



ZABBIX 5.0

ZABBIX API – LEVEL 1

MESSAGE TO SPREAD

- ✓ Everything is easy like copy and paste
- ✓ Documentation page is your friend
- ✓ Introduce what is possible

REST API CLIENT

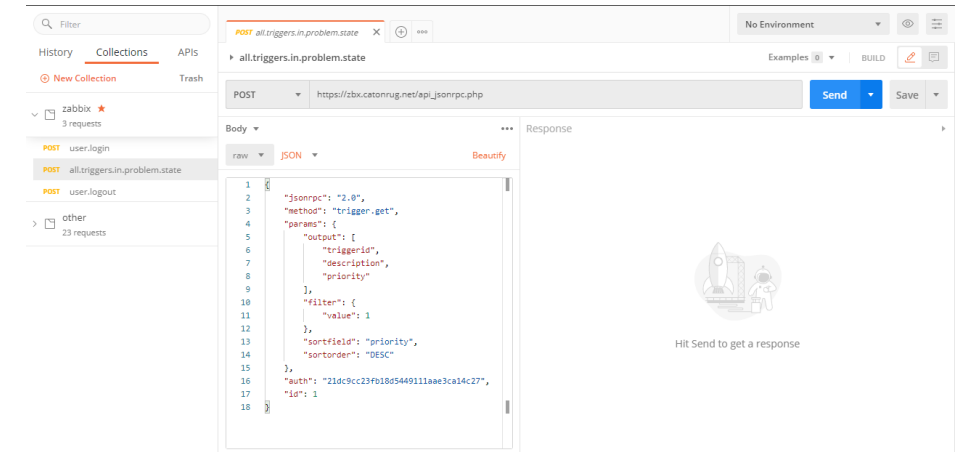
GUI to perform API calls



POSTMAN

<https://www.postman.com/downloads/>

- ✓ Available on Windows, Linux, Mac
- ✓ Store all your snippets to Google account
- ✓ Use official Zabbix documentation page



OBTAIN SESSION TOKEN

Filter

HistoryCollectionsAPIs

New CollectionTrash

zabbix★
3 requests

POST user.login

POST all.triggers.in.problem.state

POST user.logout

other
23 requests

POST user.login

user.login

POSThttps://zbx.catonrug.net/api_jsonrpc.php

Body

rawJSONBeautify

```
1 {
2   "jsonrpc": "2.0",
3   "method": "user.login",
4   "params": {
5     "user": "api",
6     "password": "zabbix"
7   },
8   "id": 1
9 }
```

No Environment

Examples 0BUILD

SendSave

Body

PrettyRawPreviewVisualizeJSON

```
1 {
2   "jsonrpc": "2.0",
3   "result": "21dc9cc23fb18d5449111aae3ca14c27",
4   "id": 1
5 }
```

200 OK239 ms680 BSave Response

PICK A FLAVOR FROM DOCUMENTATION PAGE

Filter

HistoryCollectionsAPITrash

New Collection

zabbix3 requests

POST user.login

POST all.triggers.in.problem.state

POST user.logout

other23 requests

POST all.triggers.in.problem.state

all.triggers.in.problem.state

POST

https://zbx.catonrug.net/api_jsonrpc.php

Send

Save

Body

rawJSONBeautify

```
1 {
2   "jsonrpc": "2.0",
3   "method": "trigger.get",
4   "params": {
5     "output": [
6       "triggerid",
7       "description",
8       "priority"
9     ],
10    "filter": {
11      "value": 1
12    },
13    "sortfield": "priority",
14    "sortorder": "DESC"
15  },
16  "auth": "21dc9cc23fb18d5449111aae3ca14c27",
17  "id": 1
18 }
```

Response



Hit Send to get a response

RESPONSE

Filter

HistoryCollectionsAPIS

New CollectionTrash

zabbix

3 requests

POST

user.login

POST

all.triggers.in.problem.state

POST

user.logout

other

23 requests

POST

all.triggers.in.problem.state

Examples 0BUILD

POSThttps://zbx.catonrug.net/api_jsonrpc.phpSendSave

Body

rawJSONBeautify

```
1 {
2   "jsonrpc": "2.0",
3   "method": "trigger.get",
4   "params": {
5     "output": [
6       "triggerid",
7       "description",
8       "priority"
9     ],
10    "filter": {
11      "value": 1
12    },
13    "sortfield": "priority",
14    "sortorder": "DESC"
15  },
16  "auth": "21dc9cc23fb18d5449111aae3ca14c27",
17  "id": 1
18 }
```

Body

PrettyRawPreviewVisualizeJSON

200 OK195 ms23.3 KBSave Response

```
1 {
2   "jsonrpc": "2.0",
3   "result": [
4     {
5       "triggerid": "435701",
6       "description": "Internal event for itemid where the item does
7         not exist anymore",
8       "priority": "4"
9     },
10    {
11      "triggerid": "659246",
12      "description": "{ITEM.LASTVALUE1} body errors found on Things to
13        change on MX Linux 19, Debian 10",
14      "priority": "4"
15    },
16    {
17      "triggerid": "727316",
18      "description": "{ITEM.LASTVALUE1} body errors found on Capture
19        all lines from pattern till end, regex",
20      "priority": "4"
21    }
22  ]
23 }
```


MANAGE ENVIRONMENTS

Filter

HistoryCollectionsAPIs

+ New CollectionTrash

zabbix★
3 requests

POST user.login

POST all.triggers.in.problem.state

POST user.logout

other
23 requests

POSTall.triggers.in.problem.state

No Environment

Examples 0Manage Environments

POSThttps://zbx.catonrug.net/api_jsonrpc.php

SendSave

Body

rawJSONBeautify

```
1 {
2   "jsonrpc": "2.0",
3   "method": "trigger.get",
4   "params": {
5     "output": [
6       "triggerid",
7       "description",
8       "priority"
9     ],
10    "filter": {
11      "value": 1
12    },
13    "sortfield": "priority",
14    "sortorder": "DESC"
15  },
16  "auth": "21dc9cc23fb18d5449111aae3ca14c27",
17  "id": 1
18 }
```

Body

PrettyRawPreviewVisualizeJSON

```
1 {
2   "jsonrpc": "2.0",
3   "result": [
4     {
5       "triggerid": "435701",
6       "description": "Internal event for itemid where the item does
7         not exist anymore",
8       "priority": "4"
9     },
10    {
11      "triggerid": "659246",
12      "description": "{ITEM.LASTVALUE1} body errors found on Things to
13        change on MX Linux 19, Debian 10",
14      "priority": "4"
15    },
16    {
17      "triggerid": "727316",
18      "description": "{ITEM.LASTVALUE1} body errors found on Capture
19        all lines from pattern till end, regex",
20      "priority": "4"
21    }
22  ]
23 }
```

200 OK241 ms23.3 KBSave Response

CREATE AN ENVIRONMENT

The screenshot shows the Postman application interface. On the left, there's a sidebar with a 'Filter' search bar, tabs for 'History', 'Collections', and 'APIs', and a 'New Collection' button. Below these, there are two collections: 'zabbix' (3 requests) and 'other' (23 requests). The main area displays a 'POST' request to 'all.triggers.in.problem.state'. A modal window titled 'MANAGE ENVIRONMENTS' is open in the center. The modal contains the following text:

MANAGE ENVIRONMENTS

An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. [Learn more about environments](#)

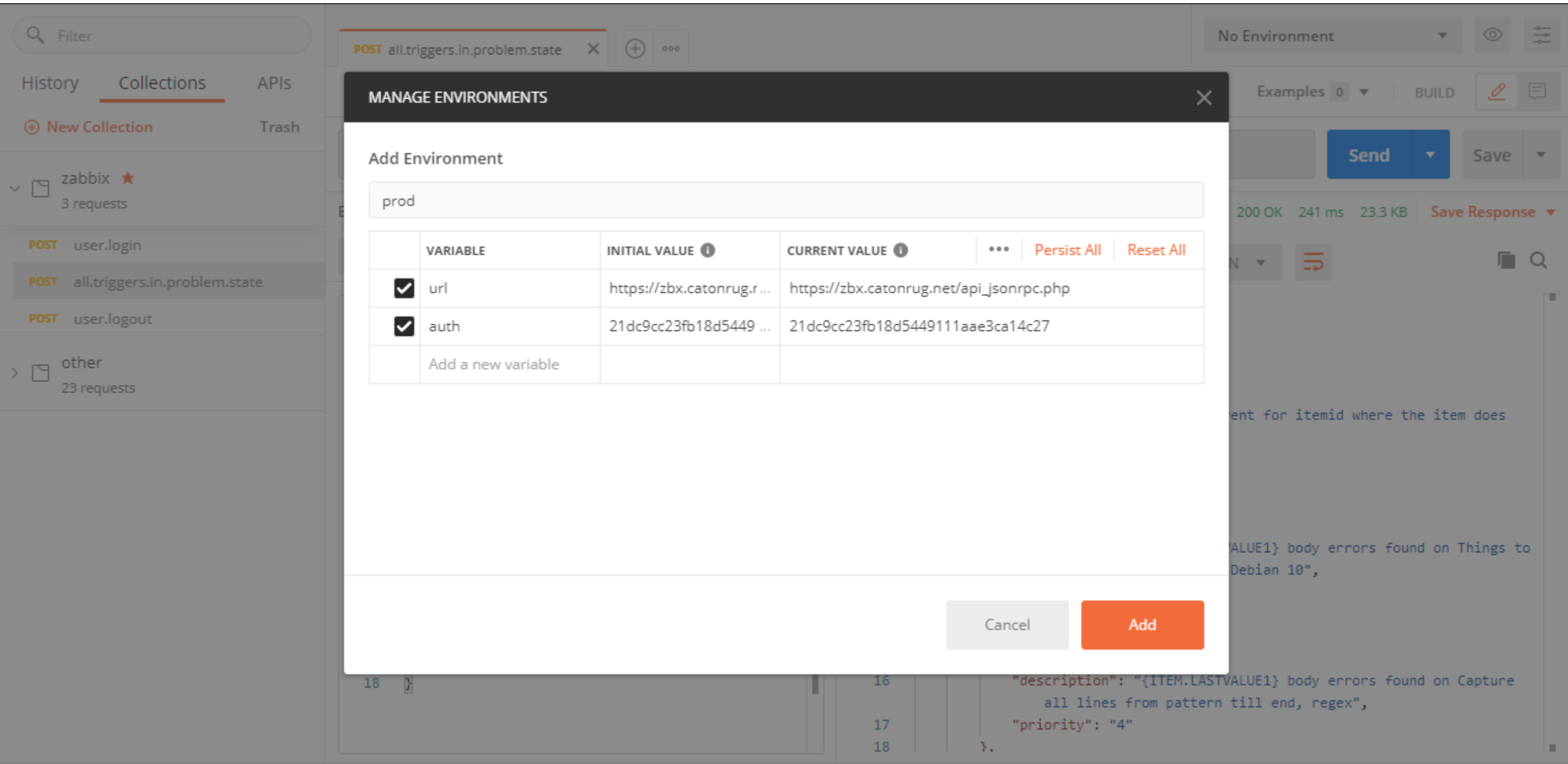
You can declare a variable in an environment and give it a starting value, then use it in a request by putting the variable name within curly-braces. [Create an environment](#) to get started.

At the bottom of the modal, there are three buttons: 'Globals', 'Import', and 'Add'.

In the background, the right sidebar shows 'No Environment' selected, 'Examples: 0', a 'BUILD' button, and a 'Send' button. Below these, there's a status bar showing '200 OK', '241 ms', and '23.3 KB', along with a 'Save Response' button. The bottom of the screen shows a JSON response snippet:

```
18 }
16 "description": "{ITEM.LASTVALUE1} body errors found on Capture
17 all lines from pattern till end, regex",
18 "priority": "4"
```

CREATE ENVIRONMENT CHARACTERISTICS



The screenshot displays the Postman application interface. In the foreground, a 'MANAGE ENVIRONMENTS' dialog box is open, titled 'Add Environment'. The environment name 'prod' is entered in the top text field. Below this is a table with columns: 'VARIABLE', 'INITIAL VALUE', and 'CURRENT VALUE'. Two variables are listed: 'url' and 'auth', both with checkboxes in the first column. The 'url' variable has an initial value of 'https://zbx.catonrug.r...' and a current value of 'https://zbx.catonrug.net/api_jsonrpc.php'. The 'auth' variable has an initial value of '21dc9cc23fb18d5449 ...' and a current value of '21dc9cc23fb18d5449111aae3ca14c27'. At the bottom of the table is a row labeled 'Add a new variable'. To the right of the table are three dots and two buttons: 'Persist All' and 'Reset All'. At the bottom of the dialog are 'Cancel' and 'Add' buttons.

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
☒	url	https://zbx.catonrug.r ...	https://zbx.catonrug.net/api_jsonrpc.php			
☒	auth	21dc9cc23fb18d5449 ...	21dc9cc23fb18d5449111aae3ca14c27			
	Add a new variable					

PROD, TEST, 4.0, 5.0

The image shows the Postman application interface with the 'MANAGE ENVIRONMENTS' dialog box open. The dialog box has a dark header with the title 'MANAGE ENVIRONMENTS' and a close button. Below the header, there is a descriptive text: 'An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. [Learn more about environments](#)'. Below this text, there is a list of environments. The first environment is named 'prod'. To the right of the 'prod' environment name, there are three buttons: 'Share', 'Copy', and 'More options'. At the bottom of the dialog box, there are three buttons: 'Globals', 'Import', and 'Add'. The background of the image shows the Postman interface with a sidebar on the left containing 'History' and 'Collections' tabs. The 'Collections' tab is active, showing a list of collections: 'zabbix' (3 requests), 'user.login', 'all.triggers.in.problem.state', 'user.logout', and 'other' (23 requests). The main area of the interface shows a request editor for a POST request to 'all.triggers.in.problem.state'. The response area shows a 200 OK status with a response time of 241 ms and a size of 23.3 KB. The response body is a JSON object with a 'description' field containing a message about body errors found on Things to Debian 10, and a 'priority' field with the value '4'.

Filter

History Collections APIs

+ New Collection Trash

zabbix ★
3 requests

POST user.login

POST all.triggers.in.problem.state

POST user.logout

other
23 requests

POST all.triggers.in.problem.state

No Environment

Examples 0 BUILD

Send Save

200 OK 241 ms 23.3 KB Save Response

ent for itemid where the item does

VALUE1} body errors found on Things to Debian 10",

18 }.

16 "description": "{ITEM.LASTVALUE1} body errors found on Capture
all lines from pattern till end, regex",
17 "priority": "4"
18 },.

MANAGE ENVIRONMENTS

An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. [Learn more about environments](#)

prod

Share Copy More options

Globals Import Add

SET ENVIRONMENT, CONFIGURE DYNAMICS

The screenshot displays the Postman application interface. On the left, a sidebar shows a 'Collections' tab with a collection named 'zabbix' containing 3 requests. The main workspace shows a REST client request for the endpoint `all.triggers.in.problem.state` with a `POST` method. The request body is a JSON object. A dropdown menu is open in the top right corner, showing the environment selection options: 'No Environment' and 'prod'. The response pane on the right shows a `200 OK` status with a response body containing a list of trigger details.

Request:

```
POST all.triggers.in.problem.state
{
  "jsonrpc": "2.0",
  "method": "trigger.get",
  "params": {
    "output": [
      "triggerid",
      "description",
      "priority"
    ],
    "filter": {
      "value": 1
    },
    "sortfield": "priority",
    "sortorder": "DESC"
  },
  "auth": "{{auth}}",
  "id": 1
}
```

Response:

```
200 OK 184 ms 23.3 KB
{
  "jsonrpc": "2.0",
  "result": [
    {
      "triggerid": "435701",
      "description": "Internal event for itemid where the item does not exist anymore",
      "priority": "4"
    },
    {
      "triggerid": "659246",
      "description": "{ITEM.LASTVALUE1} body errors found on Things to change on MX Linux 19, Debian 10",
      "priority": "4"
    },
    {
      "triggerid": "727316",
      "description": "{ITEM.LASTVALUE1} body errors found on Capture all lines from pattern till end, regex",
      "priority": "4"
    }
  ]
}
```

API scripting

An abstract digital graphic on a dark blue background. It features a glowing red ring with a grid-like texture, surrounded by streams of blue and white binary code (0s and 1s) and light particles. The elements appear to be flowing and interacting, creating a sense of dynamic energy and data processing.

via «bash»

Samples = cheat sheet

PREREQUISITES

«curl» available out from box for most of platforms

«jq» provides conveniences to locate element or just beautifully output. Install on:

CentOS7/RHEL7

```
yum install epel-release && yum install jq
```

CentOS8/RHEL8

```
dnf install jq
```

Ubuntu/Debian

```
apt install jq
```

Standalone 64-bit binary on any Linux:

```
curl -skL "https://github.com/stedolan/jq/releases/download/jq-1.5/jq-linux64" -o /usr/bin/jq && chmod +x /usr/bin/jq
```

CREATE A QUERY

Visit documentation page . Select API flavor.	Escape double quotes. Do even think to do it manually!	Replace: 038e1d7b1735c6a5436ee9eae095879e with: \$auth
<pre>{ "jsonrpc": "2.0", "method": "alert.get", "params": { "output": "extend", "actionids": "3" }, "auth": "038e1d7b1735c6a5436ee9eae095879e", "id": 1 }</pre>	Replace Find in Files Mark Find what : Replace with :	<pre>curl -s -X POST \ -H 'Content-Type: application/json-rpc' \ -d " \ { \"jsonrpc\": \"2.0\", \"method\": \"alert.get\", \"params\": { \"output\": \"extend\", \"actionids\": \"3\" }, \"auth\": \"\$auth\", \"id\": 1 } " \$url</pre>

Always execute step 2 at first and only then: step 3

Notice 2 double quotes are not escaped

BASH EXAMPLE

```
#!/bin/bash
```

1. set connection details

```
url=http://127.0.0.1/api_jsonrpc.php
user=api
password=zabbix
```

2. get authorization token

```
auth=$(curl -s -X POST \
-H 'Content-Type: application/json-rpc' \
-d " \
{
  \"jsonrpc\": \"2.0\",
  \"method\": \"user.login\",
  \"params\": {
    \"user\": \"$user\",
    \"password\": \"$password\"
  },
  \"id\": 1,
  \"auth\": null
}
" $url | \
jq -r '.result'
)
```

3. show triggers in problem state

```
curl -s -X POST \
-H 'Content-Type: application/json-rpc' \
-d " \
{
  \"jsonrpc\": \"2.0\",
  \"method\": \"trigger.get\",
  \"params\": {
    \"output\": \"extend\",
    \"selectHosts\": \"extend\",
    \"filter\": {
      \"value\": 1
    },
    \"sortfield\": \"priority\",
    \"sortorder\": \"DESC\"
  },
  \"auth\": \"$auth\",
  \"id\": 1
}
" $url | \
jq -r '.result'
```

4. logout user

```
curl -s -X POST \
-H 'Content-Type: application/json-rpc' \
-d " \
{
  \"jsonrpc\": \"2.0\",
  \"method\": \"user.logout\",
  \"params\": [],
  \"id\": 1,
  \"auth\": \"$auth\"
}
" $url
```

API task starts with 'user.login' procedure and ends with 'user.logout' procedure

LOCATE ELEMENTS - JSONPATHFINDER.COM

input

```
curl -s -X POST \
-H 'Content-Type: application/json-rpc' \
-d " \
{
  \"jsonrpc\": \"2.0\",
  \"method\": \"proxy.get\",
  \"params\": {
    \"output\": [\"host\"]
  },
  \"auth\": \"$auth\",
  \"id\": 1
}
" $url
```

output

```
{"jsonrpc": "2.0", "result": [{"host": "broceni", "proxyid": "10387"}, {"host": "mysql8mon", "proxyid": "12066"}, {"host": "riga", "proxyid": "12585"}], "id": 1}
```

The screenshot shows the JSON Path Finder web application interface. The left panel displays the input JSON with line numbers 1 through 18. The right panel shows the output JSON with the path `x.result[1].host` highlighted in green. The output JSON is also displayed below the path input field.

JSON Path Finder

Path: `x.result[1].host` Copy

jsonrpc: 2.0

result:

- 0:
- 1: **host: mysql8mon** 3
proxyid: 12066
- 2:

id: 1

LOCATE ELEMENTS - JSONPATHFINDER.COM

input

```
curl -s -X POST \  
-H 'Content-Type: application/json-rpc' \  
-d " \  
{  
  \"jsonrpc\": \"2.0\",  
  \"method\": \"proxy.get\",  
  \"params\": {  
    \"output\": [\"host\"]  
  },  
  \"auth\": \"$auth\",  
  \"id\": 1  
}  
" $url | jq -r '.result[].host'
```

output

```
broceni  
mysql8mon  
riga
```

The screenshot shows the JSON Path Finder website interface. The browser address bar displays <https://jsonpathfinder.com>. The page title is "JSON Path Finder".

The input JSON is displayed in a code editor on the left, with line numbers 1 through 18. The JSON structure is as follows:

```
{  
  "jsonrpc": "2.0",  
  "result": [  
    {  
      "host": "broceni",  
      "proxyid": "10387"  
    },  
    {  
      "host": "mysql8mon",  
      "proxyid": "12066"  
    },  
    {  
      "host": "riga",  
      "proxyid": "12585"  
    }  
  ],  
  "id": 1  
}
```

The output is displayed on the right side of the interface. The "Path" field is set to `x.result[1].host`, and the "Copy" button is visible. The output shows the following results:

```
jsonrpc: 2.0  
result:  
  0:  
  1:  
    host: mysql8mon  
    proxyid: 12066  
  2:  
id: 1
```

Three green circles with numbers 1, 2, and 3 are overlaid on the image to highlight specific elements:

- Circle 1 points to the `"host": "mysql8mon"` entry in the `result` array of the input JSON.
- Circle 2 points to the `Beautify` button in the top right of the code editor.
- Circle 3 points to the `host: mysql8mon` entry in the output results.

Popular use cases



Use case 1: Log file monitoring SNMP traps



CHARACTERISTICS

Over 600 patterns to detect

Everything is stored in one item

Event generation mode: Multiple

No recovery logic

Ambition:

Notify person and close event

OPERATIONS

ZABBIX

Monitoring

Inventory

Reports

Configuration

Host groups

Templates

Hosts

Maintenance

Actions

Event correlation

Discovery

Services

Administration

Support

Share

Help

Actions

Action

Operations

* Default operation step duration

1h

Pause operations for suppressed problems

☒

Operations

Steps	Details	Start in	Duration	Action
1	Send message to users: Admin (Zabbix Administrator) via all media	Immediately	60	Edit Remove
2	Run remote commands on current host	00:01:00	Default	Edit Remove
Add				

Recovery operations

Details

Action

Update operations

Details

Action

* At least one operation must exist.

Update

Clone

Delete

Cancel

REMOTE COMMAND

ZABBIX

Monitoring

Inventory

Reports

Configuration

Host groups

Templates

Hosts

Maintenance

Actions

Event correlation

Discovery

Services

Administration

Support

Share

Help

Actions

Operation details

Operation type Remote command

Steps 2 - 2 (0 - infinitely)

Step duration 0 (0 - use action default)

* Target list

Current host ☒

Host type here to search Select

Host group type here to search Select

Type Custom script

Execute on Zabbix agent Zabbix server (proxy) Zabbix server

* Commands

url={\$Z_API_PHP}
user={\$Z_API_USER}
password={\$Z_API_PASSWORD}

authorization
auth=\$(curl -sk -X POST -H "Content-Type: application/json"
-d "
Conditions

Label Name Action

Add

Update Cancel

Condition Action

Edit Remove

ult Edit Remove

Zabbix 5.0.3. © 2001-2020, Zabbix SIA

SOLUTION IN DETAIL

Macro	Value
{Z_API_PHP}	http://127.0.0.1/api_jsonrpc.php
{Z_API_PASSWORD}	zabbix
{Z_API_USER}	api

characteristics to access API

```
url={Z_API_PHP}
user={Z_API_USER}
password={Z_API_PASSWORD}
```

authorization

```
auth=$(curl -sk -X POST -H "Content-Type:
application/json" -d "
{
    \"jsonrpc\": \"2.0\",
    \"method\": \"user.login\",
    \"params\": {
        \"user\": \"$user\",
        \"password\": \"$password\"
    },
    \"id\": 1,
    \"auth\": null
}
" $url | \
grep -E -o "([0-9a-f]{32,32})")
```

acknowledge and close event

```
curl -sk -X POST -H "Content-Type: application/json" -d "
{
    \"jsonrpc\": \"2.0\",
    \"method\": \"event.acknowledge\",
    \"params\": {
        \"eventids\": \"{EVENT.ID}\",
        \"action\": 1,
        \"message\": \"Problem resolved.\"
    },
    \"auth\": \"$auth\",
    \"id\": 1
}
" $url
```

close api key

```
curl -sk -X POST -H "Content-Type: application/json" -d "
{
    \"jsonrpc\": \"2.0\",
    \"method\": \"user.logout\",
    \"params\": [],
    \"id\": 1,
    \"auth\": \"$auth\"
}
" $url
```

Use case 2:
Maintain global macro
via API



CHARACTERISTICS

Have managed a way to obtain session token

Will use it in "HTTP agent" item

Ambition:

Maintain session token in global macro

SOLUTION: ZABBIX_GLOBAL_MACRO_UPDATE.SH

```
#!/bin/bash

# characteristics to access API
url=http://127.0.0.1/api_jsonrpc.php
user=Admin
password=zabbix

# argument 1
macro=$1

# argument 2
value=$2

# get authorization token
auth=$(curl -s -X POST \
-H 'Content-Type: application/json-rpc' \
-d " \
{
  \"jsonrpc\": \"2.0\",
  \"method\": \"user.login\",
  \"params\": {
    \"user\": \"$user\",
    \"password\": \"$password\"
  },
  \"id\": 1,
  \"auth\": null
}
" $url | grep -E -o "([0-9a-f]{32,32})")

# get global user macro id
id=$(curl -s -X POST \
-H 'Content-Type: application/json-rpc' \
-d " \
{
  \"jsonrpc\": \"2.0\",
  \"method\": \"usermacro.get\",
  \"params\": {
    \"output\": [\"globalmacroid\"],
    \"globalmacro\": true,
    \"filter\": {\"macro\": \"$macro\"}
  },
  \"auth\": \"$auth\",
  \"id\": 1
}
" $url | jq -r ".result[].globalmacroid")

# update
curl -s -X POST \
-H 'Content-Type: application/json-rpc' \
-d " \
{
  \"jsonrpc\": \"2.0\",
  \"method\": \"usermacro.updateglobal\",
  \"params\": {
    \"globalmacroid\": \"$id\",
    \"value\": \"$value\"
  },
  \"auth\": \"$auth\",
  \"id\": 1
}
" $url

# logout user
curl -s -X POST \
-H 'Content-Type: application/json-rpc' \
-d " \
{
  \"jsonrpc\": \"2.0\",
  \"method\": \"user.logout\",
  \"params\": [],
  \"id\": 1,
  \"auth\": \"$auth\"
}
" $url
```

ZABBIX_GLOBAL_MACRO_UPDATE.SH: SAMPLE

```
zabbix_global_macro_update.sh '{$TIME.ZONE.VAR}' "Europe/Riga (EEST, +0300)"
```

```
zabbix_global_macro_update.sh '{$TIME.ZONE.VAR}' "${timedatectl | grep -oP "(Time zone: \K.*$)"}"
```

```
zabbix_global_macro_update.sh '{$TIME.ZONE.VAR}' "${zabbix_show_local_timezone.sh}"
```

Use case 3:
Zabbix API key
in a global level



DEFINE MACRO AT GLOBAL LEVEL

ZABBIX << 

 Monitoring

 Inventory

 Reports

 Configuration

 Administration

General

Proxies

Authentication

Macros [Add](#)

Update

```
curl -s -X POST -H 'Content-Type: application/json' http://127.0.0.1/api_jsonrpc.php -d \
'{"jsonrpc": "2.0", "method": "user.login", "params": {"user": "api", "password": "zabbix"}, "id": 1, "auth": null}' | \
grep -E -o "([0-9a-f]{32,32})"
```

CASE 1: IMPROVED

```
# characteristics to access API
url=${Z_API_PHP}
user=${Z_API_USER}
password=${Z_API_PASSWORD}

# authorization
auth=$(curl -sk -X POST -H "Content-Type:
application/json" -d "
{
  \"jsonrpc\": \"2.0\",
  \"method\": \"user.login\",
  \"params\": {
    \"user\": \"$user\",
    \"password\": \"$password\"
  },
  \"id\": 1,
  \"auth\": null
}
" $url | \
grep -E -o "([0-9a-f]{32,32})")
```

No need to escape double quotes anymore
Original sample still is more bulletproof

```
# acknowledge and close event
curl -sk -X POST -H 'Content-Type: application/json' -d '
{
  "jsonrpc": "2.0",
  "method": "event.acknowledge",
  "params": {
    "eventids": "{EVENT.ID}",
    "action": 1,
    "message": "Problem resolved."
  },
  "auth": "${Z_API_SESSIONID}",
  "id": 1
}
' ${Z_API_PHP}

# close api key
curl -sk -X POST -H "Content-Type: application/json" -d "
{
  \"jsonrpc\": \"2.0\",
  \"method\": \"user.logout\",
  \"params\": [],
  \"id\": 1,
  \"auth\": \"$auth\"
}
" $url
```


Action Operations

* Default operation step

Pause operations for suppressed p

Op

Recovery op

Update op

Operation details

Operation type Remote command

Steps 2 - 2 (0 - infinitely)

Step duration 0 (0 - use action default)

* Target list

Current host ☒

Host

Host group

Type Custom script

Execute on Zabbix agent Zabbix server (proxy) Zabbix server

* Commands

```
# acknowledge and close event
curl -sk -X POST -H 'Content-Type: application/json' -d '
{
  "jsonrpc": "2.0",
  "method": "event.acknowledge",
  "params": {
    "eventids": "{EVENT.ID}",
    "action": 1,
    "message": "Problem resolved."
  },
  "auth": "{Z_API_SESSIONID}",
  "id": 1
}' {Z_API_PHP}
```

Conditions

Label	Name	Action
Add		

on Action

[Edit](#) [Remove](#)

t [Edit](#) [Remove](#)

USE Case 4: API through HTTP Agent



DISCOVERY RULE

Discovery rule **Preprocessing** LLD macros Filters Overrides

* Name

Type

* Key

* URL

Request type

Timeout

Request body type

Request body

```
{
  "jsonrpc": "2.0",
  "method": "proxy.get",
  "params": {
    "output": "extend",
    "selectInterface": "extend"
  },
  "auth": "{ $Z_API_SESSIONID }",
  "id": 1
}
```

PREPROCESSING

Discovery rule **Preprocessing** LLD macros Filters Overrides

Preprocessing steps	Name	Parameters	Custom on fail	Action
1:	JSONPath	\$.result[*]	☐	Test
Add				
Update Clone Test Delete Cancel				

```
{
  "jsonrpc": "2.0",
  "result": [{
    "host": "broceni",
    "proxyid": "10387"
  }, {
    "host": "mysql8mon",
    "proxyid": "12066"
  }, {
    "host": "riga",
    "proxyid": "12585"
  }],
  "id": 1
}
```

MAPPING

Discovery rule Preprocessing **LLD macros** Filters Overrides

LLD macros

LLD macro	JSONPath	
{#PROXYNAME}	\$.host	<a data-bbox="1890 491 2002 522" href="#">Remove
<a data-bbox="545 571 614 608" href="#">Add		

Update

Clone

Test

Delete

Cancel

```
[{
  "host": "broceni",
  "proxyid": "10387"
}, {
  "host": "mysql8mon",
  "proxyid": "12066"
}, {
  "host": "riga",
  "proxyid": "12585"
}]
```

ITEM PROTOTYPE

[All templates](#) / [prx](#) [Discovery list](#) / [Discover all proxies](#) [Item prototypes 1](#) [Trigger prototypes 1](#) [Graph prototypes](#) [Host prototypes](#)

[Item prototype](#) [Preprocessing](#)

* Name

Type

* Key

Type of information

Units

* Update interval

TRIGGER PROTOTYPE

All templates / prx Discovery list / Discover all proxies Item prototypes 1 Trigger prototypes 1 Graph prototypes Host prototypes

Trigger prototype Tags Dependencies

* Name {#PROXYNAME} is not reachable for {\$PROXYFUZZYTIME}

Operational data

Severity

Not classified

Information

Warning

Average

High

Disaster

* Expression

```
{prx:zabbix[proxy,  
{#PROXYNAME},lastaccess].fuzzytime({$PROXYFUZZYTIME})}=0
```

Add

[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

Allow manual close

☐

USE Case 5: Delete inactive hosts



DELAYED ESCALATION

ZABBIX

<< 🔍

Monitoring

Inventory

Reports

Configuration

Host groups

Templates

Hosts

Maintenance

Actions

Event correlation

Discovery

Services

Administration

Support

Share

Help

Actions

Action

Operations

* Default operation step duration

1d

Pause operations for suppressed problems

☒

Operations

Steps	Details	Start in	Duration	Action
7	Send message to users: Admin (Zabbix Administrator) via all media	6 days, 00:00:00	Default	Edit Remove
8	Run remote commands on current host	7 days, 00:00:00	Default	Edit Remove
Add				

Recovery operations

Details

Action

[Add](#)

Update operations

Details

Action

[Add](#)

* At least one operation must exist.

Update

Clone

Delete

Cancel

Zabbix 5.0.3. © 2001–2020, Zabbix SIA

* Default operation step duration

Pause operations for suppressed problems

Operation

Recovery operation

Update operation

* At

Up

Operation details

Operation type Remote command

Steps 8 - 8 (0 - infinitely)

Step duration 0 (0 - use action default)

* Target list

Current host ☒

Host

Host group

Type Custom script

Execute on Zabbix agent Zabbix server (proxy) **Zabbix server**

* Commands

```
# delete host
curl -ks -X POST -H 'Content-Type: application/json-rpc' -d '
{
  "jsonrpc": "2.0",
  "method": "host.delete",
  "params": [ {HOST.ID} ],
  "auth": "{$Z_API_SESSIONID}",
  "id": 1
}' {$Z_API_PHP}
```

Conditions

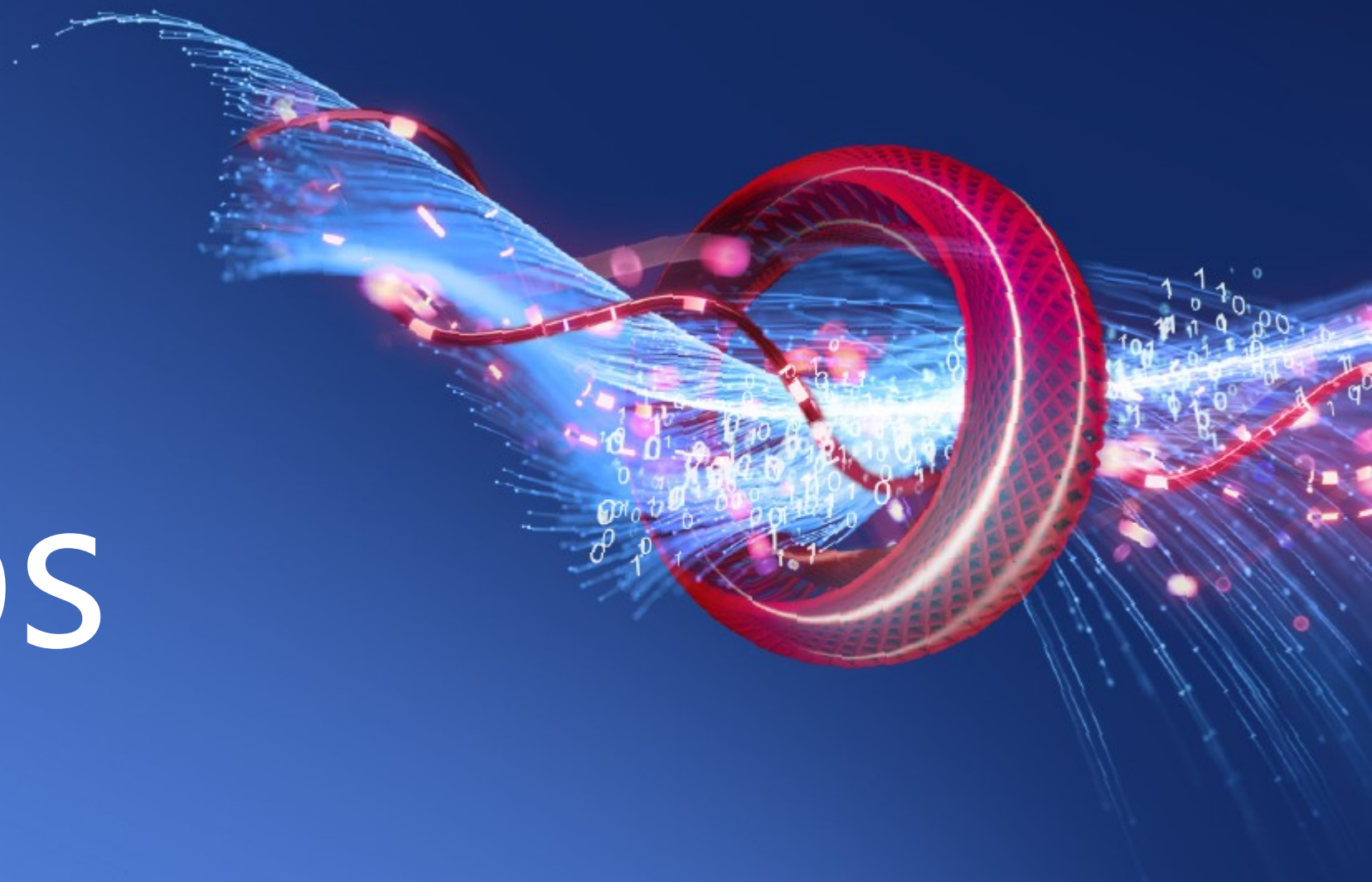
Label	Name	Action
Add		

duration Action

default [Edit](#) [Remove](#)

default [Edit](#) [Remove](#)

Tips



TIPS

- Use dedicated API user for API calls. For example, username «api»
- «curl» does not work via docker «alpine» container. Workaround is to use a «centos» container
- When «jq» utility is not available. Can try a standart grep:

```
grep -oP 'result\":"\K\w+'  
grep -oP 'itemid\":"\K\d+'  
grep -oP 'host\":"\K\w+'
```

THANK
YOU!

