



Customizing Zabbix with built in JavaScript

Who am I?



Edwin Muller
Consultant Telecom Smart Meters

Who am I?



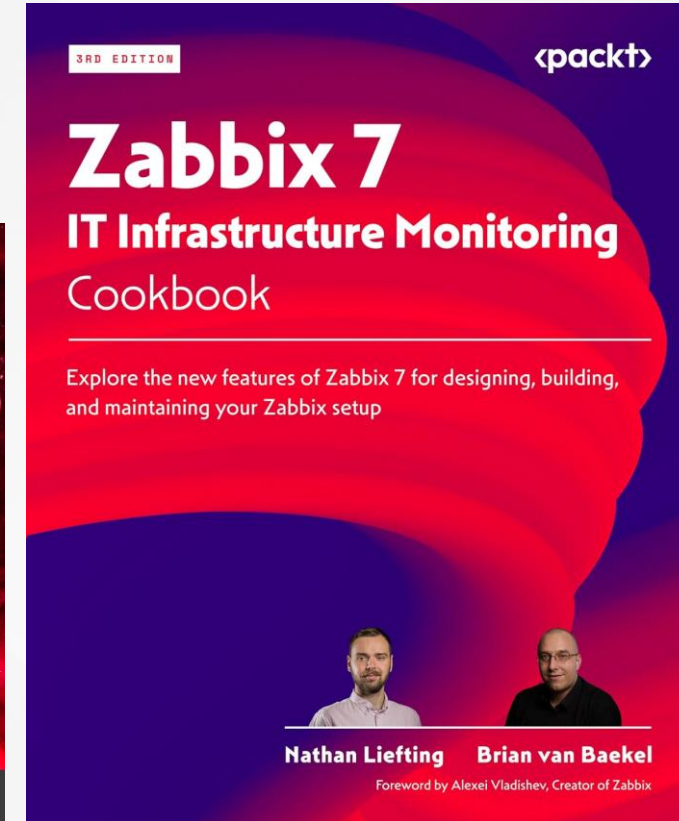
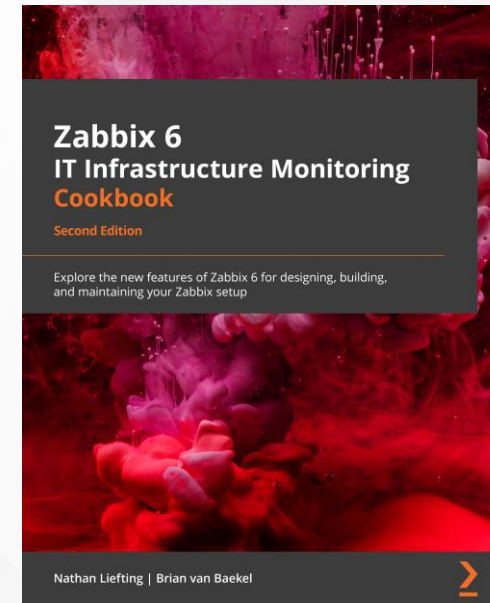
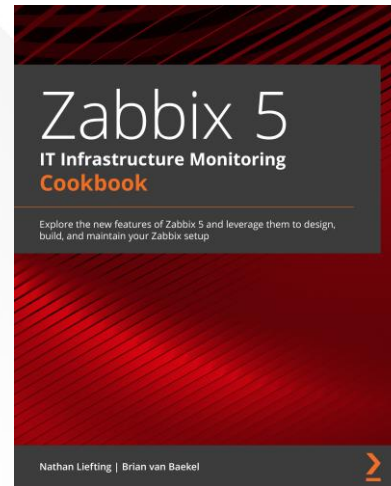
Nathan Liefing
Zabbix Consultant / Trainer



Opensource ICT Solutions



- Zabbix support
- Zabbix training
- Zabbix consultancy
- And more...



<https://www.linkedin.com/company/opensource-ict-solutions/>



oicts.com



Opensource ICT Solutions

450connect

450connect is Allianters German based joint venture of the energy and water industries based in Köln, whose mission is to securely digitize critical infrastructures.

450connect only uses standardized LTE radio technology (4G/5G) in the 450 MHz frequency band.

TReNT

TReNT is Allianters commercial Dark Fiber supplier, delivering reliable connectivity for your business-critical IT applications.

Since January 2020, the unused capacity of Allianters Telecom's fiber optic network has also been available to the market via TReNT.

allianters Telecom

Allianters Telecom (AT) is Allianters in-house M2M integrated service provider.

AT provides fiber, connectivity, devices, service and knowledge.

AT has extensive knowledge of telecommunication solutions and the energy domain.

UTILITY CONNECT

SMART DEDICATED NETWORK

Utility Connect (UC) is an operator offering wireless connectivity over 450 MHz.

UC operates a dedicated wireless network specifically designed and built for the energy utilities.



Wireless devices - SNMP



Wireless assets	Size
Smart Meters	3.200.000
Distribution automation	10.000
Street Lighting	20.000
Power Quality	53
Alarm indicators	4.000
Gas automation	120
Others	200

Why Zabbix in Alliander?

- Monitoring of **health**, **performance** and **availability** of wireless routers
- Monitoring of the communication chain
- Monitoring internal systems, detection of mass outages

- Keeping firmware up-to-date
- Historical behavior of routers
 - Wireless signal quality
 - Signal to Noise ratio
- Up-to-date wireless network provider information (why?)

Why up-to-date wireless provider information?

- Routers are wireless with eSIM
- Remote SIM Provisioning (RSP)
- Providers may vary per device
 - Roaming
 - Multi IMSI
 - Multi profile
- Dashboards and Trends per provider
- Cost management



Problem: We want to have a value from an item as a tag

<input type="checkbox"/>	Name ▲	Items	Triggers	Graphs	Availability	Agent encryption	Info	Tags
<input type="checkbox"/>	router-02	Items	Triggers	Graphs	ZBX SNMP	None		provider: ???

Displaying 1 of 1 found

End result: Dynamic item tag on the host level

<input type="checkbox"/>	Name ▲	Items	Triggers	Graphs	Availability	Agent encryption	Info	Tags
<input type="checkbox"/>	router-02	Items	Triggers	Graphs	ZBX SNMP	None		provider: T-Mobile (3)

Displaying 1 of 1 found

How do we achieve this without external scripting?

- Zabbix introduced built-in Javascript
 - Available in various locations: Global, item level
 - Extensive Javascripting capability with Duktape
 - **NOT** extendable with libraries

- Go to Alerts | Scripts
 - Type: Webhook
 - Name the script
 - Parse Parameters
 - Write a Script
 - Define other parameters

* Name

Scope

Type

Parameters

Name	Value	Action
<input type="text" value="hostId"/>	<input type="text" value="{HOST.ID}"/>	Remove
<input type="text" value="tagName"/>	<input type="text" value="Provider"/>	Remove
<input type="text" value="tagValue"/>	<input type="text" value="{ITEM.VALUE}"/>	Remove
<input type="text" value="token"/>	<input type="text" value="zabbix_api_token"/>	Remove
<input type="text" value="url"/>	<input type="text" value="http://127.0.0.1/zabbix/api_jsonrpc"/>	Remove

[Add](#)

* Script

* Timeout

Description

Host group

Let's break down the script

Parse parameters

```
var fields = {},  
    zabbixApiUrl = params.url,  
    zabbixApiToken = params.token,  
    hostId = params.hostId,  
    newTag = params.tagValue,  
    tagName = params.tagName;
```

Parameters

Name	Value	Action
hostId	{HOST.ID}	Remove
tagName	Provider	Remove
tagValue	{ITEM.VALUE}	Remove
token	zabbix_api_token	Remove
url	http://127.0.0.1/zabbix/api_jsonrpc	Remove
Add		

Let's break down the script

Define your API call(s)

```
// Fetch existing host tags
var getTagsData = {
  jsonrpc: '2.0',
  method: 'host.get',
  params: {
    output: 'extend',
    selectTags: 'extend',
    filter: {
      hostid: hostId
    }
  },
  auth: zabbixApiToken,
  id: 1,
};
```

Execute API call

```
// Make the API call to get existing tags
var request = new HttpRequest();
request.addHeader('Content-Type: application/json');
var getTagsResponse = request.post(zabbixApiUrl, JSON.stringify(getTagsData));
```

Mobile networks – Costs in Control

- Unlike fixed networks costs in mobile networks are key
- **Ping** to test network with minimal costs (200 bytes per ping)

```
"161", "2000-01-01 00:00:00,532889", "1.1.1.1", "2.2.2.2", "ICMP", "100", "Echo (ping) request id=0x53d6, seq=0/0, ttl=53 (reply in 162)"  
"162", "2000-01-01 00:00:00,533248", "2.2.2.2", "1.1.1.1", "ICMP", "100", "Echo (ping) reply id=0x53d6, seq=0/0, ttl=64 (request in 161)"  
"163", "2000-01-01 00:00:01,995479", "1.1.1.1", "2.2.2.2", "ICMP", "100", "Echo (ping) request id=0x5528, seq=0/0, ttl=53 (reply in 164)"  
"164", "2000-01-01 00:00:01,995820", "2.2.2.2", "1.1.1.1", "ICMP", "100", "Echo (ping) reply id=0x5528, seq=0/0, ttl=64 (request in 163)"
```

- Use **ping** before SNMP increases success rate, to “wake up device”
- Count **#pings** until successful with **max retries** setting
- Each mobile network technology has optimal **ping** settings!

Problem: We want to execute SNMP only if ping is successful

<input type="checkbox"/> router-13	ICMP Ping status	2s	Up (1)
------------------------------------	------------------	----	--------

End result: Execute (multiple) pings then execute SNMP walk

* Name

Type

* Key

Type of information

* Host interface

* SNMP OID

Units

* Update interval

Custom intervals

Type	Interval	Period	Action
<input type="button" value="Flexible"/> <input checked="" type="button" value="Scheduling"/>	<input style="border: 1px solid #ccc;" type="text" value="md31wd1h0m0"/>		Remove
Add			

In short, let's create the ICMP script

- Simple bash script to execute 1 ping
 - If ping successful STOP
 - If not try a max of 3 more
- Send result back with Zabbix sender
 - Send as JSON
 - Collect status, retries, time and loss

```
while [ $retry_count -lt $max_retries ]; do
    # Perform a single ping using fping
    ping_result=$(fping -c 1 -i $interval -t $timeout $hostipdns 2>&1)

    # If the output contains "timed out", exit status 0
    if [[ $ping_result == *"timed out"* ]]; then
        fping_exit_status=0

        # Ping failed, increment the retry count
        ((retry_count++))
    else
        # If "timed out" is not found, exit status 1
        fping_exit_status=1

        # Ping succeeded, extract time and packet loss from ping result
        ping_time=$(echo "$ping_result" | awk -F'bytes,' '{print $2}' | awk '{print $1}')
        packet_loss=$(echo "$ping_result" | awk -F'avg,' '{print $2}' | awk '{print $1}')

        json_output="{\"ping_status\": \"1\", \"retry_count\": $retry_count, \"ping_time\": \"$ping_time\", \"packet_loss\": \"$packet_loss\"}"
        zabbix_sender -z localhost -s $hostname -k ping.script -o "$json_output"
        #echo $json_output
        exit 0
    fi
done
```

What about the Javascript?

Trigger when ping received

* Name

Event name

Operational data

Severity

* Expression

[Expression constructor](#)

Notes:

- Nodata auto resolves after 30s, so watch the script execution time
- Tag has to be created to execute JavaScript
- Creates many events

Create tag

Trigger tags Inherited and trigger tags

Name	Value	
<input type="text" value="execute"/>	<input type="text" value="snmp"/>	Remove

[Add](#)

Now we create the script

* Name


Scope Action operation Manual host action Manual event action

Type Webhook Script SSH Telnet IPMI

Parameters

Name	Value	Action
<input type="text" value="hostId"/>	<input type="text" value="{HOST.ID}"/>	Remove
<input type="text" value="token"/>	<input type="text" value="api_token_here"/>	Remove
<input type="text" value="url"/>	<input type="text" value="http://127.0.0.1/zabbix/api_jsonrpc"/>	Remove

[Add](#)

* Script 

* Timeout

- Alerts | Scripts
- Parse **parameters** to the script
HostID, API Token, Zabbix API URL
- Communicate **internally**
(localhost on single instance Zabbix server)
- Write the **script**

Let's break down the script

Again: Parse the parameters

```
var fields = {},
    zabbixApiUrl = params.url,
    zabbixApiToken = params.token,
    hostId = params.hostId;
```

Parameters	Name	Value	Action
	hostId	{HOST.ID}	Remove
	token	api_token_here	Remove
	url	http://127.0.0.1/zabbix/api_jsonrpc	Remove
	Add		

Also: Do not forget to add error logging

```
// Log the parsed JSON
Zabbix.log(3, '[Execute SNMP script] Parsed JSON: ' + JSON.stringify(params));

if ('error' in item_result) {
    Zabbix.log(4, '[Execute SNMP script] Zabbix API item retrieval failed:' + item_info.result);
}
```

Let's break down the script

Now how do we execute SNMP through the API?

Step 1: Get item data (search for key `snmp.get`)

```
// Zabbix API request to get the item details
var item_get_data = {
  jsonrpc: '2.0',
  method: 'item.get',
  params: {
    output: 'extend',
    hostids: hostId,
    search: {
      key_: 'snmp.get', // Item key to search for
    },
  },
  auth: zabbixApiToken,
  id: 2,
};
```

Let's break down the script

Step 2: Use data to **execute now** through API with **task.create**

```
// Zabbix API request to create a task and execute the item
var task_create_data = {
  jsonrpc: '2.0',
  method: 'task.create',
  params: [
    {
      type: 6,
      request: {
        itemid: item_id,
      }
    }
  ],
  auth: zabbixApiToken,
  id: 3,
};
```

Do not forget the action

Alerts | Actions

* Name

Conditions

Label	Name
A	Value of tag <i>execute</i> equals <i>snmp</i>

[Add](#)

Execute the JavaScript through action operation

* Default operation step duration

Operations

Steps	Details	Start in	Duration	Action
1	Run script "Execute SNMP" on current host	Immediately	Default	Edit Remove

[Add](#)

Zabbix 7.0: Another way to reduce data

* Name

Type

* Key

Type of information

* Host interface

* SNMP OID

Units

* Update interval

Custom intervals	Type	Interval	Period	Action
<input type="button" value="Flexible"/> <input checked="" type="button" value="Scheduling"/>		<input type="text" value="md31wd1h0m0"/>		Remove

[Add](#)

SNMP Bulk collection

- **Reduced** data collection load
- **Increased** bytes used

SNMP get collection

- **Increased** data collection load
- **Decreased** bytes used
- Zabbix 7.0: **Single item** for many **OIDs**

* Name

Type

* Key

Type of information

* Host interface

* SNMP OID

Units

* Update interval

Custom intervals	Type	Interval	Period	Action
<input type="button" value="Flexible"/> <input checked="" type="button" value="Scheduling"/>		<input type="text" value="md31wd1h0m0"/>		Remove

[Add](#)

Benefits of Zabbix for Alliander

- **Near-Real-Time** and **more extensive** information of mobile devices
- Alarms for routers that need (immediate) action
- Network management system including all peripherals
- Historical data for identification of trends
- Customization of Zabbix enables ability to minimize data usage

Customize, customize and wait

Custom Additions:

- Zabbix built-in JavaScript
 - Global
 - Items
 - Media types
- Script items

Added by Zabbix:

- SNMP **Bulk** items
- SNMP **Get** items in single item
- More to come?
 - ICMP Ping max retry item:
ZBXNEXT-9497
 - Item execution dependencies:
ZBXNEXT-9498

Questions?

