of the webpage to be displayed in the screen item. Used by URL screen items.
valign	integer	Specifies how the screen item must be aligned vertically in the cell. Possible values: 0 - (<i>default</i>) middle; 1 - top; 2 - bottom.

Property	Type	Description
width	integer	Width of the screen item in pixels.
x	integer	Default: 320. X-coordinates of the screen item on the screen, from left to right.
y	integer	Default: 0. Y-coordinates of the screen item on the screen, from top to bottom.
		Default: 0.

screenitem.create

Description

object screenitem.create(object/array screenItems)

This method allows to create new screen items.

Parameters

(object/array) Screen items to create.

The method accepts screen items with the **standard screen item properties**.

Return values

(object) Returns an object containing the IDs of the created screen items under the screenitemids property. The order of the returned IDs matches the order of the passed screen items.

Examples

Creating a screen item

Create a screen item displaying a graph in the left-upper cell of the screen.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "screenitem.create",
  "params": {
    "screenid": 16,
    "resourcetype": 0,
    "resourceid": 612,
    "x": 0,
    "y": 0
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "screenitemids": [
      "65"
    ]
  },
  "id": 1
}
```

See also

- [screen.update](#)

Source

CScreenItem::create() in *frontends/php/include/classes/api/services/CScreenItem.php*.

screenitem.delete

Description

object screenitem.delete(array screenItemIds)

This method allows to delete screen items.

Parameters

(array) IDs of the screen items to delete.

Return values

(object) Returns an object containing the IDs of the deleted screen items under the `screenitemids` property.

Examples

Deleting multiple screen items

Delete two screen items.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "screenitem.delete",
  "params": [
    "65",
    "63"
  ],
  "auth": "3a57200802b24cda67c4e4010b50c065",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "screenitemids": [
      "65",
      "63"
    ]
  },
  "id": 1
}
```

See also

- [screen.update](#)

Source

CScreenItem::delete() in *frontends/php/include/classes/api/services/CScreenItem.php*.

screenitem.get

Description

integer/array screenitem.get(object parameters)

The method allows to retrieve screen items according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
screenitemids	string/array	Return only screen items with the given IDs.
screenids	string/array	Return only screen items that belong to the given screen.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: screenitemid and screenid. These parameters being common for all get methods are described in detail in the reference commentary page 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 screen items from screen

Retrieve all screen items from the given screen.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "screenitem.get",
  "params": {
    "output": "extend",
    "screenids": "3"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "screenitemid": "20",
      "screenid": "3",
      "resourcetype": "0",
      "resourceid": "433",
      "width": "500",
      "height": "120",
      "x": "0",
      "y": "0",
      "colspan": "1",
      "rowspan": "1",
      "elements": "0",
      "valign": "1",

```

```

    "halign": "0",
    "style": "0",
    "url": "",
    "dynamic": "0",
    "sort_triggers": "0",
    "application": "",
    "max_columns": "3"
  },
  {
    "screenitemid": "21",
    "screenid": "3",
    "resourcetype": "0",
    "resourceid": "387",
    "width": "500",
    "height": "100",
    "x": "0",
    "y": "1",
    "colspan": "1",
    "rowspan": "1",
    "elements": "0",
    "valign": "1",
    "halign": "0",
    "style": "0",
    "url": "",
    "dynamic": "0",
    "sort_triggers": "0",
    "application": "",
    "max_columns": "3"
  },
  {
    "screenitemid": "22",
    "screenid": "3",
    "resourcetype": "1",
    "resourceid": "10013",
    "width": "500",
    "height": "148",
    "x": "1",
    "y": "0",
    "colspan": "1",
    "rowspan": "1",
    "elements": "0",
    "valign": "1",
    "halign": "0",
    "style": "0",
    "url": "",
    "dynamic": "0",
    "sort_triggers": "0",
    "application": "",
    "max_columns": "3"
  },
  {
    "screenitemid": "23",
    "screenid": "3",
    "resourcetype": "1",
    "resourceid": "22181",
    "width": "500",
    "height": "184",
    "x": "1",
    "y": "1",
    "colspan": "1",
    "rowspan": "1",
    "elements": "0",

```

```

        "valign": "1",
        "halign": "0",
        "style": "0",
        "url": "",
        "dynamic": "0",
        "sort_triggers": "0",
        "application": "",
        "max_columns": "3"
    }
],
    "id": 1
}

```

Source

CScreenItem::get() in *frontends/php/include/classes/api/services/CScreenItem.php*.

screenitem.update

Description

object screenitem.update(object/array screenItems)

This method allows to update existing screen items.

Parameters

(object/array) **Screen item properties** to be updated.

The screenitemid property must be defined for each screen item, 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 screen items under the screenitemids property.

Examples

Setting the size of the screen item

Set the width of the screen item to 500px and height to 300px.

Request:

```

{
    "jsonrpc": "2.0",
    "method": "screenitem.update",
    "params": {
        "screenitemid": "20",
        "width": 500,
        "height": 300
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}

```

Response:

```

{
    "jsonrpc": "2.0",
    "result": {
        "screenitemids": [
            "20"
        ]
    },
    "id": 1
}

```

See also

- [screenitem.updatebyposition](#)

Source

CScreenItem::update() in *frontends/php/include/classes/api/services/CScreenItem.php*.

screenitem.updatebyposition

Description

object screenitem.updatebyposition(array screenItems)

This method allows to update screen items in the given screen cells. If a cell is empty, a new screen item will be created.

Parameters

(array) [Screen item properties](#) to be updated.

The x, y and screenid properties must be defined for each screen item, 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 and created screen items under the screenitemids property.

Examples

Changing a screen items resource ID

Change the resource ID for the screen element located in the upper-left cell of the screen.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "screenitem.updatebyposition",
  "params": [
    {
      "screenid": "16",
      "x": 0,
      "y": 0,
      "resourceid": "644"
    }
  ],
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "screenitemids": [
      "66"
    ]
  },
  "id": 1
}
```

See also

- [screenitem.update](#)

Source

CScreenItem::update() in *frontends/php/include/classes/api/services/CScreenItem.php*.

Script

This class is designed to work with scripts.

Object references:

- [Script](#)

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	<i>(readonly)</i> ID of the script.
command (required)	string	Command to run.
name (required)	string	Name of the script.
confirmation	string	Confirmation pop up text. The pop up will appear when trying to run the script from the Zabbix frontend.
description	string	Description of the script.
execute_on	integer	Where to run the script. Possible values: 0 - run on Zabbix agent; 1 - run on Zabbix server. 2 - <i>(default)</i> run on Zabbix server (proxy).
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.
host_access	integer	Default: 0. Host permissions needed to run the script. Possible values: 2 - <i>(default)</i> read; 3 - write.
type	integer	Script type. Possible values: 0 - <i>(default)</i> script; 1 - IPMI.
usrgrpid	string	ID of the user group that will be allowed to run the script. If set to 0, the script will be available for all user groups. Default: 0.

script.create

Description

object `script.create(object/array scripts)`

This method allows to create new scripts.

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 script

Create a 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",
    "host_access": 3,
    "confirmation": "Are you sure you would like to reboot the server?"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "scriptids": [
      "3"
    ]
  },
  "id": 1
}
```

Source

CScript::create() in *frontends/php/include/classes/api/services/CScript.php*.

script.delete

Description

object script.delete(array scriptIds)

This method allows to delete scripts.

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 *frontends/php/include/classes/api/services/CScript.php*.

script.execute

Description

object script.execute(object parameters)

This method allows to run a script on a host.

Parameters

(object) Parameters containing the ID of the script to run and the ID of the host.

Parameter	Type	Description
hostid (required)	string	ID of the host to run the script on.
scriptid (required)	string	ID of the script to run.

Return values

(object) Returns the result of script execution.

Property	Type	Description
response	string	Whether the script was run successfully.
value	string	Possible values: success or failed. Script output.

Examples

Run a 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"
  },
  "id": 1
}
```

Source

CScript::execute() in *frontends/php/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.

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.
usrgrpsids	string/array	Return only scripts that can be run by users in the given user groups.
selectGroups	query	Return host groups that the script can be run on in the groups property.
selectHosts	query	Return hosts that the script can be run on in the hosts property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: scriptid and name. These parameters being common for all get methods are described in detail in the reference commentary .
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	

Parameter	Type	Description
sortorder	string/array	
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

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"
    },
    {
      "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"
    },
    {
      "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"
    }
],
    "id": 1
}

```

See also

- [Host](#)
- [Host group](#)

Source

CScript::get() in *frontends/php/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.

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",
                "usrgrpuid": "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 *frontends/php/include/classes/api/services/CScript.php*.

script.update

Description

object script.update(object/array scripts)

This method allows to update existing scripts.

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.

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 *frontends/php/include/classes/api/services/CScript.php*.

Service

This class is designed to work with services.

Object references:

- [Service](#)
- [Service time](#)
- [Service dependency](#)
- [Service alarm](#)

Available methods:

- [service.adddependencies](#) - adding dependencies between IT services
- [service.addtimes](#) - adding service times
- [service.create](#) - creating new IT services
- [service.delete](#) - deleting IT services
- [service.deletedependencies](#) - deleting dependencies between IT services
- [service.deletetimes](#) - deleting service times
- [service.get](#) - retrieving IT services
- [service.getsla](#) - retrieving availability information about IT services
- [service.update](#) - updating IT services

> Service object

The following objects are directly related to the `service` API.

Service

The service object has the following properties.

Property	Type	Description
<code>serviceid</code>	string	(<i>readonly</i>) ID of the service.
<code>algorithm</code> (required)	integer	Algorithm used to calculate the state of the service. Possible values: 0 - do not calculate; 1 - problem, if at least one child has a problem; 2 - problem, if all children have problems.
<code>name</code> (required)	string	Name of the service.
<code>showsla</code> (required)	integer	Whether SLA should be calculated. Possible values: 0 - do not calculate; 1 - calculate.
<code>sortorder</code> (required)	integer	Position of the service used for sorting.
<code>goodsla</code>	float	Minimum acceptable SLA value. If the SLA drops lower, the service is considered to be in problem state. Default: 99.9.

Property	Type	Description
status	integer	<i>(readonly)</i> Whether the service is in OK or problem state. If the service is in problem state, status is equal either to: - the priority of the linked trigger if it is set to 2, "Warning" or higher (priorities 0, "Not classified" and 1, "Information" are ignored); - the highest status of a child service in problem state.
triggerid	string	If the service is in OK state, status is equal to 0. Trigger associated with the service. Can only be set for services that don't have children. Default: 0

Service time

The service time object defines periods, when an service is scheduled to be up or down. It has the following properties.

Property	Type	Description
timeid	string	<i>(readonly)</i> ID of the service time.
serviceid (required)	string	ID of the service.
ts_from (required)	integer	Cannot be updated. Time when the service time comes into effect.
ts_to (required)	integer	For onetime downtimes ts_from must be set as a Unix timestamp, for other types - as a specific time in a week, in seconds, for example, 90000 for Tue, 2:00 AM. Time when the service time ends.
type (required)	integer	For onetime uptimes ts_to must be set as a Unix timestamp, for other types - as a specific time in a week, in seconds, for example, 90000 for Tue, 2:00 AM. Service time type. Possible values: 0 - planned uptime, repeated every week; 1 - planned downtime, repeated every week; 2 - one-time downtime.
note	string	Additional information about the service time.

Service dependency

The service dependency object represents a dependency between services. It has the following properties.

Property	Type	Description
linkid	string	<i>(readonly)</i> ID of the service dependency.
servicedownid (required)	string	ID of the service, that a service depends on, that is, the child service. An service can have multiple children.
serviceupid (required)	string	ID of the service, that is dependent on a service, that is, the parent service. An service can have multiple parents forming a directed graph.

Property	Type	Description
soft (required)	integer	Type of dependency between services. Possible values: 0 - hard dependency; 1 - soft dependency. An service can have only one hard-dependent parent. This attribute has no effect on status or SLA calculation and is only used to create a core service tree. Additional parents can be added as soft dependencies forming a graph. An service can not be deleted if it has hard-dependent children.

Service alarm

Note:

Service alarms cannot be directly created, updated or deleted via the Zabbix API.

The service alarm objects represents an service's state change. It has the following properties.

Property	Type	Description
servicealarmid	string	ID of the service alarm.
serviceid	string	ID of the service.
clock	timestamp	Time when the service state change has happened.
value	integer	Status of the service.
Refer the the service status property for a list of possible values.		

service.adddependencies

Description

```
object service.adddependencies(object/array serviceDependencies)
```

This method allows to create dependencies between services.

Parameters

(object/array) Service dependencies to create.

Each service dependency has the following parameters.

Parameter	Type	Description
serviceid	string	ID of the service that depends on a service, that is, the parent service.
dependsOnServiceid	string	ID of the service that a service depends on, that is, the child service.
soft	string	Type of dependency.
Refer to the service dependency object page for more information on dependency types.		

Return values

(object) Returns an object containing the IDs of the affected parent services under the `serviceids` property.

Examples

Creating a hard dependency

Make service "2" a hard-dependent child of service "3".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "service.adddependencies",
  "params": {
    "serviceid": "3",
    "dependsOnServiceid": "2",
    "soft": 0
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "serviceids": [
      "3"
    ]
  },
  "id": 1
}
```

See also

- [service.update](#)

Source

CService::addDependencies() in *frontends/php/include/classes/api/services/CService.php*.

service.addtimes

Description

object service.addtimes(object/array serviceTimes)

This method allows to create new service times.

Parameters

(object/array) Service times to create.

The method accepts service times with the [standard service time properties](#).

Return values

(object) Returns an object containing the IDs of the affected services under the `serviceids` property.

Examples

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.addtimes",
  "params": {
    "serviceid": "4",
    "type": 1,
    "ts_from": 165600,
    "ts_to": 201600
  }
}
```

```

    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}

```

Response:

```

{
  "jsonrpc": "2.0",
  "result": {
    "serviceids": [
      "4"
    ]
  },
  "id": 1
}

```

See also

- [service.update](#)

Source

CService::addTimes() in *frontends/php/include/classes/api/services/CService.php*.

service.create

Description

`object service.create(object/array services)`

This method allows to create new services.

Parameters

(object/array) services to create.

Additionally to the [standard service properties](#), the method accepts the following parameters.

Parameter	Type	Description
dependencies	array	Service dependencies. Each service dependency has the following parameters: - <code>dependsOnServiceid</code> - (<i>string</i>) ID of an service the service depends on, that is, the child service. - <code>soft</code> - (<i>integer</i>) type of service dependency; refer to the service dependency object page for more information on dependency types.
parentid	string	ID of a hard-linked parent service.
times	array	Service times 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 an service

Create an service that will be switched to problem state, if at least one child has a problem. SLA calculation will be on and the minimum acceptable SLA is 99.99%.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "service.create",
  "params": {
    "name": "Server 1",
    "algorithm": 1,
    "showsla": 1,
    "goodsla": 99.99,
    "sortorder": 1
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "serviceids": [
      "5"
    ]
  },
  "id": 1
}
```

Source

CService::create() in *frontends/php/include/classes/api/services/CService.php*.

service.delete

Description

object service.delete(array serviceIds)

This method allows to delete services.

Services with hard-dependent child services cannot be deleted.

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 *frontends/php/include/classes/api/services/CService.php*.

service.deletedependencies

Description

object service.deletedependencies(string/array serviceIds)

This method allows to delete all dependencies from services.

Parameters

(string/array) IDs of the services to delete all dependencies from.

Return values

(object) Returns an object containing the IDs of the affected services under the `serviceids` property.

Examples

Deleting dependencies from an service

Delete all dependencies from service "2".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "service.deletedependencies",
  "params": [
    "2"
  ],
  "auth": "3a57200802b24cda67c4e4010b50c065",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "serviceids": [
      "2"
    ]
  },
  "id": 1
}
```

See also

- [service.update](#)

Source

CService::delete() in *frontends/php/include/classes/api/services/CService.php*.

service.deletetimes

Description

object service.deletetimes(string/array serviceIds)

This method allows to delete all service times from services.

Parameters

(string/array) IDs of the services to delete all service times from.

Return values

(object) Returns an object containing the IDs of the affected services under the serviceids property.

Examples

Deleting service times from an service

Delete all service times from service "2".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "service.deletetimes",
  "params": [
    "2"
  ],
  "auth": "3a57200802b24cda67c4e4010b50c065",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "serviceids": [
      "2"
    ]
  },
  "id": 1
}
```

See also

- [service.update](#)

Source

CService::delete() in *frontends/php/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.

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 with the given hard-dependent parent services.

Parameter	Type	Description
childids	string/array	Return only services that are hard-dependent on the given child services.
selectParent	query	Return the hard-dependent parent service in the <code>parent</code> property.
selectDependencies	query	Return child service dependencies in the <code>dependencies</code> property.
selectParentDependencies	query	Return parent service dependencies in the <code>parentDependencies</code> property.
selectTimes	query	Return service times in the <code>times</code> property.
selectAlarms	query	Return service alarms in the <code>alarms</code> property.
selectTrigger	query	Return the associated trigger in the <code>trigger</code> property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>name</code> and <code>sortorder</code> . 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",
    "selectDependencies": "extend"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "serviceid": "2",
      "name": "Server 1",
      "status": "0",
      "algorithm": "1",

```

```

        "triggerid": "0",
        "showsla": "1",
        "goodsla": "99.9000",
        "sortorder": "0",
        "dependencies": []
    },
    {
        "serviceid": "3",
        "name": "Data center 1",
        "status": "0",
        "algorithm": "1",
        "triggerid": "0",
        "showsla": "1",
        "goodsla": "99.9000",
        "sortorder": "0",
        "dependencies": [
            {
                "linkid": "11",
                "serviceupid": "3",
                "servicedownid": "2",
                "soft": "0",
                "sortorder": "0",
                "serviceid": "2"
            },
            {
                "linkid": "10",
                "serviceupid": "3",
                "servicedownid": "5",
                "soft": "0",
                "sortorder": "1",
                "serviceid": "5"
            }
        ]
    },
    {
        "serviceid": "5",
        "name": "Server 2",
        "status": "0",
        "algorithm": "1",
        "triggerid": "0",
        "showsla": "1",
        "goodsla": "99.9900",
        "sortorder": "1",
        "dependencies": []
    }
],
    "id": 1
}

```

Source

CService::get() in *frontends/php/include/classes/api/services/CService.php*.

service.getsla

Description

object service.getsla(object parameters)

This method allows to calculate availability information about services.

Parameters

(object) Parameters containing the IDs of the services and time intervals to calculate SLA.

Parameter	Type	Description
serviceids	string/array	IDs of services to return availability information for.
intervals	array	Time intervals to return service layer availability information about. Each time interval must have the following parameters: - from - (<i>timestamp</i>) interval start time; - to - (<i>timestamp</i>) interval end time.

Return values

(object) Returns the following availability information about each service under the corresponding service ID.

Property	Type	Description
status	integer	Current status of the service. Refer to the service object page for more information on service statuses.
problems	array	Triggers that are currently in problem state and are linked either to the service or one of its descendants.
sla	array	SLA data about each time period. Each SLA object has the following properties: - from - (<i>timestamp</i>) interval start time; - to - (<i>timestamp</i>) interval end time; - sla - (<i>float</i>) SLA for the given time interval; - okTime - (<i>integer</i>) time the service was in OK state, in seconds; - problemTime - (<i>integer</i>) time the service was in problem state, in seconds; - downtimeTime - (<i>integer</i>) time the service was in scheduled downtime, in seconds.

Examples

Retrieving availability information for an service

Retrieve availability information about a service during a week.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "service.getsla",
  "params": {
    "serviceids": "2",
    "intervals": [
      {
        "from": 1352452201,
        "to": 1353057001
      }
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
```

```

    "2": {
        "status": "3",
        "problems": {
            "13904": {
                "triggerid": "13904",
                "expression": "{13359}=0",
                "description": "Service unavailable",
                "url": "",
                "status": "0",
                "value": "1",
                "priority": "3",
                "lastchange": "1352967420",
                "comments": "",
                "error": "",
                "templateid": "0",
                "type": "0",
                "value_flags": "0",
                "flags": "0"
            }
        },
        "sla": [
            {
                "from": 1352452201,
                "to": 1353057001,
                "sla": 97.046296296296,
                "okTime": 586936,
                "problemTime": 17864,
                "downtimeTime": 0
            }
        ]
    },
    "id": 1
}

```

See also

- [Trigger](#)

Source

CService::getSla() in *frontends/php/include/classes/api/services/CService.php*.

service.update

Description

object service.update(object/array services)

This method allows to update existing services.

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
dependencies	array	Service dependencies to replace the current service dependencies. Each service dependency has the following parameters: - dependsOnServiceid - (<i>string</i>) ID of an service the service depends on, that is, the child service. - soft - (<i>integer</i>) type of service dependency; refer to the service dependency object page for more information on dependency types.
parentid	string	ID of a hard-linked parent service.
times	array	Service times to replace the current service times.

Return values

(object) Returns an object containing the IDs of the updated services under the `serviceids` property.

Examples

Setting the parent of an service

Make service "3" the hard-linked parent of service "5".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "service.update",
  "params": {
    "serviceid": "5",
    "parentid": "3"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "serviceids": [
      "5"
    ]
  },
  "id": 1
}
```

See also

- [service.adddependencies](#)
- [service.addtimes](#)
- [service.deletedependencies](#)
- [service.deletetimes](#)

Source

CService::update() in `frontends/php/include/classes/api/services/CService.php`.

Task

This class is designed to work with tasks.

Available methods:

- [task.create](#) - creating new tasks

task.create

Description

`object task.create(object task)`

This method allows to create new task.

Parameters

(object) A task to create.

The method accepts the following parameters.

Parameter	Type	Description
type (required)	integer	Task type. Possible values: 6 - Check now.
itemids (required)	string/array	IDs of items and low-level discovery rules.

Note that tasks can be created for the following types of items/discovery rules:

- Zabbix agent
- SNMPv1/v2/v3 agent
- Simple check
- Internal check
- Aggregate 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 `check 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",
    "itemids": ["10092", "10093"],
  },
  "auth": "700ca65537074ec963db7efabda78259",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "taskids": [
```

```

        "1",
        "2"
    ]
},
"id": 1
}

```

Source

CTask::create() in *frontends/php/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	(<i>readonly</i>) ID of the template.
host	string	Technical name of the template.
(required)		
description	text	Description of the template.
name	string	Visible name of the template.
		Default: host property value.

Template tag

The template tag object has the following properties.

Property	Type	Description
tag	string	Template tag name.
(required)		
value	string	Template tag value.

template.create

Description

object `template.create(object/array templates)`

This method allows to create new templates.

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	Host groups to add the template to. The host groups must have the <code>groupid</code> property defined.
tags	object/array	Template tags.
templates	object/array	Templates to be linked to the template. The templates must have the <code>templateid</code> property defined.
macros	object/array	User macros to be created for the template.
hosts	object/array	Hosts to link the template to. The hosts must have the <code>hostid</code> property defined.

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 it to two hosts.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "template.create",
  "params": {
    "host": "Linux template",
    "groups": {
      "groupid": 1
    },
    "hosts": [
      {
        "hostid": "10084"
      },
      {
        "hostid": "10090"
      }
    ],
    "tags": [
      {
        "tag": "Host name",
        "value": "{HOST.NAME}"
      }
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
```

```

    "result": {
        "templateids": [
            "10086"
        ]
    },
    "id": 1
}

```

Source

CTemplate::create() in *frontends/php/include/classes/api/services/CTemplate.php*.

template.delete

Description

object template.delete(array templateIds)

This method allows to delete templates.

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 *frontends/php/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.

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 host groups.
parentTemplateids	string/array	Return only templates that are children of the given templates.
hostids	string/array	Return only templates that are linked to the given hosts.
graphids	string/array	Return only templates that contain the given graphs.
itemids	string/array	Return only templates that contain the given items.
triggerids	string/array	Return only templates that contain the given triggers.
with_items	flag	Return only templates that have items.
with_triggers	flag	Return only templates that have triggers.
with_graphs	flag	Return only templates that have graphs.
with_httptests	flag	Return only templates that have web scenarios.
evaltype	integer	Rules for tag searching. Possible values: 0 - (default) And/Or; 2 - Or.
tags	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.
selectGroups	query	Return the host groups that the template belongs to in the groups property.
selectTags	query	Return template tags in the tags property.
selectHosts	query	Return the hosts that are linked to the template in the hosts property.
selectTemplates	query	Supports count. Return the child templates in the templates property.
selectParentTemplates	query	Supports count. Return the parent templates in the parentTemplates property.
selectHttpTests	query	Supports count. Return the web scenarios from the template in the httpTests property.
selectItems	query	Supports count. Return items from the template in the items property. Supports count.

Parameter	Type	Description
selectDiscoveries	query	Return low-level discoveries from the template in the <code>discoveries</code> property.
selectTriggers	query	Supports count. Return triggers from the template in the <code>triggers</code> property.
selectGraphs	query	Supports count. Return graphs from the template in the <code>graphs</code> property.
selectApplications	query	Supports count. Return applications from the template in the <code>applications</code> property.
selectMacros	query	Supports count. Return the macros from the template in the <code>macros</code> property..
selectScreens	query	Return screens from the template in the <code>screens</code> property.
limitSelects	integer	Supports count. Limits the number of records returned by subselects.
sortfield	string/array	Applies to the following subselects: <code>selectTemplates</code> - results will be sorted by name; <code>selectHosts</code> - sorted by host; <code>selectParentTemplates</code> - sorted by host; <code>selectItems</code> - sorted by name; <code>selectDiscoveries</code> - sorted by name; <code>selectTriggers</code> - sorted by description; <code>selectGraphs</code> - sorted by name; <code>selectApplications</code> - sorted by name; <code>selectScreens</code> - sorted by name. Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>hostid</code> , <code>host</code> , <code>name</code> , <code>status</code> . These parameters being common for all <code>get</code> 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 templates by name

Retrieve all data about two templates named "Template OS Linux" and "Template OS Windows".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "template.get",
  "params": {
    "output": "extend",
    "filter": {
      "host": [
        "Template OS Linux",
        "Template OS Windows"
      ]
    }
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "proxy_hostid": "0",
      "host": "Template OS 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": "Template OS Linux",
      "flags": "0",
      "templateid": "10001",
      "description": "",
      "tls_connect": "1",
      "tls_accept": "1",
      "tls_issuer": "",
      "tls_subject": "",
      "tls_psk_identity": "",
      "tls_psk": ""
    },
    {

```

```

        "proxy_hostid": "0",
        "host": "Template OS 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": "Template OS Windows",
        "flags": "0",
        "templateid": "10081",
        "description": "",
        "tls_connect": "1",
        "tls_accept": "1",
        "tls_issuer": "",
        "tls_subject": "",
        "tls_psk_identity": "",
        "tls_psk": ""
    }
],
    "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

- [Host group](#)
- [Template](#)
- [User macro](#)
- [Host interface](#)

Source

CTemplate::get() in *frontends/php/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.

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 <code>templateid</code> property defined.
groups	object/array	Host groups to add the given templates to. The host groups must have the <code>groupid</code> property defined.
hosts	object/array	Hosts and templates to link the given templates to. The hosts must have the <code>hostid</code> 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 <code>templateid</code> property defined.

Return values

(object) Returns an object containing the IDs of the updated templates under the `templateids` property.

Examples

Adding templates to a group

Add two templates to the host group "2".

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
}
```

Linking a template to hosts

Link template "10073" to two hosts.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "template.massadd",
  "params": {
    "templates": [
      {
        "templateid": "10073"
      }
    ],
    "hosts": [
      {
        "hostid": "10106"
      },
      {
        "hostid": "10104"
      }
    ]
  }
}
```



```

    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}

```

Response:

```

{
  "jsonrpc": "2.0",
  "result": {
    "templateids": [
      "10073"
    ]
  },
  "id": 1
}

```

See also

- [template.update](#)
- [Host](#)
- [Host group](#)
- [User macro](#)

Source

CTemplate::massAdd() in *frontends/php/include/classes/api/services/CTemplate.php*.

template.massremove

Description

object template.massremove(object parameters)

This method allows to remove related objects from multiple templates.

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	Host groups to remove the given templates from.
hostids	string/array	Hosts or templates to unlink the given templates from (downstream).
macros	string/array	User macros to delete from the given templates.
templateids_clear	string/array	Templates to unlink and clear from the given templates (upstream).
templateids_link	string/array	Templates to unlink from the given templates (upstream).

Return values

(object) Returns an object containing the IDs of the updated templates under the `templateids` property.

Examples

Removing templates from a group

Remove two templates from group "2".

Request:

```

{
  "jsonrpc": "2.0",
  "method": "template.massremove",
  "params": {

```

```

        "templateids": [
            "10085",
            "10086"
        ],
        "groupids": "2"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}

```

Response:

```

{
    "jsonrpc": "2.0",
    "result": {
        "templateids": [
            "10085",
            "10086"
        ]
    },
    "id": 1
}

```

Unlinking templates from a host

Unlink template "10085" from two hosts.

Request:

```

{
    "jsonrpc": "2.0",
    "method": "template.massremove",
    "params": {
        "templateids": "10085",
        "hostids": [
            "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 *frontends/php/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.

Parameters

(object) Parameters containing the IDs of the templates to update and the properties that should be updated.

Additionally to the **standard template properties**, the method accepts the following parameters.

Parameter	Type	Description
templates (required)	object/array	Templates to be updated. The templates must have the <code>templateid</code> property defined.
groups	object/array	Host groups to replace the current host groups the templates belong to. The host groups must have the <code>groupid</code> property defined.
hosts	object/array	Hosts and templates to replace the ones the templates are currently linked to. Both hosts and templates must use the <code>hostid</code> property to pass an ID.
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.
templates_link	object/array	Templates to replace the currently linked templates. The templates must have the <code>templateid</code> property defined.

Return values

(object) Returns an object containing the IDs of the updated templates under the `templateids` property.

Examples

Replacing host 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](#)
- [Host group](#)
- [User macro](#)

Source

CTemplate::massUpdate() in *frontends/php/include/classes/api/services/CTemplate.php*.

template.update

Description

`object template.update(object/array templates)`

This method allows to update existing templates.

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	Host groups to replace the current host groups the templates belong to. The host groups must have the <code>groupid</code> property defined.
tags	object/array	Template tags to replace the current template tags.
hosts	object/array	Hosts and templates to replace the ones the templates are currently linked to. Both hosts and templates must use the <code>hostid</code> property to pass an ID.
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 <code>templateid</code> property defined.

Parameter	Type	Description
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 *frontends/php/include/classes/api/services/CTemplate.php*.

Template screen

This class is designed to work with template screens.

Object references:

- [Template screen](#)

Available methods:

- [templatescreen.copy](#) - copy template screens
- [templatescreen.create](#) - create new template screens
- [templatescreen.delete](#) - delete template screens
- [templatescreen.get](#) - retrieve template screens
- [templatescreen.update](#) - update template screens

> Template screen object

The following objects are directly related to the `templatescreen` API.

Template screen

The template screen object has the following properties.

Property	Type	Description
screenid	string	(<i>readonly</i>) ID of the template screen.
name (required)	string	Name of the template screen.
templateid (required)	string	ID of the template that the screen belongs to.
hsize	integer	Width of the template screen.
vsize	integer	Default: 1 Height of the template screen.
		Default: 1

templatescreen.copy

Description

object `templatescreen.copy(object parameters)`

This method allows to copy template screens to the given templates.

Parameters

(object) Parameters defining the template screens to copy and the target templates.

Parameter	Type	Description
screenIds (required)	string/array	IDs of template screens to copy.
templateIds (required)	string/array	IDs of templates to copy the screens to.

Return values

(boolean) Returns true if the copying was successful.

Examples

Copy a template screen

Copy template screen "25" to template "30085".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "templatescreen.copy",
  "params": {
    "screenIds": "25",
    "templateIds": "30085"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": true,
  "id": 1
}
```

Source

CTemplateScreen::copy() in *frontends/php/include/classes/api/services/CTemplateScreen.php*.

templatescreen.create

Description

object templatescreen.create(object/array templateScreens)

This method allows to create new template screens.

Parameters

(object/array) Template screens to create.

Additionally to the **standard template screen properties**, the method accepts the following parameters.

Parameter	Type	Description
screenitems	array	Template screen items to create on the screen.

Return values

(object) Returns an object containing the IDs of the created template screens under the screenids property. The order of the returned IDs matches the order of the passed template screens.

Examples

Create a template screen

Create a template screen named "Graphs" with 2 rows and 3 columns and add a graph to the upper-left cell.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "templatescreen.create",
  "params": {
    "name": "Graphs",
    "templateid": "10047",
    "hsize": 3,
    "vsize": 2,
    "screenitems": [
      {
        "resourcetype": 0,
        "resourceid": "410",
        "x": 0,
        "y": 0
      }
    ]
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "screenids": [
      "45"
    ]
  },
  "id": 1
}
```

See also

- [Template screen item](#)

Source

CTemplateScreen::create() in *frontends/php/include/classes/api/services/CTemplateScreen.php*.

templatescreen.delete

Description

object templatescreen.delete(array templateScreenIds)

This method allows to delete template screens.

Parameters

(array) IDs of the template screens to delete.

Return values

(object) Returns an object containing the IDs of the deleted template screens under the `screenids` property.

Examples

Delete multiple template screens

Delete two template screens.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "templatescreen.delete",
  "params": [
    "45",
```



```

        "46"
    ],
    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}

```

Response:

```

{
    "jsonrpc": "2.0",
    "result": {
        "screenids": [
            "45",
            "46"
        ]
    },
    "id": 1
}

```

Source

CTemplateScreen::delete() in *frontends/php/include/classes/api/services/CTemplateScreen.php*.

templatescreen.get

Description

integer/array templatescreen.get(object parameters)

The method allows to retrieve template screens according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
hostids	string/array	Return only template screens that belong to the given hosts.
screenids	string/array	Return only template screens with the given IDs.
screenitemids	string/array	Return only template screens that contain the given screen items.
templateids	string/arary	Return only template screens that belong to the given templates.
noInheritance	flag	Do not return inherited template screens.
selectScreenItems	query	Return the screen items that are used in the template screen in the <code>screenitems</code> property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>screenid</code> and <code>name</code> . These parameters being common for all <code>get</code> 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 screens from template

Retrieve all screens from template "10001" and all of the screen items.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "templatescreen.get",
  "params": {
    "output": "extend",
    "selectScreenItems": "extend",
    "templateids": "10001"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "screenid": "3",
      "name": "System performance",
      "hsize": "2",
      "vsize": "2",
      "templateid": "10001",
      "screenitems": [
        {
          "screenitemid": "20",
          "screenid": "3",
          "resourcetype": "0",
          "resourceid": "433",
          "width": "500",
          "height": "120",
          "x": "0",
          "y": "0",
          "colspan": "1",
          "rowspan": "1",
          "elements": "0",
          "valign": "1",
          "halign": "0",
          "style": "0",
          "url": ""
        },
        {
          "screenitemid": "21",
          "screenid": "3",
          "resourcetype": "0",
          "resourceid": "387",
          "width": "500",
          "height": "100",
          "x": "0",
          "y": "1",
          "colspan": "1",
          "rowspan": "1",

```

```

        "elements": "0",
        "valign": "1",
        "halign": "0",
        "style": "0",
        "url": ""
    },
    {
        "screenitemid": "22",
        "screenid": "3",
        "resourcetype": "1",
        "resourceid": "10013",
        "width": "500",
        "height": "148",
        "x": "1",
        "y": "0",
        "colspan": "1",
        "rowspan": "1",
        "elements": "0",
        "valign": "1",
        "halign": "0",
        "style": "0",
        "url": ""
    },
    {
        "screenitemid": "23",
        "screenid": "3",
        "resourcetype": "1",
        "resourceid": "22181",
        "width": "500",
        "height": "184",
        "x": "1",
        "y": "1",
        "colspan": "1",
        "rowspan": "1",
        "elements": "0",
        "valign": "1",
        "halign": "0",
        "style": "0",
        "url": ""
    }
]
    }
    ],
    "id": 1
}

```

See also

- [Template screen item](#)

Source

CTemplateScreen::get() in *frontends/php/include/classes/api/services/CTemplateScreen.php*.

templatescreen.update

Description

object templatescreen.update(object/array templateScreens)

This method allows to update existing template screens.

Parameters

(object/array) Template screen properties to be updated.

The `screenid` property must be defined for each template screen, all other properties are optional. Only the passed properties will be updated, all others will remain unchanged.

Additionally to the **standard template screen properties**, the method accepts the following parameters.

Parameter	Type	Description
screenitems	array	Screen items to replace existing screen items.
		Screen items are updated by coordinates, so each screen item must have the <code>x</code> and <code>y</code> properties defined.

Return values

(object) Returns an object containing the IDs of the updated template screens under the `screenids` property.

Examples

Rename a template screen

Rename the template screen to "Performance graphs".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "templatescreen.update",
  "params": {
    "screenid": "3",
    "name": "Performance graphs"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "screenids": [
      "3"
    ]
  },
  "id": 1
}
```

Source

CTemplateScreen::update() in *frontends/php/include/classes/api/services/CTemplateScreen.php*.

Template screen item

This class is designed to work with template screen items.

Object references:

- **Template screen item**

Available methods:

- **templatescreenitem.get** - retrieve template screen items

> Template screen item object

The following objects are directly related to the `templatescreenitem` API.

Template screen item

The template screen item object defines an element displayed on a template screen. It has the following properties.

Property	Type	Description
screenitemid	string	(<i>readonly</i>) ID of the template screen item.
resourceid (required)	string	ID of the object from the parent template displayed on the template screen item. Depending on the type of screen item, the resourceid property can reference different objects. Unused by clock and URL template screen items.
resourcetype (required)	integer	<i>Note: the resourceid property always references an object used in the parent template object, even if the screen item itself is inherited on a host or template.</i> Type of template screen item. Possible values: 0 - graph; 1 - simple graph; 3 - plain text; 7 - clock; 11 - URL; 19 - simple graph prototype; 20 - graph prototype.
screenid (required)	string	ID of the template screen that the item belongs to.
colspan	integer	Number of columns the template screen item will span across.
elements	integer	Default: 1. Number of lines to display on the template screen item.
halign	integer	Default: 25. Specifies how the template screen item must be aligned horizontally in the cell. Possible values: 0 - (<i>default</i>) center; 1 - left; 2 - right.
height	integer	Height of the template screen item in pixels.
max_columns	integer	Default: 200. Specifies the maximum amount of columns a graph prototype or simple graph prototype screen element can have.
rowspan	integer	Default: 3. Number or rows the template screen item will span across.
style	integer	Default: 1. Template screen item display option. Possible values for clock screen items: 0 - (<i>default</i>) local time; 1 - server time; 2 - host time. Possible values for plain text screen items: 0 - (<i>default</i>) display values as plain text; 1 - display values as HTML.

Property	Type	Description
url	string	URL of the webpage to be displayed in the template screen item. Used by URL template screen items.
valign	integer	Specifies how the template screen item must be aligned vertically in the cell. Possible values: 0 - (<i>default</i>) middle; 1 - top; 2 - bottom.
width	integer	Width of the template screen item in pixels. Default: 320.
x	integer	X-coordinates of the template screen item on the screen, from left to right.
y	integer	Y-coordinates of the template screen item on the screen, from top to bottom. Default: 0.

templatescreenitem.get

Description

`integer/array templatescreenitem.get(object parameters)`

The method allows to retrieve template screen items according to the given parameters.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
screenids	string/array	Return only template screen items that belong to the given template screens.
screenitemids	string/array	Return only template screen items with the given IDs.
hostids	string/array	Returns an additional <code>real_resourceid</code> property for each template screen item, that belongs to a screen from the given hosts or templates. The <code>real_resourceid</code> property contains the ID of object displayed on the screen.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>screenitemid</code> and <code>screenid</code> . These parameters being common for all <code>get</code> 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 template screen items for screen

Return all template screen items from template screen "15".

Request:

```
{
  "jsonrpc": "2.0",
  "method": "templatescreenitem.get",
  "params": {
    "output": "extend",
    "screenids": "15"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "screenitemid": "42",
      "screenid": "15",
      "resourcetype": "0",
      "resourceid": "454",
      "width": "500",
      "height": "200",
      "x": "0",
      "y": "0",
      "colspan": "1",
      "rowspan": "1",
      "elements": "0",
      "valign": "1",
      "halign": "0",
      "style": "0",
      "url": "",
      "max_columns": "3"
    },
    {
      "screenitemid": "43",
      "screenid": "15",
      "resourcetype": "0",
      "resourceid": "455",
      "width": "500",
      "height": "270",
      "x": "1",
      "y": "0",
      "colspan": "1",
      "rowspan": "1",
      "elements": "0",
      "valign": "1",
      "halign": "0",
      "style": "0",
      "url": "",
      "max_columns": "3"
    }
  ],
  "id": 1
}
```

```
}
```

Source

CTemplateScreenItem::get() in *frontends/php/include/classes/api/services/CTemplateScreenItem.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	Time when that value was received.
itemid	string	ID of the related item.
num	integer	Number of values within this 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	Time when that value was received.
itemid	string	ID of the related item.
num	integer	Number of values within this hour.
value_min	integer	Hourly minimum value.
value_avg	integer	Hourly average value.
value_max	integer	Hourly maximum value.

trend.get

Description

```
integer/array trend.get(object parameters)
```

The method allows to retrieve trend data according to the given parameters.

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": "0.1650",
      "value_avg": "0.2168",
      "value_max": "0.3500",
    }
  ],
  "id": 1
}
```

Source

CTrend::get() in *frontends/php/include/classes/api/services/CTrend.php*.

Trigger

This class is designed to work with triggers.

Object references:

- [Trigger](#)

Available methods:

- [trigger.adddependencies](#) - adding new trigger dependencies
- [trigger.create](#) - creating new triggers
- [trigger.delete](#) - deleting triggers
- [trigger.deletedependencies](#) - deleting trigger dependencies
- [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	(<i>readonly</i>) ID of the trigger.
description (required)	string	Name of the trigger.
expression (required)	string	Reduced trigger expression.
comments	string	Additional description of the trigger.
error	string	(<i>readonly</i>) Error text if there have been any problems when updating the state of the trigger.
flags	integer	(<i>readonly</i>) Origin of the trigger. Possible values are: 0 - (<i>default</i>) a plain trigger; 4 - a discovered trigger.
lastchange	timestamp	(<i>readonly</i>) Time when the trigger last changed its state.
priority	integer	Severity of the trigger. Possible values are: 0 - (<i>default</i>) not classified; 1 - information; 2 - warning; 3 - average; 4 - high; 5 - disaster.
state	integer	(<i>readonly</i>) State of the trigger. Possible values: 0 - (<i>default</i>) 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 - (<i>default</i>) enabled; 1 - disabled.
templateid	string	(<i>readonly</i>) ID of the parent template trigger.

Property	Type	Description
type	integer	Whether the trigger can generate multiple problem events. Possible values are: 0 - <i>(default)</i> do not generate multiple events; 1 - generate multiple events.
url	string	URL associated with the trigger.
value	integer	<i>(readonly)</i> Whether the trigger is in OK or problem state. Possible values are: 0 - <i>(default)</i> OK; 1 - problem.
recovery_mode	integer	OK event generation mode. Possible values are: 0 - <i>(default)</i> Expression; 1 - Recovery expression; 2 - None.
recovery_expression	string	Reduced trigger recovery expression.
correlation_mode	integer	OK event closes. Possible values are: 0 - <i>(default)</i> 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 - <i>(default)</i> No; 1 - Yes.

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.adddependencies

Description

`object trigger.adddependencies(object/array triggerDependencies)`

This method allows to create new trigger dependencies.

Parameters

(object/array) Trigger dependencies to create.

Each trigger dependency has the following parameters:

Parameter	Type	Description
triggerid (required)	string	ID of the dependent trigger.
dependsOnTriggerid (required)	string	ID of the trigger that the trigger depends on.

Return values

(object) Returns an object containing the IDs of the dependent triggers under the `triggerids` property.

Examples

Add a trigger dependency

Make trigger "14092" dependent on trigger "13565."

Request:

```
{
  "jsonrpc": "2.0",
  "method": "trigger.adddependencies",
  "params": {
    "triggerid": "14092",
    "dependsOnTriggerid": "13565"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "triggerids": [
      "14092"
    ]
  },
  "id": 1
}
```

See also

- [trigger.update](#)

Source

CTrigger::addDependencies() in *frontends/php/include/classes/api/services/CTrigger.php*.

trigger.create

Description

object trigger.create(object/array triggers)

This method allows to create new triggers.

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 <code>triggerid</code> 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": "{Linux server:system.cpu.load[percpu,avg1].last()}>5",
      "dependencies": [
        {
          "triggerid": "17367"
        }
      ]
    },
    {
      "description": "Service status",
      "expression": "{Linux server:log[/var/log/system,Service .* has stopped].strlen()}<>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 *frontends/php/include/classes/api/services/CTrigger.php*.

trigger.delete

Description

object trigger.delete(array triggerIds)

This method allows to delete triggers.

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 `frontends/php/include/classes/api/services/CTrigger.php`.

trigger.deletedependencies

Description

object `trigger.deletedependencies(string/array triggers)`

This method allows to delete all trigger dependencies from the given triggers.

Parameters

(string/array) Triggers to delete the trigger dependencies from.

Return values

(object) Returns an object containing the IDs of the affected triggers under the `triggerids` property.

Examples

Deleting dependencies from multiple triggers

Delete all dependencies from two triggers.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "trigger.deleteDependencies",
  "params": [
```

```

    {
        "triggerid": "14544"
    },
    {
        "triggerid": "14545"
    }
],
"auth": "038e1d7b1735c6a5436ee9eae095879e",
"id": 1
}

```

Response:

```

{
    "jsonrpc": "2.0",
    "result": {
        "triggerids": [
            "14544",
            "14545"
        ]
    },
    "id": 1
}

```

See also

- [trigger.update](#)

Source

CTrigger::deleteDependencies() in *frontends/php/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.

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 from the given host 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.
applicationids	string/array	Return only triggers that contain items from the given applications.
functions	string/array	Return only triggers that use the given functions.
group	string	Refer to the supported trigger functions page for a list of supported functions. Return only triggers that belong to hosts from the host group with the given name.
host	string	Return only triggers that belong to host with the given name.
inherited	boolean	If set to <code>true</code> return only triggers inherited from a template.

Parameter	Type	Description
templated	boolean	If set to <code>true</code> return only triggers that belong to templates.
dependent	boolean	If set to <code>true</code> return only triggers that have dependencies. If set to <code>false</code> 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 <code>true</code> return only enabled triggers that belong to hosts in maintenance.
withUnacknowledgedEvents	flag	Return only triggers that have unacknowledged events.
withAcknowledgedEvents	flag	Return only triggers with all events acknowledged.
withLastEventUnacknowledged	flag	Return only triggers with the last event unacknowledged.
skipDependent	flag	Skip triggers in a problem state that are dependent on other triggers. Note that the other triggers are ignored if disabled, have disabled items or disabled item hosts.
lastChangeSince	timestamp	Return only triggers that have changed their state after the given time.
lastChangeTill	timestamp	Return only triggers that have changed their state before the given time.
only_true	flag	Return only triggers that have recently been in a problem state.
min_severity	integer	Return only triggers with severity greater or equal than the given severity.
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.
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.
selectGroups	query	Return the host groups that the trigger belongs to in the <code>groups</code> property.
selectHosts	query	Return the hosts that the trigger belongs to in the <code>hosts</code> property.
selectItems	query	Return items contained by the trigger in the <code>items</code> property.

Parameter	Type	Description
selectFunctions	query	Return functions used in the trigger in the <code>functions</code> property. The function objects represents the functions used in the trigger expression and has the following properties: <code>functionid</code> - (<i>string</i>) ID of the function; <code>itemid</code> - (<i>string</i>) ID of the item used in the function; <code>function</code> - (<i>string</i>) name of the function; <code>parameter</code> - (<i>string</i>) parameter passed to the function.
selectDependencies	query	Return triggers that the trigger depends on in the <code>dependencies</code> property.
selectDiscoveryRule	query	Return the low-level discovery rule that created the trigger.
selectLastEvent	query	Return the last significant trigger event in the <code>lastEvent</code> property.
selectTags	query	Return the trigger tags in <code>tags</code> property.
selectTriggerDiscovery	query	Return the trigger discovery object in the <code>triggerDiscovery</code> property. The trigger discovery objects links the trigger to a trigger prototype from which it was created. It has the following properties: <code>parent_triggerid</code> - (<i>string</i>) ID of the trigger prototype from which the trigger has been created.
filter	object	Return only those results that exactly match the given filter. Accepts an array, where the keys are property names, and the values are either a single value or an array of values to match against. Supports additional filters: <code>host</code> - technical name of the host that the trigger belongs to; <code>hostid</code> - ID of the host that the trigger belongs to.
limitSelects	integer	Limits the number of records returned by subselects. Applies to the following subselects: <code>selectHosts</code> - results will be sorted by <code>host</code> . Sort the result by the given properties.
sortfield	string/array	Possible values are: <code>triggerid</code> , <code>description</code> , <code>status</code> , <code>priority</code> , <code>lastchange</code> and <code>hostname</code> . These parameters being common for all get methods are described in detail in the reference commentary page.
countOutput	boolean	
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 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": [
    {
      "functions": [
        {
          "functionid": "13513",
          "itemid": "24350",
          "function": "diff",
          "parameter": "0"
        }
      ],
      "triggerid": "14062",
      "expression": "{13513}>0",
      "description": "/etc/passwd has been changed on {HOST.NAME}",
      "url": "",
      "status": "0",
      "value": "0",
      "priority": "2",
      "lastchange": "0",
      "comments": "",
      "error": "",
      "templateid": "10016",
      "type": "0",
      "state": "0",
      "flags": "0",
      "recovery_mode": "0",
      "recovery_expression": "",
      "correlation_mode": "0",
      "correlation_tag": "",
      "manual_close": "0"
    }
  ],
  "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](#)

Source

CTrigger::get() in *frontends/php/include/classes/api/services/CTrigger.php*.

trigger.update

Description

object trigger.update(object/array triggers)

This method allows to update existing triggers.

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
}
```

See also

- [trigger.adddependencies](#)
- [trigger.deletedependencies](#)

Source

CTrigger::update() in *frontends/php/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	(<i>readonly</i>) ID of the trigger prototype.
description (required)	string	Name of the trigger prototype.
expression (required)	string	Reduced trigger expression.
comments	string	Additional comments to the trigger prototype.
priority	integer	Severity of the trigger prototype. Possible values: 0 - (<i>default</i>) 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 - (<i>default</i>) enabled; 1 - disabled.
templateid	string	(<i>readonly</i>) ID of the parent template trigger prototype.
type	integer	Whether the trigger prototype can generate multiple problem events. Possible values: 0 - (<i>default</i>) 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 - (<i>default</i>) Expression; 1 - Recovery expression; 2 - None.
recovery_expression	string	Reduced trigger recovery expression.

Property	Type	Description
correlation_mode	integer	OK event closes.
correlation_tag	string	Possible values are: 0 - <i>(default)</i> All problems; 1 - All problems if tag values match.
manual_close	integer	Tag for matching. Allow manual close. Possible values are: 0 - <i>(default)</i> No; 1 - Yes.

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.

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": "{Zabbix server:vfs.fs.size[{#FSNAME},pfree].last()}<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 *frontends/php/include/classes/api/services/CTriggerPrototype.php*.

triggerprototype.delete

Description

object triggerprototype.delete(array triggerPrototypeIds)

This method allows to delete trigger prototypes.

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 *frontends/php/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.

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.
applicationids	string/array	Return only trigger prototypes that contain items from the given applications.
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.
group	string	Refer to the supported trigger functions page for a list of supported functions. Return only trigger prototypes that belong to hosts from the host groups with the given name.
groupids	string/array	Return only trigger prototypes that belong to hosts from the given host 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 <code>true</code> return only trigger prototypes inherited from a template.
maintenance	boolean	If set to <code>true</code> 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 <code>true</code> 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.

Parameter	Type	Description
selectDiscoveryRule	query	Return the LLD rule that the trigger prototype belongs to.
selectFunctions	query	Return functions used in the trigger prototype in the <code>functions</code> property. The function objects represents the functions used in the trigger expression and has the following properties: <code>functionid</code> - (<i>string</i>) ID of the function; <code>itemid</code> - (<i>string</i>) ID of the item used in the function; <code>function</code> - (<i>string</i>) name of the function; <code>parameter</code> - (<i>string</i>) parameter passed to the function.
selectGroups	query	Return the host groups that the trigger prototype belongs to in the <code>groups</code> property.
selectHosts	query	Return the hosts that the trigger prototype belongs to in the <code>hosts</code> property.
selectItems	query	Return items and item prototypes used the trigger prototype in the <code>items</code> property.
selectDependencies	query	Return trigger prototypes and triggers that the trigger prototype depends on in the <code>dependencies</code> property.
selectTags	query	Return the trigger prototype tags in <code>tags</code> 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: <code>host</code> - technical name of the host that the trigger prototype belongs to; <code>hostid</code> - ID of the host that the trigger prototype belongs to.
limitSelects	integer	Limits the number of records returned by subselects.
sortfield	string/array	Applies to the following subselects: <code>selectHosts</code> - results will be sorted by <code>host</code> . Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>triggerid</code> , <code>description</code> , <code>status</code> and <code>priority</code> . 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

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": [
    {
      "functions": [
        {
          "functionid": "12598",
          "itemid": "22454",
          "function": "last",
          "parameter": "0"
        }
      ],
      "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"
    },
    {
      "functions": [
        {
          "functionid": "13500",
          "itemid": "22686",
          "function": "last",
          "parameter": "0"
        }
      ],
      "triggerid": "13266",
      "expression": "{13500}<201",
      "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"
    }
],
    "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](#)

Source

CTriggerPrototype::get() in *frontends/php/include/classes/api/services/CTriggerPrototype.php*.

triggerprototype.update

Description

object triggerprototype.update(object/array triggerPrototypes)

This method allows to update existing trigger prototypes.

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 *frontends/php/include/classes/api/services/CTriggerPrototype.php*.

User

This class is designed to work with users.

Object references:

- [User](#)

Available methods:

- [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.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	(<i>readonly</i>) ID of the user.

Property	Type	Description
alias (required)	string	User alias.
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.
lang	string	Default: 15m. Language code of the user's language.
name	string	Default: en_GB. Name of the user.
refresh	string	Automatic refresh period. Accepts seconds and time unit with suffix.
rows_per_page	integer	Default: 30s. Amount of object rows to show per page.
surname	string	Default: 50. Surname of the user.
theme	string	User's theme.
type	integer	Possible values: default - (default) system default; blue-theme - Blue; dark-theme - Dark. Type of the user.
url	string	Possible values: 1 - (default) Zabbix user; 2 - Zabbix admin; 3 - Zabbix super admin. URL of the page to redirect the user to after logging in.

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.

Property	Type	Description
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.
period	string	Default: 63 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.

Parameters

The method accepts the following parameters.

Parameter	Type	Description
<code>extend</code>	boolean	Default value: "true". Setting it's value to "false" allows to check session without extending it's lifetime. Supported since Zabbix 4.0.
<code>sessionid</code>	string	User session id.

Attention:

Calling **user.checkAuthentication** method prolongs user session by default.

Return values

(object) Returns an object containing information about user.

Examples

Request:

```
{
  "jsonrpc": "2.0",
  "method": "user.checkAuthentication",
  "params": {
    "sessionid": "8C8447FF6F61D134CEAC740CCA1BC90D"
  },
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "userid": "1",
    "alias": "Admin",
    "name": "Zabbix",
  }
}
```



```

        "surname": "Administrator",
        "url": "",
        "autologin": "1",
        "autologout": "0",
        "lang": "ru_RU",
        "refresh": "0",
        "type": "3",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "127.0.0.1",
        "attempt_clock": "1355919038",
        "rows_per_page": "50",
        "debug_mode": true,
        "userip": "127.0.0.1",
        "sessionid": "8C8447FF6F61D134CEAC740CCA1BC90D",
        "gui_access": "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 *frontends/php/include/classes/api/services/CUser.php*.

user.create

Description

object user.create(object/array users)

This method allows to create new users.

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.
usrgrps (required)	array	Can be omitted if user is added only to groups that have LDAP access. User groups to add the user to.
user_medias	array	The user groups must have the usrgrp_id property defined. Medias to create for the user.

Return values

(object) Returns an object containing the IDs of the created users under the **user_ids** 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": {
    "alias": "John",
    "passwd": "Doe123",
    "usrgrps": [
      {
        "usrgrpid": "7"
      }
    ],
    "user_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

- [Media](#)
- [User group](#)

Source

CUser::create() in *frontends/php/include/classes/api/services/CUser.php*.

user.delete

Description

object user.delete(array users)

This method allows to delete users.

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 *frontends/php/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.

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.
usrgrpsids	string/array	Return only users that belong to the given user groups.
getAccess	flag	Adds additional information about user permissions. Adds the following properties for each user: gui_access - (<i>integer</i>) user's frontend authentication method. Refer to the gui_access property of the user group object for a list of possible values. debug_mode - (<i>integer</i>) indicates whether debug is enabled for the user. Possible values: 0 - debug disabled, 1 - debug enabled. users_status - (<i>integer</i>) 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.

Parameter	Type	Description
selectUsrgrps	query	Return user groups that the user belongs to in the <code>usrgrps</code> property.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: <code>userid</code> and <code>alias</code> . These parameters being common for all <code>get</code> 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",
      "alias": "Admin",
      "name": "Zabbix",
      "surname": "Administrator",
      "url": "",
      "autologin": "1",
      "autologout": "0s",
      "lang": "ru_RU",
      "refresh": "0s",
      "type": "3",
      "theme": "default",
      "attempt_failed": "0",
      "attempt_ip": "",
      "attempt_clock": "0",
      "rows_per_page": "50"
    }
  ]
}
```

```

    },
    {
        "userid": "2",
        "alias": "guest",
        "name": "Default2",
        "surname": "User",
        "url": "",
        "autologin": "0",
        "autologout": "15m",
        "lang": "en_GB",
        "refresh": "30s",
        "type": "1",
        "theme": "default",
        "attempt_failed": "0",
        "attempt_ip": "",
        "attempt_clock": "0",
        "rows_per_page": "50"
    }
],
"id": 1
}

```

See also

- [Media](#)
- [Media type](#)
- [User group](#)

Source

CUser::get() in *frontends/php/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.

Parameters

Attention:

This method is available to unauthenticated users and must be called without the `auth` parameter in the JSON-RPC request.

(object) Parameters containing the user name and password.

The method accepts the following parameters.

Parameter	Type	Description
password (required)	string	User password.
user (required)	string	User name.
userData	flag	Return information about the authenticated user.

Return values

(string/object) If the `userData` parameter is used, returns an object containing information about the authenticated user.

Additionally to the [standard user properties](#), the following information is returned:

Property	Type	Description
debug_mode	boolean	Whether debug mode is enabled for the user.
gui_access	integer	User's authentication method to the frontend.
		Refer to the <code>gui_access</code> property of the <code>user group object</code> 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": {
    "user": "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": {
    "user": "Admin",
    "password": "zabbix",
    "userData": true
  },
  "id": 1
}
```

Response:

```

{
  "jsonrpc": "2.0",
  "result": {
    "userid": "1",
    "alias": "Admin",
    "name": "Zabbix",
    "surname": "Administrator",
    "url": "",
    "autologin": "1",
    "autologout": "0",
    "lang": "ru_RU",
    "refresh": "0",
    "type": "3",
    "theme": "default",
    "attempt_failed": "0",
    "attempt_ip": "127.0.0.1",
    "attempt_clock": "1355919038",
    "rows_per_page": "50",
    "debug_mode": true,
    "userip": "127.0.0.1",
    "sessionid": "5b56eee8be445e98f0bd42b435736e42",
    "gui_access": "0"
  },
  "id": 1
}

```

See also

- [user.logout](#)

Source

CUser::login() in *frontends/php/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.

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 *frontends/php/include/classes/api/services/CUser.php*.

user.update

Description

`object user.update(object/array users)`

This method allows to update existing users.

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.
usrgrps	array	Can be empty string if user belongs to or is moved only to groups that have LDAP access. User groups to replace existing user groups.
user_medias	array	The user groups must have the <code>usrgrp_id</code> property defined. Medias to replace existing medias.

Return values

(object) Returns an object containing the IDs of the updated users under the `user_ids` 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
}
```

Source

CUser::update() in *frontends/php/include/classes/api/services/CUser.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	<i>(readonly)</i> ID of the user group.
name (required)	string	Name of the user group.
debug_mode	integer	Whether debug mode is enabled or disabled.
gui_access	integer	Possible values are: 0 - <i>(default)</i> disabled; 1 - enabled. Frontend authentication method of the users in the group. Possible values: 0 - <i>(default)</i> 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 - <i>(default)</i> enabled; 1 - disabled.

Permission

The permission object has the following properties.

Property	Type	Description
id (required)	string	ID of the host group to add permission to.
permission (required)	integer	Access level to the host 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.

Parameters

(object/array) User groups to create.

Additionally to the **standard user group properties**, the method accepts the following parameters.

Parameter	Type	Description
rights	object/array	Permissions to assign to the group
tag_filters	array	Tag based permissions to assign to the group
userids	string/array	IDs of users to add 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, which denies access to host group "2", and add a user to it.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "usergroup.create",
  "params": {
    "name": "Operation managers",
    "rights": {
      "permission": 0,
      "id": "2"
    },
    "userids": "12"
  }
}
```

```

    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}

```

Response:

```

{
    "jsonrpc": "2.0",
    "result": {
        "usrgrpids": [
            "20"
        ]
    },
    "id": 1
}

```

See also

- [Permission](#)

Source

CUserGroup::create() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.delete

Description

object usergroup.delete(array userGroupIds)

This method allows to delete user groups.

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 *frontends/php/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.

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.
userid	string/array	Return only user groups that contain the given users.
usrgrpids	string/array	Return only user groups with the given IDs.
with_gui_access	integer	Return only user groups with the given frontend authentication method. Refer to the user group page for a list of supported methods.
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.
selectRights	query	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.
limitSelects	integer	Refer to the user group page for a list of access levels to host groups. Limits the number of records returned by subselects.
sortfield	string/array	Sort the result by the given properties.
countOutput	boolean	Possible values are: usrgrpId, name. 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 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"
    },
    {
      "usrgrpid": "8",
      "name": "Guests",
      "gui_access": "0",
      "users_status": "0",
      "debug_mode": "0"
    },
    {
      "usrgrpid": "11",
      "name": "Enabled debug mode",
      "gui_access": "0",
      "users_status": "0",
      "debug_mode": "1"
    },
    {
      "usrgrpid": "12",
      "name": "No access to the frontend",
      "gui_access": "2",
      "users_status": "0",
      "debug_mode": "0"
    },
    {
      "usrgrpid": "14",
```

```

        "name": "Read only",
        "gui_access": "0",
        "users_status": "0",
        "debug_mode": "0"
    },
    {
        "usrgrp_id": "18",
        "name": "Deny",
        "gui_access": "0",
        "users_status": "0",
        "debug_mode": "0"
    }
],
"id": 1
}

```

See also

- [User](#)

Source

CUserGroup::get() in *frontends/php/include/classes/api/services/CUserGroup.php*.

usergroup.update

Description

object usergroup.update(object/array userGroups)

This method allows to update existing user groups.

Parameters

(object/array) User group properties to be updated.

The `usrgrp_id` 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
rights	object/array	Permissions to replace the current permissions assigned to the user group.
tag_filters	array	Tag based permissions to assign to the group
userids	string/array	IDs of the users to replace the users in the group.

Return values

(object) Returns an object containing the IDs of the updated user groups under the `usrgrp_ids` property.

Examples

Disabling a user group

Disable a user group.

Request:

```

{
    "jsonrpc": "2.0",
    "method": "usergroup.update",
    "params": {
        "usrgrp_id": "17",
        "users_status": "1"
    },
    "auth": "038e1d7b1735c6a5436ee9eae095879e",
    "id": 1
}

```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "usrgrpids": [
      "17"
    ]
  },
  "id": 1
}
```

See also

- [Permission](#)

Source

CUserGroup::update() in *frontends/php/include/classes/api/services/CUserGroup.php*.

User macro

This class is designed to work with host and global 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	(<i>readonly</i>) ID of the global macro.
macro (required)	string	Macro string.
value (required)	string	Value of the macro.

Host macro

The host macro object defines a macro available on a host or template. It has the following properties.

Property	Type	Description
hostmacroid	string	(<i>readonly</i>) ID of the host macro.
hostid (required)	string	ID of the host that the macro belongs to.
macro (required)	string	Macro string.

Property	Type	Description
value (required)	string	Value of the macro.

usermacro.create

Description

object usermacro.create(object/array hostMacros)

This method allows to create new host macros.

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

Creat 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 *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.createglobal

Description

object usermacro.createglobal(object/array globalMacros)

This method allows to create new global macros.

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 *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.delete

Description

object usermacro.delete(array hostMacroIds)

This method allows to delete host macros.

Parameters

(array) IDs of the host macros to delete.

Return values

(object) Returns an object containing the IDs of the deleted host macros under the `hostmacroids` property.

Examples

Deleting multiple host macros

Delete two host macros.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "usermacro.delete",
  "params": [
    "32",
    "11"
  ],
  "id": 1
}
```

```

    "auth": "3a57200802b24cda67c4e4010b50c065",
    "id": 1
}

```

Response:

```

{
    "jsonrpc": "2.0",
    "result": {
        "hostmacroids": [
            "32",
            "11"
        ]
    },
    "id": 1
}

```

Source

CUserMacro::delete() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.deleteglobal

Description

object usermacro.deleteglobal(array globalMacroIds)

This method allows to delete global macros.

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 *frontends/php/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.

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.
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.
selectGroups	query	Return host groups that the host macro belongs to in the <code>groups</code> property.
selectHosts	query	Used only when retrieving host macros. Return hosts that the host macro belongs to in the <code>hosts</code> property.
selectTemplates	query	Used only when retrieving host macros. Return templates that the host macro belongs to in the <code>templates</code> property.
sortfield	string/array	Used only when retrieving host macros. Sort the result by the given properties.
countOutput	boolean	Possible value: <code>macro</code> . These parameters being common for all <code>get</code> 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 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": "{$INTERFACE}",
      "value": "eth0"
    },
    {
      "hostmacroid": "11",
      "hostid": "10198",
      "macro": "{$SNMP_COMMUNITY}",
      "value": "public"
    }
  ],
  "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"
    }
  ],
  "id": 1
}
```

Source

CUserMacro::get() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.update

Description

object usermacro.update(object/array hostMacros)

This method allows to update existing host macros.

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
}
```

Source

CUserMacro::update() in *frontends/php/include/classes/api/services/CUserMacro.php*.

usermacro.updateglobal

Description

object usermacro.updateglobal(object/array globalMacros)

This method allows to update existing global macros.

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 *frontends/php/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.
name	string	Name of the value map.
(required)		
mappings	array	Value mappings for current value map. The mapping object is described in detail below.
(required)		

Value mappings

The value mappings object defines value mappings of the value map. It has the following properties.

Property	Type	Description
value (required)	string	Original value.
newvalue (required)	string	Value to which the original value is mapped to.

valuemap.create

Description

object valuemap.create(object/array valuemaps)

This method allows to create new value maps.

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": {
    "name": "Service state",
    "mappings": [
      {
        "value": "0",
        "newvalue": "Down"
      },
      {
        "value": "1",
        "newvalue": "Up"
      }
    ]
  },
  "auth": "57562fd409b3b3b9a4d916d45207bbcb",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "valuemapids": [
      "1"
    ]
  },
  "id": 1
}
```

Source

CValueMap::create() in *frontends/php/include/classes/api/services/CValueMap.php*.

valuemap.delete

Description

`object valuemap.delete(array valuemapid)`

This method allows to delete value maps.

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 `valuemapid` 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": {
    "valuemapid": [
      "1",
      "2"
    ]
  },
  "id": 1
}
```

Source

CValueMap::delete() in *frontends/php/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.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
valuemapid	string/array	Return only value maps with the given IDs.

Parameter	Type	Description
selectMappings	query	Return the value mappings for current value map in the mappings property.
sortfield	string/array	Supports count. Sort the result by the given properties.
countOutput	boolean	Possible values are: valuemapid, name. These parameters being common for all get methods are described in detail in the reference commentary .
editable	boolean	
excludeSearch	boolean	
filter	object	
limit	integer	
output	query	
preservekeys	boolean	
search	object	
searchByAny	boolean	
searchWildcardsEnabled	boolean	
sortorder	string/array	
startSearch	boolean	

Return values

(integer/array) Returns either:

- an array of objects;
- the count of retrieved objects, if the countOutput parameter has been used.

Examples

Retrieving value maps

Retrieve all configured value maps.

Request:

```
{
  "jsonrpc": "2.0",
  "method": "valuemap.get",
  "params": {
    "output": "extend"
  },
  "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": [
        {
          "value": "1",
          "newvalue": "unknown"
        },
        {
          "value": "2",
          "newvalue": "notInstalled"
        },
        {
          "value": "3",
          "newvalue": "ok"
        },
        {
          "value": "4",
          "newvalue": "failed"
        },
        {
          "value": "5",
          "newvalue": "highTemperature"
        },
        {
          "value": "6",
          "newvalue": "replaceImmediately"
        },
        {
          "value": "7",
          "newvalue": "lowCapacity"
        }
      ]
    }
  ],
  "id": 1
}
```

Source

CValueMap::get() in *frontends/php/include/classes/api/services/CValueMap.php*.

valuemap.update

Description

object valuemap.update(object/array valuemaps)

This method allows to update existing value maps.

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": [
      {
        "value": "0",
        "newvalue": "Online"
      },
      {
        "value": "1",
        "newvalue": "Offline"
      }
    ]
  },
  "auth": "57562fd409b3b3b9a4d916d45207bbcb",
}
```

```
}
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": {
    "valuemapids": [
      "2"
    ]
  },
  "id": 1
}
```

Source

CValueMap::update() in *frontends/php/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	(<i>readonly</i>) 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.
applicationid	string	Default: Zabbix ID of the application that the web scenario belongs to.
authentication	integer	Authentication method that will be used by the web scenario. Possible values: 0 - (<i>default</i>) none; 1 - basic HTTP authentication; 2 - NTLM authentication.

Property	Type	Description
delay	string	Execution interval of the web scenario. Accepts seconds, time unit with suffix and user macro.
headers	array of HTTP fields	Default: 1m. 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 <i>http://[username[:password]@]proxy.example.com[:port]</i> .
http_user	string	User name used for basic HTTP or NTLM authentication.
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.
ssl_cert_file	string	Default: 1. 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.
templateid	string	Possible values are: 0 - (default) enabled; 1 - disabled. (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.
verify_peer	integer	Possible values are: 0 - (default) skip host verification; 1 - verify host. Whether to verify the SSL certificate of the web server.
		Possible values are: 0 - (default) skip peer verification; 1 - verify peer.

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.
headers	array of HTTP fields	Possible values are: 0 - don't follow redirects; 1 - (default) follow redirects. HTTP headers that will be sent when performing a request. Scenario step headers will overwrite headers specified for the web scenario.

Property	Type	Description
httpstestid	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.
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.

httpstest.create

Description

object httpstest.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.

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.

Return values

(object) Returns an object containing the IDs of the created web scenarios under the httpstestids 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://mycompany.com",
        "status_codes": "200",
        "no": 1
      },
      {
        "name": "Homepage / About",
        "url": "http://mycompany.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 *frontends/php/include/classes/api/services/CHttpTest.php*.

httptest.delete

Description

object httptest.delete(array webScenarioIds)

This method allows to delete web scenarios.

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 *frontends/php/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.

Parameters

(object) Parameters defining the desired output.

The method supports the following parameters.

Parameter	Type	Description
applicationids	string/array	Return only web scenarios that belong to the given applications.
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 <code>true</code> return only web scenarios inherited from a template.
monitored	boolean	If set to <code>true</code> return only enabled web scenarios that belong to monitored hosts.
templated	boolean	If set to <code>true</code> 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.
selectHosts	query	Return the host that the web scenario belongs to as an array in the <code>hosts</code> property.

Parameter	Type	Description
selectSteps	query	Return web scenario steps in the steps property.
sortfield	string/array	Supports count. Sort the result by the given properties.
countOutput	boolean	Possible values are: httpstestid and name. 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": "httpstest.get",
  "params": {
    "output": "extend",
    "selectSteps": "extend",
    "httpstestids": "9"
  },
  "auth": "038e1d7b1735c6a5436ee9eae095879e",
  "id": 1
}
```

Response:

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "httpstestid": "9",
      "name": "Homepage check",
      "applicationid": "0",
      "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://mycompany.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://mycompany.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 *frontends/php/include/classes/api/services/CHttpTest.php*.

httptest.update

Description

`object httptest.update(object/array webScenarios)`

This method allows to update existing web scenarios.

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
<code>steps</code>	array	Scenario steps to replace existing steps.

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 *frontends/php/include/classes/api/services/CHttpTest.php*.

Appendix 1. Reference commentary

Notation Data types

The Zabbix API supports the following data types:

Type	Description
boolean	A boolean value, accepts either <code>true</code> or <code>false</code> .
flag	The value is considered to be <code>true</code> if it is passed and not equal to <code>null</code> and <code>false</code> 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: <code>extend</code> - returns all object properties; <code>count</code> - returns the number of retrieved records, supported only by certain subselects.

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
<code>countOutput</code>	boolean	Return the number of records in the result instead of the actual data.
<code>editable</code>	boolean	If set to <code>true</code> return only objects that the user has write permissions to.
<code>excludeSearch</code>	boolean	Default: <code>false</code> . Return results that do not match the criteria given in the <code>search</code> parameter.
<code>filter</code>	object	Return only those results that exactly match the given filter.
<code>limit</code>	integer	Doesn't work for text fields. Limit the number of records returned.
<code>output</code>	query	Object properties to be returned.
<code>preservekeys</code>	boolean	Default: <code>extend</code> . Use IDs as keys in the resulting array.
<code>search</code>	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.

Parameter	Type	Description
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.
searchWildcardsEnabled	boolean	Default: false. If set to true enables the use of "*" as a wildcard character in the search parameter.
sortfield	string/array	Default: false. 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.
sortorder	string/array	If no value is specified, data will be returned unsorted. Order of sorting. If an array is passed, each value will be matched to the corresponding property given in the sortfield parameter.
startSearch	boolean	Possible values are: ASC - (default) ascending; DESC - descending. 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"
        }
      ]
    }
  ],
  "id": 1
}
```

```

{
  "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 4.0 to 4.2

Backward incompatible changes General

httptest

Changes:

[ZBXNEXT-4933](#) `httptest.create`, `httptest.update`: dropped support of string format for headers and variables fields in both web scenario and web scenario step objects.

map

Changes:

[ZBXNEXT-4933](#) `map.create`, `map.update`: dropped support of `elementid` in map element object.

Other changes and bug fixes General

dcheck

Changes:

[ZBXNEXT-4938](#) `dcheck.get`: returns new properties `host_source` and `name_source`.

discoveryrule

Changes:

[ZBXNEXT-4865](#) `discoveryrule.create`, `discoveryrule.update`, `discoveryrule.get`: added support for Prometheus agents in preprocessing steps.

[ZBXNEXT-5083](#) `discoveryrule.create`, `discoveryrule.update`, `discoveryrule.get`: added support of dependent discovery rules.

[ZBXNEXT-4087](#) `discoveryrule.get`: added new query option `selectPreprocessing`.

[ZBXNEXT-4087](#) `discoveryrule.create`, `discoveryrule.update`: added new property `preprocessing`.

[ZBXNEXT-4877](#) `discoveryrule.get`: added new query option `selectLLDMacroPaths`.

[ZBXNEXT-1190](#) `discoveryrule.get`: added new query option `selectApplicationPrototypes`.

drule

Changes:

[ZBXNEXT-4938](#) `drule.create`, `drule.update`: added new options `host_source` and `name_source` in `dchecks` array.

host

Changes:

[ZBXNEXT-4731](#) `host.create`, `host.update`: added new property `tags`.

[ZBXNEXT-4731](#) `host.get`: added new parameters `evaltype` and `tags` that allow to filter hosts by tags.

[ZBXNEXT-4731](#) `host.get`: added new option `selectTags` that returns host tags in the `tags` property.

item

Changes:

[ZBXNEXT-4865](#) `item.create`, `item.update`, `item.get`: added support for Prometheus agents in preprocessing steps.

[ZBXNEXT-4742](#) `item.create`, `item.update`, `item.get`: added new item preprocessing types; extended item preprocessing with custom error handlers.

itemprototype

Changes:

[ZBXNEXT-4865](#) `itemprototype.create`, `itemprototype.update`, `itemprototype.get`: added support for Prometheus agents in preprocessing steps.

[ZBXNEXT-4742](#) `itemprototype.create`, `itemprototype.update`, `itemprototype.get`: added new item preprocessing types; extended item preprocessing with custom error handlers.

trigger

Changes:

[ZBXNEXT-221](#) `trigger.get`: added new query option `dependent`

template

Changes:

[ZBXNEXT-4731](#) `template.create`, `template.update`: added new property `tags`.

[ZBXNEXT-4731](#) `template.get`: added new parameters `evaltype` and `tags` that allow to filter templates by tags.
[ZBXNEXT-4731](#) `template.get`: added new option `selectTags` that returns template tags in the `tags` property.

Zabbix API changes in 4.2

4.2.7 valuemap

Changes:

[ZBX-3783](#) `valuemap.get`: added strict validation of input parameters

4.2.2 alert

Changes:

[ZBX-11272](#) `alert.get`: added support of two new options `mediatypeid` and `sendto` in the `sortfield` parameter

4.2.1 discoveryrule

Bug fixes:

[ZBX-10861](#) `discoveryrule.create`, `discoveryrule.update`: changed "username" and "password" fields to optional for HTTP agent type discovery rules and using authentication

`httptest`

Bug fixes:

[ZBX-10861](#) `httptest.create`, `httptest.update`: changed "http_user" and "http_password" fields to optional when using authentication

`item`

Bug fixes:

[ZBX-10861](#) `item.create`, `item.update`: changed "username" and "password" fields to optional for HTTP agent type items and using authentication

`itemprototype`

Bug fixes:

[ZBX-10861](#) `itemprototype.create`, `itemprototype.update`: changed "username" and "password" fields to optional for HTTP agent type item prototypes and using authentication

20. 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 colour 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, httpstest.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.

See also

* [Troubleshooting page on zabbix.org](#)

2 Installation

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.

Note:

schema.sql, images.sql and data.sql files are located in the *database* subdirectory of Zabbix sources. If Zabbix was installed from distribution packages, refer to the distribution documentation.

Attention:

For a Zabbix proxy database, **only** schema.sql should be imported (no images.sql nor data.sql)

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.

MySQL

Character set utf8 and utf8_bin collation is required for Zabbix server to work properly with MySQL database.

```

shell> mysql -uroot -p<password>
mysql> create database zabbix character set utf8 collate utf8_bin;
mysql> grant all privileges on zabbix.* to zabbix@localhost identified by '<password>';
mysql> quit;

```

If you use Zabbix packages continue with instructions for [Debian/Ubuntu](#) or [RHEL/CentOS](#) to import the data into the database.

```

shell> cd database/mysql
shell> mysql -uzabbix -p<password> zabbix < schema.sql
# stop here if you are creating database for Zabbix proxy
shell> mysql -uzabbix -p<password> zabbix < images.sql
shell> mysql -uzabbix -p<password> zabbix < data.sql

```

PostgreSQL

You need to have database user with permissions to create database objects. The following shell command will create user zabbix. Specify password when prompted and repeat password (note, you may first be asked for sudo password):

```

shell> sudo -u postgres createuser --pwprompt zabbix

```

Now we will set up the database zabbix (last parameter) with the previously created user as the owner (-O zabbix) and import initial schema and data (assuming you are in the root directory of Zabbix sources):

```

shell> sudo -u postgres createdb -O zabbix -E Unicode -T template0 zabbix

```

If you use Zabbix packages continue with instructions for [Debian/Ubuntu](#) or [RHEL/CentOS](#) to import the data into the database.

```

shell> cd database/postgresql
shell> cat schema.sql | sudo -u zabbix psql zabbix
# stop here if you are creating database for Zabbix proxy
shell> cat images.sql | sudo -u zabbix psql zabbix
shell> 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, e. g. "psql -U <username>" depending on how your system/database are configured. If you have troubles setting up the database please consult your Database administrator.

TimescaleDB

Support of TimescaleDB has been added on **experimental** basis in Zabbix 4.2.0.

Warning:

Currently TimescaleDB is not supported by Zabbix proxy.

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
```

The timescaledb.sql script is located in database/postgresql. The script must be run after the regular PostgreSQL database has been created with initial schema/data (see section above):

```

cat timescaledb.sql | sudo -u zabbix psql zabbix

```

Note that the timescaledb.sql script sets the following housekeeping parameters (from *Administration* → *General* → *House-keeping*):

- 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's possible to use TimescaleDB partitioning only for trends (by setting *Override item trend period*) or only for history (*Override item history period*).

Note:

You may want to run the `timescaledb-tune` tool provided by TimescaleDB to optimize PostgreSQL configuration parameters in your `postgresql.conf`.

Oracle

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 with a `user` shell user having write access to `/tmp` directory. 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
```

If you are creating a database for Zabbix server you need to have images from Zabbix sources on the host where Oracle is running. Copy them to a directory `/tmp/zabbix_images` on the Oracle host:

```
shell> cd /path/to/zabbix-sources
shell> ssh user@oracle_host "mkdir /tmp/zabbix_images"
shell> scp -r misc/images/png_modern user@oracle_host:/tmp/zabbix_images/
```

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.

Now the temporary directory can be removed:

```
shell> ssh user@oracle_host "rm -rf /tmp/zabbix_images"
```

IBM DB2

```
shell> db2 "create database zabbix using codeset utf-8 territory us pagesize 32768"
shell> cd database/ibm_db2
shell> db2batch -d zabbix -f schema.sql
# stop here if you are creating database for Zabbix proxy
shell> db2batch -d zabbix -f images.sql
shell> db2batch -d zabbix -f data.sql
```

Note:

It is important to set UTF-8 locale for Zabbix server, Zabbix proxy and the web server running Zabbix frontend. Otherwise text information from Zabbix will be interpreted by IBM DB2 server as non-UTF-8 and will be additionally converted on the way from Zabbix to the database and back. The database will store corrupted non-ASCII characters.

Zabbix frontend uses `OFFSET` and `LIMIT` clauses in SQL queries. For this to work, IBM DB2 server must have `DB2_COMPATIBILITY_VECTOR` variable be set to 3. Run the following command before starting the database server:

```
shell> db2set DB2_COMPATIBILITY_VECTOR=3
```

SQLite

Using SQLite is supported for **Zabbix proxy** only!

Note:

If using SQLite with Zabbix proxy, database will be automatically created if it does not exist.

```
shell> cd database/sqlite3
shell> sqlite3 /var/lib/sqlite/zabbix.db < schema.sql
```

Return to the [installation section](#).

2 Zabbix agent on Microsoft Windows

Configuring agent

Zabbix agent runs as a Windows service.

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 **Hostnameltem** parameter in the agent [configuration file](#) - or the default values are used if any of these parameters are not specified.

The default value for **Hostnameltem** parameter is the value returned by the "system.hostname" agent key and for Windows platform it returns the NetBIOS host name.

The default value for **Hostname** is the value returned by the **Hostnameltem** 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.

Attention:

The **system.hostname** key always returns the NetBIOS host name which is limited to 15 symbols and in UPPERCASE only - regardless of the length and lowercase/uppercase characters in the real host name.

Starting from Zabbix agent 1.8.6 version for Windows the "system.hostname" key supports an optional parameter - *type* of the name. The default value of this parameter is "netbios" (for backward compatibility) and the other possible value is "host".

Attention:

The **system.hostname[host]** key always returns the full, real (case sensitive) Windows host name.

So, to simplify the configuration of `zabbix_agentd.conf` file and make it unified, two different approaches could be used.

1. leave **Hostname** or **Hostnameltem** parameters undefined and Zabbix agent will use NetBIOS host name as the hostname;
2. leave **Hostname** parameter undefined and define **Hostnameltem** like this:

Hostnameltem=system.hostname[host]

and Zabbix agent will use the full, real (case sensitive) Windows host name as the hostname.

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
```

3 Elasticsearch setup

Attention:

Elasticsearch support is experimental!

Setup procedure considered in this section is applicable to the following Elasticsearch versions: **5.0.x → 6.1.x**. In case an earlier or later version of Elasticsearch is used, some functionality may not work as intended.

Zabbix has recently started to support storage of historical data by means of Elasticsearch instead of a database. Users are now given the possibility to choose the storage place for historical data between a compatible database and Elasticsearch.

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" : {
      "values" : {
        "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 '{
    "template": "uint*",
    "index_patterns": ["uint*"],
    "settings" : {
      "index" : {
```

```

        "number_of_replicas" : 1,
        "number_of_shards" : 5
    }
},
"mappings" : {
    "values" : {
        "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 "template" and "index_patterns" fields to match index name and to set valid mapping that 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 '{
    "template": "text*",
    "index_patterns": ["text*"],
    "settings" : {
        "index" : {
            "number_of_replicas" : 1,
            "number_of_shards" : 5
        }
    },
    "mappings" : {
        "values" : {
            "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.)

4 Migration to TimescaleDB

Overview

Support of a time-series database has been added since Zabbix 4.2 in the form of **experimental** support of 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.

This section provides the steps necessary for 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`:

```
cat timescaledb.sql | sudo -u zabbix psql zabbix
```

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 (from *Administration* → *General* → *Housekeeping*):

- Override item history period
- Override item trend period

In order to use partitioned housekeeping for history and trends both these options must be on. It's possible to use TimescaleDB partitioning only for trends (by setting *Override item trend period*) or only for history (*Override item history period*).

5 Real-time export of events, 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 data cannot be written to the export file or the export file cannot be renamed or new one cannot be created after renaming it, Zabbix will retry with 10 second interval until success.

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](#).

Another parameter - `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.

3 Daemon configuration

1 Zabbix server

Note:

The default values reflect daemon defaults, not the values in the shipped configuration files.

The parameters supported in a Zabbix server configuration file:

Parameter	Mandatory	Range	Default	Description
AlertScriptsPath	no		/usr/local/share/zabbix/alertscripts	Location of custom alert scripts (depends on compile-time installation variable <i>datadir</i>).
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 This parameter is supported since Zabbix 2.2.0.
CacheSize	no	128K-8G	8M	Size of configuration cache, in bytes. Shared memory size for storing host, item and trigger data. Upper limit used to be 2GB before Zabbix 2.2.3.
CacheUpdateFrequency	no	1-3600	60	How often Zabbix will perform update of configuration cache, in seconds. See also runtime control options.
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.
DBName	yes			Database name.
DBPassword	no			Database password. Comment this line if no password is used.
DBPort	no	1024-65535		Database port when not using local socket.
DBSchema	no			Schema name. Used for IBM DB2 and PostgreSQL.
DBSocket	no			Path to MySQL socket file.
DBUser	no			Database user.
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.

Parameter	Mandatory	Range	Default	Description
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.
ExternalScripts	no		/usr/local/share/zabbix/external_scripts	Location of external scripts (depends on compile-time installation variable <i>datadir</i>).
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!
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	no		0	Enable preprocessing of history values in history storage to store values in different indices based on date: 0 - disable 1 - enable
HistoryStorageURL	no			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	How often Zabbix will perform housekeeping procedure (in hours). Housekeeping is removing outdated information from the database. <i>Note:</i> 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. <i>Note:</i> 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. This postponing behavior is in place since Zabbix 2.4.0. 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 <i>housekeeper_execute</i> 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.

Parameter	Mandatory	Range	Default	Description
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: <code>/absolute/path/to/config/files/*. Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.</code>
JavaGateway	no			IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started. This parameter is supported since Zabbix 2.0.0.
JavaGatewayPort	no	1024-32767	10052	Port that Zabbix Java gateway listens on. This parameter is supported since Zabbix 2.0.0.
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. Multiple IP addresses are supported since Zabbix 1.8.3.
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> (since 4.2.3) LoadModule=</abs_path/module.so> Either the module must be located in directory specified by LoadModulePath or (since 4.2.3) 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.
LogFile	yes, if LogType is set to <i>file</i> , otherwise no			Name of log file.

Parameter	Mandatory	Range	Default	Description
LogFileSize	no	0-1024	1	Maximum size of log file in MB. 0 - disable automatic log rotation. <i>Note:</i> 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: <i>file</i> - write log to file specified by LogFile parameter, <i>system</i> - write log to syslog, <i>console</i> - 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. This parameter is supported since Zabbix 1.8.2.
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! This parameter is supported since Zabbix 1.8.2 and applies only to deleting history and trends of already deleted items.
PidFile	no		/tmp/zabbix_server.pid	Name of PID file.
ProxyConfigFrequency	no	1-604800	3600	How often Zabbix server sends configuration data to a Zabbix proxy in seconds. Used only for proxies in a passive mode. This parameter is supported since Zabbix 1.8.3.
ProxyDataFrequency	no	1-3600	1	How often Zabbix server requests history data from a Zabbix proxy in seconds. Used only for proxies in a passive mode. This parameter is supported since Zabbix 1.8.3.

Parameter	Mandatory	Range	Default	Description
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. This parameter is supported since Zabbix 2.0.0.
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.
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 and is supported since Zabbix 2.4.
SSLKeyLocation	no			Location of SSL private key files for client authentication. This parameter is used in web monitoring only and is supported since Zabbix 2.4.
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.
StartDBSyncers	no	1-100	4	Number of pre-forked instances of DB Syncers. The upper limit used to be 64 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.
StartAlerters	no	1-100	3	Number of pre-forked instances of alerters. This parameter is supported since Zabbix 3.4.0.
StartDiscoverers	no	0-250	1	Number of pre-forked instances of discoverers. The upper limit used to be 255 before version 1.8.5.

Parameter	Mandatory	Range	Default	Description
StartEscalators	no	1-100	1	Number of pre-forked instances of escalators. This parameter is supported since Zabbix 3.0.0.
StartHTTPPollers	no	0-1000	1	Number of pre-forked instances of HTTP pollers ¹ . The upper limit used to be 255 before version 1.8.5.
StartIPMIPollers	no	0-1000	0	Number of pre-forked instances of IPMI pollers. The upper limit used to be 255 before version 1.8.5.
StartJavaPollers	no	0-1000	0	Number of pre-forked instances of Java pollers ¹ . This parameter is supported since Zabbix 2.0.0.
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.
StartPingers	no	0-1000	1	Number of pre-forked instances of ICMP pingers ¹ . The upper limit used to be 255 before version 1.8.5.
StartPollersUnreachable	no	0-1000	1	Number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java) ¹ . Since Zabbix 2.4.0, at least one poller for unreachable hosts must be running if regular, IPMI or Java pollers are started. The upper limit used to be 255 before version 1.8.5. This option is missing in version 1.8.3.
StartPollers	no	0-1000	5	Number of pre-forked instances of pollers ¹ . <i>Note that a non-zero value is required for internal, aggregated and calculated items to work.</i>
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.

Parameter	Mandatory	Range	Default	Description
StartProxyPollers	no	0-250	1	Number of pre-forked instances of pollers for passive proxies ¹ . The upper limit used to be 255 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.
StartSNMPTrapper	no	0-1	0	If set to 1, SNMP trapper process will be started. This parameter is supported since Zabbix 2.0.0.
StartTimers	no	1-1000	1	Number of pre-forked instances of timers. Timers process maintenance periods. This parameter is supported since Zabbix 2.2.0.
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. At least one trapper process must be running to display server availability and view queue in the frontend. The upper limit used to be 255 before version 1.8.5.
StartVMwareCollectors	no	0-250	0	Number of pre-forked vmware collector instances. This parameter is supported since Zabbix 2.2.0.
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:::1 This parameter is supported since Zabbix 4.0.5.
Timeout	no	1-30	3	Specifies how long we wait for agent, SNMP device or external check (in seconds).

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. 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.
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.
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.
User	no		zabbix	Drop privileges to a specific, existing user on the system. Only has effect if run as 'root' and AllowRoot is disabled. This parameter is supported since Zabbix 2.4.0.

Parameter	Mandatory	Range	Default	Description
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. This parameter is supported since Zabbix 2.2.0.
VMwareCacheSize	no	256K-2G	8M	Shared memory size for storing VMware data. A VMware internal check <code>zabbix[vmware,buffer,...]</code> 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. This parameter is supported since Zabbix 2.2.0.
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. This parameter is supported since Zabbix 2.2.0.
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. This parameter is supported since Zabbix 2.2.9, 2.4.4
VMwareTimeout	no	1-300	10	The maximum number of seconds vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor). This parameter is supported since Zabbix 2.2.9, 2.4.4

Footnotes

Warning:

(1) Note that too many data gathering processes (pollers, unreachable 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.

Note:

Zabbix supports configuration files only in UTF-8 encoding without BOM.

Comments starting with "#" are only supported in the beginning of the line.

2 Zabbix proxy

Note:

The default values reflect daemon defaults, not the values in the shipped configuration files.

The parameters supported in a Zabbix proxy configuration file:

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 This parameter is supported since Zabbix 2.2.0.
CacheSize	no	128K-8G	8M	Size of configuration cache, in bytes. Shared memory size, for storing host and item data. Upper limit used to be 2GB before Zabbix 2.2.3.
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).
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.

Parameter	Mandatory	Range	Default	Description
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. <code>:memory:</code> , <code>file::memory:?cache=shared</code> or <code>file:memdb1?mode=memory&cache=sha</code>)
DBPassword	no			Warning: Do not attempt to use the same database Zabbix server is using. Database password. Ignored for SQLite. Comment this line if no password is used.
DBSchema	no			Schema name. Used for IBM DB2 and PostgreSQL.
DBSocket	no		3306	Path to MySQL socket. Database port when not using local socket. Ignored for SQLite.
DBUser				Database user. Ignored for SQLite.
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 <code>ProxyMode</code> parameter).
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		<code>/usr/local/share/zabbix/externalscripts</code>	Location of external scripts (depends on compile-time installation variable <code>datadir</code>).

Parameter	Mandatory	Range	Default	Description
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!
HeartbeatFrequency	no	0-3600	60	Frequency of heartbeat messages in seconds. Used for monitoring availability of proxy on server side. 0 - heartbeat messages disabled.
HistoryCacheSize	no	128K-2G	16M	Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter). Size of history cache, in bytes.
HistoryIndexCacheSize	no	128K-2G	4M	Shared memory size for storing history data. Size of history index cache, in bytes.
Hostname	no		Set by HostnameItem	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. Unique, case sensitive Proxy name. Make sure the proxy name is known to the server! Allowed characters: alphanumeric, '.', '_', '-' and '-'. Maximum length: 64
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. This parameter is supported since Zabbix 1.8.6.

Parameter	Mandatory	Range	Default	Description
HousekeepingFrequency	no	0-24	1	How often Zabbix will perform housekeeping procedure (in hours). Housekeeping is removing outdated information from the database. <i>Note:</i> 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. <i>Note:</i> 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. This postponing behavior is in place since Zabbix 2.4.0. 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 <i>housekeeper_execute</i> 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.

Parameter	Mandatory	Range	Default	Description
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: <code>/absolute/path/to/config/files/*. Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.</code>
JavaGateway	no			IP address (or hostname) of Zabbix Java gateway. Only required if Java pollers are started. This parameter is supported since Zabbix 2.0.0.
JavaGatewayPort	no	1024-32767	10052	Port that Zabbix Java gateway listens on. This parameter is supported since Zabbix 2.0.0.
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. Multiple IP addresses are supported since Zabbix 1.8.3.
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> (since 4.2.3) LoadModule=</abs_path/module.so> Either the module must be located in directory specified by LoadModulePath or (since 4.2.3) 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 <i>file</i> , otherwise no			Name of log file.

Parameter	Mandatory	Range	Default	Description
LogFileSize	no	0-1024	1	Maximum size of log file in MB. 0 - disable automatic log rotation. <i>Note:</i> 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
LogType	no		file	This parameter is supported since Zabbix 3.4.0. Log output type: <i>file</i> - write log to file specified by LogFile parameter, <i>system</i> - write log to syslog, <i>console</i> - write log to standard output.
LogSlowQueries	no	0-3600000	0	This parameter is supported since Zabbix 3.0.0. 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	This parameter is supported since Zabbix 1.8.2. Proxy will keep data locally for N hours, even if the data have already been synced with the server.
ProxyMode	no	0-1	0	This parameter may be used if local data will be used by third party applications. Proxy operating mode. 0 - proxy in the active mode 1 - proxy in the passive mode
ProxyOfflineBuffer	no	1-720	1	This parameter is supported since Zabbix 1.8.3. <i>Note</i> 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. Proxy will keep data for N hours in case of no connectivity with Zabbix server. Older data will be lost.

Parameter	Mandatory	Range	Default	Description
ServerPort	no	1024-32767	10051	Port of Zabbix trapper on Zabbix server. Active proxy parameter. Ignored for passive proxies (see ProxyMode parameter). If ProxyMode is set to <i>active mode</i> : IP address or DNS name of Zabbix server to get configuration data from and send data to.
Server	yes			 If ProxyMode is set to <i>passive mode</i> : 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 and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. <i>Example:</i> Server=127.0.0.1,192.168.1.0/24,::1,2001::
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. This parameter is supported since Zabbix 2.0.0.
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.
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 and is supported since Zabbix 2.4.0.
SSLKeyLocation	no			Location of SSL private key files for client authentication. This parameter is used in web monitoring only and is supported since Zabbix 2.4.0.

Parameter	Mandatory	Range	Default	Description
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 DB Syncers. The upper limit used to be 64 before version 1.8.5. This parameter is supported since Zabbix 1.8.3.
StartDiscoverers	no	0-250	1	Number of pre-forked instances of discoverers. The upper limit used to be 255 before version 1.8.5.
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. The upper limit used to be 255 before version 1.8.5.
StartJavaPollers	no	0-1000	0	Number of pre-forked instances of Java pollers. This parameter is supported since Zabbix 2.0.0.
StartPingers	no	0-1000	1	Number of pre-forked instances of ICMP pingers. The upper limit used to be 255 before version 1.8.5.
StartPollersUnreachable	no	0-1000	1	Number of pre-forked instances of pollers for unreachable hosts (including IPMI and Java). Since Zabbix 2.4.0, at least one poller for unreachable hosts must be running if regular, IPMI or Java pollers are started. The upper limit used to be 255 before version 1.8.5. This option is missing in version 1.8.3.
StartPollers	no	0-1000	5	Number of pre-forked instances of pollers. The upper limit used to be 255 before version 1.8.5.

Parameter	Mandatory	Range	Default	Description
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. This parameter is supported since Zabbix 2.0.0.
StartTrappers	no	0-1000	5	Number of pre-forked instances of trappers. Trappers accept incoming connections from Zabbix sender and active agents. The upper limit used to be 255 before version 1.8.5.
StartVMwareCollectors	no	0-250	0	Number of pre-forked vmware collector instances. This parameter is supported since Zabbix 2.2.0.
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:::1 This parameter is supported since Zabbix 4.0.5.
Timeout	no	1-30	3	Specifies how long we wait for agent, SNMP device or external check (in seconds).

Parameter	Mandatory	Range	Default	Description
TLSAccept	yes for passive proxy, if TLS certificate or PSK parameters are defined (even for <i>unencrypted</i> 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: <i>unencrypted</i> - accept connections without encryption (default) <i>psk</i> - accept connections with TLS and a pre-shared key (PSK) <i>cert</i> - 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.
TLSCConnect	yes for active proxy, if TLS certificate or PSK parameters are defined (even for <i>unencrypted</i> 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: <i>unencrypted</i> - connect without encryption (default) <i>psk</i> - connect using TLS and a pre-shared key (PSK) <i>cert</i> - 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.

Parameter	Mandatory	Range	Default	Description
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. This parameter is supported since Zabbix 2.4.0.
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.
VMwareCacheSize	no	256K-2G	8M	Shared memory size for storing VMware data. A VMware internal check <code>zabbix[vmware,buffer,...]</code> 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. This parameter is supported since Zabbix 2.2.0.

Parameter	Mandatory	Range	Default	Description
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. This parameter is supported since Zabbix 2.2.0.
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. This parameter is supported since Zabbix 2.2.9, 2.4.4
VMwareTimeout	no	1-300	10	The maximum number of seconds vmware collector will wait for a response from VMware service (vCenter or ESX hypervisor). This parameter is supported since Zabbix 2.2.9, 2.4.4

Note:

Zabbix supports configuration files only in UTF-8 encoding without **BOM**.

Comments starting with “#” are only supported in the beginning of the line.

3 Zabbix agent (UNIX)

Note:

The default values reflect daemon defaults, not the values in the shipped configuration files.

The parameters supported in a Zabbix agent configuration file (*zabbix_agentd.conf*):

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 <i>Alias</i> parameters may be present. Multiple parameters with the same <i>Alias</i> key are allowed. Different <i>Alias</i> keys may reference the same item key. Aliases can be used in <i>HostMetadataItem</i> but not in <i>HostnameItem</i> parameters. Examples: - Retrieving the ID of user 'zabbix'. Alias=zabbix.userid:vfs.file.regexp[/etc/passwd/9]+)"",\1] Now shorthand key zabbix.userid may be used to retrieve data. - 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 utilisation percentage with default parameters as well as use cpu.util[all, idle, avg15] to get specific data about CPU utilisation. - 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.
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.

Parameter	Mandatory	Range	Default	Description
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)
EnableRemoteCommands	no		0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed
HostMetadata	no	0-255 characters		Optional parameter that defines host metadata. Host metadata is used only at host auto-registration 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. This option is supported in version 2.2.0 and higher.

Parameter	Mandatory	Range	Default	Description
HostMetadataItem	no			Optional parameter that defines a <i>Zabbix agent</i> item used for getting host metadata. This option is only used when HostMetadata is not defined. Supports UserParameters and aliases. Supports <i>system.run[]</i> regardless of <i>EnableRemoteCommands</i> value. HostMetadataItem value is retrieved on each auto-registration attempt and is used only at host auto-registration process (active agent). During an auto-registration 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. This option is supported in version 2.2.0 and higher.
Hostname	no		Set by HostnameItem	Unique, case sensitive hostname. Required for active checks and must match hostname as configured on the server. Allowed characters: alphanumeric, '.', '-', '_' and '-'. Maximum length: 64
HostnameItem	no		system.hostname	Optional parameter that defines a <i>Zabbix agent</i> 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 <i>system.run[]</i> regardless of <i>EnableRemoteCommands</i> value. This option is supported in version 1.8.6 and higher.

Parameter	Mandatory	Range	Default	Description
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: <code>/absolute/path/to/config/files/*. Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.</code>
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> (since 4.2.3) LoadModule=</abs_path/module.so> Either the module must be located in directory specified by LoadModulePath or (since 4.2.3) 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 <i>file</i> , otherwise no			Name of log file.
LogFileSize	no	0-1024	1	Maximum size of log file in MB. 0 - disable automatic log rotation. <i>Note:</i> If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

Parameter	Mandatory	Range	Default	Description
LogType	no		file	Log output type: <i>file</i> - write log to file specified by LogFile parameter, <i>system</i> - write log to syslog, <i>console</i> - 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 when processing 'log' and 'eventlog' active checks. The provided value will be overridden by the parameter 'maxlines', provided in 'log' or 'eventlog' item key. <i>Note:</i> Zabbix will process 10 times more new lines than set in <i>MaxLinesPerSecond</i> 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. <i>Note</i> 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. <i>Note</i> , 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: Spaces are allowed.

Parameter	Mandatory	Range	Default	Description
ServerActive	no			IP:port (or hostname:port) of Zabbix server or Zabbix proxy for active checks. 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.
SourceIP	no			Source IP address for outgoing connections.
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. The upper limit used to be 16 before version 1.8.5.
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 <i>unencrypted</i> connection), otherwise no			What incoming connections to accept. Used for a passive checks. Multiple values can be specified, separated by comma: <i>unencrypted</i> - accept connections without encryption (default) <i>psk</i> - accept connections with TLS and a pre-shared key (PSK) <i>cert</i> - 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.

Parameter	Mandatory	Range	Default	Description
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.
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for <i>unencrypted</i> connection), otherwise no			How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified: <i>unencrypted</i> - connect without encryption (default) <i>psk</i> - connect using TLS and a pre-shared key (PSK) <i>cert</i> - 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.

Parameter	Mandatory	Range	Default	Description
UnsafeUserParameters	no	0,1	0	Allow all characters to be passed in arguments to user-defined parameters. Supported since Zabbix 1.8.2. 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. This parameter is supported since Zabbix 2.4.0.
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. Example: UserParameter=system.test,who wc -l

Note:

In Zabbix agent 2.0.0 version configuration parameters related to active and passive checks have been changed. See the ["See also"](#) section at the bottom of this page to read more details about these changes.

Note:

Zabbix supports configuration files only in UTF-8 encoding without [BOM](#).

Comments starting with "#" are only supported in the beginning of the line.

See also

1. [Differences in the Zabbix agent configuration for active and passive checks starting from version 2.0.0](#)

4 Zabbix agent (Windows)

Note:

The default values reflect daemon defaults, not the values in the shipped configuration files.

The parameters supported in a Zabbix agent (Windows) configuration file:

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 <i>Alias</i> parameters may be present. Multiple parameters with the same <i>Alias</i> key are allowed. Different <i>Alias</i> keys may reference the same item key. Aliases can be used in <i>HostMetadataItem</i> but not in <i>HostnameItem</i> or <i>PerfCounter</i> parameters. Examples: - 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. - 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 utilisation percentage with default parameters as well as use cpu.load[percpu,avg15] to get specific data about CPU load. - 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.
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)
EnableRemoteCommands	no		0	Whether remote commands from Zabbix server are allowed. 0 - not allowed 1 - allowed
HostMetadata	no	0-255 characters		Optional parameter that defines host metadata. Host metadata is used only at host auto-registration 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. This option is supported in version 2.2.0 and higher.
HostMetadataItem	no			Optional parameter that defines a <i>Zabbix agent</i> item used for getting host metadata. This option is only used when HostMetadata is not defined. Supports UserParameters, performance counters and aliases. Supports <i>system.run[]</i> regardless of <i>EnableRemoteCommands</i> value. HostMetadataItem value is retrieved on each auto-registration attempt and is used only at host auto-registration process (active agent). During an auto-registration 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. This option is supported in version 2.2.0 and higher.

Parameter	Mandatory	Range	Default	Description
Hostname	no		Set by HostnameItem	Unique, case sensitive hostname. Required for active checks and must match hostname as configured on the server. Allowed characters: alphanumeric, '.', '-', '_' and '-'. Maximum length: 64
HostnameItem	no		system.hostname	Optional parameter that defines a <i>Zabbix agent</i> 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 <i>system.run[]</i> regardless of <i>EnableRemoteCommands</i> value. This option is supported in version 1.8.6 and higher. See also a more detailed description .
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: <code>/absolute/path/to/config/files/*</code> . Pattern matching is supported since Zabbix 2.4.0. See special notes about limitations.
ListenIP	no		0.0.0.0	List of comma-delimited IP addresses that the agent should listen on. Multiple IP addresses are supported since Zabbix 1.8.3.
ListenPort	no	1024-32767	10050	Agent will listen on this port for connections from the server.
LogFile	yes, if LogType is set to <i>file</i> , 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. <i>Note:</i> If the log file size limit is reached and file rotation fails, for whatever reason, the existing log file is truncated and started anew.

Parameter	Mandatory	Range	Default	Description
LogType	no		file	Log output type: <i>file</i> - write log to file specified by LogFile parameter, <i>system</i> - write log Windows Event Log, <i>console</i> - 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. <i>Note:</i> Zabbix will process 10 times more new lines than set in <i>MaxLinesPerSecond</i> to seek the required string in log items.
PerfCounter	no			Syntax: <parameter_name>,"<perf_counter_path>",<period> Defines new parameter <parameter_name> which is an average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds). For example, if you wish to receive average number of processor interrupts per second for last minute, you can define new parameter "interrupts" as 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			Syntax: <parameter_name>,"<perf_counter_path>",<period> Defines new parameter <parameter_name> which is an average value for system performance counter <perf_counter_path> for the specified time period <period> (in seconds). 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 new parameter "interrupts" as 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\\CurrentVersion\\Perflib\\009. This parameter is supported since Zabbix 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.

Parameter	Mandatory	Range	Default	Description
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::
ServerActive	no	(*)		Spaces are allowed. IP:port (or hostname:port) of Zabbix server or Zabbix proxy for active checks. 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.
SourceIP	no			Source IP address for outgoing connections.
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. The upper limit used to be 16 before version 1.8.5.
Timeout	no	1-30	3	Spend no more than Timeout seconds on processing

Parameter	Mandatory	Range	Default	Description
TLSAccept	yes, if TLS certificate or PSK parameters are defined (even for <i>unencrypted</i> connection), otherwise no			What incoming connections to accept. Used for a passive checks. Multiple values can be specified, separated by comma: <i>unencrypted</i> - accept connections without encryption (default) <i>psk</i> - accept connections with TLS and a pre-shared key (PSK) <i>cert</i> - 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.
TLSConnect	yes, if TLS certificate or PSK parameters are defined (even for <i>unencrypted</i> connection), otherwise no			How the agent should connect to Zabbix server or proxy. Used for active checks. Only one value can be specified: <i>unencrypted</i> - connect without encryption (default) <i>psk</i> - connect using TLS and a pre-shared key (PSK) <i>cert</i> - 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.

Parameter	Mandatory	Range	Default	Description
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				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. Example: UserParameter=system.test,echo 1

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.

Note:

In Zabbix agent 2.0.0 version configuration parameters related to active and passive checks have been changed. See the ["See also"](#) section at the bottom of this page to read more details about these changes.

Note:

Zabbix supports configuration files only in UTF-8 encoding without [BOM](#).

Comments starting with "#" are only supported in the beginning of the line.

See also

1. Differences in the Zabbix agent configuration for active and passive checks starting from version 2.0.0.

5 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	<code>zabbix.listenIP</code>	no		0.0.0.0	IP address to listen on.
LISTEN_PORT	<code>zabbix.listenPort</code>	no	1024-32767	10052	Port to listen on.
PID_FILE	<code>zabbix.pidFile</code>	no		<code>/tmp/zabbix_java.pid</code>	Name of PID file. If omitted, Zabbix Java Gateway is started as a console application.
START_POLLERS	<code>zabbix.startPollers</code>	no	1-1000	5	Number of worker threads to start.
TIMEOUT	<code>zabbix.timeout</code>	no	1-30	3	How long to wait for network operations. This parameter is supported since Zabbix 2.0.15, 2.2.10 and 2.4.5.

Warning:

Port 10052 is not [IANA registered](#).

6 Special notes on "Include" parameter

If an Include parameter is used for including a file, the file must be readable.

If an 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)
- 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 '...

If an 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)

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	the request success information ('success' or 'failed')
version	string	the 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,
        "Template OS Linux",
        3,
        -1,
        2,

```

```

        "",
        "",
        "Template OS Linux",
        1,
        1,
        "",
        "",
        "",
        ""
    ],
    [
        10050,
        "Template App Zabbix Agent",
        3,
        -1,
        2,
        "",
        "",
        "Template App 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": "3.4.0"
}

```

Proxy request

The proxy data request is used to obtain host 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
host availability	array	(optional) array of host availability data objects
	hostid number	host identifier
	available number	Zabbix agent availability
		0, HOST_AVAILABLE_UNKNOWN - unknown 1, HOST_AVAILABLE_TRUE - available 2, HOST_AVAILABLE_FALSE - unavailable
	error string	Zabbix agent error message or empty string
	snmp_available number	SNMP agent availability
		0, HOST_AVAILABLE_UNKNOWN - unknown 1, HOST_AVAILABLE_TRUE - available 2, HOST_AVAILABLE_FALSE - unavailable
	snmp_error string	SNMP agent error message or empty string
	ipmi_available number	IPMI agent availability
		0, HOST_AVAILABLE_UNKNOWN - unknown 1, HOST_AVAILABLE_TRUE - available 2, HOST_AVAILABLE_FALSE - unavailable
	ipmi_error string	IPMI agent error message or empty string
	jmx_available number	JMX agent availability
		0, HOST_AVAILABLE_UNKNOWN - unknown 1, HOST_AVAILABLE_TRUE - available 2, HOST_AVAILABLE_FALSE - unavailable
	jmx_error string	JMX agent 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

name	value type	description
discovery data	eventid number	(optional) eventlog item eventid value
	state string	(optional) item state 0 , <i>ITEM_STATE_NORMAL</i> 1 , <i>ITEM_STATE_NOTSUPPORTED</i>
	lastlogsize number	(optional) last log size of log type items
	mtime number	(optional) modify time of log type items
	data array	(optional) array of discovery data objects
	clock number	the discovery data timestamp
	druleid number	the discovery rule identifier
	dcheckid number	the discovery check identifier or null for discovery rule data
	type number	the discovery check type: -1 discovery rule data 0 , <i>SVC_SSH</i> - SSH service check 1 , <i>SVC_LDAP</i> - LDAP service check 2 , <i>SVC_SMTP</i> - SMTP service check 3 , <i>SVC_FTP</i> - FTP service check 4 , <i>SVC_HTTP</i> - HTTP service check 5 , <i>SVC_POP</i> - POP service check 6 , <i>SVC_NNTP</i> - NNTP service check 7 , <i>SVC_IMAP</i> - IMAP service check 8 , <i>SVC_TCP</i> - TCP port availability check 9 , <i>SVC_AGENT</i> - Zabbix agent 10 , <i>SVC_SNMPv1</i> - SNMPv1 agent 11 , <i>SVC_SNMPv2</i> - SNMPv2 agent 12 , <i>SVC_ICMP</i> - ICMP ping 13 , <i>SVC_SNMPv3</i> - SNMPv3 agent 14 , <i>SVC_HTTPS</i> - HTTPS service check 15 , <i>SVC_TELNET</i> - Telnet availability check
	ip string	the host IP address
	dns string	the host DNS name
	port number	(optional) service port number
	key_ string	(optional) the item key for discovery check of type 9 <i>SVC_AGENT</i>
	value string	(optional) value received from the service, can be empty for most of services
	status number	(optional) service status: 0 , <i>DOBJECT_STATUS_UP</i> - Service UP 1 , <i>DOBJECT_STATUS_DOWN</i> - Service DOWN
auto registration	data array	(optional) array of auto registration data objects
	clock number	the auto registration data timestamp
	host string	the host name
	ip string	(optional) the host IP address
	dns string	(optional) the resolved DNS name from IP address
	port string	(optional) the host port
tasks	host_metadata string	(optional) the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)
	tasks array	(optional) array of tasks
	type number	the task type: 0 , <i>ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT</i> - remote command result

name	value type	description
status	number	the remote command execution status: 0 , ZBX_TM_REMOTE_COMMAND_COMPLETED - the remote command completed successfully 1 , ZBX_TM_REMOTE_COMMAND_FAILED - the remote command failed
error	string	(optional) the error message
parent_task_id	number	the 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	the proxy version (<major>.<minor>.<build>)
server→proxy: response	string	the request success information ('success' or 'failed')
tasks	array	(optional) array of tasks
type	number	the task type: 1 , ZBX_TM_TASK_PROCESS_REMOTE_COMMAND - remote command the task creation time the time in seconds after which task expires the 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) the remote command to execute the 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
clock	number	(optional) the port for telnet and ssh commands
ttl	number	(optional) the authentication type for ssh commands
command	string	(optional) the user name for telnet and ssh commands
execute_on	number	(optional) the password for telnet and ssh commands
port	number	(optional) the public key for ssh commands
auth_type	number	(optional) the private key for ssh commands
username	string	the parent task id
password	string	target hostid
public_key	string	
private_key	string	
parent_task_id	number	
hostid	number	

Example:

server→proxy:


```
{
  "request": "proxy data"
}
```

proxy→server:

```
{
  "session": "12345678901234567890123456789012"
  "host availability":[
    {
      "hostid":10106,
      "available":1,
      "error": "",
      "snmp_available":0,
      "snmp_error": "",
      "ipmi_available":0,
      "ipmi_error": "",
      "jmx_available":0,
      "jmx_error": ""
    },
    {
      "hostid":10107,
      "available":1,
      "error": "",
      "snmp_available":0,
      "snmp_error": "",
      "ipmi_available":0,
      "ipmi_error": "",
      "jmx_available":0,
      "jmx_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":"3.4.0"
}

```

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": "MIGfMAOGCSqGSIB3DQEBAQUAA4GNADCBiQKBgQCqGKuk01De7zhZj6+H0qtjTkVxwTCpvKe",
      "privatekey": "lsuusFncCzWBQ7RKNUSeSmQRMSGkVb1/3j+skZ6UtW+5u09lHNSj6tQ5QCqGKuk01De7zhd",
      "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	the proxy name
version	string	the proxy version (<major>.<minor>.<build>)
server→proxy:		
response	string	the request success information ('success' or 'failed')

proxy→server:

```

{
  "request": "proxy heartbeat",
  "host": "Proxy #12",
  "version": "3.4.0"
}

```

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	the 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 column type in database schema
proxy→server:		

name	value type	description
response	<i>string</i>	the request success information ('success' or 'failed')

Example:

proxy→server:

```
{
  "request": "proxy config",
  "host": "Proxy #12",
  "version": "3.4.0"
}
```

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,
        "Template OS Linux",
        3,
        -1,
        2,
        "",
        "",
        "Template OS Linux",
        1,
        1,
        "",
        "",
        ""
      ]
    ]
  }
}
```

```

        ""
    ],
    [
        10050,
        "Template App Zabbix Agent",
        3,
        -1,
        2,
        "",
        "",
        "Template App 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 availability, history, discovery and auto registration data. This request is sent every DataSenderFrequency (proxy configuration parameter) seconds.

name	value type	description
proxy→server:		
request	string	'proxy data'
host	string	the proxy name
session	string	data session token
host availability	array	(optional) array of host availability data objects
hostid	number	host identifier
available	number	Zabbix agent availability 0 , HOST_AVAILABLE_UNKNOWN - unknown 1 , HOST_AVAILABLE_TRUE - available 2 , HOST_AVAILABLE_FALSE - unavailable
error	string	Zabbix agent error message or empty string
snmp_available	number	SNMP agent availability 0 , HOST_AVAILABLE_UNKNOWN - unknown 1 , HOST_AVAILABLE_TRUE - available 2 , HOST_AVAILABLE_FALSE - unavailable
snmp_error	string	SNMP agent error message or empty string
ipmi_available	number	IPMI agent availability 0 , HOST_AVAILABLE_UNKNOWN - unknown 1 , HOST_AVAILABLE_TRUE - available 2 , HOST_AVAILABLE_FALSE - unavailable
ipmi_error	string	IPMI agent error message or empty string
jmx_available	number	JMX agent availability 0 , HOST_AVAILABLE_UNKNOWN - unknown 1 , HOST_AVAILABLE_TRUE - available 2 , HOST_AVAILABLE_FALSE - unavailable
jmx_error	string	JMX agent 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) modify time of log type items
discovery data	array	(optional) array of discovery data objects
clock	number	the discovery data timestamp
druleid	number	the discovery rule identifier

name	value type	description
id	number	the discovery check identifier or null for discovery rule data
dcheckid	number	the discovery check type:
type	number	-1 discovery rule data 0, <i>SVC_SSH</i> - SSH service check 1, <i>SVC_LDAP</i> - LDAP service check 2, <i>SVC_SMTP</i> - SMTP service check 3, <i>SVC_FTP</i> - FTP service check 4, <i>SVC_HTTP</i> - HTTP service check 5, <i>SVC_POP</i> - POP service check 6, <i>SVC_NNTP</i> - NNTP service check 7, <i>SVC_IMAP</i> - IMAP service check 8, <i>SVC_TCP</i> - TCP port availability check 9, <i>SVC_AGENT</i> - Zabbix agent 10, <i>SVC_SNMPv1</i> - SNMPv1 agent 11, <i>SVC_SNMPv2</i> - SNMPv2 agent 12, <i>SVC_ICMPING</i> - ICMP ping 13, <i>SVC_SNMPv3</i> - SNMPv3 agent 14, <i>SVC_HTTPS</i> - HTTPS service check 15, <i>SVC_TELNET</i> - Telnet availability check
ip	string	the host IP address
dns	string	the host DNS name
port	number	(optional) service port number
key_	string	(optional) the item key for discovery check of type 9 <i>SVC_AGENT</i>
value	string	(optional) value received from the service, can be empty for most of services
status	number	(optional) service status:
		0, <i>DOBJECT_STATUS_UP</i> - Service UP
		1, <i>DOBJECT_STATUS_DOWN</i> - Service DOWN
auto registration	array	(optional) array of auto registration data objects
clock	number	the auto registration data timestamp
host	string	the host name
ip	string	(optional) the host IP address
dns	string	(optional) the resolved DNS name from IP address
port	string	(optional) the host port
host_metadata	string	(optional) the host metadata sent by agent (based on HostMetadata or HostMetadataItem agent configuration parameter)
tasks	array	(optional) array of tasks
type	number	the task type:
		0, <i>ZBX_TM_TASK_PROCESS_REMOTE_COMMAND_RESULT</i> - remote command result
status	number	the remote command execution status:
		0, <i>ZBX_TM_REMOTE_COMMAND_COMPLETED</i> - the remote command completed successfully
		1, <i>ZBX_TM_REMOTE_COMMAND_FAILED</i> - the remote command failed
error	string	(optional) the error message
parent_taskid	number	the 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)

name	value type	description
version	<i>string</i>	the proxy version (<major>.<minor>.<build>)
server→proxy:		
response	<i>string</i>	the request success information ('success' or 'failed')
tasks	<i>array</i>	(optional) array of tasks
type	<i>number</i>	the task type:
		1, ZBX_TM_TASK_PROCESS_REMOTE_COMMAND - remote command the task creation time the time in seconds after which task expires the 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) the remote command to execute the 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 (optional) the port for telnet and ssh commands (optional) the authentication type for ssh commands (optional) the user name for telnet and ssh commands (optional) the password for telnet and ssh commands (optional) the public key for ssh commands (optional) the private key for ssh commands the parent task id target hostid
clock	<i>number</i>	
ttl	<i>number</i>	
command	<i>string</i>	
execute_on	<i>number</i>	
port	<i>number</i>	
auth_type	<i>number</i>	
username	<i>string</i>	
password	<i>string</i>	
public_key	<i>string</i>	
private_key	<i>string</i>	
parent_task_id	<i>number</i>	
hostid	<i>number</i>	

Example:

proxy→server:

```
{
  "request": "proxy data",
  "host": "Proxy #12",
  "session": "12345678901234567890123456789012",
  "host_availability": [
    {
      "hostid": 10106,
      "available": 1,
      "error": "",
      "snmp_available": 0,
      "snmp_error": "",
      "ipmi_available": 0,
      "ipmi_error": "",
      "jmx_available": 0,
      "jmx_error": ""
    }
  ]
}
```



```

    },
    {
        "hostid":10107,
        "available":1,
        "error":"",
        "snmp_available":0,
        "snmp_error":"",
        "ipmi_available":0,
        "ipmi_error":"",
        "jmx_available":0,
        "jmx_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": "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKuk01De7zhZj6+H0qtjTkVxwTCpvKe",
        "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u091HNsj6tQ5QCqGKuk01De7zhd",
        "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": "3.4.0"
}

```

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": "MIGfMAOGCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKuk01De7zhZj6+H0qtjTkVxwTCpvKe",
        "privatekey": "lsuusFncCzWBQ7RKNUSesmQRMSGkVb1/3j+skZ6UtW+5u091HNsj6tQ5QCqGKuk01De7zhd",
        "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 sender protocol

Please refer to [Trapper items](#) page for more information.

4 Header

Overview

Header is present in response and request messages between Zabbix components. It is required to determine the length of message. The header consists of:

<PROTOCOL> - "ZBXD" (4 bytes).

<FLAGS> - the protocol flags, (1 byte). 0x01 - Zabbix communications protocol, 0x02 - compression).

<DATALEN> - data length (4 bytes). 1 will be formatted as 01/00/00/00 (four bytes, 32 bit number in little-endian format).

<RESERVED> - reserved for protocol extensions (4 bytes).

When compression is enabled (0x02 flag) the <RESERVED> bytes contains uncompressed data size, 32 bit number in little-endian format.

To not exhaust memory (potentially) Zabbix protocol is limited to accept only 128MB in one connection.

Implementation

Here are code snippets showing how to add Zabbix protocol header to the data you *want* to send in order to obtain packet you *should* send to Zabbix so it is interpreted correctly.

Language	Code
bash	<code>printf -v LENGTH '%016x' "\${#DATA}"PACK=</code>
Java	<code>byte[] header = new byte[] {'Z', 'B', 'X', 'D'}</code>
PHP	<code>\$packet = "ZBXD\1" . pack('P', strlen(\$d</code>

Language	Code
Perl	<code>my \$packet = "ZBXD\1" . pack('<Q', length;</code>
Python	<code>packet = "ZBXD\1" + struct.pack('<Q', le</code>

5 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
<i>hosts</i>	array	List of hosts involved in the trigger expression; there should be at least one element in array.
-	string	Visible host name.
<i>groups</i>	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.
<i>tags</i>	array	List of problem tags (can be empty).
-	object	
-	<i>tag</i> string	Tag name.
-	<i>value</i> string	Tag value (can be empty).
<i>name</i>	string	Problem event name.
<i>clock</i>	number	Number of seconds since Epoch to the moment when problem was detected (integer part).
<i>ns</i>	number	Number of nanoseconds to be added to <code>clock</code> to get a precise problem detection time.
<i>eventid</i>	number	Problem event ID.
<i>value</i>	number	1 (always).

The following information is exported for a recovery event:

Field	Type	Description
<i>clock</i>	number	Number of seconds since Epoch to the moment when problem was resolved (integer part).
<i>ns</i>	number	Number of nanoseconds to be added to <code>clock</code> to get a precise problem resolution time.
<i>eventid</i>	number	Recovery event ID.
<i>p_eventid</i>	number	Problem event ID.
<i>value</i>	number	0 (always).

Examples

Problem:

```
{ "hosts": ["Host B", "Zabbix Server"], "groups": ["Group X", "Group Y", "Group Z", "Zabbix servers"], "tags": [{"ta
```

Recovery:

```
{ "clock": 1519304345, "ns": 987654321, "eventid": 43, "p_eventid": 42, "value": 0 }
```

Problem (multiple problem event generation):

```
{ "hosts": ["Host B", "Zabbix Server"], "groups": ["Group X", "Group Y", "Group Z", "Zabbix servers"], "tags": [{"ta
```

```
{ "hosts": ["Host B", "Zabbix Server"], "groups": ["Group X", "Group Y", "Group Z", "Zabbix servers"], "tags": [{"ta
```

Recovery:

```
{"clock":1519304346,"ns":987654321,"eventid":44,"p_eventid":43,"value":0}
```

```
{"clock":1519304346,"ns":987654321,"eventid":44,"p_eventid":42,"value":0}
```

Item values

The following information is exported for a collected item value:

Field	Type	Description
<i>host</i>	string	Visible host name of the item host.
<i>groups</i>	array	List of host groups of the item host; there should be at least one element in array.
<i>applications</i>	- string	Host group name.
	array	List of the item applications; empty if there are none.
<i>itemid</i>	- string	Application name.
	number	Item ID.
<i>name</i>	string	Visible item name.
<i>clock</i>	number	Number of seconds since Epoch to the moment when value was collected (integer part).
<i>ns</i>	number	Number of nanoseconds to be added to <i>clock</i> to get a precise value collection time.
<i>timestamp</i> (Log only)	number	0 if not available.
<i>source</i> (Log only)	string	Empty string if not available.
<i>severity</i> (Log only)	number	0 if not available.
<i>eventid</i> (Log only)	number	0 if not available.
<i>value</i>	number (for numeric items) or string (for text items)	Collected item value.

Examples

Numeric (unsigned) value:

```
{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["Zabbix Agent","Availability"],"itemid":4,"value":123456789}
```

Numeric (float) value:

```
{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["CPU","Performance"],"itemid":4,"value":12.3456789}
```

Character, text value:

```
{"host":"Host B","groups":["Group X","Group Y","Group Z"],"applications":["Zabbix Agent","Installed software"],"itemid":4,"value":"Zabbix Agent 5.0.10 (64-bit)"}
```

Log value:

```
{"host":"Host A","groups":["Group X","Group Y","Group Z"],"applications":["Log files","Critical"],"itemid":4,"value":"2017-09-01 12:34:56 [Critical] Zabbix Agent 5.0.10 (64-bit)"}>
```

Trends

The following information is exported for a calculated trend value:

Field	Type	Description
<i>host</i>	string	Visible host name of the item host.
<i>groups</i>	array	List of host groups of the item host; there should be at least one element in array.
<i>applications</i>	- string	Host group name.
	array	List of the item applications; empty if there are none.
<i>itemid</i>	- string	Application name.
	number	Item ID.
<i>name</i>	string	Visible item name.
<i>clock</i>	number	Number of seconds since Epoch to the moment when value was collected (integer part).

Field	Type	Description
<i>count</i>	number	Number of values collected for a given hour.
<i>min</i>	number	Minimum item value for a given hour.
<i>avg</i>	number	Average item value for a given hour.
<i>max</i>	number	Maximum item value for a given hour.

Examples

Numeric (unsigned) value:

```
{"host": "Host B", "groups": ["Group X", "Group Y", "Group Z"], "applications": ["Zabbix Agent", "Availability"], "itemid": 4,
```

Numeric (float) value:

```
{"host": "Host B", "groups": ["Group X", "Group Y", "Group Z"], "applications": ["CPU", "Performance"], "itemid": 4,
```

5 Items

1 Items supported by platform

The table displays support for Zabbix **agent items** on various platforms:

- Items marked with "X" are supported, the ones marked with "-" are not supported.
- If an item is marked with "?", it is not known whether it is supported or not.
- If an item is marked with "r", it means that it requires root privileges.
- Parameters that are included in angle brackets <like_this> are optional.

Note:

Windows-only Zabbix agent items are not included in this table.

NetBSD											
OpenBSD											▼▼
Mac										▼▼	
OS X											
Tru64									▼▼		
AIX								▼▼			
HP-UX							▼▼				
Solaris								▼▼			
FreeBSD					▼▼						
Linux			▼▼								
2.6											
(and later)											
Linux			▼▼								
2.4											
Windows		▼▼									
Parameter	▼▼										
/ sys-tem											
▼▼	1	2	3	4	5	6	7	8	9	10	11
agent.hostname	X	X	X	X	X	X	X	X	X	X	X
agent.ping	X	X	X	X	X	X	X	X	X	X	X
agent.version	X	X	X	X	X	X	X	X	X	X	X
kernel.maxfiles	-	X	X	X	-	-	-	?	X	X	X
kernel.maxproc	-	-	X	X	X	-	-	?	X	X	X
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>]	X	X	X	X	X	X	X	X	X	X	X
log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>]	X	X	X	X	X	X	X	X	X	X	X
logrt[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>]	X	X	X	X	X	X	X	X	X	X	X
logrt.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>]	X	X	X	X	X	X	X	X	X	X	X
net.dns[<ip>,<zone>,<type>,<timeout>,<count>]	X	X	X	X	X	X	X	X	X	X	X

net.dns.record[~~<ip>~~,~~zone~~,~~<type>~~,~~<timeout>~~,~~<count>~~]											
		X	X	X	X	X	X	X	X	X	X
net.if.collisions[if]											
		X	X	X	X	-	X	-	X	X	r
net.if.discovery											
		X	X	X	X	X	X	-	-	X	X
net.if.in[if,~~<mode>~~]											
		X	X	X	X	X ¹	X	-	X	X	r
mode	bytes	X	X	X	X	X ²	X	X	-	X	r
▲ (de-fault)											
	packets	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X ²	X	X	-	X	r
	dropped	X	X	X	X	-	X	-	-	X	r
net.if.out[if,~~<mode>~~]											
		X	X	X	X	X ¹	X	-	X	X	r
mode	bytes	X	X	X	X	X ²	X	X	-	X	r
▲ (de-fault)											
	packets	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X ²	X	X	-	X	r
	dropped	X	X	X	-	-	X	-	-	-	-
net.if.total[if,~~<mode>~~]											
		X	X	X	X	X ¹	X	-	X	X	r
mode	bytes	X	X	X	X	X ²	X	X	-	X	r
▲ (de-fault)											
	packets	X	X	X	X	X	X	-	X	X	r
	errors	X	X	X	X	X ²	X	X	-	X	r
	dropped	X	X	X	-	-	X	-	-	-	-
net.tcp.listen[port]											
		X	X	X	X	-	-	-	X	-	-
net.tcp.port[~~<ip>~~,port]											
		X	X	X	X	X	X	X	X	X	X
net.tcp.service[~~service~~,~~ip~~,<port>]											
		X	X	X	X	X	X	X	X	X	X
net.tcp.service.perf[~~service~~,~~ip~~,<port>]											
		X	X	X	X	X	X	X	X	X	X
net.udp.listen[port]											
		X	X	X	X	-	-	-	X	-	-
net.udp.service[~~service~~,~~ip~~,<port>]											
		X	X	X	X	X	X	X	X	X	X
net.udp.service.perf[~~service~~,~~ip~~,<port>]											
		X	X	X	X	X	X	X	X	X	X
		1	2	3	4	5	6	7	8	9	10
proc.cpu.util[~~<name>~~,~~<user>~~,~~<type>~~,~~<cmdline>~~,~~<mode>~~,~~<zone>~~]											
type	total	-	X	X	-	X	-	-	-	-	-
▲ (de-fault)											
	user	-	X	X	-	X	-	-	-	-	-
	system	-	X	X	-	X	-	-	-	-	-
mode	avg1	-	X	X	-	X	-	-	-	-	-
▲ (de-fault)											
	avg5	-	X	X	-	X	-	-	-	-	-
	avg15	-	X	X	-	X	-	-	-	-	-
zone	current	-	-	-	-	X	-	-	-	-	-
▲ (de-fault)											
	all	-	-	-	-	X	-	-	-	-	-
proc.mem[~~<name>~~,~~<user>~~,~~<mode>~~,~~<cmdline>~~,~~<memtype>~~]											
mode	sum	-	X	X	X	X	-	X	X	-	X
▲ (de-fault)											
	avg	-	X	X	X	X	-	X	X	-	X
	max	-	X	X	X	X	-	X	X	-	X
	min	-	X	X	X	X	-	X	X	-	X
memtype		-	X	X	X	X	-	X	-	-	-
▲											
proc.num[~~<name>~~,~~<user>~~,~~<state>~~,~~<cmdline>~~]											
state	all	-	X	X	X	X	X	X	X	-	X
▲ (de-fault)											
	disk	-	X	X	X	-	-	-	-	-	X
	sleep	-	X	X	X	X	X	X	X	-	X

	zomb	-	X	X	X	X	X	X	X	-	X	X
	run	-	X	X	X	X	X	X	X	-	X	X
	trace	-	X	X	X	-	-	-	-	-	X	X
cmdline		-	X	X	X	X	X	X	X	-	X	X
▲												
	sensor[device,sensor,<mode>]		X	-	-	-	-	-	-	-	X	-
	system.boottime		X	X	X	X	-	-	-	X	X	X
	system.cpu.discovery		X	X	X	X	X	X	X	X	X	X
	system.cpu.intr	-	X	X	X	X	-	X	-	-	X	X
	system.cpu.load[*cpu>,<mode>]	X	X	X	X	X	X	X	X	X	X	X
cpu ▲	all	X	X	X	X	X	X	X	X	X	X	X
	(de-fault)											
	percpu	X	X	X	X	X	X	X	-	X	X	X
mode	avg1	X	X	X	X	X	X	X	X	X	X	X
▲	(de-fault)											
	avg5	X	X	X	X	X	X	X	X	X	X	X
	avg15	X	X	X	X	X	X	X	X	X	X	X
	system.cpu.num[*type>]	X	X	X	X	X	X	X	-	X	X	X
type	online	X	X	X	X	X	X	X	-	X	X	X
▲	(de-fault)											
	max	-	X	X	X	X	-	-	-	X	-	-
	system.cpu.switches		X	X	X	X	-	X	-	-	X	X
	system.cpu.util[*cpu>,<type>,<mode>]	X	X	X	X	X	X	X	X	-	X	X
type	user	-	X	X	X	X	X	X	X	-	X	X
▲	(de-fault)											
	nice	-	X	X	X	-	X	-	X	-	X	X
	idle	-	X	X	X	X	X	X	X	-	X	X
	system	X	X	X	X	X	X	X	X	-	X	X
	iowait	-	-	X	-	X	-	X	-	-	-	-
	interrupt	-	-	X	X	-	-	-	-	-	X	-
	softirq	-	-	X	-	-	-	-	-	-	-	-
	steal	-	-	X	-	-	-	-	-	-	-	-
	guest	-	-	X	-	-	-	-	-	-	-	-
	guest_nice	-	-	X	-	-	-	-	-	-	-	-
mode	avg1	X	X	X	X	X	X	X	X	-	X	X
▲	(de-fault)											
	avg5	X	X	X	X	X	X	X	-	-	X	X
	avg15	X	X	X	X	X	X	X	-	-	X	X
	1	2	3	4	5	6	7	8	9	10	11	
	system.hostname[*type>]		X	X	X	X	X	X	X	X	X	X
	system.hw.chassis[*info>]	X	X	-	-	-	-	-	-	-	-	-
	system.hw.cpu[*cpu>,<info>]	X	X	-	-	-	-	-	-	-	-	-
	system.hw.devices[*type>]	X	X	-	-	-	-	-	-	-	-	-
	system.hw.macaddr[*interface>,<format>]		X	X	X	X	X	X	X	X	X	X
	system.localtime[*type>]	X	X	X	X	X	X	X	X	X	X	X
type	utc	X	X	X	X	X	X	X	X	X	X	X
▲	(de-fault)											
	local	X	X	X	X	X	X	X	X	X	X	X
	system.run[command,<mode>]	X	X	X	X	X	X	X	X	X	X	X
mode	wait	X	X	X	X	X	X	X	X	X	X	X
▲	(de-fault)											
	nowait	X	X	X	X	X	X	X	X	X	X	X
	system.stat[resource,<type>]	-	-	-	-	-	X	-	-	-	-	-
	system.sw.arch	X	X	X	X	X	X	X	X	X	X	X
	system.sw.os[*info>]	X	X	-	-	-	-	-	-	-	-	-

system.sw.packages[<package>,<manager>,<format>]		-	-	-	-	-	-	-	-	-	-	-
system.swap.in[<device>,<type>]		-	X		-	-	-	-	-	X		-
(specifying a de- vice is only sup- ported under Linux)												
type	count	-	X	X	-	X	-	-	-	-	X	-
▲ (pages will only work if device was not speci- fied)												
	sectors	-	X	X	-	-	-	-	-	-	-	-
	pages	-	X	X	-	X	-	-	-	-	X	-
(de- fault under Linux)												
system.swap.out[<device>,<type>]		-	X		-	-	-	-	-	X		-
(specifying a de- vice is only sup- ported under Linux)												
type	count	-	X	X	-	X	-	-	-	-	X	-
▲ (pages will only work if device was not speci- fied)												
	sectors	-	X	X	-	-	-	-	-	-	-	-
	pages	-	X	X	-	X	-	-	-	-	X	-
(de- fault under Linux)												

system.swap.size[<device>,<type>]		X		X		-		X		X		-
(specifying a device is only supported under FreeBSD, for other platforms must be empty or "all")												
type	free	X		X		X		X		X		-
▲	(de-fault)											
	total	X		X		X		X		X		-
	used	X		X		X		X		X		-
	pfree	X		X		X		X		X		-
	pusd	-		X		X		X		X		-
system.uname		X		X		X		X		X		X
system.uptime		X		X		X		-		X		X
system.users.num				X		X		X		X		X
	1			2		3		4		5		6
vfs.dev.read[<device>,<type>,<mode>]		X		X		-		X		-		X
type	sectors	-		X		X		-		-		-
▲												
	operations			X		X		X		-		X
	(de-fault for OpenBSD, AIX)											
	bytes	-		-		-		X		-		X
	(de-fault for Solaris)											
	sps	-		X		X		-		-		-
	(de-fault for Linux)											
	ops	-		X		X		X		-		-
	bps	-		-		-		X		-		-
	(de-fault for FreeBSD)											

mode	avg1	-	X	X	X	-	-	-	-	-	-	-
▲	(de- fault)											
(compatibility)												
only												
with type												
in:												
sps,												
ops,												
bps)												
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.dev.write[<device>,<type>,<mode>]		X				X		X			X	
type	sectors	-	X	X	-	-	-	-	-	-	-	-
▲												
operations			X	X	X	X	-	X	-	-	X	-
(de- fault for OpenBSD, AIX)												
bytes		-	-	-	X	X	-	X	-	-	X	-
(de- fault for So- laris)												
sps		-	X	X	-	-	-	-	-	-	-	-
(de- fault for Linux)												
ops		-	X	X	X	-	-	-	-	-	-	-
bps		-	-	-	X	-	-	-	-	-	-	-
(de- fault for FreeBSD)												
mode	avg1	-	X	X	X	-	-	-	-	-	-	-
▲	(de- fault)											
(compatibility)												
only												
with type												
in:												
sps,												
ops,												
bps)												
	avg5	-	X	X	X	-	-	-	-	-	-	-
	avg15	-	X	X	X	-	-	-	-	-	-	-
vfs.dir.count[dir,<regex_incl>,<regex_excl>,<types_incl>,<types_excl>,<max_depth>,<min_size>,<max_size>,<min_age>]												
vfs.dir.size[dir,<regex_incl>,<regex_excl>,<mode>,<max_depth>,<regex_excl_dir>]												
vfs.file.cksum[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.contents[file,<encoding>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.exists[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.md5sum[file]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.regexp[file,regexp,<encoding>,<output>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.regmatch[file,regexp,<encoding>]		X	X	X	X	X	X	X	X	X	X	X
vfs.file.size[file]		X	X	X	X	X	X	X	X	X	X	X
	1	2	3	4	5	6	7	8	9	10	11	
vfs.file.time[file,<mode>]		X	X	X	X	X	X	X	X	X	X	X
mode	modify	X	X	X	X	X	X	X	X	X	X	X
▲	(de- fault)											
access		X	X	X	X	X	X	X	X	X	X	X

	change	X	X	X	X	X	X	X	X	X	X	X
vfs.fs.discovery	X	X	X	X	X	X	X	-	X	X	X	X
vfs.fs.inode[fs,<mode>]	X	X	X	X	X	X	X	X	X	X	X	X
mode	total	-	X	X	X	X	X	X	X	X	X	X
▲	(de-fault)											
	free	-	X	X	X	X	X	X	X	X	X	X
	used	-	X	X	X	X	X	X	X	X	X	X
	pfree	-	X	X	X	X	X	X	X	X	X	X
	pusd	-	X	X	X	X	X	X	X	X	X	X
vfs.fs.size[fs,<mode>]	X	X	X	X	X	X	X	X	X	X	X	X
mode	total	X	X	X	X	X	X	X	X	X	X	X
▲	(de-fault)											
	free	X	X	X	X	X	X	X	X	X	X	X
	used	X	X	X	X	X	X	X	X	X	X	X
	pfree	X	X	X	X	X	X	X	X	X	X	X
	pusd	X	X	X	X	X	X	X	X	X	X	X
vm.memory.size[*mode*]	X	X	X	X	X	X	X	X	X	X	X	X
mode	total	X	X	X	X	X	X	X	X	X	X	X
▲	(de-fault)											
	active	-	-	-	X	-	X	-	-	X	X	X
	anon	-	-	-	-	-	-	-	-	-	-	X
	buffers	-	X	X	X	-	-	-	-	-	X	X
	cached	X	X	X	X	-	-	X	-	-	X	X
	exec	-	-	-	-	-	-	-	-	-	-	X
	file	-	-	-	-	-	-	-	-	-	-	X
	free	X	X	X	X	X	X	X	X	X	X	X
	inactive	-	-	-	X	-	-	-	-	X	X	X
	pinned	-	-	-	-	-	-	X	-	-	-	-
	shared	-	X	-	X	-	-	-	-	-	X	X
	wired	-	-	-	X	-	-	-	-	X	X	X
	used	X	X	X	X	X	X	X	X	X	X	X
	pusd	X	X	X	X	X	X	X	X	X	X	X
	available	X	X	X	X	X	X	X	X	X	X	X
	pavailable	X	X	X	X	X	X	X	X	X	X	X
web.page.get[host,<path>,<port>]	X	X	X	X	X	X	X	X	X	X	X	X
web.page.perf[host,<path>,<port>]	X	X	X	X	X	X	X	X	X	X	X	X
web.page.regex[host,<path>,<port>,regexp,<length>,<output>]	X	X	X	X	X	X	X	X	X	X	X	X
	1	2	3	4	5	6	7	8	9	10	11	

Note:

See also a description of **vm.memory.size** parameters.

Footnotes

¹ net.if.in, net.if.out and net.if.total items do not provide statistics of loopback interfaces (e.g. lo0).

² These values for these items are not supported for loopback interfaces on Solaris systems up to and including Solaris 10 6/06 as byte, error and utilisation 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.

³ 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.

⁴ Not supported on Windows Event Log.

2 vm.memory.size parameters

Overview

This section provides more details and platform-specific information on the parameters of 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** - inactive + cached + free memory as percentage of 'total'
- **pinned** - same as 'wired'
- **pused** - active + wired memory as percentage of 'total'
- **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.

Attention:
Some of these parameters are platform-specific and might not be available on your platform. Please refer to [this page](#) 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+	/proc/meminfo, see "MemAvailable" in Linux kernel	total - free
(also backported to 3.10 on RHEL 7)	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.	
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 the ["See also"](#) section at the bottom of this page to find more detailed information about memory calculation in different OS.

Platform-specific notes

- on Linux **shared** works only on kernel 2.4

See also

1. [Detailed information about memory calculation in different OS](#)

3 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.

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).

Getting the list of items

Agent request

```
{
  "request": "active checks",
  "host": "<hostname>"
}
```

Server response

```
{
  "response": "success",
  "data": [
```

```

    {
      "key": "log[/home/zabbix/logs/zabbix_agentd.log]",
      "delay": 30,
      "lastlogsize": 0,
      "mtime": 0
    },
    {
      "key": "agent.version",
      "delay": 600,
      "lastlogsize": 0,
      "mtime": 0
    },
    {
      "key": "vfs.fs.size[/nono]",
      "delay": 600,
      "lastlogsize": 0,
      "mtime": 0
    }
  ]
}

```

The server must respond with success. For each returned item, all properties **key**, **delay**, **lastlogsize** and **mtime** must exist, regardless of whether item is a log item or not.

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

```

{
  "request": "agent data",
  "session": "12345678901234567890123456789012",
  "data": [
    {
      "host": "<hostname>",
      "key": "agent.version",
      "value": "2.4.0",
      "id": 1,
      "clock": 1400675595,
      "ns": 76808644
    },
    {
      "host": "<hostname>",
      "key": "log[/home/zabbix/logs/zabbix_agentd.log]",
      "lastlogsize": 112,
      "value": " 19845:20140621:141708.521 Starting Zabbix Agent [<hostname>]. Zabbix 2.4.0 (revision",
      "id": 2,
      "clock": 1400675595,
      "ns": 77053975
    }
  ]
}

```

```

        "host": "<hostname>",
        "key": "vfs.fs.size[/nono]",
        "state": 1,
        "value": "Cannot obtain filesystem information: [2] No such file or directory",
        "id": 3,
        "clock": 1400675595,
        "ns": 78154128
    }
],
"clock": 1400675595,
"ns": 78211329
}

```

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

```

{
  "response": "success",
  "info": "processed: 3; failed: 0; total: 3; seconds spent: 0.003534"
}

```

Attention:

If sending of some values fails on the server (for example, because host or item has been disabled or deleted), agent will not retry sending of those values.

For example:

1. Agent opens a TCP connection
2. Agent sends a list of values
3. Server processes the data and sends the status back
4. TCP connection is closed

Note how in the example above the not supported status for `vfs.fs.size[/nono]` is indicated by the "state" value of 1 and the error message in "value" property.

Attention:

Error message will be trimmed to 2048 symbols on server side.

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.

4 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"
}
```

Alternatively Zabbix sender can send request with a timestamp

```
{
  "request": "sender data",
  "data": [
    {
      "host": "<hostname>",
      "key": "trap",
      "value": "test value",
      "clock": 1516710794
    },
    {
      "host": "<hostname>",
      "key": "trap",
      "value": "test value",
      "clock": 1516710795
    }
  ],
  "clock": 1516712029,
  "ns": 873386094
}
```

Zabbix server response

```
{
  "response": "success",
  "info": "processed: 2; failed: 0; total: 2; seconds spent: 0.060904"
}
```

5 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.

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.version	Guests	Guests
log	Administrators	Guests
log.count	Administrators	Guests

Item key	User group	
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.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
service.discovery	Guests	Guests
service.info	Guests	Guests
services	Guests	Guests
wmi.get	Administrators	Guests
vm.vmemory.size	Guests	Guests

6 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 back-ends 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

7 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.

8 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[sm5c47b397-isa-0880,in,avg] or sensor[sm5c47b397.2176,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.

⁵ According to [specification](#) these are voltages on chip pins and generally speaking may need scaling.

9 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 **vsiz**. 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 proccentry64 structure	Tries to be compatible with
vsiz ((- default value))	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
dsiz	Data size	pi_dsiz	"ps gvw" TSIZ column
tsiz	Text (code) size	pi_tsiz	
sdsiz	Data size from shared library	pi_sdsiz	
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
vsiz	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 ((- default value))	Size of process (code + data + stack)	tsiz + dsiz + ssiz	
tsiz	Text (code) size	kp_eproc.e_vm.vm_tsiz or ki_tsiz	ps -o tsiz
dsiz	Data size	kp_eproc.e_vm.vm_dsiz or ki_dsiz	ps -o dsiz
ssiz	Stack size	kp_eproc.e_vm.vm_ssiz or ki_ssiz	ps -o ssiz

Linux

See values supported for 'memtype' parameter on Linux in the table.

Supported value	Description	Source in /proc/<pid>/status file
vsiz ((- default value))	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

Supported value	Description	Source in /proc/<pid>/status file
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 ((- default value))	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

10 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

[illegible]

/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.

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.mem[]` and `proc.num[]` items for monitoring programs which modify their commandlines.

Before putting name and cmdline parameters into `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.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.

11 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 multi-line 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\r\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.

12 Item value preprocessing details

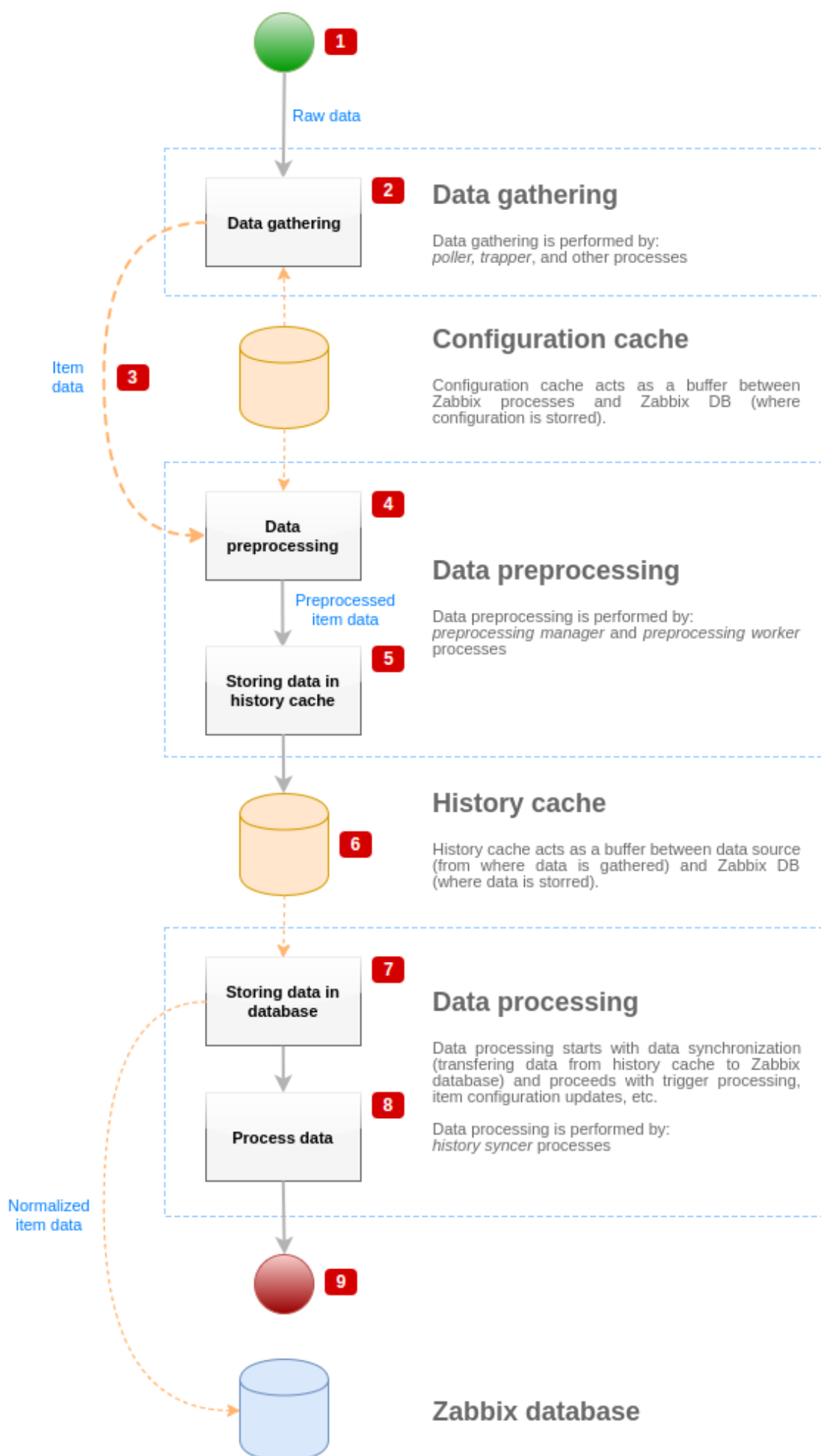
Overview

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:



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 pre-defined 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.

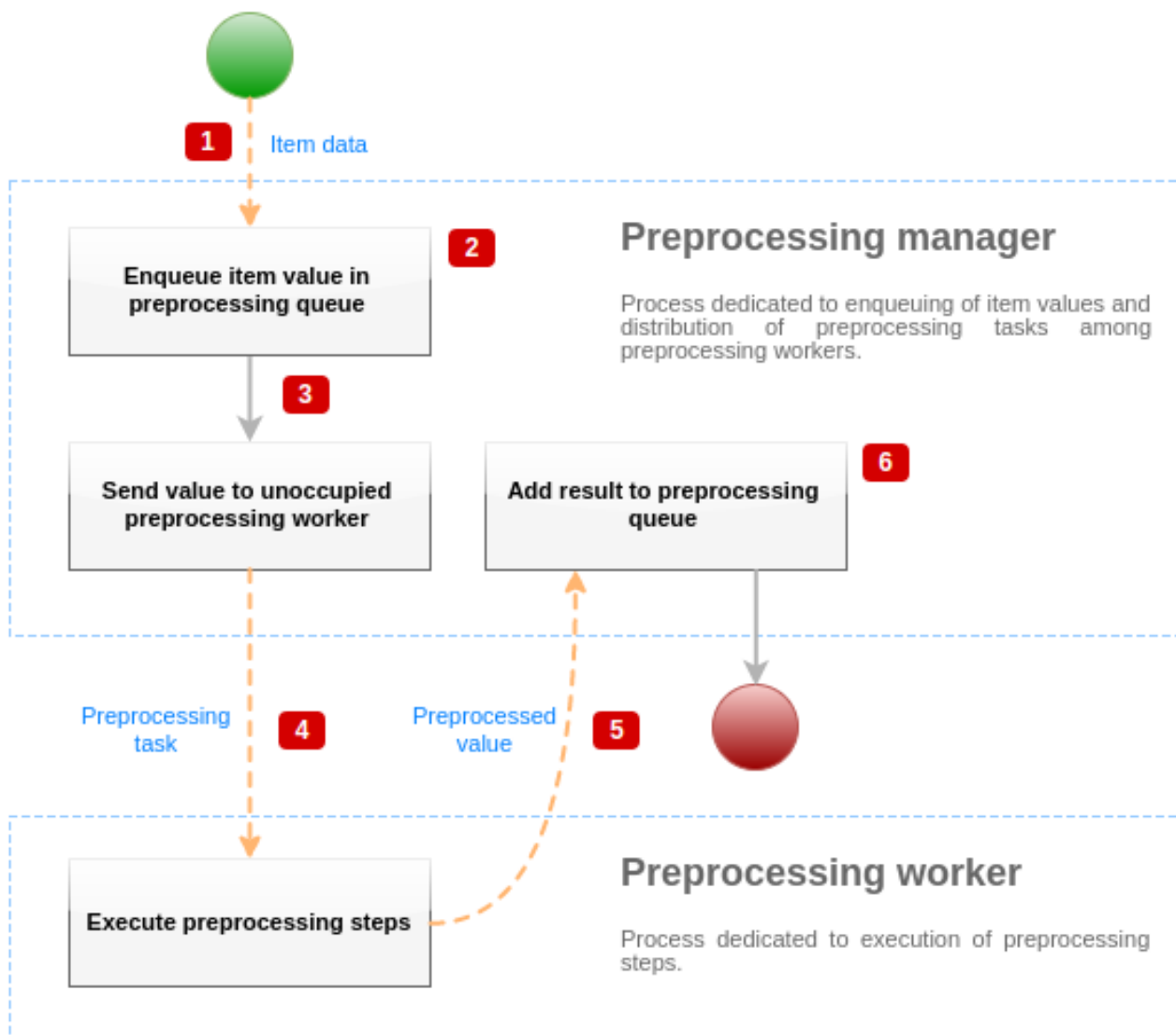
Note:

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
- * When both master and dependent items reach history synchronization phase, master item becomes NOT SUP

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

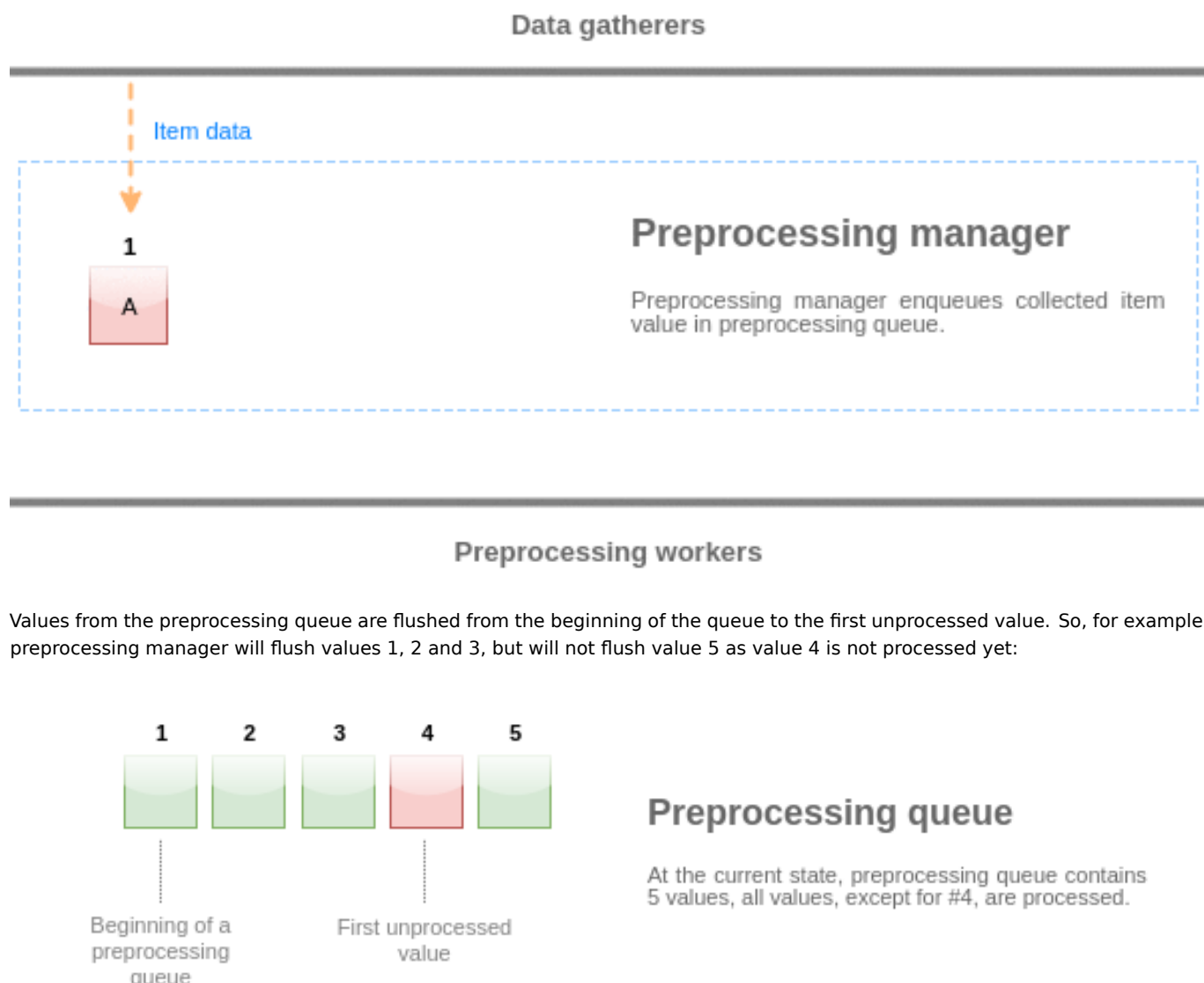
- Dependent item receives value that is not present in master item history. The use case is very similar to the previous one, except for the master item type. For example, if `CHAR` type is used for master item, then master item value will be truncated at the history synchronization phase, while dependent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



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, 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.

13 Supported JSONPath functionality

Attention:

The full JSONPath functionality as described in this page is available starting with Zabbix 4.2.5.

Overview

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.

Supported segments

Segment	Description
<name>	Match object property by name.
*	Match all object properties.
['<name>']	Match object property by name.
['<name>', '<name>', ...]	Match object property by any of the listed names.
[<index>]	Match array element by the index.
[<number>, <number>, ...]	Match array element by any of the listed indexes.
[*]	Match all object properties or array elements.
[<start>:<end>]	Match array elements by the defined range: <start> - the first index to match (including). If not specified matches all array elements from the beginning. If negative specifies starting offset from the end of array. <end> - the last index to match (excluding). If not specified matches all array elements to the end. If negative specifies starting offset from the end of array.
[?(<expression>)]	Match objects/array elements by applying 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.

Filter expression

Filter expression is a 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.
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 since Zabbix 4.2.8. 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": "Checmical cleaning",
          "price": 46
        },
        {
          "description": "Pressing pages damaged by moisture",
          "price": 24.5
        },
        {
          "description": "Rebinding torn book",
          "price": 99.49
        }
      ]
    }
  },
  "filters": {
    "price": 10,
    "category": "fiction",
    "no filters": "no \"filters\""
  },
  "closed message": "Store is closed",
  "tags": [
    "a",
    "b",
    "c",
    "d",
    "e"
  ]
}

```

JSONPath	Type	Result	Comments
\$.filters.price	definite	10	
\$.filters.category	definite	fiction	
\$.filters['no filters']	definite	no "filters"	
\$.filters	definite	{ "price": 10, "category": "fiction", "no filters": "no \"filters\"" }	
\$.books[1].title	definite	Sword of Honour	
\$.books[-1].author	definite	J. R. R. Tolkien	
\$.books.length()	definite	4	
\$.tags[:]	indefinite	["a", "b", "c", "d", "e"]	
\$.tags[2:]	indefinite	["c", "d", "e"]	
\$.tags[:3]	indefinite	["a", "b", "c"]	
\$.tags[1:4]	indefinite	["b", "c", "d"]	
\$.tags[-2:]	indefinite	["d", "e"]	
\$.tags[: -3]	indefinite	["a", "b"]	
\$.tags[: -3].length()	definite	2	
\$.books[0, 2].title	indefinite	["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]	

JSONPath	Type	Result	Comments
<code>\$.books[?(@.id == 4 - 0.4 * 5)].title</code>	indefinite	["Sword of Honour"]	This query shows that arithmetical operations can be used in queries. Of course this query can be simplified to <code>\$.books[?(@.id == 2)].title</code>
<code>\$.books[?(@.id == 2 @.id == 4)].title</code>	indefinite	["Sword of Honour", "The Lord of the Rings"]	
<code>\$.books[?!(@.id == 2)].title</code>	indefinite	["Sayings of the Century", "Moby Dick", "The Lord of the Rings"]	
<code>\$.books[?(@.id != 2)].title</code>	indefinite	["Sayings of the Century", "Moby Dick", "The Lord of the Rings"]	
<code>\$.books[?(@.title =~ " of ")] .title</code>	indefinite	["Sayings of the Century", "Sword of Honour", "The Lord of the Rings"]	
<code>\$.books[?(@.price > 12.99)].title</code>	indefinite	["The Lord of the Rings"]	
<code>\$.books[?(@.author > "Herman Melville")].title</code>	indefinite	["Sayings of the Century", "The Lord of the Rings"]	
<code>\$.books[?(@.price > \$.filters.price)].title</code>	indefinite	["Sword of Honour", "The Lord of the Rings"]	
<code>\$.books[?(@.category == \$.filters.category)].title</code>	indefinite	["Sword of Honour", "Moby Dick", "The Lord of the Rings"]	

JSONPath	Type	Result	Comments
<code>\$..[?(@.id)]</code>	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 }]	
<code>\$.services..[?(@.price > 50)].description</code>	indefinite	['Printing and assembling book in A5 format', 'Rebinding torn book']	
<code>\$..id.length()</code>	definite	4	
<code>\$.books[?(@.id == 2)].title.first()</code>	definite	Sword of Honour	
<code>\$..tags.first().length()</code>	definite	5	<code>\$..tags</code> is indefinite path, so it returns an array of matched elements - ["a", "b", "c", "d", "e"], <code>first()</code> returns the first element - ["a", "b", "c", "d", "e"] and finally <code>length()</code> calculates its length - 5.
<code>\$.books[*].price.min()</code>	definite	8.95	
<code>\$.price.max()</code>	definite	154.99	
<code>\$.books[?(@.category == "fiction")].price.avg()</code>	definite	14.99	
<code>\$.books[?(@.category == \$.filters.xyz)].title</code>	indefinite		A query without match returns NULL for definite and indefinite paths.
<code>\$.services[?(@.active=="true")].servicegroup [1000,1001]</code>	indefinite		Text constants must be used in boolean value comparisons.

JSONPath	Type	Result	Comments
<code>\$.services[?(@.active=="false")]</code>	Array	<code>["Discard servicegroup1002"]</code>	Text constants must be used in boolean value comparisons.

14 Preprocessing by Javascript details

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. Also a 10 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.

15 Unreachable/unavailable host settings

Overview

Several configuration **parameters** define how Zabbix server should behave when an agent check (Zabbix, SNMP, IPMI, JMX) fails and a host becomes unreachable.

Unreachable host

A host 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 host availability in any way.

From that moment **UnreachableDelay** defines how often a host 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 a host 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 a host is rechecked more than once before a host becomes unavailable.

If the unreachable host reappears, the monitoring returns to normal automatically:

```
resuming Zabbix agent checks on host "New host": connection restored
```

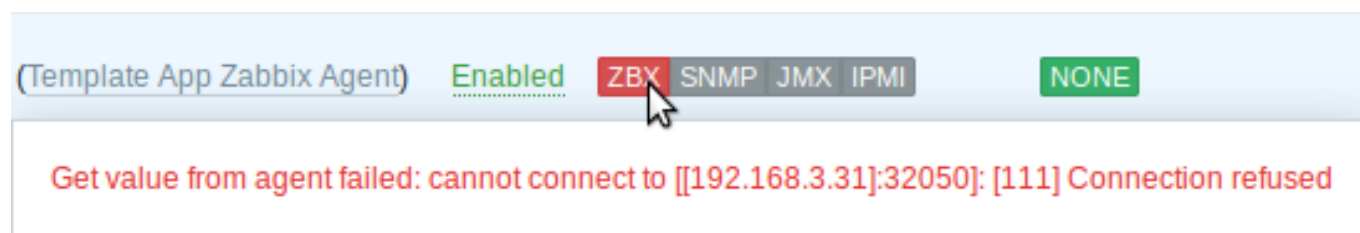
Unavailable host

After the UnreachablePeriod ends and the host has not reappeared, the host is treated as unavailable.

In the server log it is indicated by messages like these:

```
temporarily disabling Zabbix agent checks on host "New host": host unavailable
```

and in the **frontend** the host availability icon for the respective interface goes from green (or gray) to red (note that on mouseover a tooltip with the error description is displayed):



The **UnavailableDelay** parameter defines how often a host is checked during host 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 host is restored, the monitoring returns to normal automatically, too:

```
enabling Zabbix agent checks on host "New host": host became available
```

16 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.

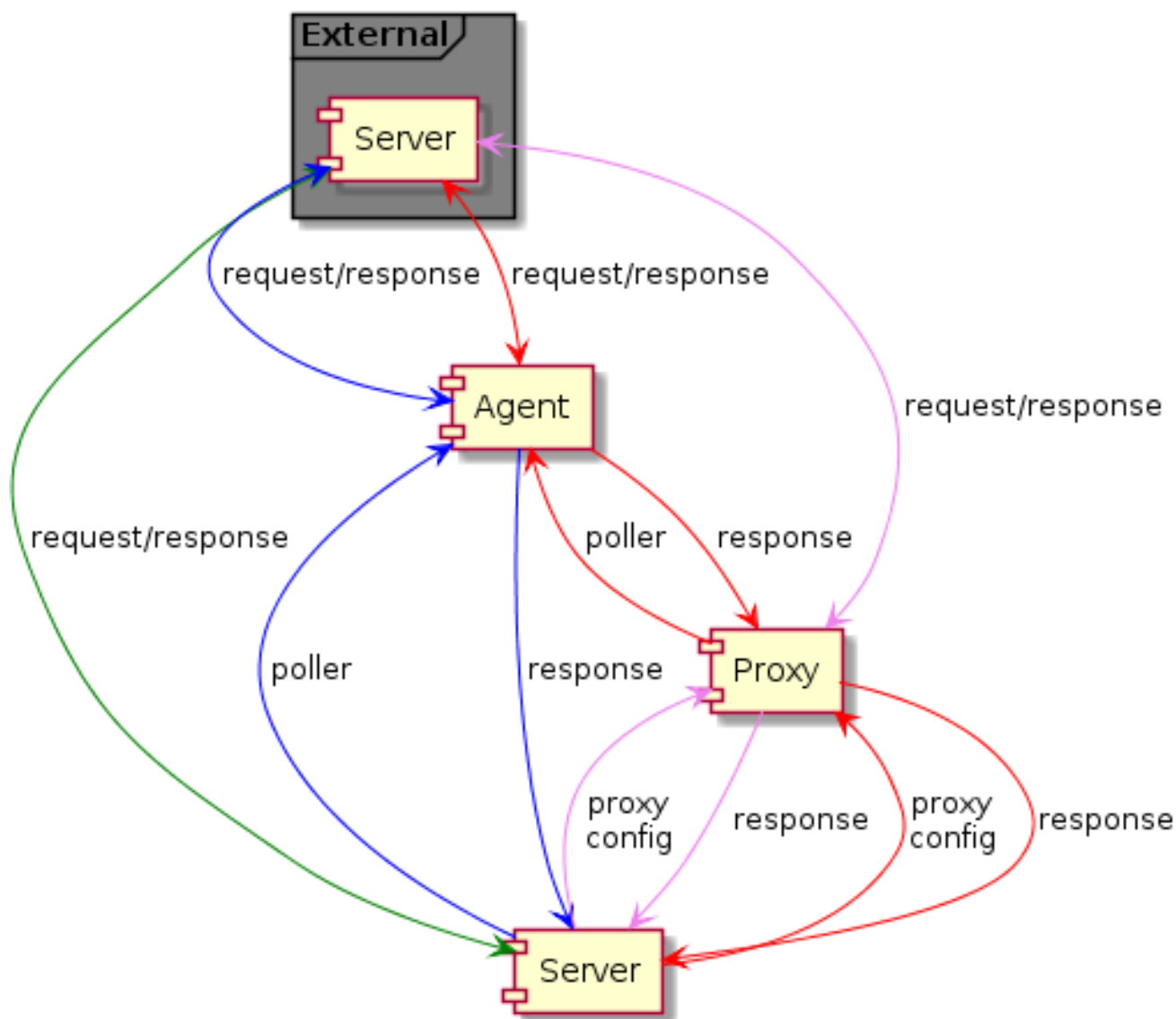
Items

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[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:

- Template App Remote Zabbix server
- Template App 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,
    "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
  },
  "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](#)

6 Triggers

1 Supported trigger functions

All functions supported in [trigger expressions](#) are listed here.

	Description	Parameters	Comments
abschange	The amount of absolute difference between last and previous values.		Supported value types: float, int, str, text, log For example: (previous value;last value=abschange) 1;5=4 3;1=2 0;- 2.5=2.5 For strings returns: 0 - values are equal 1 - values differ
avg (sec #num,<time_shift>)			

Average value of an item within the defined evaluation period.	sec or #num - maximum evaluation period ¹ in seconds or in latest collected values (preceded by a hash mark) time_shift (optional) - evaluation point is moved the number of seconds back in time	Supported value types: float, int Examples: => avg(#5) → average value for the five latest values => avg(1h) → average value for an hour => avg(1h,1d) → average value for an hour one day ago. The <code>time_shift</code> parameter is supported since Zabbix 1.8.2. It is useful when there is a need to compare the current average value with the average value <code>time_shift</code> seconds back.

band (<sec|#num>,mask,<time_shift>)

Value of "bitwise AND" of an item value and mask.	sec (ignored, equals #1) or #num (optional) - the Nth most recent value mask (manda- tory) - 64-bit unsigned integer (0 - 18446744073709551615) time_shift (optional) - see avg()	Supported value types: int Take note that #num works differently here than with many other functions (see last()). Although the com- parison 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 compar- ing to 4, not 100. Examples: => band(,12)=8 or band(,12)=4 → 3rd or 4th bit set, but not both at the same time => band(,20)=16 → 3rd bit not set and 5th bit set. This function is supported since Zabbix 2.2.0.

change

The amount of difference between last and previous values.

Supported value types: float, int, str, text, log

For example:
(previous value;last value=change)
1;5=+4
3;1=-2
0;-2.5=-2.5

See also:
[ab-schange](#)
for comparison

For strings returns:
0 - values are equal
1 - values differ

count (sec|#num,<pattern>,<operator>,<time_shift>)

Number of values within the defined evaluation period.	sec or #num - maximum evaluation period ¹ in seconds or in latest collected values (preceded by a hash mark) pattern (optional) - required pattern operator (optional)	Supported value types: float, integer, string, text, log Float items match with the precision of 0.000001. With <i>band</i> as third parameter, the second pattern parameter can be specified as two numbers, separated by '/': number_to_compare_with count() calculates "bitwise AND" from the value and the <i>mask</i> and compares the result to <i>number_to_compare_with</i> . If the result of "bitwise AND" is equal to <i>number_to_compare_with</i> , the value is counted. If <i>number_to_compare_with</i> and <i>mask</i> are equal, only the <i>mask</i> need be specified (without '/').
	Supported operators: <i>eq</i> - equal <i>ne</i> - not equal <i>gt</i> - greater <i>ge</i> - greater or equal <i>lt</i> - less <i>le</i> - less or equal <i>like</i> - matches if contains pattern (case-sensitive) <i>band</i> - bitwise AND <i>regex</i> - case sensitive match of regular expression given in pattern <i>iregexp</i> - case insensitive match of regular expression given in pattern	Note that: <i>eq</i>

FUNCTION			
date		Current date in YYYY-MM-DD format.	Supported value types: <i>any</i> Example of returned value: 20150731
dayofmonth		Day of month in range of 1 to 31.	Supported value types: <i>any</i> This function is supported since Zabbix 1.8.5.
dayofweek		Day of week in range of 1 to 7 (Mon - 1, Sun - 7).	Supported value types: <i>any</i>
delta (sec #num,<time_shift>)		Difference between the maximum and minimum values within the defined evaluation period ('max()' minus 'min()').	sec or #num - maximum evaluation period ¹ in seconds or in latest collected values specified (preceded by a hash mark) time_shift (optional) - see avg()
diff			Supported value types: float, int The time_shift parameter is supported since Zabbix 1.8.2.

FUNCTION		
	Checking if last and previous values differ.	Supported value types: float, int, str, text, log Returns: 1 - last and previous values differ 0 - otherwise
forecast (sec #num,<time_shift>,time,<fit>,<mode>)		

Future value, max, min, delta or avg of the item.	sec or #num - maximum evaluation period ¹ in seconds or in latest collected values specified (preceded by a hash mark) time_shift (optional) - see avg() time - forecasting horizon in seconds fit (optional) - function used to fit historical data Supported fits: <i>linear</i> - linear function <i>polynomialN</i> - polynomial of degree N (1 ≤ N ≤ 6) <i>exponential</i> - exponential function <i>logarithmic</i> - logarithmic function <i>power</i> - power function Note that: <i>linear</i> is default, <i>polynomial1</i> is equivalent to <i>linear</i> mode (optional) - de-	Supported value types: float, int If value to return is larger than 999999999999.9999 or less than -999999999999.9999, return value is cropped to 999999999999.9999 or -999999999999.9999 correspondingly. Becomes not supported only if misused in expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors. Examples: => forecast(#10,,1h) → forecast of item value after one hour based on last 10 values => forecast(1h,,30m) → forecast of item value after 30 minutes based on last hour data => forecast(1h,1d,12h) → forecast

FUNCTION

fuzzytime (sec)

FUNCTION

Checking how much an item value (as times- tamp) differs from the Zabbix server time.	sec - seconds	Supported value types: float, int Returns: 1 - difference between item value (as times- tamp) and Zabbix server times- tamp is less than or equal to T seconds 0 - otherwise Usually used with the 'sys- tem.localtime' item to check that local time is in sync with the local time of Zabbix server. <i>Note that 'sys- tem.localtime' must be config- ured as a passive check.</i> Can be used also with vfs.file.time[/path/file,m key to check that file didn't get updates for long time. Example: => fuzzy- time(60)=0 → detect a problem if time difference is over 60

FUNCTION			
iregexp (<pattern>,<sec #num>)	This function is a non case-sensitive analogue of regexp().	see regexp()	Supported value types: str, log, text
last (<sec #num>,<time_shift>)			

The most recent value.	sec (ignored, equals #1) or #num (optional) - the Nth most recent value time_shift (optional) - see avg()	Supported value types: float, int, str, text, log Take note that #num works differently here than with many other functions. For example: last() is always equal to last(#1) last(#3) - third most recent value (<i>not</i> three latest values) Zabbix does not guarantee exact order of values if more than two values exist within one second in history. The #num parameter is supported since Zabbix 1.6.2. The time_shift parameter is supported since Zabbix 1.8.2.

logeventid (<pattern>)

FUNCTION

	Checking if event ID of the last log entry matches a regular expression.	pattern (optional) - regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style.	Supported value types: log Returns: 0 - does not match 1 - matches This function is supported since Zabbix 1.8.5.
logseverity	Log severity of the last log entry.		Supported value types: log Returns: 0 - default severity N - severity (integer, useful for Windows event logs: 1 - Information, 2 - Warning, 4 - Error, 7 - Failure Audit, 8 - Success Audit, 9 - Critical, 10 - Verbose). Zabbix takes log severity from Information field of Windows event log.
logsource (<pattern>)			

	Checking if log source of the last log entry matches a regular expression.	pattern (optional) - regular expression describing the required pattern, Perl Compatible Regular Expression (PCRE) style.	Supported value types: log Returns: 0 - does not match 1 - matches Normally used for Windows event logs. For example, log-source("VMware Server").
max (sec #num,<time_shift>)	Highest value of an item within the defined evaluation period.	sec or #num - maximum evaluation period ¹ in seconds or in latest collected values (preceded by a hash mark) time_shift (optional) - see avg()	Supported value types: float, int The time_shift parameter is supported since Zabbix 1.8.2.
min (sec #num,<time_shift>)	Lowest value of an item within the defined evaluation period.	sec or #num - maximum evaluation period ¹ in seconds or in latest collected values (preceded by a hash mark) time_shift (optional) - see avg()	Supported value types: float, int The time_shift parameter is supported since Zabbix 1.8.2.
nodata (sec)			

FUNCTION

Checking for no data received.	sec - eval- uation period in seconds. The period should not be less than 30 seconds because the history syncer process calculates this function only every 30 seconds. nodata(0) is disal- lowed.	Supported value types: <i>any</i> Returns: 1 - if no data received during the defined period of time 0 - otherwise Note that this function will display an error if, within the period of the 1st pa- rameter: - there's no data and Zabbix server was restarted - there's no data and main- tenance was com- pleted - there's no data and the item was added or re- enabled Errors are displayed in the <i>Info</i> column in trigger <i>configura-</i> <i>tion</i> .

now

FUNCTION

	Number of seconds since the Epoch (00:00:00 UTC, January 1, 1970).	Supported value types: <i>any</i>
percentile (sec #num,<time_shift>,percentage)	P-th percentile of a period, where P (percentage) is specified by the third parameter. sec or #num - maximum evaluation period¹ in seconds or in latest collected values (preceded by a hash mark) time_shift (optional) - see avg() percentage - a floating-point number between 0 and 100 (inclusive) with up to 4 digits after the decimal point	Supported value types: float, int This function is supported since Zabbix 3.0.0.
prev	Previous value.	Supported value types: float, int, str, text, log Returns the same as last(#2).
regexp (<pattern>,<sec #num>)		

	Checking if the latest (most recent) value matches regular expres- sion.	pattern (optional) - regular expres- sion, Perl Compatible Regular Expression (PCRE) style. sec or #num (optional) - maximum evalua- tion period ¹ in seconds or in latest collected values (preceded by a hash mark). In this case, more than one value may be pro- cessed.	Supported value types: str, text, log Returns: 1 - found 0 - otherwise If more than one value is pro- cessed, '1' is returned if there is at least one matching value. This function is case- sensitive.
str (<pattern>,<sec #num>)	Finding a string in the latest (most recent) value.	pattern (optional) - required string sec or #num (optional) - maximum evalua- tion period ¹ in seconds or in latest collected values (preceded by a hash mark). In this case, more than one value may be pro- cessed.	Supported value types: str, text, log Returns: 1 - found 0 - otherwise If more than one value is pro- cessed, '1' is returned if there is at least one matching value. This function is case- sensitive.

FUNCTION

strlen (<sec|#num>,<time_shift>)

Length of the latest (most recent) value in characters (not bytes).	sec (ignored, equals #1) or #num (optional) - the Nth most recent value time_shift (optional) - see avg()	Supported value types: str, text, log Take note that #num works differently here than with many other functions.

Examples:

=>
strlen()(is equal to strlen(#1))
→ length of the latest value
=>
strlen(#3)
→ length of the third most recent value
=>
strlen(,1d)
→ length of the most recent value one day ago.

This function is supported since Zabbix 1.8.4.

sum (sec|#num,<time_shift>)

FUNCTION

	Sum of collected values within the defined evaluation period.	sec or #num - maximum evaluation period ¹ in seconds or in latest collected values (preceded by a hash mark) time_shift (optional) - see avg()	Supported value types: float, int The time_shift parameter is supported since Zabbix 1.8.2.
time	Current time in HHMMSS format.		Supported value types: <i>any</i> Example of returned value: 123055
timeleft	(sec #num,<time_shift>,threshold,<fit>)		

Time in seconds needed for an item to reach a specified threshold.	sec or #num - maximum evaluation period ¹ in seconds or in latest collected values (preceded by a hash mark) time_shift (optional) - see avg() threshold - value to reach fit (optional) - see forecast()	Supported value types: float, int If value to return is larger than 999999999999.9999, return value is cropped to 999999999999.9999. Returns 999999999999.9999 if threshold cannot be reached. Becomes not supported only if misused in expression (wrong item type, invalid parameters), otherwise returns -1 in case of errors. Examples: => timeleft(#10,,0) → time until item value reaches zero based on last 10 values => timeleft(1h,,100) → time until item value reaches 100 based on last hour data => timeleft(1h,1d,0) → time until item value

Warning:

Important notes:

- 1) All functions return numeric values only. Comparison to strings is not supported.
- 2) Some of the functions cannot be used for non-numeric values!
- 3) String arguments should be double quoted. Otherwise, they might get misinterpreted.
- 4) For all trigger functions **sec** and **time_shift** must be an integer with an optional **time unit suffix** and has absolutely nothing to do with the item's data type.

Footnotes

- ¹ The function is evaluated starting with the first received value (unless the `timeshift` parameter is used).

Functions and unsupported items

Since Zabbix 3.2, **nodata()**, **date()**, **dayofmonth()**, **dayofweek()**, **now()** and **time()** functions are calculated for unsupported items, too. Other functions require that the referenced item is in a supported state.

7 Macros**1 Supported macros**

Overview

The table contains a complete list of macros supported by Zabbix.

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 → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	<i>Numeric ID of the triggered action.</i> Supported since 2.2.0.
{ACTION.NAME}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	<i>Name of the triggered action.</i> Supported since 2.2.0.
{ALERT.MESSAGE}	→ Alert script parameters	<i>'Default message' value from action configuration.</i> Supported since 3.0.0.
{ALERT.SENDTO}	→ Alert script parameters	<i>'Send to' value from user media configuration.</i> Supported since 3.0.0.
{ALERT.SUBJECT}	→ Alert script parameters	<i>'Default subject' value from action configuration.</i> Supported since 3.0.0.
{DATE}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	<i>Current date in yyyy.mm.dd. format.</i>
{DISCOVERY.DEVICE.IPADDRESS}	→ Discovery notifications	<i>IP address of the discovered device.</i> Available always, does not depend on host being added.

Macro	Supported in	Description
{DISCOVERY.DEVICE.DNS}	Discovery notifications	DNS name of the discovered device. Available always, does not depend on host being added.
{DISCOVERY.DEVICE.STATUS}	Discovery notifications	Status of the discovered device: can be either UP or DOWN.
{DISCOVERY.DEVICE.UPTIME}	Discovery notifications	Time since the last change of discovery status for a particular device. For example: 1h 29m. For devices with status DOWN, this is the period of their downtime.
{DISCOVERY.RULE.NAME}	Discovery notifications	Name of the discovery rule that discovered the presence or absence of the device or service.
{DISCOVERY.SERVICE.NAME}	Discovery notifications	Name of the service that was discovered. For example: HTTP.
{DISCOVERY.SERVICE.PORT}	Discovery notifications	Port of the service that was discovered. For example: 80.
{DISCOVERY.SERVICE.STATUS}	Discovery notifications	Status of the discovered <i>service</i> : can be either UP or DOWN. {DISCOVERY.SERVICE.UPTIME} → Discovery notifications Time since the last change of discovery status for a particular service. For example: 1h 29m. For services with status DOWN, this is the period of their downtime. {ESC.HISTORY} → Trigger-based notifications and commands → Internal notifications → Problem update 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	Acknowledgement status of the event (Yes/No).
{EVENT.AGE}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	Age of the event that triggered an action. Useful in escalated messages.
{EVENT.DATE}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	Date of the event that triggered an action.
{EVENT.ID}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	Numeric ID of the event that triggered an action.
{EVENT.NAME}	→ Trigger-based notifications and commands → Problem update notifications	Name of the problem event that triggered an action. Supported since 4.0.0.
{EVENT.NSEVERITY}	→ Trigger-based notifications and commands → Problem update notifications	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.RECOVERY.DATE}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications	Date of the recovery event. Can be used in <i>recovery</i> messages only. Supported since 2.2.0.
{EVENT.RECOVERY.ID}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications	Numeric ID of the recovery event. Can be used in <i>recovery</i> messages only. Supported since 2.2.0.

Macro	Supported in	Description
{EVENT.RECOVERY.NAME}	→ Trigger-based notifications and commands → Problem update notifications	<i>Name of the recovery event.</i> Supported since 4.2.8.
{EVENT.RECOVERY.STATUS}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications	<i>Verbal value of the recovery event.</i> Can be used in recovery messages only. Supported since 2.2.0.
{EVENT.RECOVERY.TAGS}	→ Trigger-based notifications and commands → Problem update notifications	<i>A comma separated list of recovery event tags.</i> Expanded to an empty string if no tags exist. Supported since 3.2.0.
{EVENT.RECOVERY.TIME}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications	<i>Time of the recovery event.</i> Can be used in recovery messages only. Supported since 2.2.0.
{EVENT.RECOVERY.VALUE}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications	<i>Numeric value of the recovery event.</i> Can be used in recovery messages only. Supported since 2.2.0.
{EVENT.SEVERITY}	→ Trigger-based notifications and commands → Problem update notifications	<i>Name of the event severity.</i> Supported since 4.0.0.
{EVENT.STATUS}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	<i>Verbal value of the event that triggered an action.</i> Supported since 2.2.0.
{EVENT.TAGS}	→ Trigger-based notifications and commands → Problem update notifications	<i>A comma separated list of event tags.</i> Expanded to an empty string if no tags exist. Supported since 3.2.0.
{EVENT.TIME}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	<i>Time of the event that triggered an action.</i>
{EVENT.UPDATE.ACTION}	→ Problem update notifications	<i>Human-readable name of the action(s) performed during problem update.</i> Resolves to the following values: <i>acknowledged, commented, changed severity from (original severity) to (updated severity) and closed</i> (depending on how many actions are performed in one update). Supported since 4.0.0.
{EVENT.UPDATE.DATE}	→ Problem update notifications	<i>Date of problem update (acknowledgement, etc).</i> Deprecated name: {ACK.DATE}
{EVENT.UPDATE.HISTORY}	→ Trigger-based notifications and commands → Problem update notifications	<i>Log of problem updates (acknowledgements, etc).</i> Deprecated name: {EVENT.ACK.HISTORY}
{EVENT.UPDATE.MESSAGE}	→ Problem update notifications	<i>Problem update message.</i> Deprecated name: {ACK.MESSAGE}
{EVENT.UPDATE.TIME}	→ Problem update notifications	<i>Time of problem update (acknowledgement, etc).</i> Deprecated name: {ACK.TIME}
{EVENT.VALUE}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	<i>Numeric value of the event that triggered an action (1 for problem, 0 for recovering).</i> Supported since 2.2.0.

Macro	Supported in	Description
{HOST.CONN<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Global scripts (including confirmation text) → Map element labels, map URL names and values → Item key parameters¹ → Host interface IP/DNS → Trapper item "Allowed hosts" field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring⁴ → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget/screen element → Trigger names and descriptions → Trigger URLs → Tag 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.	<i>Host IP address or DNS name, depending on host settings².</i> Supported in trigger names since 2.0.0.
{HOST.DESCRPTION<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Map element labels	<i>Host description.</i> Supported since 2.4.0.
{HOST.DNS<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Global scripts (including confirmation text) → Map element labels, map URL names and values → Item key parameters¹ → Host interface IP/DNS → Trapper item "Allowed hosts" field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring⁴ → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget/screen element → Trigger names and descriptions → Trigger URLs → Tag 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.	<i>Host DNS name².</i> Supported in trigger names since 2.0.0.

Macro	Supported in	Description
{HOST.HOST<1-9>}	→ Trigger-based notifications and commands → Auto registration notifications → Internal notifications → Problem update notifications → Global scripts (including confirmation text) → 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 monitoring⁴ → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget/screen element → Trigger names and descriptions → Trigger URLs → Tag 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.	<i>Host name.</i> {HOSTNAME<1-9>} is deprecated.
{HOST.ID<1-9>}	→ Map element labels, map URL names and values → URL field of dynamic URL dashboard widget/screen element → Trigger URLs → Tag names and values	<i>Host ID.</i>
{HOST.IP<1-9>}	→ Trigger-based notifications and commands → Auto registration notifications → Internal notifications → Problem update notifications → Global scripts (including confirmation text) → Map element labels, map URL names and values → Item key parameters¹ → Host interface IP/DNS → Trapper item "Allowed hosts" field → Database monitoring additional parameters → SSH and Telnet scripts → JMX item endpoint field → Web monitoring⁴ → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget/screen element → Trigger names and descriptions → Trigger URLs → Tag 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.	<i>Host IP address².</i> Supported since 2.0.0. {IPADDRESS<1-9>} is deprecated.
{HOST.METADATA}	→ Auto registration notifications	<i>Host metadata.</i> Used only for active agent auto-registration. Supported since 2.2.0.

Macro	Supported in	Description
{HOST.NAME<1-9>}	→ Trigger-based notifications and commands → Problem update notifications → Internal notifications → Global scripts (including confirmation text) → 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 monitoring⁴ → Low-level discovery rule filter regular expressions → URL field of dynamic URL dashboard widget/screen element → Trigger names and descriptions → Trigger URLs → Tag 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.	<i>Visible host name.</i> Supported since 2.0.0.
{HOST.PORT<1-9>}	→ Trigger-based notifications and commands → Auto registration notifications → Internal notifications → Problem update notifications → Trigger names and descriptions → Trigger URLs → JMX item endpoint field → Tag names and values	<i>Host (agent) port².</i> Supported in auto-registration since 2.0.0. Supported in trigger names, trigger descriptions, internal and trigger-based notifications since 2.2.2.
{HOSTGROUP.ID}	→ Map element labels, map URL names and values	<i>Host group ID.</i>
{INVENTORY.ALIAS<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	<i>Alias field in host inventory.</i>
{INVENTORY.ASSET.TAG<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	<i>Asset tag field in host inventory.</i>
{INVENTORY.CHASSIS<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	<i>Chassis field in host inventory.</i>
{INVENTORY.CONTACT<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	<i>Contact field in host inventory.</i> {PROFILE.CONTACT<1-9>} is deprecated.
{INVENTORY.CONTRACT.NUMBER<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	<i>Contract number field in host inventory.</i>
{INVENTORY.DEPLOYMENT.STATUS<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	<i>Deployment status field in host inventory.</i>

Macro	Supported in	Description
<code>{INVENTORY.HARDWARE<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Hardware field in host inventory. {PROFILE.HARDWARE<1-9>} is deprecated.
<code>{INVENTORY.HARDWARE-FULL<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Hardware (Full details) field in host inventory.
<code>{INVENTORY.HOST.NETMASK<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Host subnet mask field in host inventory.
<code>{INVENTORY.HOST.NETWORKS<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Host networks field in host inventory.
<code>{INVENTORY.HOST.ROUTER<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Host router field in host inventory.
<code>{INVENTORY.HW.ARCH<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Hardware architecture field in host inventory.
<code>{INVENTORY.HW.DATE.DECOMMISSIONED<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Date hardware decommissioned field in host inventory.
<code>{INVENTORY.HW.DATE.EXPIRES<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Date hardware maintenance expires field in host inventory.
<code>{INVENTORY.HW.DATE.INSTALLED<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Date hardware installed field in host inventory.
<code>{INVENTORY.HW.DATE.PURCHASED<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Date hardware purchased field in host inventory.
<code>{INVENTORY.INSTALLER.NAME<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Installer name field in host inventory.
<code>{INVENTORY.LOCATION<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Location field in host inventory. {PROFILE.LOCATION<1-9>} is deprecated.

Macro	Supported in	Description
{INVENTORY.LOCATION.LATITUDE<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Location latitude field in host inventory.
{INVENTORY.LOCATION.LONGITUDE<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Location longitude field in host inventory.
{INVENTORY.MACADDRESS.A<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	MAC address A field in host inventory. {PROFILE.MACADDRESS<1-9>} is deprecated.
{INVENTORY.MACADDRESS.B<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	MAC address B field in host inventory.
{INVENTORY.MODEL<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Model field in host inventory.
{INVENTORY.NAME<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Name field in host inventory. {PROFILE.NAME<1-9>} is deprecated.
{INVENTORY.NOTES<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Notes field in host inventory. {PROFILE.NOTES<1-9>} is deprecated.
{INVENTORY.OOB.IP<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	OOB IP address field in host inventory.
{INVENTORY.OOB.NETMASK<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	OOB subnet mask field in host inventory.
{INVENTORY.OOB.ROUTER<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	OOB router field in host inventory.
{INVENTORY.OS<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	OS field in host inventory. {PROFILE.OS<1-9>} is deprecated.
{INVENTORY.OS.FULL<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	OS (Full details) field in host inventory.

Macro	Supported in	Description
~~{INVENTORY.OS.SHORT~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~OS (Short) field in host inventory.~~
~~{INVENTORY.POC.PRIMARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Primary POC cell field in host inventory.~~
~~{INVENTORY.POC.PRIMARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Primary POC email field in host inventory.~~
~~{INVENTORY.POC.PRIMARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Primary POC name field in host inventory.~~
~~{INVENTORY.POC.PRIMARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Primary POC notes field in host inventory.~~
~~{INVENTORY.POC.PRIMARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Primary POC phone A field in host inventory.~~
~~{INVENTORY.POC.PRIMARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Primary POC phone B field in host inventory.~~
~~{INVENTORY.POC.PRIMARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Primary POC screen name field in host inventory.~~
~~{INVENTORY.POC.SECONDARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Secondary POC cell field in host inventory.~~
~~{INVENTORY.POC.SECONDARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Secondary POC email field in host inventory.~~
~~{INVENTORY.POC.SECONDARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Secondary POC name field in host inventory.~~
~~{INVENTORY.POC.SECONDARY~~ ~~9>}~~	~~Trigger-based notifications and commands~~ → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	~~Secondary POC notes field in host inventory.~~

Macro	Supported in	Description
<code>{INVENTORY.POC.SECONDARYPHONEA<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Secondary POC phone A field in host inventory.
<code>{INVENTORY.POC.SECONDARYPHONEB<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Secondary POC phone B field in host inventory.
<code>{INVENTORY.POC.SECONDARYSCREEN<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Secondary POC screen name field in host inventory.
<code>{INVENTORY.SERIALNO.A<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Serial number A field in host inventory. <code>{PROFILE.SERIALNO<1-9>}</code> is deprecated.
<code>{INVENTORY.SERIALNO.B<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Serial number B field in host inventory.
<code>{INVENTORY.SITE.ADDRESSA<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Site address A field in host inventory.
<code>{INVENTORY.SITE.ADDRESSB<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Site address B field in host inventory.
<code>{INVENTORY.SITE.ADDRESSC<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Site address C field in host inventory.
<code>{INVENTORY.SITE.CITY<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Site city field in host inventory.
<code>{INVENTORY.SITE.COUNTRY<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Site country field in host inventory.
<code>{INVENTORY.SITE.NOTES<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Site notes field in host inventory.
<code>{INVENTORY.SITE.RACK<1-9>}</code>	Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Site rack location field in host inventory.

Macro	Supported in	Description
~~{INVENTORY.SITE.STATE<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Site state/province field in host inventory.
~~{INVENTORY.SITE.ZIP<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Site ZIP/postal field in host inventory.
~~{INVENTORY.SOFTWARE<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Software field in host inventory. {PROFILE.SOFTWARE<1-9>} is deprecated.
~~{INVENTORY.SOFTWARE.APPLICATION.A<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Software application A field in host inventory.
~~{INVENTORY.SOFTWARE.APPLICATION.B<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Software application B field in host inventory.
~~{INVENTORY.SOFTWARE.APPLICATION.C<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Software application C field in host inventory.
~~{INVENTORY.SOFTWARE.APPLICATION.D<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Software application D field in host inventory.
~~{INVENTORY.SOFTWARE.APPLICATION.E<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Software application E field in host inventory.
~~{INVENTORY.SOFTWARE.FULL<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Software (Full details) field in host inventory.
~~{INVENTORY.TAG<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Tag field in host inventory. {PROFILE.TAG<1-9>} is deprecated.
~~{INVENTORY.TYPE<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Type field in host inventory. {PROFILE.DEVICETYPE<1-9>} is deprecated.
~~{INVENTORY.TYPE.FULL<1-9>}~~	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	Type (Full details) field in host inventory.

Macro	Supported in	Description
<code>{INVENTORY.URL.A<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	<i>URL A field in host inventory.</i>
<code>{INVENTORY.URL.B<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	<i>URL B field in host inventory.</i>
<code>{INVENTORY.URL.C<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	<i>URL C field in host inventory.</i>
<code>{INVENTORY.VENDOR<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → Tag names and values → Map element labels, map URL names and values	<i>Vendor field in host inventory.</i>
<code>{ITEM.DESCRPTION<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications	<i>Description of the Nth item in the trigger expression that caused a notification.</i> Supported since 2.0.0.
<code>{ITEM.ID<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	<i>Numeric ID of the Nth item in the trigger expression that caused a notification.</i> Supported since 1.8.12.
<code>{ITEM.KEY<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications → HTTP agent type item, item prototype and discovery rule fields: URL, query fields, request body, headers, proxy, SSL certificate file, SSL key file.	<i>Key of the Nth item in the trigger expression that caused a notification.</i> Supported since 2.0.0. {TRIGGER.KEY} is deprecated.
<code>{ITEM.KEY.ORIG<1-9>}</code>	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications	<i>Original key (with macros not expanded) of the Nth item in the trigger expression that caused a notification.</i> Supported since 2.0.6.
<code>{ITEM.LASTVALUE<1-9>}</code>	→ Trigger-based notifications and commands → Problem update notifications → Trigger names and descriptions → Tag names and values → Trigger URLs⁷	<i>The latest value of the Nth item in the trigger expression that caused a notification.</i> It will resolve to *UNKNOWN* in the frontend if the latest history value has been collected more than the ZBX_HISTORY_PERIOD time ago (defined in defines.inc.php). 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. Supported since 1.4.3. It is alias to <code>{{HOST.HOST}}:{{ITEM.KEY}}.last()</code> . Customizing the macro value is supported for this macro; starting with Zabbix 3.2.0.
<code>{ITEM.LOG.AGE<1-9>}</code>	→ Trigger-based notifications and commands → Problem update notifications	<i>Age of the log item event.</i>
<code>{ITEM.LOG.DATE<1-9>}</code>	→ Trigger-based notifications and commands → Problem update notifications	<i>Date of the log item event.</i>
<code>{ITEM.LOG.EVENTID<1-9>}</code>	→ Trigger-based notifications and commands → Problem update notifications	<i>ID of the event in the event log.</i> For Windows event log monitoring only.

Macro	Supported in	Description
{ITEM.LOG.NSEVERITY< ~~1~~ -9>}	→ Trigger-based notifications and commands → Problem update notifications	<i>Numeric severity of the event in the event log.</i> For Windows event log monitoring only.
{ITEM.LOG.SEVERITY<1-9>}	→ Trigger-based notifications and commands → Problem update notifications	<i>Verbal severity of the event in the event log.</i> For Windows event log monitoring only.
{ITEM.LOG.SOURCE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications	<i>Source of the event in the event log.</i> For Windows event log monitoring only.
{ITEM.LOG.TIME<1-9>}	→ Trigger-based notifications and commands → Problem update notifications	<i>Time of the log item event.</i>
{ITEM.NAME<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications	<i>Name of the Nth item (with macros resolved) in the trigger expression that caused a notification.</i>
{ITEM.NAME.ORIG<1-9>}	→ Trigger-based notifications and commands → Internal notifications → Problem update notifications	<i>Original name (i.e. without macros resolved) of the Nth item in the trigger expression that caused a notification.</i> Supported since 2.0.6.
{ITEM.STATE<1-9>}	→ Item-based internal notifications	<i>The latest state of the Nth item in the trigger expression that caused a notification.</i> Possible values: Not supported and Normal . Supported since 2.2.0.
{ITEM.VALUE<1-9>}	→ Trigger-based notifications and commands → Problem update notifications → Trigger names and descriptions → Tag names and values → Trigger URLs ⁷	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 ZBX_HISTORY_PERIOD time ago (defined in defines.inc.php). Supported since 1.4.3.
{LLDRULE.DESCRPTION}	→ LLD-rule based internal notifications	<i>Description of the low-level discovery rule which caused a notification.</i> Supported since 2.2.0.
{LLDRULE.ID}	→ LLD-rule based internal notifications	<i>Numeric ID of the low-level discovery rule which caused a notification.</i> Supported since 2.2.0.
{LLDRULE.KEY}	→ LLD-rule based internal notifications	<i>Key of the low-level discovery rule which caused a notification.</i> Supported since 2.2.0.
{LLDRULE.KEY.ORIG}	→ LLD-rule based internal notifications	<i>Original key (with macros not expanded) of the low-level discovery rule which caused a notification.</i> Supported since 2.2.0.
{LLDRULE.NAME}	→ LLD-rule based internal notifications	<i>Name of the low-level discovery rule (with macros resolved) that caused a notification.</i> Supported since 2.2.0.

Macro	Supported in	Description
{LLDRULE.NAME.ORIG}	→ LLD-rule based internal notifications	<i>Original name (i.e. without macros resolved) of the low-level discovery rule that caused a notification.</i> Supported since 2.2.0.
{LLDRULE.STATE}	→ LLD-rule based internal notifications	<i>The latest state of the low-level discovery rule.</i> Possible values: Not supported and Normal . Supported since 2.2.0.
{MAP.ID}	→ Map element labels, map URL names and values	<i>Network map ID.</i>
{MAP.NAME}	→ Map element labels, map URL names and values → Text field in map shapes	<i>Network map name.</i> Supported since 3.4.0.
{PROXY.DESCRPTION<1-9>}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	<i>Description of the proxy.</i> 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.DESCRPTION} here, without indexing. 3) proxy to which an active agent registered (in auto-registration notifications). Use {PROXY.DESCRPTION} here, without indexing. Supported since 2.4.0.
{PROXY.NAME<1-9>}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	<i>Name of the proxy.</i> 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 auto-registration notifications). Use {PROXY.NAME} here, without indexing. Supported since 1.8.4.
{TIME}	→ Trigger-based notifications and commands → Discovery notifications → Auto-registration notifications → Internal notifications → Problem update notifications	<i>Current time in hh:mm:ss.</i>
{TRIGGER.DESCRPTION}	→ Trigger-based notifications and commands → Trigger-based internal notifications → Problem update notifications	<i>Trigger description.</i> Supported since 2.0.4. Starting with 2.2.0, all macros supported in a trigger description will be expanded if {TRIGGER.DESCRPTION} is used in notification text. {TRIGGER.COMMENT} is deprecated.
{TRIGGER.EVENTS.ACK}	→ Trigger-based notifications and commands → Problem update notifications → Map element labels	<i>Number of acknowledged events for a map element in maps, or for the trigger which generated current event in notifications.</i> Supported since 1.8.3.
{TRIGGER.EVENTS.PROBLEMACK}	→ Trigger-based notifications and commands → Problem update notifications → Map element labels	<i>Number of acknowledged PROBLEM events for all triggers disregarding their state.</i> Supported since 1.8.3.
{TRIGGER.EVENTS.PROBLEMUNACK}	→ Trigger-based notifications and commands → Problem update notifications → Map element labels	<i>Number of unacknowledged PROBLEM events for all triggers disregarding their state.</i> Supported since 1.8.3.
{TRIGGER.EVENTS.UNACK}	→ Trigger-based notifications and commands → Problem update notifications → Map element labels	<i>Number of unacknowledged events for a map element in maps, or for the trigger which generated current event in notifications.</i> Supported in map element labels since 1.8.3.
{TRIGGER.HOSTGROUP.NAMES}	→ Trigger-based notifications and commands → Problem update notifications → Trigger-based internal notifications	<i>A sorted (by SQL query), comma-space separated list of host groups in which the trigger is defined.</i> Supported since 2.0.6.

Macro	Supported in	Description
{TRIGGER.PROBLEM.EVENTS.PROBLEM.ACK}	→ Trigger-based notifications and commands → Problem update notifications	Number of acknowledged PROBLEM events for triggers in PROBLEM state. Supported since 1.8.3.
{TRIGGER.PROBLEM.EVENTS.PROBLEM.UNACK}	→ Trigger-based notifications and commands → Problem update notifications	Number of unacknowledged PROBLEM events for triggers in PROBLEM state. Supported since 1.8.3.
{TRIGGER.EXPRESSION}	→ Trigger-based notifications and commands → Trigger-based internal notifications → Problem update notifications	Trigger expression. Supported since 1.8.12.
{TRIGGER.EXPRESSION.RECOVERY}	→ Trigger-based notifications and commands → Trigger-based internal notifications → Problem update notifications	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 → Trigger-based internal notifications → Problem update notifications → Map element labels, map URL names and values → Trigger URLs → Trigger tag values	Numeric trigger ID which triggered this action. Supported in trigger URLs since Zabbix 1.8.8, in trigger tag values since 4.2.8.
{TRIGGER.NAME}	→ Trigger-based notifications and commands → Trigger-based internal notifications → Problem update notifications	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 → Trigger-based internal notifications → Problem update notifications	Original name of the trigger (i.e. without macros resolved). Supported since 2.0.6.
{TRIGGER.NSEVERITY}	→ Trigger-based notifications and commands → Trigger-based internal notifications → Problem update notifications	Numerical trigger severity. Possible values: 0 - Not classified, 1 - Information, 2 - Warning, 3 - Average, 4 - High, 5 - Disaster. Supported starting from Zabbix 1.6.2.
{TRIGGER.SEVERITY}	→ Trigger-based notifications and commands → Trigger-based internal notifications → Problem update notifications	Trigger severity name. Can be defined in <i>Administration</i> → <i>General</i> → <i>Trigger severities</i> .
{TRIGGER.STATE}	→ Trigger-based internal notifications	The latest state of the trigger. Possible values: Unknown and Normal . Supported since 2.2.0.
{TRIGGER.STATUS}	→ Trigger-based notifications and commands → Problem update notifications	Current trigger value. Can be either PROBLEM or OK . {STATUS} is deprecated.
{TRIGGER.TEMPLATE.NAME}	→ Trigger-based notifications and commands → Trigger-based internal notifications → Problem update notifications	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. Supported since 2.0.6.
{TRIGGER.URL}	→ Trigger-based notifications and commands → Trigger-based internal notifications → Problem update notifications	Trigger URL.
{TRIGGER.VALUE}	→ Trigger-based notifications and commands → Trigger expressions → Problem update notifications	Current trigger numeric value: 0 - trigger is in OK state, 1 - trigger is in PROBLEM state.
{TRIGGERS.UNACK}	→ Map element labels	Number of unacknowledged triggers for a map element, disregarding trigger state. A trigger is considered to be unacknowledged if at least one of its PROBLEM events is unacknowledged.
{TRIGGERS.PROBLEM.UNACK}	→ Map element labels	Number of unacknowledged PROBLEM triggers for a map element. A trigger is considered to be unacknowledged if at least one of its PROBLEM events is unacknowledged. Supported since 1.8.3.

Macro	Supported in	Description
{TRIGGERS.ACK}	→ Map element labels	<i>Number of acknowledged triggers for a map element, disregarding trigger state.</i> A trigger is considered to be acknowledged if all of it's PROBLEM events are acknowledged. Supported since 1.8.3.
{TRIGGERS.PROBLEM.ACK}	Map element labels	<i>Number of acknowledged PROBLEM triggers for a map element.</i> A trigger is considered to be acknowledged if all of it's PROBLEM events are acknowledged. Supported since 1.8.3.
{USER.FULLNAME}	→ Problem update notifications	<i>Name and surname of the user who added event acknowledgement.</i> Supported since 3.4.0.
{host:key.func(param)}	→ Trigger-based notifications and commands → Problem update notifications → Map element/shape labels ³ → Link labels in maps ³ → Graph names ⁵ → Trigger expressions ⁶	<i>Simple macros, as used in building trigger expressions.</i> Supported for shape labels in maps since 3.4.2.
{ \$MACRO }	→ See: User macros supported by location	<i>User-definable macros.</i>
{ #MACRO }	→ See: Low-level discovery macros	<i>Low-level discovery macros.</i> Supported since 2.0.0.
		Customizing the macro value is supported for this macro, starting with Zabbix 4.0.0.

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.

² In remote commands, 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.

³ Only the **avg**, **last**, **max** and **min** functions, with seconds as parameter are supported in this macro in map labels.

⁴ Supported since Zabbix 2.2.0, {HOST.*} macros are supported in web scenario *Name*, *Variables*, *Headers*, *SSL certificate file* and *SSL key file* fields and in scenario step *Name*, *URL*, *Post*, *Headers* and *Required string* fields.

⁵ Supported since Zabbix 2.2.0. 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:

```
* {Cisco switch:ifAlias[{#SNMPINDEX}].last()}
* {{HOST.HOST}:ifAlias[{#SNMPINDEX}].last()}
```

⁶ While supported to build trigger expressions, simple macros may not be used inside each other.

⁷ Supported since 4.0.0.

Indexed macros

The indexed macro syntax of {MACRO<1-9>} is limited to the context of **trigger expressions**. It can be used to reference hosts 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).

Additionally the {HOST.HOST<1-9>} macro is also supported within the {host:key.func(param)} macro in **graph names**. For example, {{HOST.HOST2}:key.func()} in the graph name will refer to the host of the second item in the graph.

Warning:

Use macros **without** index (i. e. {HOST.HOST}, {HOST.IP}, etc) in all other contexts.

2 User macros supported by location

Overview

User-definable macros are supported in the following locations:

- Hosts
 - Interface IP/DNS
 - Interface port
 - Tag names and values
- Templates
 - Tag names and values
- Passive proxy
 - Interface port
- Items and item prototypes
 - Name (*deprecated*)
 - Key parameters
 - Update interval
 - Custom intervals
 - History storage period
 - Trend storage period
 - Item value preprocessing steps
 - SNMPv3 context name
 - SNMPv3 security name
 - SNMPv3 auth pass
 - SNMPv3 priv pass
 - SNMPv1/v2 community
 - SNMP OID
 - SNMP port
 - SSH username
 - SSH public key
 - SSH private key
 - SSH password
 - SSH script
 - Telnet username
 - Telnet password
 - Telnet script
 - Calculated item formula
 - Trapper item "Allowed hosts" field
 - Database monitoring additional parameters
 - JMX item endpoint field
 - 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 SSL certificate file field
 - HTTP agent SSL key file field
 - HTTP agent SSL key password field
 - HTTP agent HTTP timeout field
 - HTTP agent HTTP allowed hosts field
- Discovery
 - Update interval
 - SNMPv3 context name
 - SNMPv3 security name
 - SNMPv3 auth pass
 - SNMPv3 priv pass
 - SNMPv1/v2 community
 - SNMP OID
- Low-level discovery rule
 - Name
 - Key parameters

- Update interval
- Custom intervals
- SNMPv3 context name
- SNMPv3 security name
- SNMPv3 auth pass
- SNMPv3 priv pass
- SNMPv1/v2 community
- SNMP OID
- SNMP port
- SSH username
- SSH public key
- SSH private key
- SSH password
- SSH script
- Telnet username
- Telnet password
- Telnet script
- Trapper item "Allowed hosts" field
- Database monitoring additional parameters
- JMX item endpoint field
- Keep lost resources period
- Filter regular expressions
- 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 timeout field
- Web scenario
 - Name
 - Update interval
 - Agent
 - HTTP proxy
 - Variables
 - Headers
 - Step name
 - Step URL
 - Step post variables
 - Step headers
 - Step timeout
 - Required string
 - Required status codes
 - Authentication (user and password)
 - SSL certificate file
 - SSL key file
 - SSL key password
- Triggers
 - Name
 - Expression (only in constants and function parameters)
 - Description
 - URLs
 - Tag names and values
 - Tag for matching
- Trigger-based notifications and commands
- Trigger-based internal notifications
- Problem update notifications
- Action operations
 - Default operation step duration
 - Step duration

- Action conditions
 - Time period condition
- Global scripts (including confirmation text)
- URL field of dynamic URL screen element
- Administration → Users → Media: 'When active' field
- Administration → General → Working time: 'Working time' field

For a complete list of all macros supported in Zabbix, see [supported macros](#).

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

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**
- last parameter of **aggregate checks**
- 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)
- slide show configuration ('Default delay' field)
- 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* ('Refresh unsupported items' field)

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:

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>120
{host:system.uptime[]}.last()}<86400
{host:system.cpu.load.avg(600)}<10
{host:vm.memory.size[available].last()}<20971520
```

could be changed to:

```
{host:zabbix[proxy,zabbix_proxy,lastaccess]}>2m
{host:system.uptime.last()}<1d
{host:system.cpu.load.avg(10m)}<10
{host:vm.memory.size[available].last()}<20M
```

9 Setting time periods

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
<i>d</i>	Day of the week: 1 - Monday, 2 - Tuesday ,... , 7 - Sunday
<i>hh</i>	Hours: 00-24
<i>mm</i>	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. This is true starting from version 1.8.7, for everything, while **Working time** has always worked this way.

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
- *Additional log entry is created for remote commands on Zabbix server to save script execution output and

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 Recipes for monitoring

General

Monitoring server availability

At least three methods (or combination of all methods) may be used in order to monitor availability of a server.

- ICMP ping ("icmpping" key)
- "zabbix[host,agent,available]" item
- trigger function `nodata()` for monitoring the availability of hosts that use active checks only

Sending alerts via WinPopUps

WinPopUps maybe very useful if you're running Windows OS and want to get quick notification from Zabbix. It could be good addition for email-based alert messages. Details about enabling of WinPopUps can be found at <http://www.zabbix.com/forum/showthread.php?t=2147>.

Monitoring specific applications

AS/400

IBM AS/400 platform can be monitored using SNMP. More information is available at <http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg244504.html?Open>.

MySQL

Several user parameters can be used for the monitoring of MySQL in the agent configuration file: `/usr/local/etc/zabbix_agentd.conf`

```
### Set of parameters for monitoring MySQL server (v3.23.42 and later)
### Change -u and add -p if required
#UserParameter=mysql.ping,mysqladmin -uroot ping|grep alive|wc -l
#UserParameter=mysql.uptime,mysqladmin -uroot status|cut -f2 -d":"|cut -f2 -d" "
#UserParameter=mysql.threads,mysqladmin -uroot status|cut -f3 -d":"|cut -f2 -d" "
#UserParameter=mysql.questions,mysqladmin -uroot status|cut -f4 -d":"|cut -f2 -d" "
#UserParameter=mysql.slowqueries,mysqladmin -uroot status|cut -f5 -d":"|cut -f2 -d" "
#UserParameter=mysql.qps,mysqladmin -uroot status|cut -f9 -d":"|cut -f2 -d" "
#UserParameter=mysql.version,mysql -V
```

- *mysql.ping*

Check whether MySQL is alive.

Result: 0 - not started 1 - alive

- *mysql.uptime*

Number of seconds MySQL is running.

- *mysql.threads*

Number of MySQL threads.

- *mysql.questions*

Number of processed queries.

- *mysql.slowqueries*

Number of slow queries.

- *mysql.qps*

Queries per second.

- *mysql.version*

Version of MySQL. For example: `mysql Ver 14.14 Distrib 5.1.53, for pc-linux-gnu (i686)`

For additional information see also the `userparameter_mysql.conf` file in `conf/zabbix_agentd` directory.

Mikrotik routers

Use SNMP agent provided by Mikrotik. See <http://www.mikrotik.com> for more information.

Windows

Use Zabbix Windows agent included (pre-compiled) into Zabbix distribution.

Tuxedo

Tuxedo command line utilities `tmadmin` and `qmadm` can be used in definition of a `UserParameter` in order to return per server/service/queue performance counters and availability of Tuxedo resources.

Informix

Standard Informix utility **onstat** can be used for monitoring of virtually every aspect of Informix database. Also, Zabbix can retrieve information provided by Informix SNMP agent.

HP OpenView

Zabbix can be configured to send messages to OpenView server. The following steps must be performed:

Step 1

Define new media.

The media will execute a script which will send required information to OpenView.

Step 2

Define new user.

The user has to be linked with the media.

Step 3

Configure actions.

Configure actions to send all (or selected) trigger status changes to the user.

Step 4

Write media script.

The script will have the following logic. If trigger is ON, then execute OpenView command *opcmsg -id application=<application> msg_grp=<msg_grp> object=<object> msg_text=<text>*. The command will return unique message ID which has to be stored somewhere, preferably in a new table of ZABBIX database. If trigger is OFF then *opcmack <message id>* has to be executed with message ID retrieved from the database.

Refer to OpenView official documentation for more details about *opcmsg* and *opcmack*. The media script is not given here.

12 Performance tuning

Attention:

This is a work in progress.

Overview

It is very important to have Zabbix system properly tuned for maximum performance.

Hardware

General advice on hardware:

- Use fastest processor available
- SCSI or SAS is better than IDE (performance of IDE disks may be significantly improved by using utility *hdparm*) and SATA
- 15K RPM is better than 10K RPM which is better than 7200 RPM
- Use fast RAID storage
- Use fast Ethernet adapter
- Having more memory is always better

Operating system

- Use latest (stable!) version of OS
- Exclude unnecessary functionality from kernel
- Tune kernel parameters

Zabbix configuration parameters

Many parameters may be tuned to get optimal performance.

zabbix_server

StartPollers

General rule - keep value of this parameter as low as possible. Every additional instance of *zabbix_server* adds known overhead, in the same time, parallelism is increased. Optimal number of instances is achieved when queue, on average, contains minimum number of parameters (ideally, 0 at any given moment). This value can be monitored by using internal check *zabbix[queue]*.

Note:

See the "See also" section at the bottom of this page to find out how to configure optimal count of zabbix processes.

DebugLevel

Optimal value is 3.

DBSocket

MySQL only. It is recommended to use DBSocket for connection to the database. That is the fastest and the most secure way.

Database engine

This is probably the most important part of Zabbix tuning. Zabbix heavily depends on the availability and performance of database engine.

- use fastest database engine, i.e. MySQL or PostgreSQL
- use stable release of a database engine
- rebuild MySQL or PostgreSQL from sources to get maximum performance
- follow performance tuning instructions taken from MySQL or PostgreSQL documentation
- for MySQL, use InnoDB table structure
- ZABBIX works at least 1.5 times faster (comparing to MyISAM) if InnoDB is used. This is because of increased parallelism. However, InnoDB requires more CPU power.
- tuning the database server for the best performance is highly recommended.
- keep database tables on different hard disks
- 'history', 'history_str', 'items', 'functions', 'triggers', and 'trends' are most heavily used tables.
- for large installations keeping MySQL temporary files in tmpfs is:
 - MySQL >= 5.5: not recommended ([MySQL bug #58421](#))
 - MySQL < 5.5: recommended

GUI debugging

Problems related to the frontend performance may be diagnosed using the frontend **debug mode**.

General advice

- monitor required parameters only
- tune 'Update interval' for all items. Keeping a small update interval may be good for nice graphs, however, this may overload Zabbix
- tune parameters for default templates
- tune housekeeping parameters
- do not monitor parameters which return the same information.
- avoid the use of triggers with long period given as function argument. For example, max(3600) will be calculated significantly slower than max(60).

Viewing Zabbix process performance with "ps" and "top"

Since Zabbix 2.2 processes change their commandlines to display current activity and meaningful statistics, like:

UID	PID	PPID	C	STIME	TTY	TIME	CMD
zabbix22	4584	1	0	14:55	?	00:00:00	zabbix_server -c /home/zabbix22/zabbix_server.conf
zabbix22	4587	4584	0	14:55	?	00:00:00	zabbix_server: configuration syncer [syncd configuration in 0.018748 s]
zabbix22	4588	4584	0	14:55	?	00:00:00	zabbix_server: db watchdog [syncd alerts config in 0.018748 s]
zabbix22	4608	4584	0	14:55	?	00:00:00	zabbix_server: timer #1 [updated 0 hosts, suppressed 0 events]
zabbix22	4637	4584	0	14:55	?	00:00:01	zabbix_server: history syncer #3 [processed 0 values, 0 triggers]
zabbix22	4657	4584	0	14:55	?	00:00:00	zabbix_server: vmware collector #1 [updated 0, removed 0 VMware items]
zabbix22	4670	1	0	14:55	?	00:00:00	zabbix_proxy -c /home/zabbix22/zabbix_proxy.conf
zabbix22	4673	4670	0	14:55	?	00:00:00	zabbix_proxy: configuration syncer [syncd config 15251 bytes]
zabbix22	4674	4670	0	14:55	?	00:00:00	zabbix_proxy: heartbeat sender [sending heartbeat message success]
zabbix22	4688	4670	0	14:55	?	00:00:00	zabbix_proxy: icmp pinger #1 [got 1 values in 1.811128 sec, id=1]
zabbix22	4690	4670	0	14:55	?	00:00:00	zabbix_proxy: housekeeper [deleted 9870 records in 0.233491 sec]
zabbix22	4701	4670	0	14:55	?	00:00:08	zabbix_proxy: http poller #2 [got 1 values in 0.024105 sec, id=1]
zabbix22	4707	4670	0	14:55	?	00:00:00	zabbix_proxy: history syncer #4 [processed 0 values, 0 triggers]
zabbix22	4738	1	0	14:55	?	00:00:00	zabbix_agentd -c /home/zabbix22/zabbix_agentd.conf
zabbix22	4739	4738	0	14:55	?	00:00:00	zabbix_agentd: collector [idle 1 sec]
zabbix22	4740	4738	0	14:55	?	00:00:00	zabbix_agentd: listener #1 [waiting for connection]
zabbix22	4741	4738	0	14:55	?	00:00:00	zabbix_agentd: listener #2 [processing request]

The main process is an exception. Instead of current activity the original commandline is shown. This helps to distinguish processes on systems with multiple Zabbix instances.

This feature is not implemented for Microsoft Windows.

If logging level is set to **DebugLevel=4** these activity and statistics messages are also written into log file.

Linux

On Linux systems `ps` command can be used together with `watch` command for observing how Zabbix is doing. For example, to run `ps` command 5 times per second to see process activities:

```
watch -n 0.2 ps -fu zabbix
```

To show only Zabbix proxy and agent processes:

```
watch -tn 0.2 'ps -f -C zabbix_proxy -C zabbix_agentd'
```

To show only history syncer processes:

```
watch -tn 0.2 'ps -fC zabbix_server | grep history'
```

The `ps` command produces a wide output (approximately 190 columns) as some activity messages are long. If your terminal has less than 190 columns of text you can try

```
watch -tn 0.2 'ps -o cmd -C zabbix_server -C zabbix_proxy -C zabbix_agentd'
```

to display only commandlines without UID, PID, start time etc.

`top` command also can be used for observing Zabbix performance. Pressing 'c' key in `top` shows processes with their commandlines. In our tests on Linux `top` and `atop` correctly displayed changing activities of Zabbix processes, but `htop` was not displaying changing activities.

BSD systems

If `watch` command is not installed, a similar effect can be achieved with

```
while [ 1 ]; do ps x; sleep 0.2; clear; done
```

AIX, HP-UX

If `watch` command is not available, one can try

```
while [ 1 ]; do ps -fu zabbix; sleep 1; clear; done
```

Solaris

By default the `ps` command does not show changing activities. One option is to use `/usr/ucb/ps` instead. If `watch` command is not installed, a periodically updated list of processes can be shown with

```
while [ 1 ]; do /usr/ucb/ps gxww; sleep 1; clear; done
```

On Solaris 11:

- `/usr/ucb/ps` is not installed by default. You may need to install `ucb` package, e.g. `pkg install compatibility/ucb`,
- if Zabbix daemon has been started by privileged user its activities are not shown to non-privileged user.
- the `sleep` command accepts not only whole seconds but also fractions of second (e.g. `sleep 0.2`).

See also

1. [How to configure optimal count of zabbix processes](#)

13 Version compatibility

Supported agents

Zabbix agents starting with version 1.4 are compatible with Zabbix 4.2. However, 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 new and improved items, improved performance and reduced memory usage, use the latest 4.2 agent.

Note that Zabbix 4.2 agent cannot be used with Zabbix server older than 4.2.

Supported Zabbix proxies

Zabbix 4.2.x server can only work with Zabbix 4.2.x proxies. Zabbix 4.2.x proxies can only work with Zabbix 4.2.x server.

Attention:

It is no longer possible to start the upgraded server and have older, yet 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. For more information, see the [upgrade procedure](#).

Warnings about using incompatible Zabbix daemon versions are logged since Zabbix 4.2.2.

Supported XML files

XML files, exported with 1.8, 2.0, 2.2, 2.4, 3.0, 3.2, 3.4 and 4.0 are supported for import in Zabbix 4.2.

Attention:

In Zabbix 1.8 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.

14 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 recognised.

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

15 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;

```

16 Issues with SELinux

Socket-based inter-process communication has been added since Zabbix 3.4. 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 are present while the process is running.

17 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.

Zabbix manpages

These are Zabbix manpages for Zabbix processes.

zabbix_agentd

Section: Maintenance Commands (8)

Updated: 2018-10-11

[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 of 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

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

zabbix_get(8), **zabbix_proxy**(8), **zabbix_sender**(8), **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: 15:28:13 GMT, October 18, 2018

zabbix_get

Section: User Commands (1)

Updated: 2018-10-11

[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*] -**k** *item-key*

zabbix_get -s *host-name-or-IP* [-**p** *port-number*] [-**I** *IP-address*] --**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* -**k** *item-key*

zabbix_get -s *host-name-or-IP* [-**p** *port-number*] [-**I** *IP-address*] --**tls-connect** **psk** --**tls-psk-identity** *PSK-identity* --**tls-psk-file** *PSK-file* -**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.

-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.

-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

[zabbix_agentd\(8\)](#), [zabbix_proxy\(8\)](#), [zabbix_sender\(8\)](#), [zabbix_server\(8\)](#)

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: 15:28:13 GMT, October 18, 2018

zabbix_proxy

Section: Maintenance Commands (8)

Updated: 2018-10-11

[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.

housekeeper_execute

Execute the housekeeper. Ignored if housekeeper is being currently executed.

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

[zabbix_agentd\(8\)](#), [zabbix_get\(8\)](#), [zabbix_sender\(8\)](#), [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: 15:28:13 GMT, October 18, 2018

zabbix_sender

Section: User Commands (1)

Updated: 2019-07-10

[Index](#) [Return to Main Contents](#)

NAME

zabbix_sender - Zabbix sender utility

SYNOPSIS

zabbix_sender [-v] -z server [-p port] [-I IP-address] -s host -k key -o value

zabbix_sender [-v] -z server [-p port] [-I IP-address] [-s host] [-T] [-r] -i input-file

zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address] [-s host] -k key -o value

zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address] [-s host] [-T] [-r] -i input-file

zabbix_sender [-v] -z server [-p port] [-I IP-address] -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 -k key -o value

zabbix_sender [-v] -z server [-p port] [-I IP-address] [-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 [-T] [-r] -i input-file

zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address] [-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 -k key -o value

zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address] [-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 [-T] [-r] -i input-file

zabbix_sender [-v] -z server [-p port] [-I IP-address] -s host --tls-connect psk --tls-psk-identity PSK-identity --tls-psk-file PSK-file -k key -o value

zabbix_sender [-v] -z server [-p port] [-I IP-address] [-s host] --tls-connect psk --tls-psk-identity PSK-identity --tls-psk-file

PSK-file [-T] [-r] -i input-file

zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address] [-s host] --tls-connect psk --tls-psk-identity PSK-identity
--tls-psk-file PSK-file -k key -o value
zabbix_sender [-v] -c config-file [-z server] [-p port] [-I IP-address] [-s host] --tls-connect psk --tls-psk-identity PSK-identity
--tls-psk-file PSK-file [-T] [-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**, **TLSCConnect**, **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.

-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. 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.

-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.

-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

zabbix_agentd(8), **zabbix_get(8)**, **zabbix_proxy(8)**, **zabbix_server(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: 13:48:52 GMT, July 10, 2019

zabbix_server

Section: Maintenance Commands (8)

Updated: 2018-10-11

[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.

housekeeper_execute

Execute the housekeeper. Ignored if housekeeper is being currently executed.

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, 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

zabbix_agentd(8), **zabbix_get**(8), **zabbix_proxy**(8), **zabbix_sender**(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: 15:28:13 GMT, October 18, 2018