17 HTTP agent

Overview

This item type allows data polling using the HTTP/HTTPS protocol. Trapping is also possible using Zabbix sender or Zabbix sender protocol.

HTTP item check is executed by Zabbix server. However, when hosts are monitored by a Zabbix proxy, HTTP item checks are executed by the proxy.

HTTP item checks do not require any agent running on a host being monitored.

HTTP agent supports both HTTP and HTTPS. Zabbix will optionally follow redirects (see the *Follow redirects* option below). Maximum number of redirects is hard-coded to 10 (using cURL option CURLOPT_MAXREDIRS).

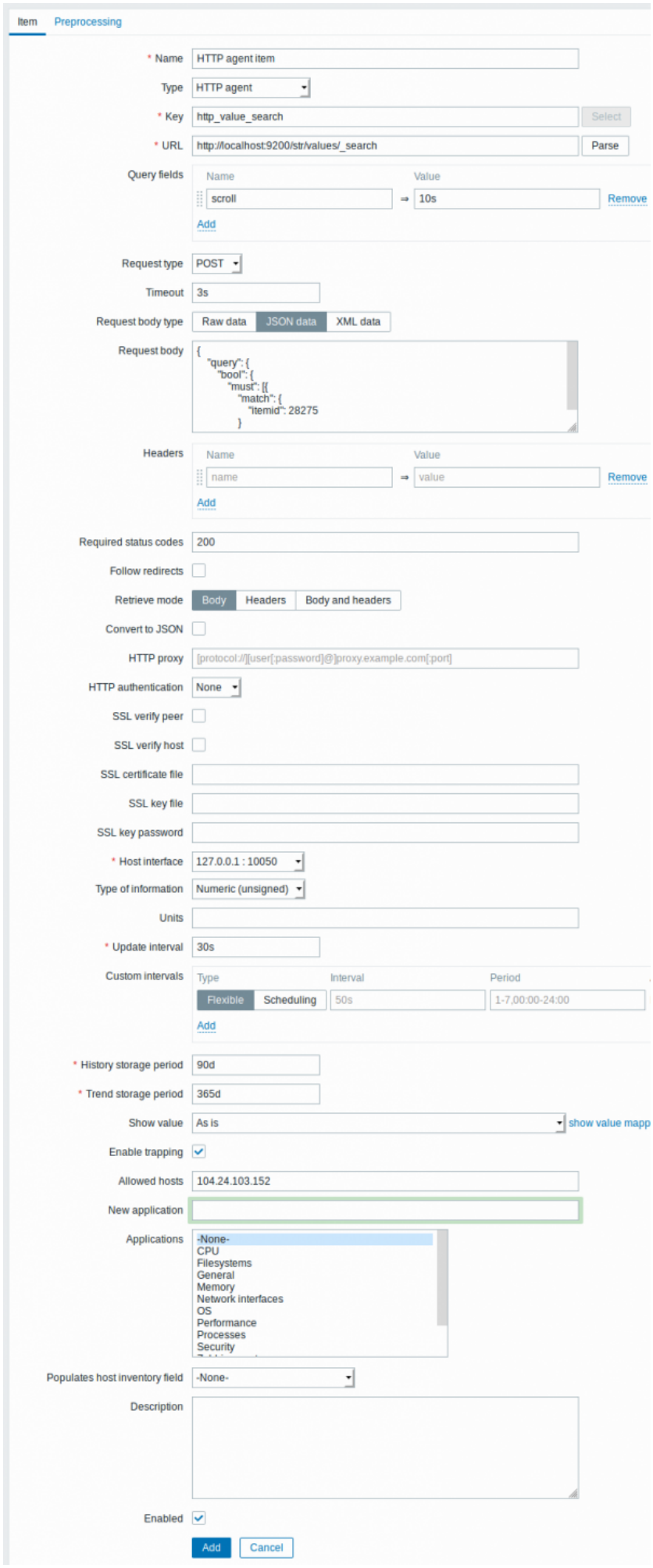
See also known issues for when using HTTPS protocol.

Zabbix server/proxy must be initially configured with cURL (libcurl) support.

Configuration

To configure an HTTP item:

- Go to: *Configuration* → *Hosts*
- Click on *Items* in the row of the host
- Click on *Create item*
- Enter parameters of the item in the form


Item
Preprocessing
* Name
HTTP agent item
Type
HTTP agent
* Key
http_value_search
Select
* URL
http://localhost:9200/str/values/_search
Parse
Query fields
Name
Value
scroll
10s
Remove
Add
Request type
POST
Timeout
3s
Request body type
Raw data
JSON data
XML data
Request body
{
"query": {
"bool": {
"must": [{
"match": {
"itemid": 28275
}
Headers
Name
Value
name
value
Remove
Add
Required status codes
200
Follow redirects
Retrieve mode
Body
Headers
Body and headers
Convert to JSON
HTTP proxy
[protocol://][user[:password]@]proxy.example.com[:port]
HTTP authentication
None
SSL verify peer
SSL verify host
SSL certificate file
SSL key file
SSL key password
* Host interface
127.0.0.1 : 10050
Type of information
Numeric (unsigned)
Units
* Update interval
30s
Custom intervals
Type
Interval
Period
Flexible
Scheduling
50s
1-7,00:00-24:00
Add
* History storage period
90d
* Trend storage period
365d
Show value
As is
show value mapp
Enable trapping
Allowed hosts
104.24.103.152
New application
Applications
-None-
CPU
Filesystems
General
Memory
Network interfaces
OS
Performance
Processes
Security
Populates host inventory field
-None-
Description
Enabled
Add
Cancel

All mandatory input fields are marked with a red asterisk.

The fields that require specific information for HTTP items are:

Type	Select **HTTP agent** here.
Key	Enter a unique item key.
URL	URL to connect to and retrieve data. For example: https://www.google.com http://www.zabbix.com/download Domain names can be specified in Unicode characters. They are automatically punycode-converted to ASCII when executing the HTTP check. The *Parse* button can be used to separate optional query fields (like ?name=Admin&password=mypassword) from the URL, moving the attributes and values into *Query fields* for automatic URL-encoding. Limited to 2048 characters. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros. This sets the CURLOPT_URL cURL option.
Query fields	Variables for the URL (see above). Specified as attribute and value pairs. Values are URL-encoded automatically. Values from macros are resolved and then URL-encoded automatically. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros. This sets the CURLOPT_URL cURL option.
Request type	Select request method type: *GET*, *POST*, *PUT* or *HEAD*
Timeout	Zabbix will not spend more than the set amount of time on processing the URL (maximum is 1 minute). Actually this parameter defines the maximum time for making a connection to the URL and maximum time for performing an HTTP request. Therefore, Zabbix will not spend more than 2 x Timeout seconds on one check. Time suffixes are supported, e.g. 30s, 1m. Supported macros: user macros, low-level discovery macros. This sets the CURLOPT_TIMEOUT cURL option.
Request body type	Select the request body type: **Raw data** - custom HTTP request body, macros are substituted but no encoding is performed **JSON data** - HTTP request body in JSON format. Macros can be used as string, number, true and false; macros used as strings must be enclosed in double quotes. Values from macros are resolved and then escaped automatically. If "Content-Type" is not specified in headers then it will default to "Content-Type: application/json" **XML data** - HTTP request body in XML format. Macros can be used as a text node, attribute or CDATA section. Values from macros are resolved and then escaped automatically in a text node and attribute. If "Content-Type" is not specified in headers then it will default to "Content-Type: application/xml" *Note* that selecting *XML data* requires libxml2.
Request body	Enter the request body. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros.
Headers	Custom HTTP headers that will be sent when performing a request. Specified as attribute and value pairs. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros. This sets the CURLOPT_HTTPHEADER cURL option.

Required status codes	List of expected HTTP status codes. If Zabbix gets a code which is not in the list, the item will become unsupported. If empty, no check is performed. For example: 200,201,210-299 Supported macros in the list: user macros, low-level discovery macros. This uses the CURLINFO_RESPONSE_CODE cURL option.
Follow redirects	Mark the checkbox to follow HTTP redirects. This sets the CURLOPT_FOLLOWLOCATION cURL option.
Retrieve mode	Select the part of response that must be retrieved: **Body** - body only **Headers** - headers only **Body and headers** - body and headers

Convert to JSON	Headers are saved as attribute and value pairs under the "header" key.
	If 'Content-Type: application/json' is encountered then body is saved as an object, otherwise it is stored as string, for example:
	`{`
	`"header": {`
	`"<key>": "<value>",`
	`"<key2>": "<value>"`
	`},`
	`"body": <body>`
	`}`

HTTP proxy	You can specify an HTTP proxy to use, using the format `[protocol://][username[:password]@]proxy.mycompany.com[:port]`. The optional `protocol://` prefix may be used to specify alternative proxy protocols (the protocol prefix support was added in cURL 7.21.7). With no protocol specified, the proxy will be treated as an HTTP proxy. By default, 1080 port will be used. If specified, the proxy will overwrite proxy related environment variables like http_proxy, HTTPS_PROXY. If not specified, the proxy will not overwrite proxy-related environment variables. The entered value is passed on "as is", no sanity checking takes place. You may also enter a SOCKS proxy address. If you specify the wrong protocol, the connection will fail and the item will become unsupported. *Note* that only simple authentication is supported with HTTP proxy. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros. This sets the CURLOPT_PROXY cURL option.
HTTP authentication	Authentication type: **None** - no authentication used. **Basic authentication** - basic authentication is used. **NTLM authentication** - NTLM (Windows NT LAN Manager) authentication is used. Selecting an authentication method will provide two additional fields for entering a user name and password, where user macros and low-level discovery macros are supported. This sets the CURLOPT_HTTPAUTH cURL option.
SSL verify peer	Mark the checkbox to verify the SSL certificate of the web server. The server certificate will be automatically taken from system-wide certificate authority (CA) location. You can override the location of CA files using Zabbix server or proxy configuration parameter SSLCALocation. This sets the CURLOPT_SSL_VERIFYPEER cURL option.
SSL verify host	Mark the checkbox to verify that the Common Name field or the Subject Alternate Name field of the web server certificate matches. This sets the CURLOPT_SSL_VERIFYHOST cURL option.

SSL certificate file	Name of the SSL certificate file used for client authentication. The certificate file must be in PEM[1] format. If the certificate file contains also the private key, leave the SSL key file field empty. If the key is encrypted, specify the password in SSL key password field. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLCertLocation. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros. This sets the CURLOPT_SSLCERT cURL option.
SSL key file	Name of the SSL private key file used for client authentication. The private key file must be in PEM[1] format. The directory containing this file is specified by Zabbix server or proxy configuration parameter SSLKeyLocation. Supported macros: {HOST.IP}, {HOST.CONN}, {HOST.DNS}, {HOST.HOST}, {HOST.NAME}, {ITEM.ID}, {ITEM.KEY}, user macros, low-level discovery macros. This sets the CURLOPT_SSLKEY cURL option.
SSL key password	SSL private key file password. Supported macros: user macros, low-level discovery macros. This sets the CURLOPT_KEYPASSWD cURL option.
Enable trapping	With this checkbox marked, the item will also function as trapper item and will accept data sent to this item by Zabbix sender or using Zabbix sender protocol.
Allowed hosts	Visible only if *Enable trapping* checkbox is marked. List of comma delimited IP addresses, optionally in CIDR notation, or hostnames. If specified, incoming connections will be accepted only from the hosts listed here. If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated equally and '::/0' will allow any IPv4 or IPv6 address. '0.0.0.0/0' can be used to allow any IPv4 address. Note, that “IPv4-compatible IPv6 addresses” (0000::/96 prefix) are supported but deprecated by RFC4291. Example: Server=127.0.0.1, 192.168.1.0/24, 192.168.3.1-255, 192.168.1-10.1-255, ::1,2001:db8::/32, zabbix.domain Spaces and user macros are allowed in this field. Host macros: {HOST.HOST}, {HOST.NAME}, {HOST.IP}, {HOST.DNS}, {HOST.CONN} are allowed in this field.

If the *HTTP proxy* field is left empty, another way for using an HTTP proxy is to set proxy-related environment variables.

For HTTP - set the `http_proxy` environment variable for the Zabbix server user. For example: *http_proxy=http://proxy_ip:proxy_port*.

For HTTPS - set the HTTPS_PR0XY environment variable. For example: *HTTPS_PROXY=http://proxy_ip:proxy_port*. More details are available by running a shell command: *# man curl*.

[1] Zabbix supports certificate and private key files in PEM format only. In case you have your certificate and private key data in PKCS #12 format file (usually with extention *.p12 or *.pfx) you may generate the PEM file from it using the following commands:

```
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes  -out ssl-cert.key
```

Examples

Example 1

Send simple GET requests to retrieve data from services such as Elasticsearch:

- Create a GET item with URL: `localhost:9200/?pretty`
- Notice the response:

```
{
  "name" : "YQ2VAY-",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "kH4CYqh5QfqgeTsjh2F9zg",
  "version" : {
    "number" : "6.1.3",
    "build_hash" : "af51318",
    "build_date" : "2018-01-26T18:22:55.523Z",
    "build_snapshot" : false,
    "lucene_version" : "7.1.0",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You know, for search"
}
```

- Now extract the version number using a JSONPath preprocessing step: `$.version.number`

Example 2

Send simple POST requests to retrieve data from services such as Elasticsearch:

- Create a POST item with URL: `http://localhost:9200/str/values/_search?scroll=10s`
- Configure the following POST body to obtain the processor load (1 min average per core)

```
{
    "query": {
        "bool": {
            "must": [{
                "match": {
                    "itemid": 28275
                }
            }],
            "filter": [{
                "range": {
                    "clock": {
                        "gt": 1517565836,
                        "lte": 1517566137
                    }
```

```
                }
            }]
        }
    }
}
```

- Received:

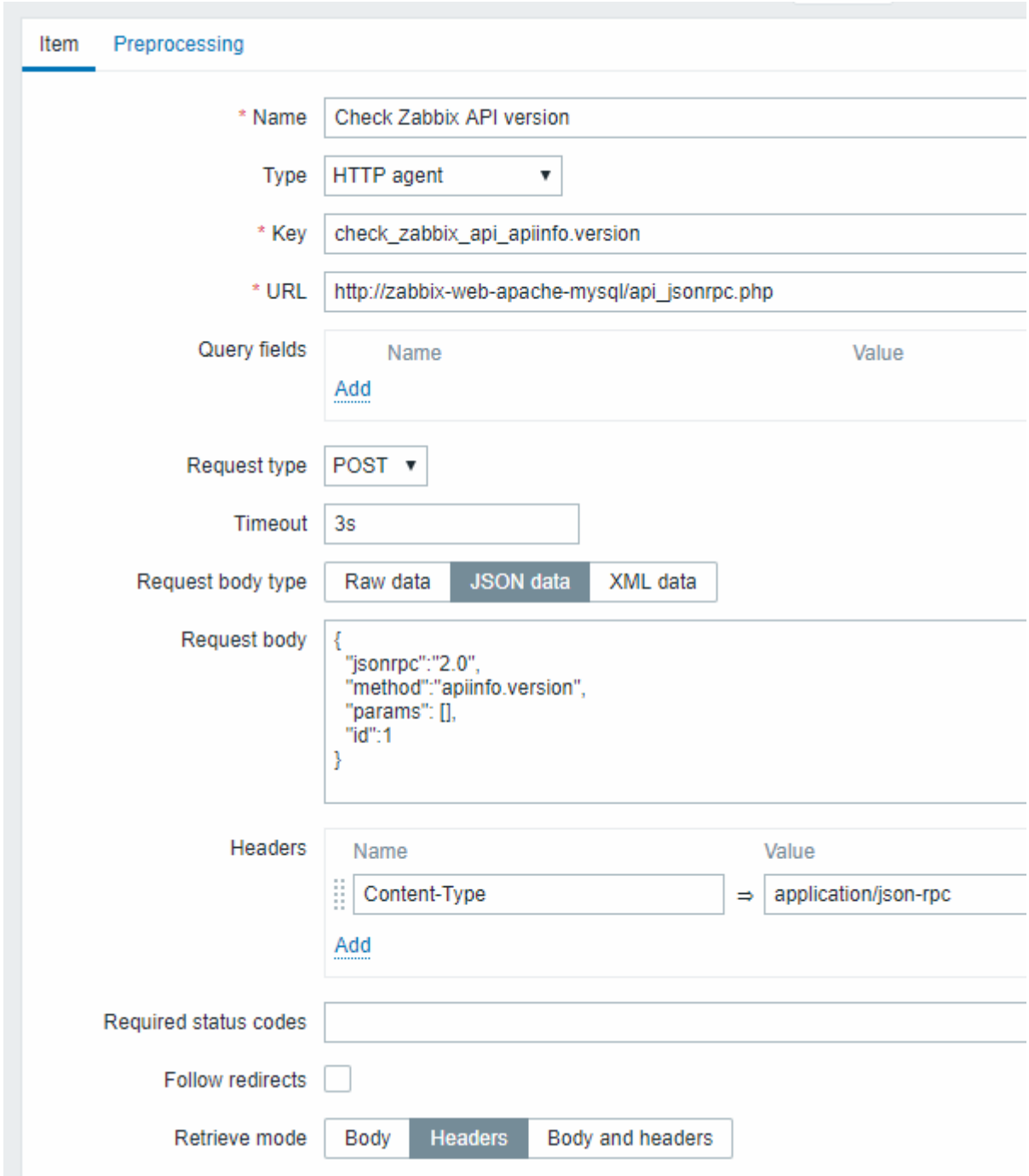
```
{
    "_scroll_id":
"DnF1ZXJ5VGhlbkZldGNoBQAAAAAAAAkFllRMlZBWS1UU1pxTmdEeGVwQjRBTFEAAAAAAAAAJRZ
ZUTJWQVktVFNacU5nRHhlcEI0QUxRAAAAAAAAACYWWVEyVkFZLVRTWnFOZ0R4ZXBCNEFMUQAAAAA
AAAAnFllRMlZBWS1UU1pxTmdEeGVwQjRBTFEAAAAAAAAAKBZZUTJWQVktVFNacU5nRHhlcEI0QUx
R",
    "took": 18,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 1,
        "max_score": 1.0,
        "hits": [{
            "_index": "dbl",
            "_type": "values",
            "_id": "dqX9VWEBV6sEKSMyk6sw",
            "_score": 1.0,
            "_source": {
                "itemid": 28275,
                "value": "0.138750",
                "clock": 1517566136,
                "ns": 25388713,
                "ttl": 604800
            }
        }]
    }
}
```

- Now use a JSONPath preprocessing step to get the item value: `$.hits.hits[0]._source.value`

Example 3

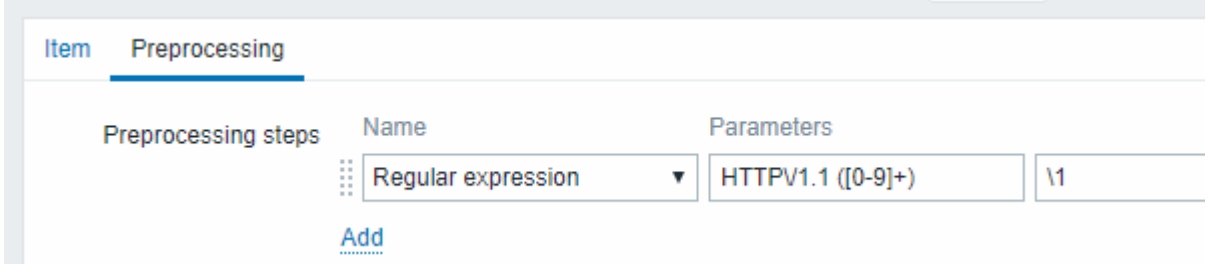
Checking if Zabbix API is alive, using `apiinfo.version`.

- Item configuration:

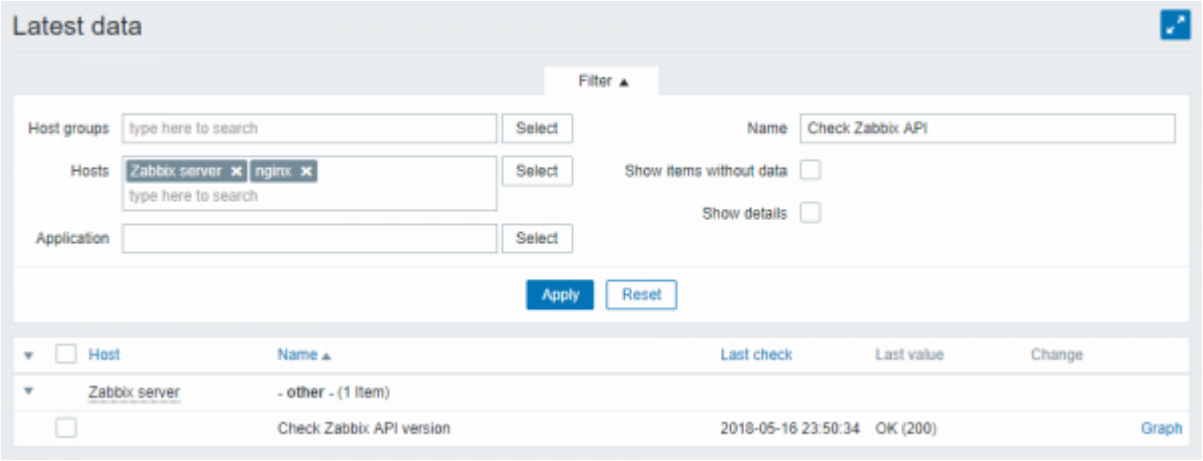


Note the use of the POST method with JSON data, setting request headers and asking to return headers only:

- Item value preprocessing with regular expression to get HTTP code:



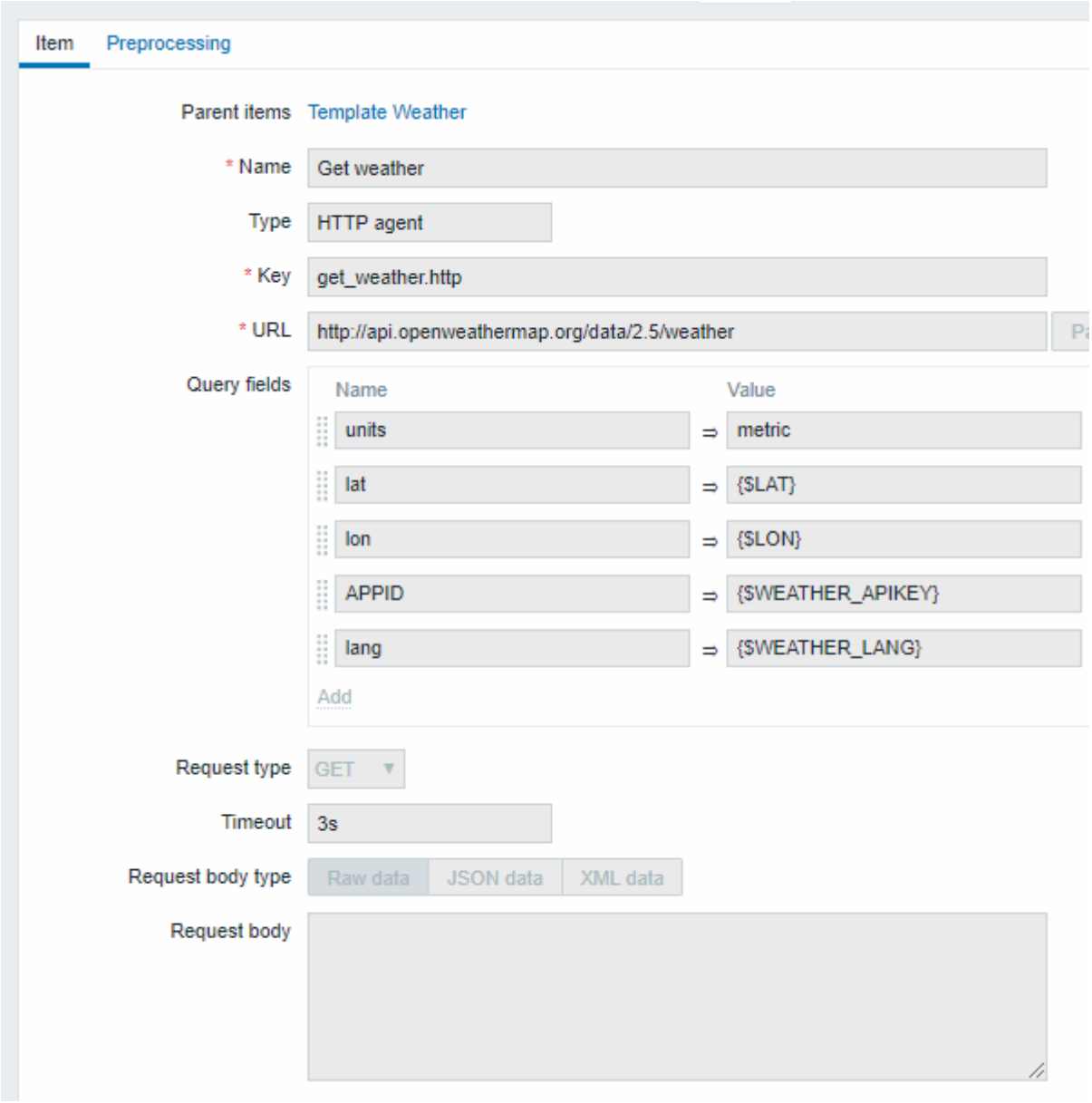
- Checking the result in *Latest data*:



Example 4

Retrieving weather information by connecting to the Openweathermap public service.

- Configure a master item for bulk data collection in a single JSON:



Note the usage of macros in query fields. Refer to the Openweathermap API for how to fill them.

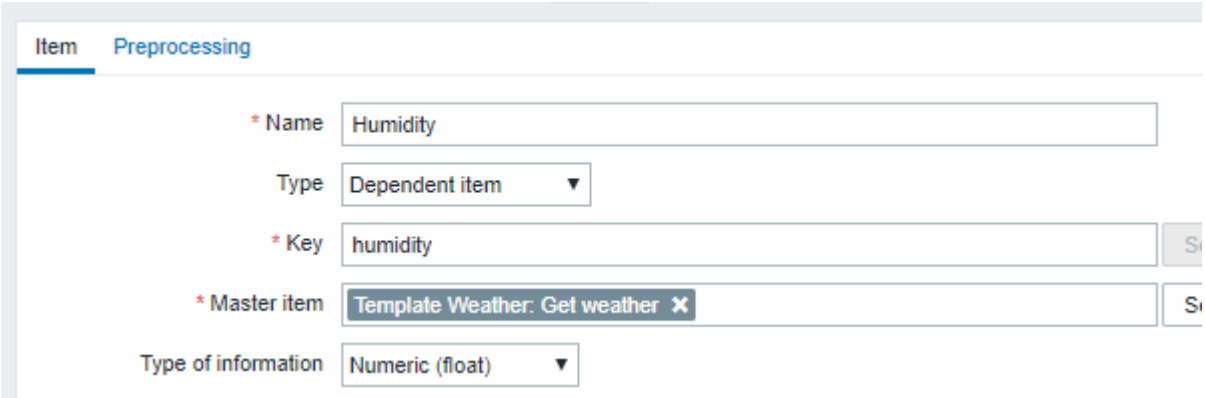
Sample JSON returned in response to HTTP agent:

```
{
    "body": {
        "coord": {
            "lon": 40.01,
            "lat": 56.11
        },
        "weather": [{
            "id": 801,
            "main": "Clouds",
            "description": "few clouds",
            "icon": "02n"
        }],
        "base": "stations",
        "main": {
```

```
            "temp": 15.14,
            "pressure": 1012.6,
            "humidity": 66,
            "temp_min": 15.14,
            "temp_max": 15.14,
            "sea_level": 1030.91,
            "grnd_level": 1012.6
        },
        "wind": {
            "speed": 1.86,
            "deg": 246.001
        },
        "clouds": {
            "all": 20
        },
        "dt": 1526509427,
        "sys": {
            "message": 0.0035,
            "country": "RU",
            "sunrise": 1526432608,
            "sunset": 1526491828
        },
        "id": 487837,
        "name": "Stavrovo",
        "cod": 200
    }
}
```

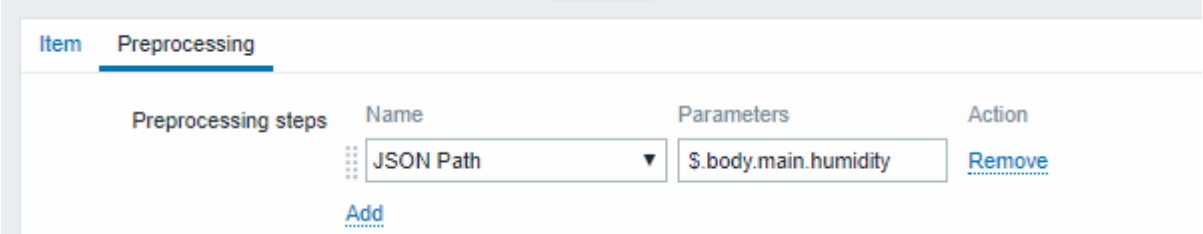

The next task is to configure dependent items that extract data from the JSON.

- Configure a sample dependent item for humidity:

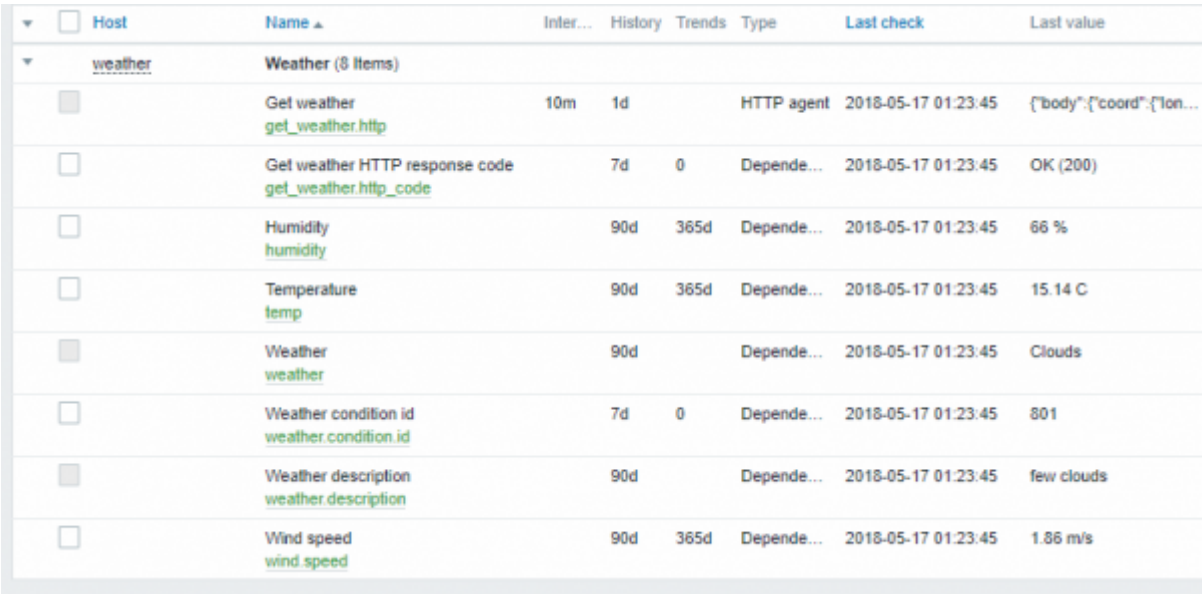


Other weather metrics such as 'Temperature' are added in the same manner.

- Sample dependent item value preprocessing with JSONPath:



- Check the result of weather data in *Latest data*:

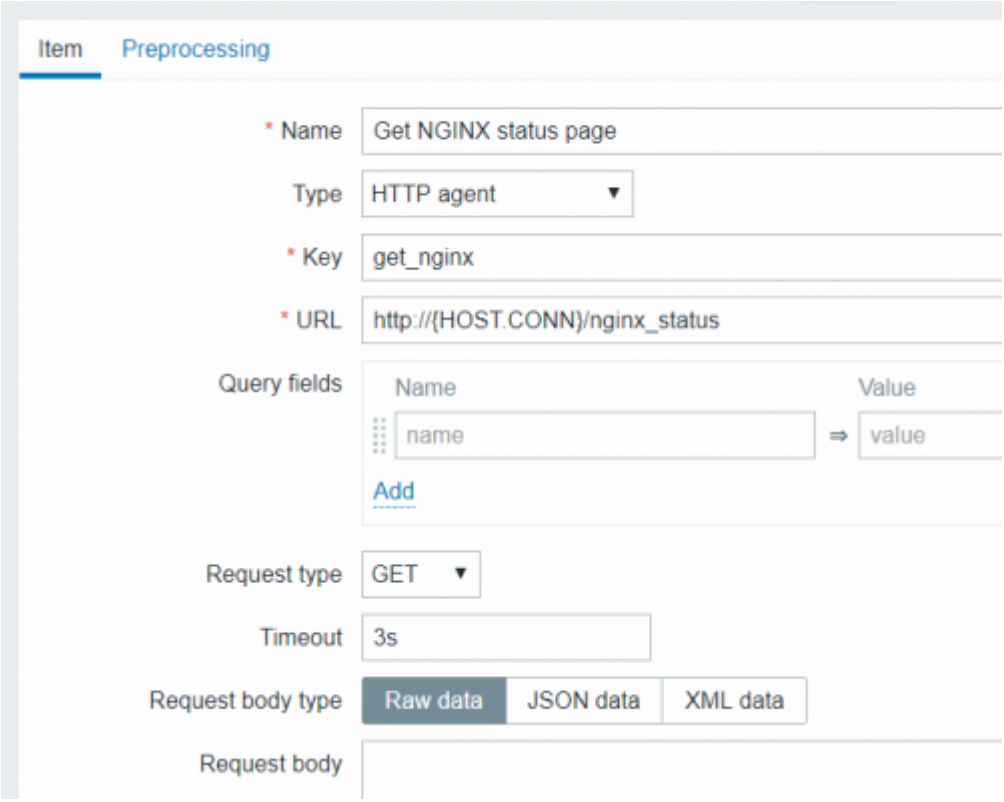


Host	Name	Inter...	History	Trends	Type	Last check	Last value
weather	**Weather** (8 Items)						
	Get weather get_weather.http	10m	1d		HTTP agent	2018-05-17 01:23:45	{"body":{"coord":{"lon...
	Get weather HTTP response code get_weather.http_code		7d	0	Depende...	2018-05-17 01:23:45	OK (200)
	Humidity humidity		90d	365d	Depende...	2018-05-17 01:23:45	66 %
	Temperature temp		90d	365d	Depende...	2018-05-17 01:23:45	15.14 C
	Weather weather		90d		Depende...	2018-05-17 01:23:45	Clouds
	Weather condition id weather.condition.id		7d	0	Depende...	2018-05-17 01:23:45	801
	Weather description weather.description		90d		Depende...	2018-05-17 01:23:45	few clouds
	Wind speed wind.speed		90d	365d	Depende...	2018-05-17 01:23:45	1.86 m/s

Example 5

Connecting to Nginx status page and getting its metrics in bulk.

- Configure Nginx following the official guide.
- Configure a master item for bulk data collection:

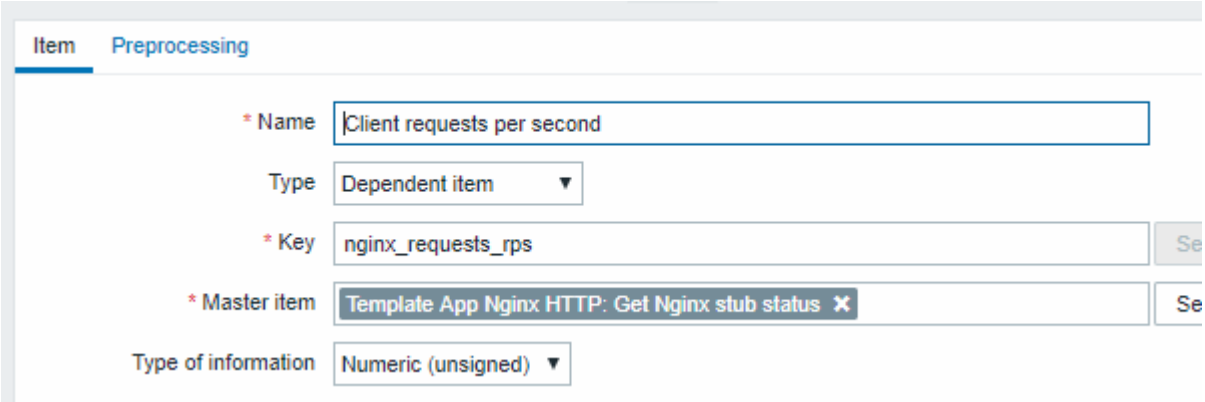


Sample Nginx stub status output:

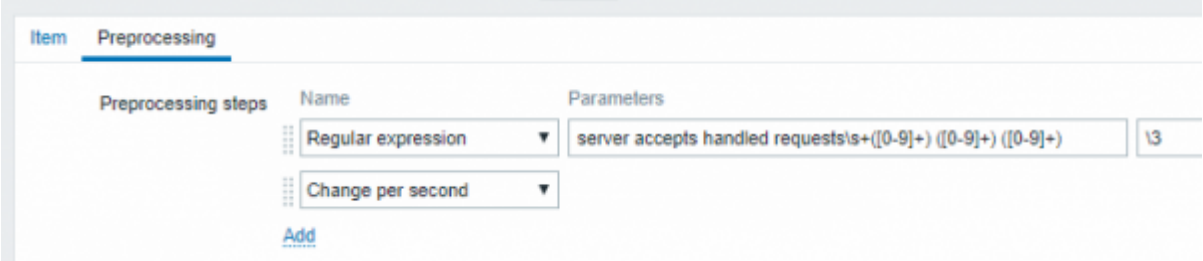
```
Active connections: 1 Active connections:
server accepts handled requests
 52 52 52
Reading: 0 Writing: 1 Waiting: 0
```

The next task is to configure dependent items that extract data.

- Configure a sample dependent item for requests per second:



- Sample dependent item value preprocessing with regular expression:



- Check the complete result from stub module in *Latest data*: