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
ZABBIX 5.0 webinar

OBTAIN SESSION TOKEN

POST user.login

https://zabbix.cetonrug.net/api_jsonrpc.php

```json
{
  "jsonrpc": "2.0",
  "method": "user.login",
  "params": {
    "user": "api",
    "password": "zabbix"
  },
  "id": 1
}
```
ZABāBIX 5.0 webinar

PICK A FLAVOR FROM DOCUMENTATION PAGE

POST `all.triggers.in.problem.state`

POST `https://zbx.cetonrug.net/api_jsonrpc.php`

```json
"jsonrpc": "2.0",
"method": "trigger.get",
"params": {
"output": [
{"triggerid", 
"description",
"priority"
],
"filter": {
"value": 1
},
"sortfield": "priority",
"sortorder": "DESC"
},
"auth": "21dc9cc234b18d544011aae3ca14c27",
"id": 1
```

Hit Send to get a response
CREATE AN ENVIRONMENT

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.
## CREATE ENVIRONMENT CHARACTERISTICS

### Add Environment

- **prod**

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>INITIAL VALUE</th>
<th>CURRENT VALUE</th>
<th>Persist All</th>
<th>Reset All</th>
</tr>
</thead>
<tbody>
<tr>
<td>auth</td>
<td>21dc9cc23fb18d5449</td>
<td>21dc9cc23fb18d5449</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Add a new variable

[Sentinel]
PROD, TEST, 4.0, 5.0

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
SET ENVIRONMENT, CONFIGURE DYNAMICS
API scripting

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

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

CentOS8/RHEL8

```bash
dnf install jq
```

Ubuntu/Debian

```bash
apt install jq
```

Standalone 64-bit binary on any Linux:

```bash
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

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

Always execute step 2 at first and only then: step 3
Notice 2 double quotes are not escaped
# 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

# BASH EXAMPLE

API task starts with 'user.login' procedure and ends with 'user.logout' procedure
# 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"
# input
curl -s -X POST \
-H 'Content-Type: application/json-rpc' \n-d "
{
  "jsonrpc": "2.0",
  "method": "proxy.get",
  "params": {
    "output": ["host"]
  },
  "auth": "$auth",
  "id": 1
}
" $url | jq -r '.result[].host'

# output
broceni
mysql8mon
riga
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

**Actions**

<table>
<thead>
<tr>
<th>Action</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Default operation step duration: 1h
- Pause operations for suppressed problems: unchecked

**Operations**

<table>
<thead>
<tr>
<th>Steps</th>
<th>Details</th>
<th>Start in</th>
<th>Duration</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Send message to users: Admin (Zabbix Administrator) via all media immediately</td>
<td>60</td>
<td></td>
<td>Edit, Remove</td>
</tr>
<tr>
<td>2</td>
<td>Run remote commands on current host</td>
<td>00:01:00</td>
<td>Default</td>
<td>Edit, Remove</td>
</tr>
</tbody>
</table>

**Recovery operations**

<table>
<thead>
<tr>
<th>Details</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Add</td>
</tr>
</tbody>
</table>

**Update operations**

<table>
<thead>
<tr>
<th>Details</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Add</td>
</tr>
</tbody>
</table>

* At least one operation must exist.

[Update, Clone, Delete, Cancel]
REMOTE COMMAND

Operation details

Operation type: Remote command

Steps: 2

Step duration: 0

Target list:
- Current host

Type: Custom script

Execute on:
- Zabbix agent
- Zabbix server (proxy)
- Zabbix server

Commands:

```
$url=(\$API_PHP)
$username=(\$API_USER)
$password=(\$API_PASSWORD)

# authorization
auth=(curl -X POST -H "Content-Type: application/json" -d "
```

Conditions:

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

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

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

SOLUTION: ZABBIX_GLOBAL_MACRO_UPDATE.SH
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

```bash
curl -s -X POST -H 'Content-Type: application/json' http://127.0.0.1/api_jsonrpc.php -d "\n{"jsonrpc":"2.0","method":"user.login","params":{"user":"api","password":"zabbix"},"id":1,"auth":null}\n" | 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})"

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

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

Operation type: Remote command

Steps: 2
Step duration: 0

Target list:
- Current host
- Host: type to search
- Host group: type to search

Type: Custom script

Execute on:
- Zabbix agent
- Zabbix server (proxy)
- Zabbix server

Commands:
```
# acknowledge and close event
curl -sX POST -H "Content-Type: application/json" -d { "jsonrpc": "2.0", "method": "event.acknowledge", "params": [ { "eventids": ["{EVENT_ID}"], "action": 1, "message": "Problem resolved." }, { "auth": "\$\{ZAPI_USER\}\$\{ZAPI_PASSWORD\}" } ] } (curl $\{ZAPI_HTTP\})
```

Conditions

Label | Name | Action
--- | --- | ---
Add

Update | Cancel
USE Case 4: API through HTTP Agent
**DISCOVERY RULE**

Name: Discover all proxies  
Type: HTTP agent  
Key: get.proxy.list  
URL: ${Z_API_PHP}  
Request type: POST  
Timeout: 8s  
Request body type: Raw data, JSON data, XML data  
Request body:
```
{
    "jsonrpc": "2.0",
    "method": "proxy.get",
    "params": {
        "output": "extend",
        "selectInterface": "extend"
    },
    "auth": "${Z_API_SESSIONID}",
    "id": 1
}
```
{  
  "jsonrpc": "2.0",  
  "result": [  
    {  
      "host": "broceni",  
      "proxyid": "10387"  
    },  
    {  
      "host": "mysql8mon",  
      "proxyid": "12066"  
    },  
    {  
      "host": "riga",  
      "proxyid": "12585"  
    }  
  ],  
  "id": 1  
}
[[
  {
    "host": "broceni",
    "proxyid": "10387"
  },
  {
    "host": "mysql8mon",
    "proxyid": "12066"
  },
  {
    "host": "riga",
    "proxyid": "12585"
  }
]]
<table>
<thead>
<tr>
<th>Item prototype</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Last access for [#PROXYNAME]</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>Zabbix internal</td>
</tr>
<tr>
<td><strong>Key</strong></td>
<td>zabbix[proxy.[#PROXYNAME].lastaccess]</td>
</tr>
<tr>
<td><strong>Type of information</strong></td>
<td>Numeric (unsigned)</td>
</tr>
<tr>
<td><strong>Units</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Update interval</strong></td>
<td>1m</td>
</tr>
</tbody>
</table>
TRIGGER PROTOTYPE

Name: [#PROXYNAME] is not reachable for [$PROXYFUZZYTIME]

Severity: Not classified, Information, Warning, Average, High, Disaster

Expression: |px:zabbix.proxy, [#PROXYNAME], lastaccess].fuzzytime({$PROXYFUZZYTIME})|=0

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
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!