**What is Zabbix**

**https://www.zabbix.com/documentation/current/manual/introduction/about**

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, virtual machines, applications, services, databases, websites, the cloud and more. Zabbix uses a flexible notification mechanism that allows users to configure e-mail based alerts for virtually any event. This allows a fast reaction to server problems. Zabbix offers excellent reporting and data visualization features based on the stored data. This makes Zabbix ideal for capacity planning.

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

[**Data gathering**](https://www.zabbix.com/documentation/current/manual/config/items)

* 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**](https://www.zabbix.com/documentation/current/manual/config/triggers)

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

[**Highly configurable alerting**](https://www.zabbix.com/documentation/current/manual/config/notifications)

* 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**](https://www.zabbix.com/documentation/current/manual/config/visualization/graphs/simple)

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

[**Web monitoring capabilities**](https://www.zabbix.com/documentation/current/manual/web_monitoring)

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

[**Extensive visualization options**](https://www.zabbix.com/documentation/current/manual/config/visualization)

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

[**Historical data storage**](https://www.zabbix.com/documentation/current/manual/installation/requirements#database_size)

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

[**Easy configuration**](https://www.zabbix.com/documentation/current/manual/config/hosts)

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

[**Use of templates**](https://www.zabbix.com/documentation/current/manual/config/templates)

* grouping checks in templates
* templates can inherit other templates

[**Network discovery**](https://www.zabbix.com/documentation/current/manual/discovery)

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

[**Fast web interface**](https://www.zabbix.com/documentation/current/manual/web_interface)

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

[**Zabbix API**](https://www.zabbix.com/documentation/current/manual/api)

* Zabbix API provides **programmable interface** to Zabbix for mass manipulations, 3rd party software integration and other purposes.

[**Permissions system**](https://www.zabbix.com/documentation/current/manual/config/users_and_usergroups)

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

[**Full featured and easily extensible agent**](https://www.zabbix.com/documentation/current/manual/concepts/agent)

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

[**Binary daemons**](https://www.zabbix.com/documentation/current/manual/concepts/server)

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

[**Ready for complex environments**](https://www.zabbix.com/documentation/current/manual/distributed_monitoring)

* remote monitoring made easy by using a Zabbix proxy

**Architecture**

<https://www.zabbix.com/documentation/current/manual/introduction/overview#architecture>

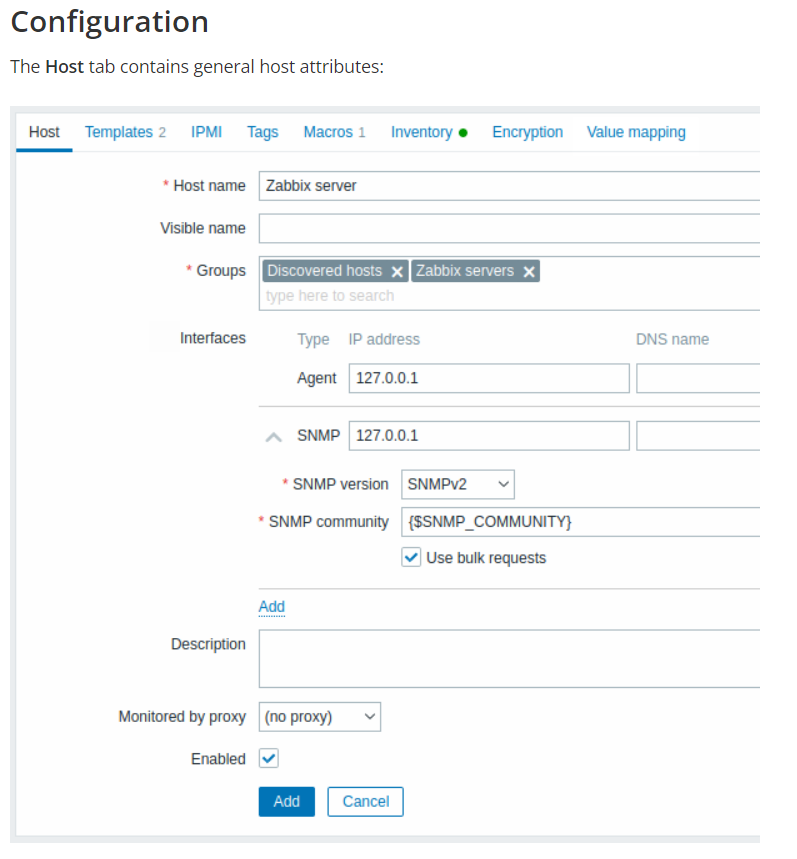
* Zabbix consists of several major software components, the responsibilities of which are outlined below.
* **Server**
* [Zabbix server](https://www.zabbix.com/documentation/current/manual/concepts/server) is the central component to which **agents report** availability and integrity information and 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](https://www.zabbix.com/documentation/current/manual/concepts/proxy) can collect performance and availability data on behalf of Zabbix server. A proxy is an optional part of Zabbix deployment; however, it may be very beneficial to distribute the load of a single Zabbix server.
* **Agent**
* Zabbix agents are **deployed on monitoring targets** to actively monitor local resources and applications and **report the gathered data to Zabbix server**. Since Zabbix 4.4, there are two types of agents available: the [Zabbix agent](https://www.zabbix.com/documentation/current/manual/concepts/agent) (lightweight, supported on many platforms, written in C) and the [Zabbix agent 2](https://www.zabbix.com/documentation/current/manual/concepts/agent) (extra-flexible, easily extendable with plugins, written in Go).
* **Data flow**
* In addition it is important to take a step back and have a look at the overall data flow within Zabbix.
* In order to create an item that gathers data you must first **create a host**.
* Moving to the other end of the Zabbix spectrum you must first have an item to create a trigger.
* You must have a trigger to create an action.
* Thus if you want to receive an alert that your CPU load is too high on *Server X* you must first create a host entry for *Server X* followed by an item for monitoring its CPU, then a trigger which activates if the CPU is too high, followed by an action which sends you an email.
* While that may seem like a lot of steps, with the use of templating it really isn't. However, due to this design it is possible to create a very flexible setup.
* **Definitions**
* [***host***](https://www.zabbix.com/documentation/current/manual/config/hosts)
* - *a networked device that you want to monitor, with IP/DNS.*

https://www.zabbix.com/documentation/current/manual/config/hosts/host

* **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.

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



Several host interface types are supported for a host: Agent, SNMP, JMX and IPMI.

|  |  |
| --- | --- |
| *Port* | TCP/UDP port number. Default values are: 10050 for Zabbix agent, 161 for SNMP agent, 12345 for JMX and 623 for IPMI. |

* [***host group***](https://www.zabbix.com/documentation/current/manual/config/hosts)
* - *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***](https://www.zabbix.com/documentation/current/manual/config/items)
* - *a particular piece of data that you want to receive off of a host, a metric of data.*
* 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](https://www.zabbix.com/documentation/current/manual/config/items/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.
* [***value preprocessing***](https://www.zabbix.com/documentation/current/manual/config/items/item#item_value_preprocessing)
* - *a transformation of received metric value* before saving it to the database.
* [***trigger***](https://www.zabbix.com/documentation/current/manual/config/triggers)
* - *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.

**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. |
| PROBLEM | Normally means that something happened. For example, the processor load is too high. |

In a basic trigger we may want to set a threshold for a five minute average of some data, for example, the CPU load. This is accomplished by defining a trigger expression where:

* we apply the 'avg' function to the value received in the item key
* we use a five minute period for evaluation
* we set a threshold of '2'

avg(host/key,5m)>2

This trigger will “fire” (become PROBLEM) if the five-minute average is over 2.

In a more complex trigger, the expression may include a **combination** of multiple functions and multiple thresholds. See also: [Trigger expression](https://www.zabbix.com/documentation/current/manual/config/triggers/expression).

Most trigger functions are evaluated based on [history](https://www.zabbix.com/documentation/current/manual/config/items/history_and_trends) data, while some trigger functions for long-term analytics, e.g. **trendavg()**, **trendcount()**, etc, use trend data.

**Calculation time**

A trigger is recalculated every time Zabbix server receives a new value that is part of the expression. When a new value is received, each function that is included in the expression is recalculated (not just the one that received the new value).

Additionally, a trigger is recalculated each time when a new value is received **and** every 30 seconds if time-based functions are used in the expression.

Time-based functions are **nodata()**, **date()**, **dayofmonth()**, **dayofweek()**, **time()**, **now()**); they are recalculated every 30 seconds by the Zabbix history syncer process.

**Evaluation period**

An evaluation period is used in functions referencing the item history. It allows to specify the interval we are interested in. It can be specified as time period (30s, 10m, 1h) or as a value range (#5 - for five latest values).

The evaluation period is measured up to "now" - where "now" is the latest recalculation time of the trigger (see [Calculation time](https://www.zabbix.com/documentation/current/manual/config/triggers#calculation_time) above); "now" is not the "now" time of the server.

The evaluation period specifies either:

* To consider all values between "now-time period" and "now" (or, with time shift, between "now-time shift-time period" and "now-time\_shift")
* To consider no more than the num count of values from the past, up to "now"
  + If there are 0 available values for the time period or num count specified - then the trigger or calculated item that uses this function becomes unsupported

Note that:

* If only a single function (referencing data history) is used in the trigger, “now” is always the latest received value. For example, if the last value was received an hour ago, the evaluation period will be regarded as up to the latest value an hour ago.
* A new trigger is calculated as soon as the first value is received (history functions); it will be calculated within 30 seconds for time-based functions. Thus the trigger will be calculated even though perhaps the set evaluation period (for example, one hour) has not yet passed since the trigger was created. The trigger will also be calculated after the first value, even though the evaluation range was set, for example, to ten latest values.
* [***event***](https://www.zabbix.com/documentation/current/manual/config/events)
* - *a single occurrence of something that deserves attention such as a trigger changing state or a discovery/agent autoregistration taking place.*
* [***event tag***](https://www.zabbix.com/documentation/current/manual/config/event_correlation/trigger/event_tags)
* - *a pre-defined marker for the event.* It may be used in event correlation, permission granulation, etc.
* [***event correlation***](https://www.zabbix.com/documentation/current/manual/config/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***](https://www.zabbix.com/documentation/current/manual/web_interface/frontend_sections/monitoring/problems)
* - *a trigger that is in “Problem” state.*
* [***problem update***](https://www.zabbix.com/documentation/current/manual/acknowledges#updating_problems)
* - *problem management options provided by Zabbix, such as adding comment, acknowledging, changing severity or closing manually.*
* [***action***](https://www.zabbix.com/documentation/current/manual/config/notifications/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***](https://www.zabbix.com/documentation/current/manual/config/notifications/action/escalations)
* - *a custom scenario for executing operations within an action; a sequence of sending notifications/executing remote commands.*
* [***media***](https://www.zabbix.com/documentation/current/manual/config/notifications/media)
* - *a means of delivering notifications; delivery channel.*

Media are the delivery channels used for sending notifications and alerts from Zabbix.

You can configure several media types:

* [E-mail](https://www.zabbix.com/documentation/current/manual/config/notifications/media/email)
* [SMS](https://www.zabbix.com/documentation/current/manual/config/notifications/media/sms)
* [Custom alertscripts](https://www.zabbix.com/documentation/current/manual/config/notifications/media/script)
* [Webhook](https://www.zabbix.com/documentation/current/manual/config/notifications/media/webhook)

Media types are configured in Administration → Media types.

**4 Webhook**

<https://www.zabbix.com/documentation/current/manual/config/notifications/media/webhook>

* [***notification***](https://www.zabbix.com/documentation/current/manual/config/notifications/action/operation/message)
* - *a message about some event sent to a user via the chosen media channel.*

**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](https://www.zabbix.com/documentation/current/manual/config/notifications/media) to send a message to

The default trigger severity ('Not classified') **must be** checked in user media [configuration](https://www.zabbix.com/documentation/current/manual/config/notifications/media/email#user_media) if you want to receive notifications for non-trigger events such as discovery, active agent autoregistration or internal evens.

* [configure an action operation](https://www.zabbix.com/documentation/current/manual/config/notifications/action/operation) that sends a message to one of the defined media

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](https://www.zabbix.com/documentation/current/manual/config/notifications/action/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](https://www.zabbix.com/documentation/current/manual/config/notifications/action/operation/remote_command).

**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.

* [***remote command***](https://www.zabbix.com/documentation/current/manual/config/notifications/action/operation/remote_command)
* - *a pre-defined command that is automatically executed on a monitored host upon some condition.*
* [***template***](https://www.zabbix.com/documentation/current/manual/config/templates)
* - *a set of entities (items, triggers, graphs, low-level discovery rules, web scenarios) ready to be applied to one or several hosts.*
* The job of templates is to speed up the deployment of monitoring tasks on a host; also to make it easier to apply mass changes to monitoring tasks. Templates are linked directly to individual hosts.
* [***web scenario***](https://www.zabbix.com/documentation/current/manual/web_monitoring)
* - *one or several HTTP requests to check the availability of a web site.*
* [***frontend***](https://www.zabbix.com/documentation/current/manual/introduction/overview#architecture)
* - *the web interface provided with Zabbix.*
* [***dashboard***](https://www.zabbix.com/documentation/current/manual/web_interface/frontend_sections/monitoring/dashboard)
* - *customizable section of the web interface displaying summaries and vizualizations* of important information in visual units called widgets.
* [***widget***](https://www.zabbix.com/documentation/current/manual/web_interface/frontend_sections/monitoring/dashboard/widgets)
* - *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***](https://www.zabbix.com/documentation/current/manual/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***](https://www.zabbix.com/documentation/current/manual/concepts/server)
* - *a central process of Zabbix software that performs monitoring, interacts with Zabbix proxies and agents, calculates triggers, sends notifications; a central repository of data.*
* [***Zabbix proxy***](https://www.zabbix.com/documentation/current/manual/concepts/proxy)
* - *a process that may collect data on behalf of Zabbix server, taking some processing load off of the server.*
* [***Zabbix agent***](https://www.zabbix.com/documentation/current/manual/concepts/agent)
* - *a process deployed on monitoring targets to actively monitor local resources and applications.*
* [***Zabbix agent 2***](https://www.zabbix.com/documentation/current/manual/concepts/agent2)
* - *a new generation of Zabbix agent to actively monitor local resources and applications, allowing to use custom plugins for monitoring.*
* Because Zabbix agent 2 shares much functionality with Zabbix agent, the term "Zabbix agent" in documentation stands for both - Zabbix agent and Zabbix agent 2, if the functional behavior is the same. Zabbix agent 2 is only specifically named where its functionality differs.
* [***encryption***](https://www.zabbix.com/documentation/current/manual/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***](https://www.zabbix.com/documentation/current/manual/discovery/network_discovery)
* - *automated discovery of network devices*.
* [***low-level discovery***](https://www.zabbix.com/documentation/current/manual/discovery/low_level_discovery)
* - *automated discovery of low-level entities on a particular device* (e.g. file systems, network interfaces, etc).
* [***low-level discovery rule***](https://www.zabbix.com/documentation/current/manual/discovery/low_level_discovery#discovery_rule)
* - *set of definitions for automated discovery of low-level entities* on a device.
* [***item prototype***](https://www.zabbix.com/documentation/current/manual/discovery/low_level_discovery#item_prototypes)
* - *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***](https://www.zabbix.com/documentation/current/manual/discovery/low_level_discovery#trigger_prototypes)
* - *a trigger with certain parameters as variables, ready for low-level discovery*. After low-level discovery the variables are automatically substituted with the real discovered parameters and the trigger automatically starts evaluating data.
* *Prototypes* of some other Zabbix entities are also in use in low-level discovery - graph prototypes, host prototypes, host group prototypes.
* [***agent auto-registration***](https://www.zabbix.com/documentation/current/manual/discovery/auto_registration)
* - *automated process whereby a Zabbix agent itself is registered* as a host and started to monitor.

<https://www.zabbix.com/documentation/current/manual/introduction/overview#architecture>

### 2 Agent

<https://www.zabbix.com/documentation/current/manual/concepts/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**](https://www.zabbix.com/documentation/current/manual/appendix/items/activepassive#passive_checks)**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](https://www.zabbix.com/documentation/current/manual/appendix/items/activepassive#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](https://www.zabbix.com/documentation/current/manual/config/items/itemtypes/zabbix_agent). Zabbix agent processes items of type 'Zabbix agent' or 'Zabbix agent (active)'.

#### 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.

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

### 4 Header[Old revisions](https://www.zabbix.com/documentation/current/manual/appendix/protocols/header_datalen?do=revisions)[Backlinks](https://www.zabbix.com/documentation/current/manual/appendix/protocols/header_datalen?do=backlink)[Export to PDF](https://www.zabbix.com/documentation/current/manual/appendix/protocols/header_datalen?do=export_pdf)

#### Overview

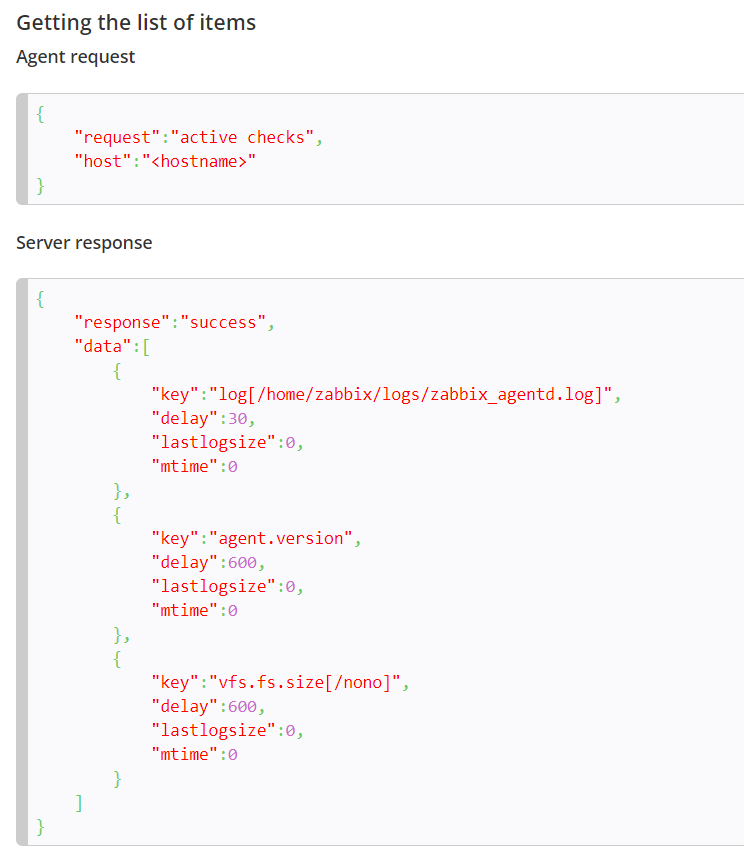
1. The header is present in response and request messages between Zabbix components. It is required to determine the length of message, if it is compressed or not and the format of message length fields. The header consists of:
2. <PROTOCOL> - "ZBXD" (4 bytes).
3. <FLAGS> - the protocol flags, (1 byte). 0x01 - Zabbix communications protocol, 0x02 - compression, 0x04 - large packet).
4. <DATALEN> - data length (4 bytes or 8 bytes for large packet). 1 will be formatted as 01/00/00/00 (four bytes, 32 bit number in little-endian format) or 01/00/00/00/00/00/00/00 (eight bytes, 64 bit number in little-endian format) for large packet.
5. <RESERVED> - uncompressed data length (4 bytes or 8 bytes for large packet). 1 will be formatted as 01/00/00/00 (four bytes, 32 bit number in little-endian format) or 01/00/00/00/00/00/00/00 (eight bytes, 64 bit number in little-endian format) for large packet.
6. When compression is enabled (0x02 flag) the <RESERVED> bytes contains uncompressed data size.
7. Zabbix protocol has 1GB packet size limit per connection. The limit of 1GB is applied for received packet data length and for uncompressed data length, however, when large packet is enabled (0x04 flag) it is possible for Zabbix proxy to receive configuration with size up to 16GB; note that large packet can only be used for Zabbix proxy configuration (since Zabbix 5.4.5), and Zabbix server will automatically set (0x04 flag) and send length fields as 8 bytes each when data length before compression exceeds 4GB.

#### 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](https://www.zabbix.com/documentation/current/manual/appendix/config/zabbix_agentd). 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)**.



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

**Passive and active agent checks**

<https://www.zabbix.com/documentation/current/manual/appendix/items/activepassive#passive_checks>

#### 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](https://www.zabbix.com/documentation/current/manual/installation/install_from_packages) section for instructions on how to install Zabbix agent as package.

Alternatively see instructions for [manual installation](https://www.zabbix.com/documentation/current/manual/installation/install#installing_zabbix_daemons) if you do not want to use packages.

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

**Agent on Windows systems**

Zabbix agent on Windows runs as a Windows service.

**Preparation**

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

C:\zabbix

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

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

**Installation**

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

C:\> c:\zabbix\zabbix\_agentd.exe -c c:\zabbix\zabbix\_agentd.conf -i

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

See [more details](https://www.zabbix.com/documentation/current/manual/appendix/install/windows_agent#installing_agent_as_windows_service) on installing and running Zabbix agent on Windows.

**Zabbix agent on Microsoft Windows**

<https://www.zabbix.com/documentation/current/manual/appendix/install/windows_agent#installing_agent_as_windows_service>

**6 Sender**[**Old revisions**](https://www.zabbix.com/documentation/current/manual/concepts/sender?do=revisions)[**Backlinks**](https://www.zabbix.com/documentation/current/manual/concepts/sender?do=backlink)[**Export to PDF**](https://www.zabbix.com/documentation/current/manual/concepts/sender?do=export_pdf)[**Back to top**](https://www.zabbix.com/documentation/current/manual/concepts/sender#dokuwiki__top)

**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](https://www.zabbix.com/documentation/current/manual/config/items/itemtypes/trapper) 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

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

**ZABBIX\_SENDER**[**Old**](https://www.zabbix.com/documentation/current/manpages/zabbix_sender?do=revisions)

**https://www.zabbix.com/documentation/current/manpages/zabbix\_sender**

## 1 Proxies[Old revisions](https://www.zabbix.com/documentation/current/manual/distributed_monitoring/proxies?do=revisions)[Backlinks](https://www.zabbix.com/documentation/current/manual/distributed_monitoring/proxies?do=backlink)[Export to PDF](https://www.zabbix.com/documentation/current/manual/distributed_monitoring/proxies?do=export_pdf)[Back to top](https://www.zabbix.com/documentation/current/manual/distributed_monitoring/proxies#dokuwiki__top)

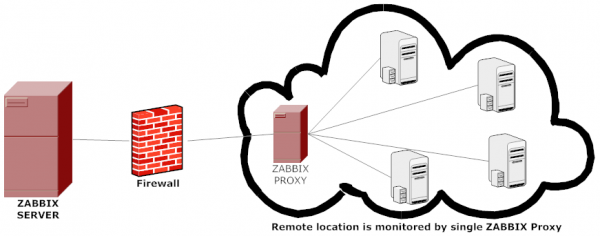
#### Overview

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

[](https://www.zabbix.com/documentation/current/_detail/manual/proxies/proxy.png?id=manual%3Adistributed_monitoring%3Aproxies)

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.

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](https://www.zabbix.com/documentation/current/manual/appendix/config/zabbix_proxy) control for how long the data are kept locally.

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](https://www.zabbix.com/documentation/current/manual/appendix/config/zabbix_server). As a result, proxy may start gathering data and send them to Zabbix server that ignores these data.

### **16 HTTP agent**

# 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](http://www.jsonrpc.org/specification) and the [JSON format homepage](https://json.org/).

<https://www.jsonrpc.org/specification>

**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](https://www.zabbix.com/documentation/current/manual/api/reference/user/login) method. Let us suppose that you want to log in as a standard 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](https://www.zabbix.com/documentation/current/manual/api/reference/host/get) method to retrieve the IDs, host names and interfaces of all configured [hosts](https://www.zabbix.com/documentation/current/manual/api/reference/host/object):
* {
* "jsonrpc": "2.0",
* "method": "host.get",
* "params": {
* "output": [
* "hostid",
* "host"
* ],
* "selectInterfaces": [
* "interfaceid",
* "ip"
* ]
* },
* "id": 2,
* "auth": "0424bd59b807674191e7d77572075f33"
* }
* 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
* }

#### Creating a new item

Let's create a new [item](https://www.zabbix.com/documentation/current/manual/api/reference/item/object) on “Zabbix server” using the data we've obtained from the previous host.get request. This can be done by using the [item.create](https://www.zabbix.com/documentation/current/manual/api/reference/item/create) method:

{

"jsonrpc": "2.0",

"method": "item.create",

"params": {

"name": "Free disk space on /home/joe/",

"key\_": "vfs.fs.size[/home/joe/,free]",

"hostid": "10084",

"type": 0,

"value\_type": 3,

"interfaceid": "1",

"delay": 30

},

"auth": "0424bd59b807674191e7d77572075f33",

"id": 3

}

A successful response will contain the ID of the newly created item, which can be used to reference the item in the following requests:

{

"jsonrpc": "2.0",

"result": {

"itemids": [

"24759"

]

},

"id": 3

}

**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](https://www.zabbix.com/documentation/current/manual/api/reference/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.

# 18. Web interface[Old revisions](https://www.zabbix.com/documentation/current/manual/web_interface?do=revisions)[Backlinks](https://www.zabbix.com/documentation/current/manual/web_interface?do=backlink)[Export to PDF](https://www.zabbix.com/documentation/current/manual/web_interface?do=export_pdf)

https://www.zabbix.com/documentation/current/manual/web\_interface

#### Overview

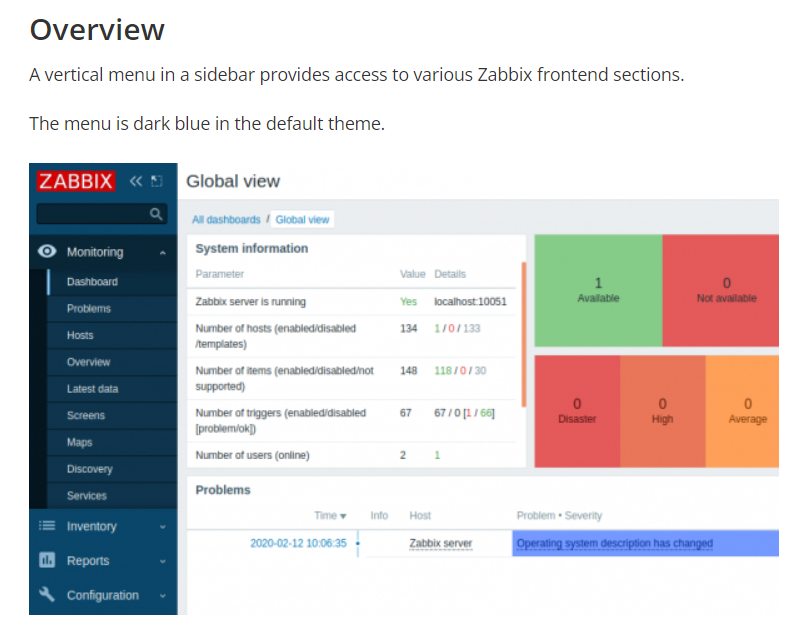
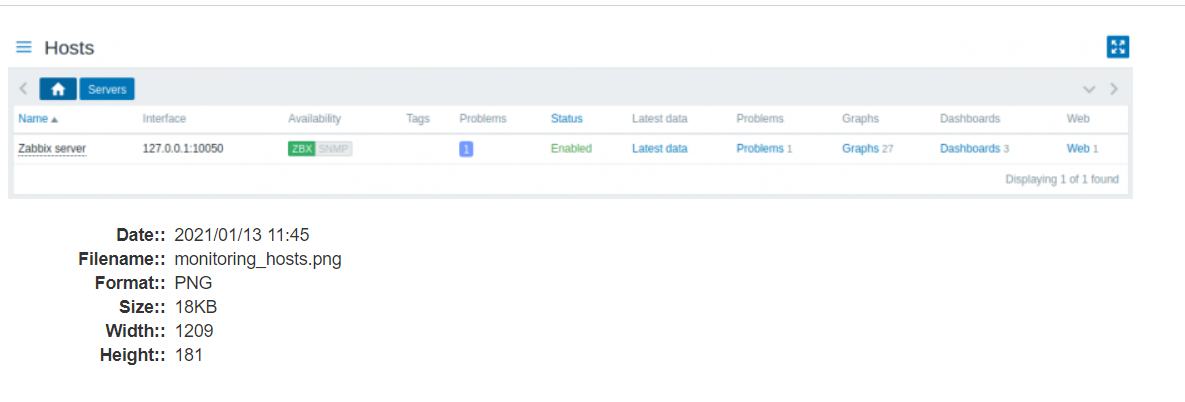
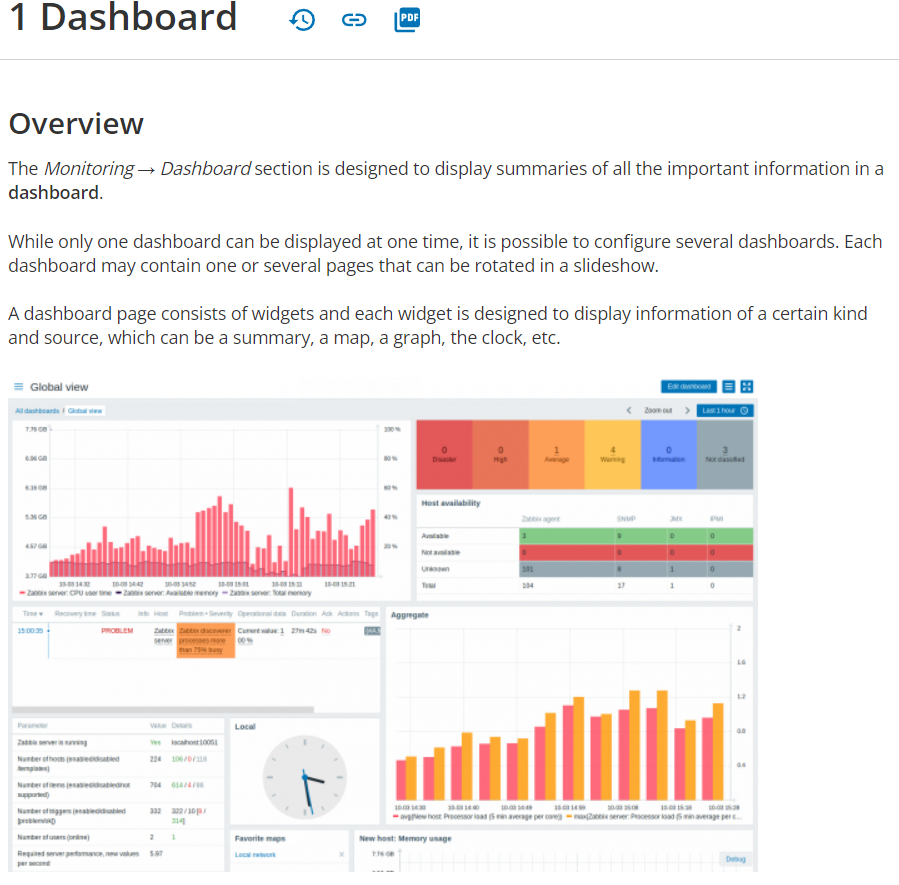
For an easy access to Zabbix from anywhere and from any platform, the web-based interface is provided.

If using more than one frontend instance make sure that **the locales and libraries (LDAP, SAML etc.) are installed** and configured identically for all frontends.

**1 Monitoring**[**Old revisions**](https://www.zabbix.com/documentation/current/manual/web_interface/frontend_sections/monitoring?do=revisions)[**Backlinks**](https://www.zabbix.com/documentation/current/manual/web_interface/frontend_sections/monitoring?do=backlink)[**Export to PDF**](https://www.zabbix.com/documentation/current/manual/web_interface/frontend_sections/monitoring?do=export_pdf)

**Overview**

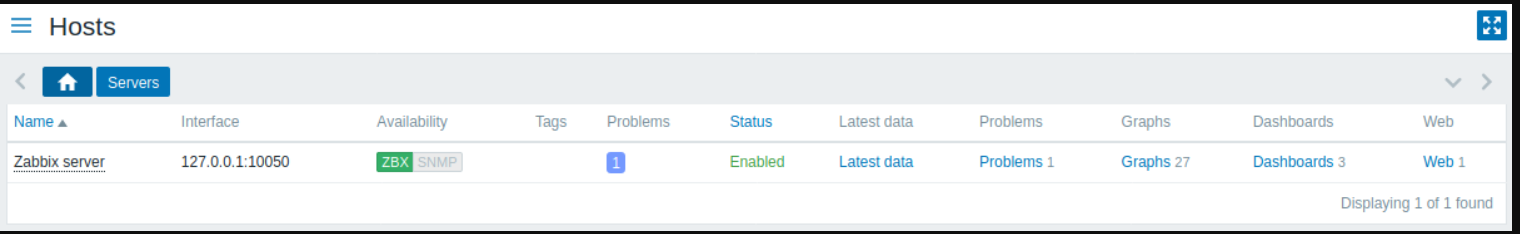
The Monitoring menu is all about displaying data. Whatever information Zabbix is configured to gather, visualize and act upon, it will be displayed in the various sections of the Monitoring menu.

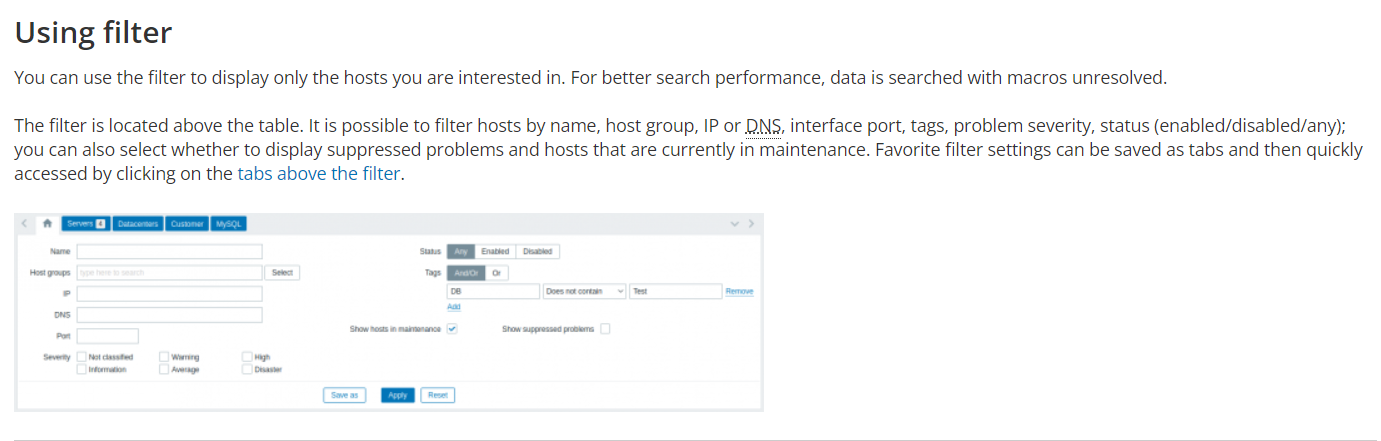


### 3 Hosts[Old revisions](https://www.zabbix.com/documentation/current/manual/web_interface/frontend_sections/monitoring/hosts?do=revisions)[Backlinks](https://www.zabbix.com/documentation/current/manual/web_interface/frontend_sections/monitoring/hosts?do=backlink)[Export to PDF](https://www.zabbix.com/documentation/current/manual/web_interface/frontend_sections/monitoring/hosts?do=export_pdf)

#### Overview

The Monitoring → Hosts section displays a full list of monitored hosts with detailed information about host interface, availability, tags, current problems, status (enabled/disabled), and links to easily navigate to the host's latest data, problem history, graphs, dashboards and web scenarios.





### 3 User settings[Old revisions](https://www.zabbix.com/documentation/current/manual/web_interface/user_profile?do=revisions)[Backlinks](https://www.zabbix.com/documentation/current/manual/web_interface/user_profile?do=backlink)[Export to PDF](https://www.zabbix.com/documentation/current/manual/web_interface/user_profile?do=export_pdf)

#### Overview

Depending on user role permissions, the User settings section may contain the following pages:

* User profile - for customizing certain Zabbix frontend features;
* API tokens - for managing API tokens assigned to the current user.

The list of available pages appears upon pressing on the [](https://www.zabbix.com/documentation/current/_detail/manual/web_interface/user_profile.png?id=manual%3Aweb_interface%3Auser_profile) user icon near the bottom of the Zabbix menu (not available for a guest user). It is also possible to switch between pages by using a title dropdown in the top left corner.

|  |  |
| --- | --- |
|  |  |
| Third-level menu. | Title dropdown. |

