# initMAX

Webinar

Zabbix API

all our microphones are muted

ask your questions in Q&A, not in the Chat

use Chat for discussion, networking or applause

1

Introduction

Tasks

› A datacenter was built a month ago and you need to add all new devices to Zabbix.

› In order to increase the availability of business critical systems, you need to integrate Zabbix with an Incident Management System.

› The manager asked to send a report on the availability of key systems every day at 8 am.

› In order to minimize manual work, you decided to create maintenance modes for network hosts automatically.

› You need to compare the Host list in CMDB and the Host list in Zabbix DB.

› You need to create thousands of hosts and items in Zabbix.

› You need to fix $ 1 in all templates.

….. And more.

Solution

ZABBIX comes with an API that provides access to almost all functions available in Zabbix.

Existence of a Zabbix API opens up a lot of opportunities for even greater efficiency in monitoring:

› easy two-way integration: set up a two-way integration with popular issue-tracking systems;

› third party software: integrate Zabbix with third party software like automation and graphing tools, notification and ticketing systems, configuration management, etc.

› configuration management: integrate configuration management systems with Zabbix API to add, remove or upgrade hardware or software easily;

› data retrieval: re-use information about organization's environment in other applications;

› mobile applications: create new applications to work with Zabbix.
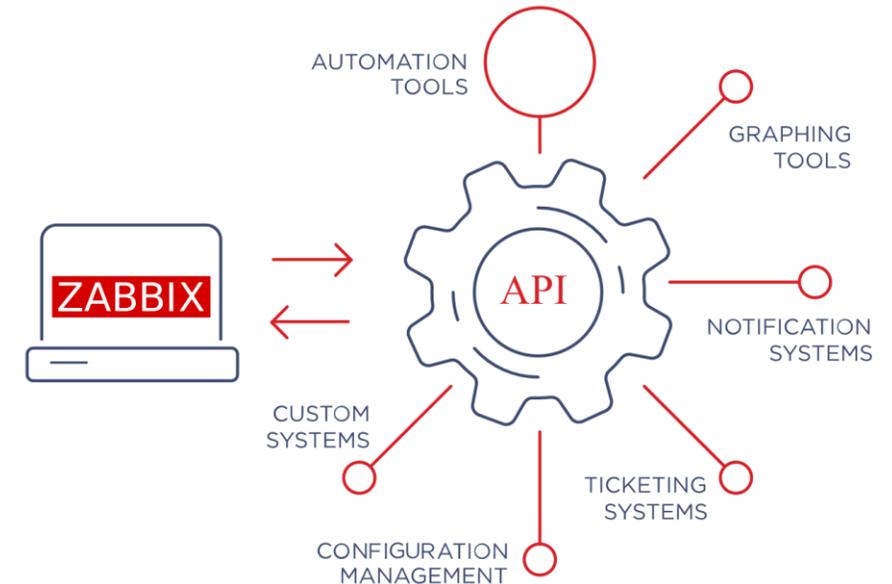
2

What is Zabbix API?

What is Zabbix API?

Interface is based on web server

Uses JSON RPC v2.0 specification:

› API consists of a set of separate methods

› requests and responses are encoded using JSON

Examples:

› host.create - creates a new host

› history.get - retrieves history data

› item.update - updates existing items
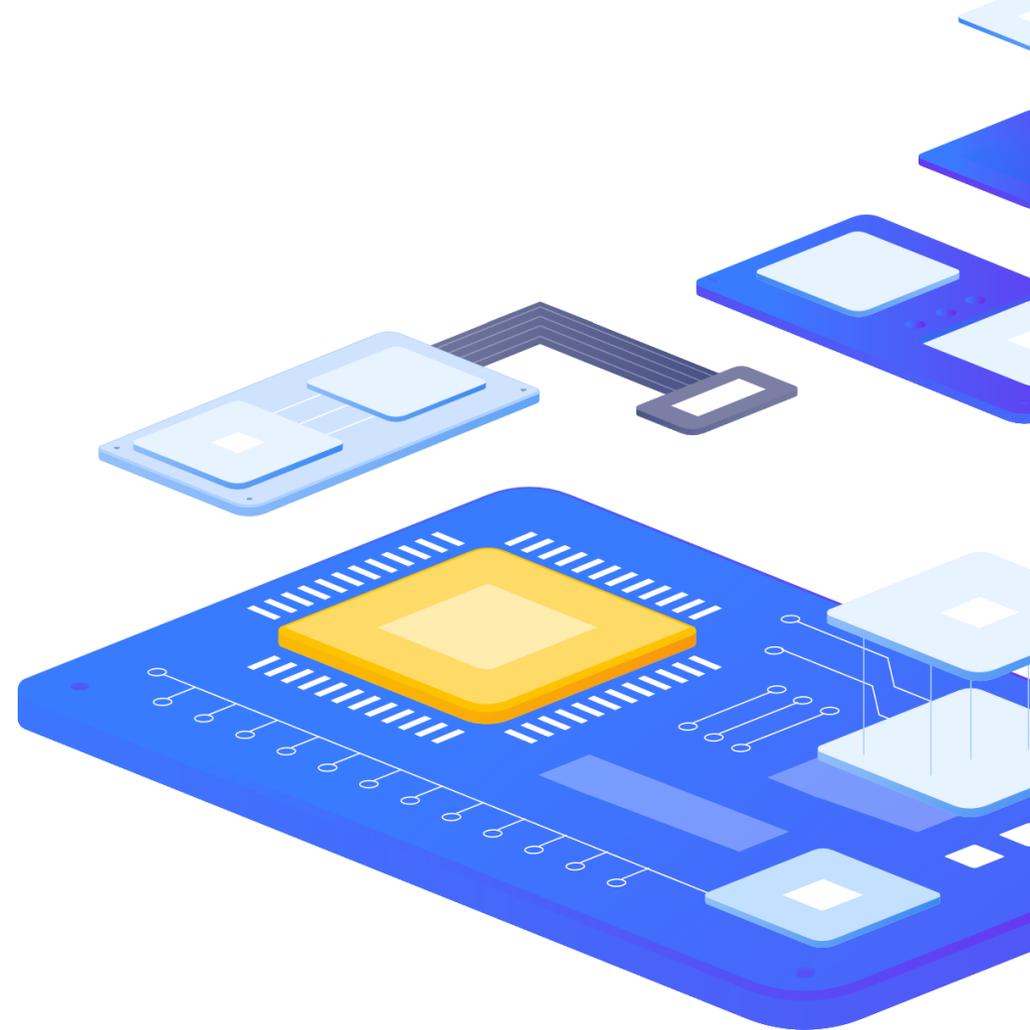
What is Zabbix API?

More than 230 different methods

Each of the methods performs one specific task
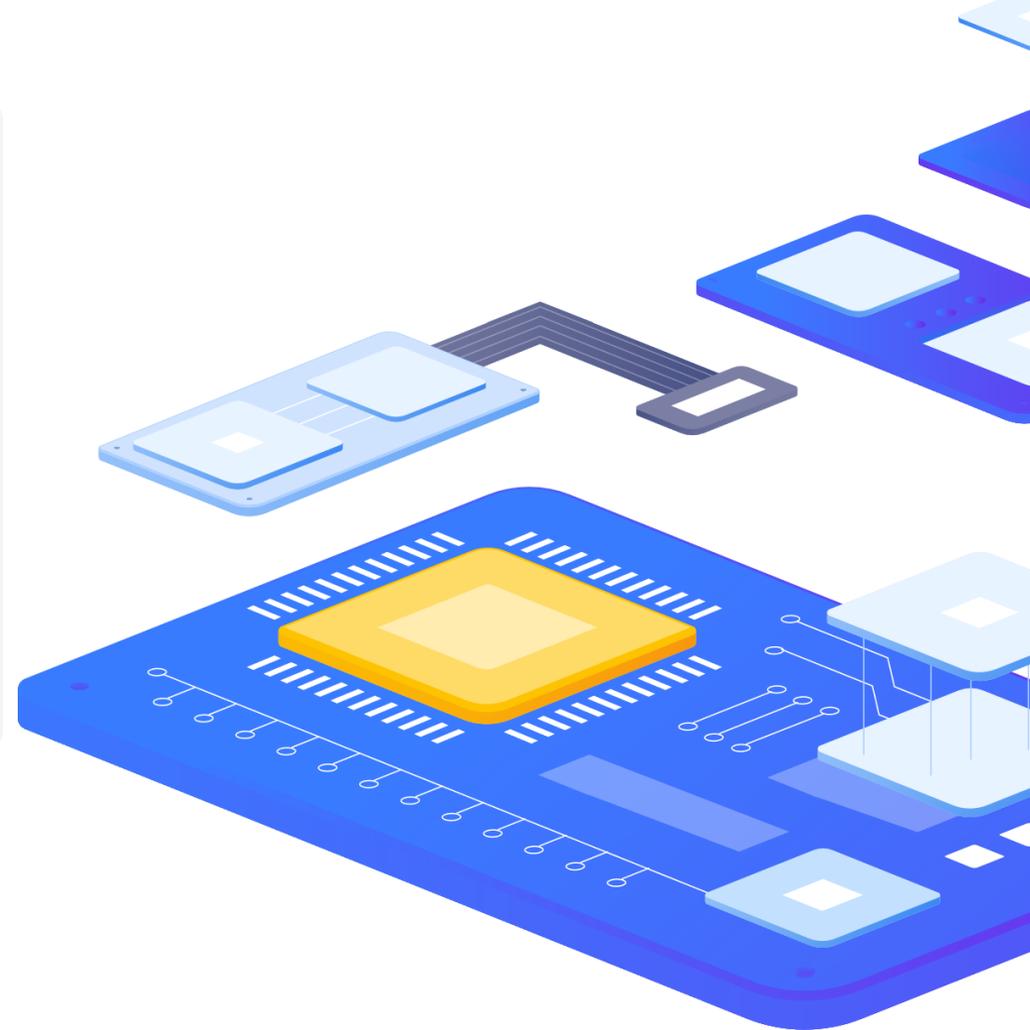
3

Structure of Zabbix API
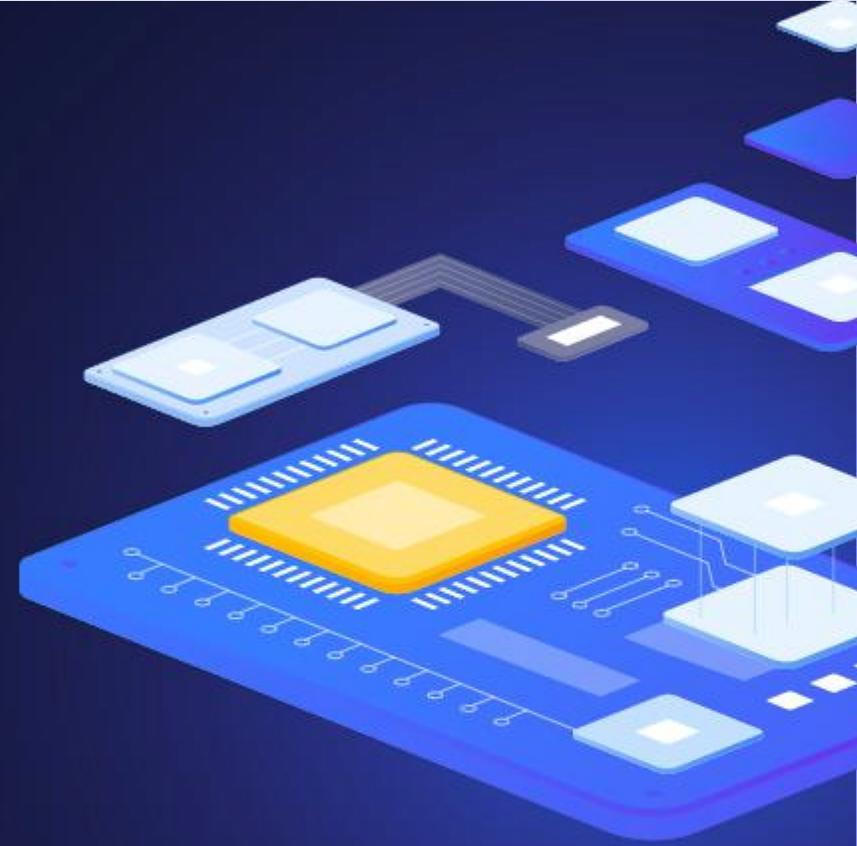
Structure of Zabbix API

Zabbix API is made of 3 building blocks: JSON, JSON-RPC and transport (e.g. HTTP)

› JSON is a simple format used to send and receive data

› JSON-RPC is a remote procedure call protocol encoded in JSON. JSON-RPC builds on top of JSON by adding a few standard keys to track requests

› The transport provides a real-time connection to API. The request/response/error is carried in the body of the HTTP message. HTTP is a stateless protocol: not persisting sessions between requests

Structure of Zabbix API

API request must contain certain properties:

- ❯ jsonrpc - version of the JSON-RPC protocol (must be 2.0)
- ❯ method - name of the method to be invoked
- ❯ params - Object or Array of values to be passed as parameters
- ❯ id - value used to match the response with the request that it is replying to (must be a number)
- ❯ auth - user authentication token or null

```
{
    "jsonrpc": "2.0",
    "method": "host.get",
    "params": {
        "proxyids": "15886", "output": ["hostid","name"]

    },
    "auth": "123aabc123ca321598a",
    "id": 1
}
```

Structure of Zabbix API

You will get either a response with a result, or an error if the request was unsuccessful

❯ The response object contains the following properties:

❯ jsonrpc - version of the JSON-RPC protocol

❯ result - the data returned by the method

❯ id - identifier of the corresponding request

```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "hostid": "15933",
      "name": "debian12"
    },
    {
      "hostid": "15883",
      "name": "DEMO Linux"
    }
  ]
}
```

Structure of Zabbix API

When rpc call encounters an error, the response contains the following properties:

› code - number that indicates the error type that occurred

› message - string providing a short description of the error

› data - value that contains additional information about the error

```
{
  "jsonrpc": "2.0",
  "error": {
    "code": -32601,
    "message": "Method not found.",
    "data": "Incorrect API \"hosst\"."
  },
  "id": 1
}
```

Structure of Zabbix API

The rpc call error codes:

❯ -32700 - invalid JSON. An error occurred on the server while parsing the JSON text (typo, wrong quotes, etc.)

❯ -32600 - received JSON is not a valid JSON-RPC Request

❯ -32601 - requested remote-procedure does not exist

❯ -32602 - invalid method parameters

❯ -32603 - Internal JSON-RPC error

❯ -32400 - System error

❯ -32300 - Transport error

❯ -32500 - Application error

4

How does API work?

# How does API work?

API Client

Zabbix API

JSON

WEB

DB
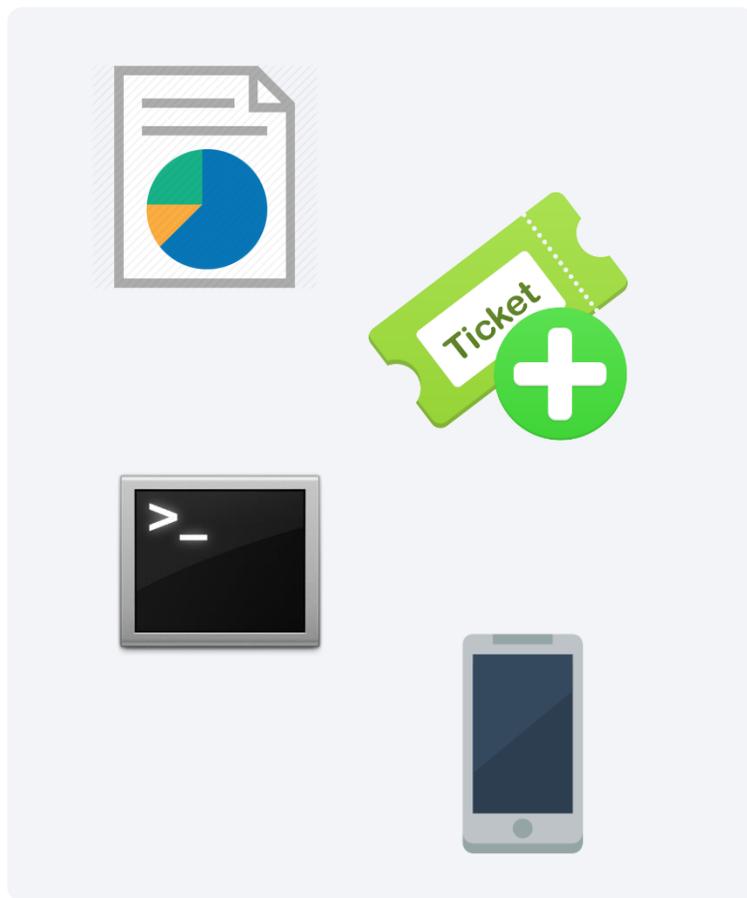
How does API work?

Authentication ⟩

API Client

API Server

How does API work?

Authentication

ID sessions

API Client

API Server

How does API work?

Authentication ⟩

⟨ ID sessions

Method A ⟩

API Client

API Server

How does API work?

Authentication >

< ID sessions

Method A >

< **Result A**
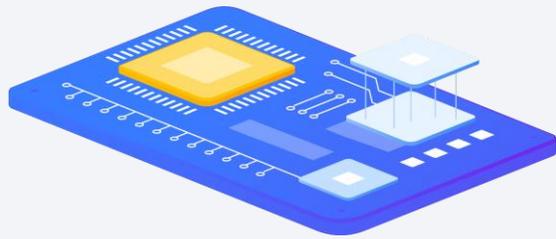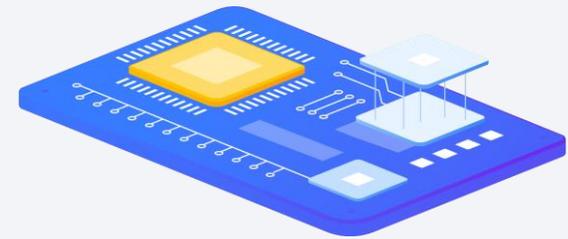
API Client

API Server
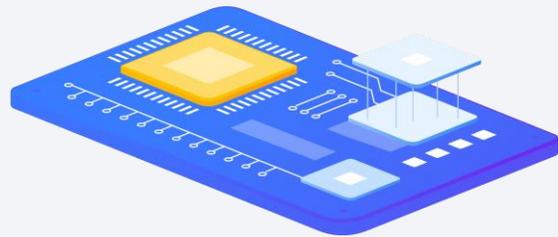
How does API work?


API Client

Authentication

ID sessions
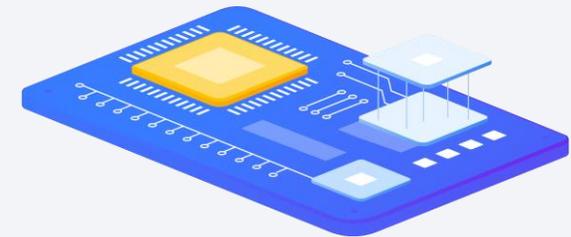
Method A

Result A

Method B
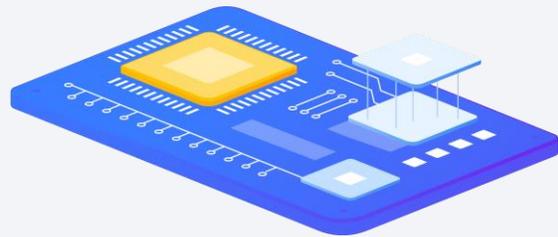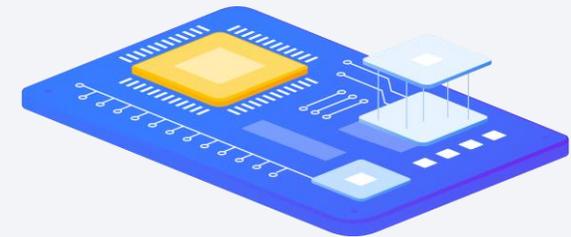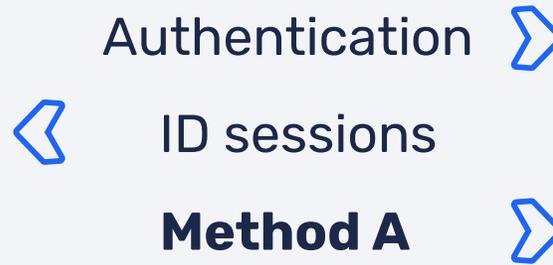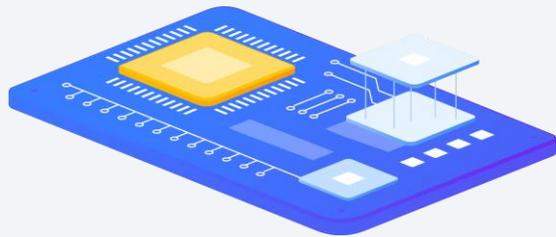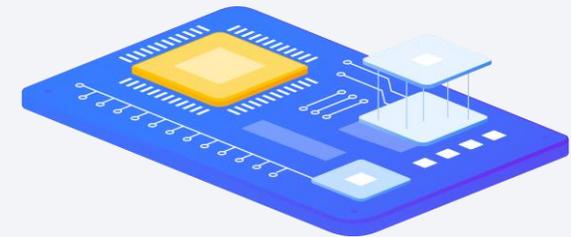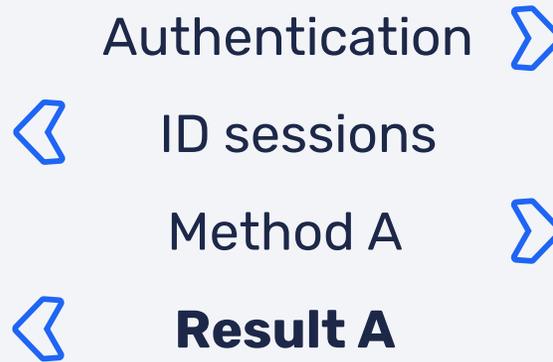
API Server

How does API work?

API Client

API Server
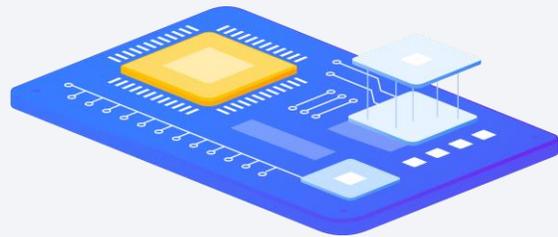
Authentication

ID sessions

Method A

Result A

Method B

Result B
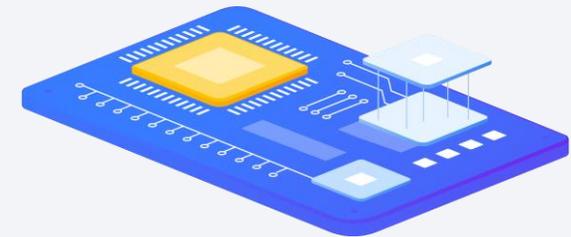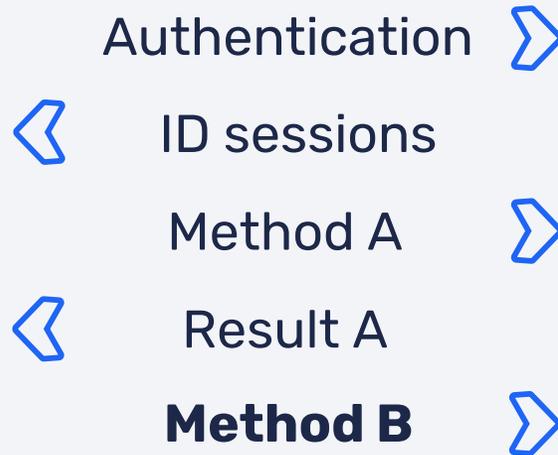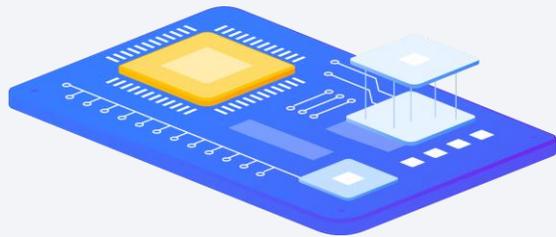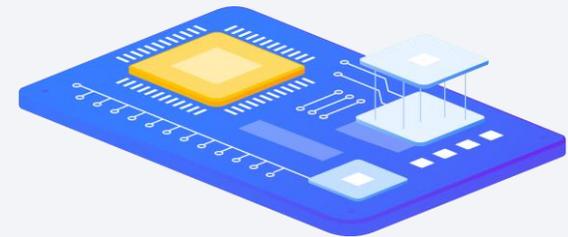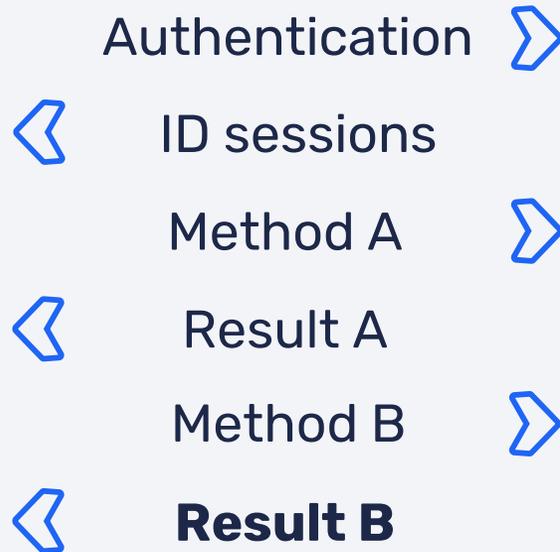
API Tokens

User based API Tokens

> Expiration Time
> User and Global managed
> Available from 5.4 version
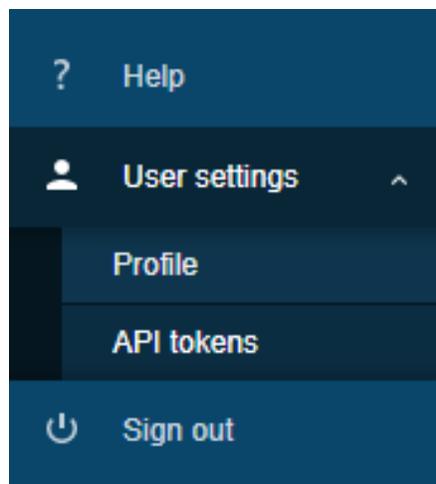
API Tokens and User Role Restrictions

API tokens ⌄

✓ API token updated

Name: Webinar

User: api

Auth token: 646043d847effdfa097ad4434e1974988b3f64f8c1e150a26eae8bff58547fee ⓘ Copy to clipboard

Expires at: 2022-04-10 00:00:00

Description: účet pro API přístup Webináře API

Enabled: ☑

Close

Access to API

Enabled ☑

API methods   Allow list   Deny list

correlation.delete ✕ dashboard.delete ✕ discoveryrule.delete ✕   Select
graphprototype.delete ✕ hostinterface.delete ✕

type here to search

How does API work?
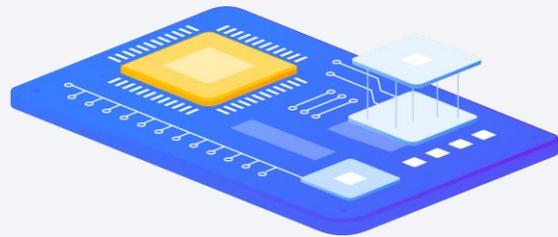
Method A

Result A

Method B

Result B

API Client
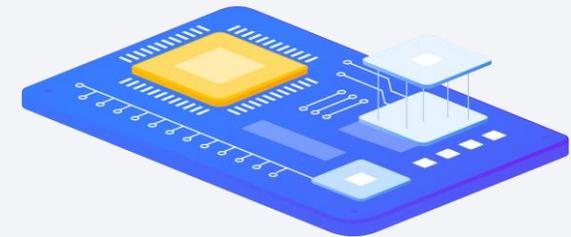
API Server

with API Token:

7ecc990488f71a5a48eeeb14b7b281f7c4b1738e2169d411efe

4

API connection examples

Zabbix API

Various programming or scripting languages :

› Use programming language you are familiar with

› Control workflow using built-in operands

› Some programming languages have Zabbix API plugins – community based, not supported by Zabbix

Zabbix API - Option 1. By using a client

Simple REST Client:

Response:

Simple REST Client

Request

URL: http://192.168.7.105/zabbix/api_jsonrpc.php

Method: ○ GET ● POST ○ PUT ○ DELETE

Headers: Content-Type: application/json

Data:
```
{"jsonrpc": "2.0",
"method":"user.login",
"params":{
        "user":"Admin",
        "password":"zabbix"},
"auth": null,"id":0}
```

Response

Status: 200 OK

Headers:
```
date: Tue, 10 Apr 2018 14:01:12 GMT
server: Apache/2.4.6 (CentOS) PHP/5.4.16
x-powered-by: PHP/5.4.16
access-control-max-age: 1000
access-control-allow-methods: POST
content-type: application/json
access-control-allow-origin: *
connection: Keep-Alive
access-control-allow-headers: Content-Type
content-length: 68
keep-alive: timeout=5, max=100
```

Data: {"jsonrpc":"2.0","result":"a2b2ec1eb81eec4de9136148e620974d","id":0}

Response

Status: 200 OK

Headers:
```
date: Tue, 10 Apr 2018 14:06:29 GMT
server: Apache/2.4.6 (CentOS) PHP/5.4.16
x-powered-by: PHP/5.4.16
access-control-max-age: 1000
access-control-allow-methods: POST
content-type: application/json
access-control-allow-origin: *
connection: Keep-Alive
access-control-allow-headers: Content-Type
content-length: 804
keep-alive: timeout=5, max=100
```

Data: {"jsonrpc":"2.0","result":[{"hostid":"10084","proxy_hostid":"0","host":"Zabbix server","status":"0",

Zabbix API – Option 2. From command line

Request via curl:

```
$ curl -i -X POST -H 'Content-
Type:application/json' -d'
{"jsonrpc": "2.0",
"method":"user.login",
"params":{
            "user":"Admin",
        "password":"zabbix"},
"auth": null,"id":0}
' http://127.0.0.1/zabbix/api_jsonrpc.php
```

> Authentication

> Credentials

> URL including host, port and API path

Response:

```
HTTP/1.1 200 OK
Date: Wed, 11 Nov 2015 09:32:41 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Access-Control-Allow-Origin: *
Access-Control-Allow-Headers: Content-Type
Access-Control-Allow-Methods: POST
Access-Control-Max-Age: 1000
Content-Length: 68
Connection: close
Content-Type: application/json

{"jsonrpc":"2.0",
"result":"2f2ec4720863281c34cdd3c4c8a5de46","id"
:0}
```

> Response including session ID

Zabbix API – Option 2. From command line

initMAX

Request via curl:

```
$ curl -i -X POST -H 'Content-Type:
application/json' -d '
{"jsonrpc":"2.0",
    "method":"host.get",
    "params":{
        "filter":{
            "host":"Zabbix server"
        }
    },

"auth":"2f2ec4720863281c34cdd3c4c8a5de46",
    "id":1}
' http://127.0.0.1/zabbix/api_jsonrpc.php
```

Response:

```
{
    "jsonrpc":"2.0",
    "result":[
        {
            "hostid":"10126",
            "proxy_hostid":"0",
            "host":"Zabbix server",
            "status":"0",
            "disable_until":"0",
            ...
            "name":"Zabbix server",
            "flags":"0",
            "templateid":"0",
            "description":""
        }
    ],
    "id":1
}
```

Filtering - Option 2. From command line

Use "filter" to return only those results that exactly match the given filter

Use "output" parameter to query specific object property:

Request via curl:

```
$ curl -X POST -H 'Content-Type:
application/json' -d'
{"jsonrpc":"2.0",
  "method":"host.get",
  "params":{
      "filter":{
          "host":"Zabbix server"
      },
      "output": "hostid"
  },
  "auth":"2f2ec4720863281c34cdd3c4c8a5de46",
  "id":1}
'
http://192.168.7.105/zabbix/api_jsonrpc.php
```

Response:

```
{
    "jsonrpc":"2.0",
    "result":[
            {"hostid":"10084"}
    ],
    "id":1
}
```

Filtering – Option 2. From command line

› Parameters accept arrays (begins with [and ends with]):

```
$ curl -X POST -H 'Content-Type: application/json' -d'
{"jsonrpc":"2.0",
  "method":"host.get",
  "params":{
      "filter":{
          "host": [
                "Zabbix server",
                "MySQL 01"
          ]
      },
      "output": [
                "hostid",
                "status"
      ]
  },
  "auth":"2f2ec4720863281c34cdd3c4c8a5de46",
  "id":1}
' http://192.168.7.105/zabbix/api_jsonrpc.php
```

Response:

```
{
    "jsonrpc":"2.0",
    "result":[

{"hostid":"10084","status":"0"},

{"hostid":"10627","status":"1"}
    ],
    "id":1
}
```

Zabbix API – Option 3. From Python

```
Install PyZabbix Python library using pip:
# yum install python-pip
# pip install pyzabbix
```

```
Get auth & host via custom script:
#!/usr/bin/env python
from pyzabbix import ZabbixAPI

zapi = ZabbixAPI("http://127.0.0.1/zabbix")
zapi.login("Admin", "z@hg*1aD")

result = zapi.host.get(filter={"host" : "Zabbix server"})
for h in result:
        for key in sorted(h):
                print "%s: %s " % (key, h[key])
```

```
$ ./host_get.py
available: 1
description:
disable_until: 0
error:
errors_from: 0
flags: 0
host: Zabbix server
hostid: 10126
...
snmp_available: 0
snmp_disable_until: 0
snmp_error:
snmp_errors_from: 0
status: 0
templateid: 0
```

Zabbix API – Option 4. From Powershell

```
For ($i=1; $i -le 1000; $i++) {
    $host_str = "TEST_API_" + $i;
    $item = "Test_item." + $i;
    $JsonItem ='{"jsonrpc": "2.0",
        "method": "host.create",
        "params": { "host": "' + $host_str +'",
            "groups": [ { "groupid": "19" } ] },
        "auth": "' + $auth + '",
        "id": 1}'

Invoke-RestMethod $ZabbixServer -ContentType "application/json-rpc; charset=utf-8" -Method Post
-Body ([System.Text.Encoding]::UTF8.GetBytes($jsonItem))
```

Zabbix API – Option 5. From Javascript

```javascript
var request = new HttpRequest(),
auth = "",
params = JSON.parse(value);

//Parse input params
apiToken = params.apiToken;
hostName = params.hostName;
Zabbix_URL = params.URL + "/api_jsonrpc.php";
hostID = params.ID;

//Construct JSON request
json_request = '{ "jsonrpc": "2.0", "method": "hostinterface.get", \
        "params": { "hostids":"' + hostId + '", "output": "extend", \
        "filter" :{ "main":"1","type":"1"}\
    }, "auth": "' + apiToken +'", "id": 1 }';
iface_result =  request.post(Zabbix_URL,json_request);

return iface_result;
```

Zabbix API

Result:

```
{
    "jsonrpc":"2.0",
    "result":[
        {
            "hostid":"10084",
            "proxy_hostid":"0",
            "host":"Zabbix server",
            "status":"0",
            "available":"1",
            ...
            "name":"Zabbix server",
            "flags":"0",
            "templateid":"0",
            "description":""
        }
    ],
    "id":1
}
```

What does "status: 0" or "available: 1" mean?

Use API Documentation:

status	integer	Status and function of the host. Possible values are: 0 - (default) monitored host; 1 - unmonitored host.

available	integer	(readonly) Availability of Zabbix agent. Possible values are: 0 - (default) unknown; 1 - available; 2 - unavailable.

Zabbix 7.0

history.push

› allows to send item history data to Zabbix server via API (you don't need Zabbix sender anymore in some situation)

› receiving sent data requires a configured trapper item or an HTTP agent item (with trapping enabled)

› This method is available to users of any type. Permissions to call the method can be revoked in user role settings. See User roles for more information.

```
{
     "jsonrpc": "2.0",
     "method": "history.push",
     "params": [
         {
             "itemid": 10600,
             "value": 0.5,
             "clock": 1690891294,
             "ns": 45440940
         }
     ]
}
```

5

API features

API vs DATABASE

Getting host from a database.

› In SQL, it would look like this:

› SELECT * FROM hosts WHERE host = 'Zabbix server'\G

Great, but what should I use? - **API!**

```
hostid: 10084
proxy_hostid: NULL
host: Zabbix server
status: 0
disable_until: 0
error:
available: 1
errors_from: 0
lastaccess: 0
ipmi_authtype: -1
ipmi_privilege: 2
ipmi_username:
jmx_disable_until: 0
jmx_available: 0
jmx_errors_from: 0
jmx_error:
…
name: Zabbix server
proxy_address:
```

ZABBIX API security

- SSL
- Password authentication, API keys
- Respects permissions
- Audit

ZABBIX API security

Backward compatibility

Rollback operation in case of an error
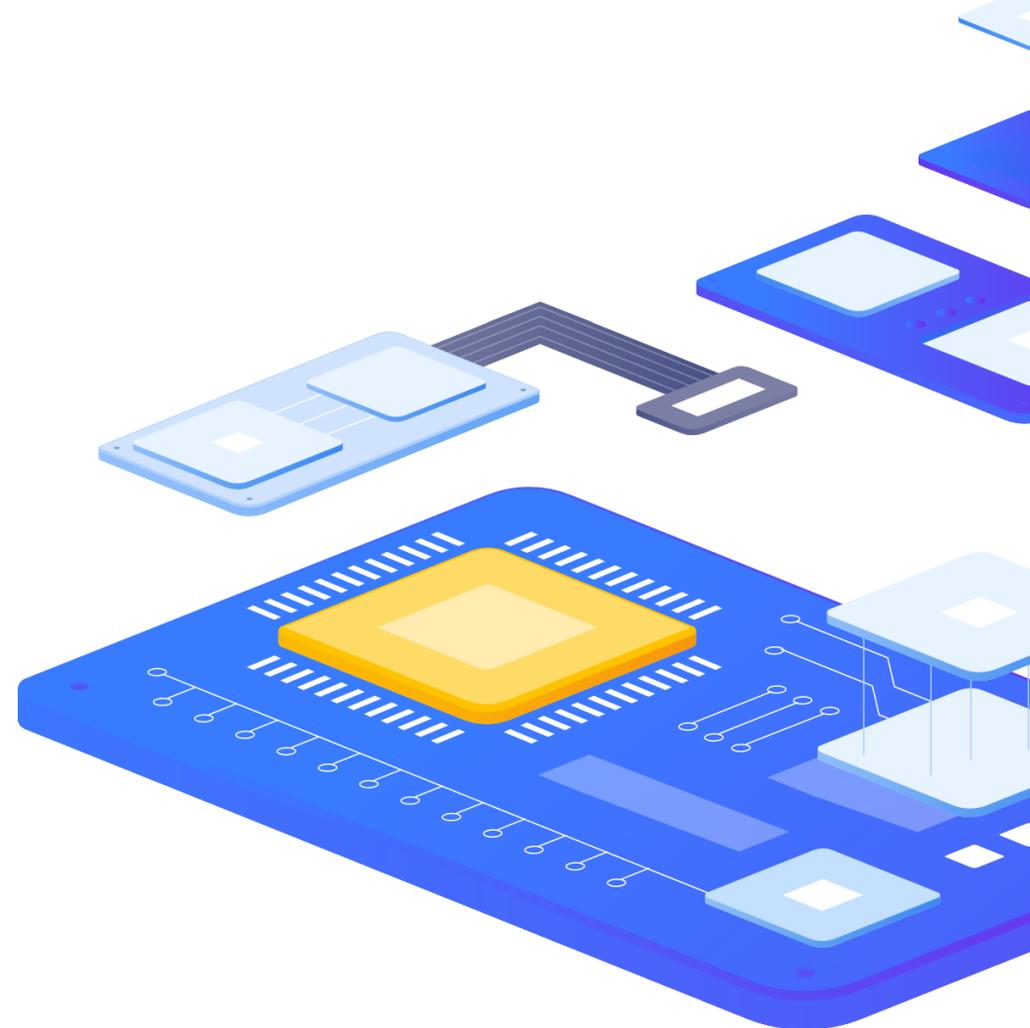
ZABBIX API Performance

- DB performance issues
- Heavy requests/huge number of objects
- Old Zabbix version

Documentation

Full list of API methods is available in ZABBIX documentation
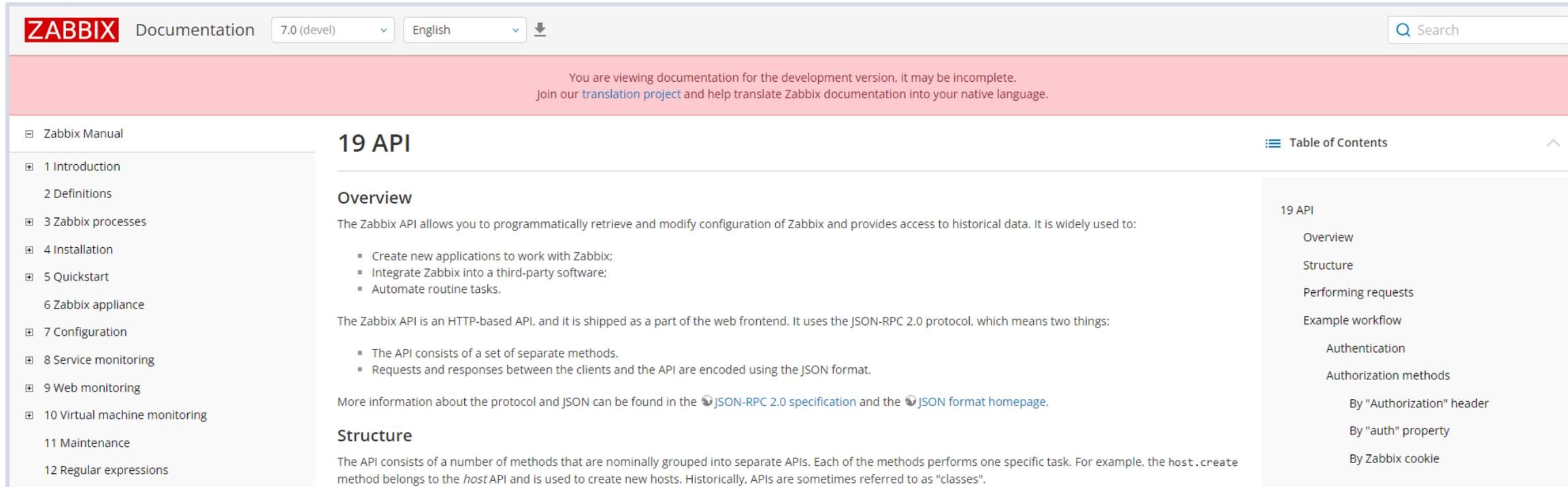
initMAX

ZABBIX Documentation   7.0 (devel) ⌄   English ⌄ ⬇   🔍 Search

You are viewing documentation for the development version, it may be incomplete.
Join our translation project and help translate Zabbix documentation into your native language.

Zabbix Manual

⊟ Zabbix Manual
  ⊞ 1 Introduction
    2 Definitions
    3 Zabbix processes
  ⊞ 4 Installation
  ⊞ 5 Quickstart
    6 Zabbix appliance
  ⊞ 7 Configuration
  ⊞ 8 Service monitoring
  ⊞ 9 Web monitoring
  ⊞ 10 Virtual machine monitoring
    11 Maintenance
    12 Regular expressions

19 API

≣ Table of Contents ⌃

Overview

The Zabbix API allows you to programmatically retrieve and modify configuration of Zabbix and provides access to historical data. It is widely used to:

- Create new applications to work with Zabbix;
- Integrate Zabbix into a third-party software;
- Automate routine tasks.

The Zabbix API is an HTTP-based API, and it is shipped as a part of the web frontend. It uses the JSON-RPC 2.0 protocol, which means two things:

- The API consists of a set of separate methods.
- Requests and responses between the clients and the API are encoded using the JSON format.

More information about the protocol and JSON can be found in the 🌐 JSON-RPC 2.0 specification and the 🌐 JSON format homepage.

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".

6

Questions?

ZABBIX API

Contact us:

Phone:	>	+420 800 244 442
Web:	>	https://www.initmax.cz
Email:	>	tomas.hermanek@initmax.cz
LinkedIn:	>	https://www.linkedin.com/company/initmax
Twitter:	>	https://twitter.com/initmax
Tomáš Heřmánek:	>	+420 732 447 184

initMAX