

Pessoal tudo bem com todos? Espero que sim.

Peço desculpas por não ter feito mais nenhum post mas final de ano a correria fica imensa como sabem, principalmente para nós que trabalhamos com TI.

Bem, já que tivemos esse tempo parados vamos retomar com um assunto muito legal. O monitoramento de APIs direto no Zabbix sem precisar usar nenhum Script para processar os dados.

Como todos nós sabemos, monitorar o ambiente produtivo é uma questão de sucesso; imaginem o seu usuário acessando um aplicativo qualquer e a API ao qual o mesmo se conecta está fora ou com algum problema de acesso, provavelmente o seu “cliente” não vai ficar feliz e vai te ligar bem chateado.

Então que tal aproveitar que a nova versão do Zabbix 4.0 tem suporte a monitores/sensores HTTP?

Ok, sem mais papo vamos colocar a mão na massa.

## **visão global**

Este tipo de item permite a pesquisa de dados usando o protocolo HTTP / HTTPS. O trapping também é possível usando o remetente Zabbix ou o protocolo do emissor Zabbix.

A verificação do item HTTP é executada pelo servidor Zabbix. No entanto, quando os hosts são monitorados por um proxy Zabbix, as verificações de itens HTTP são executadas pelo proxy.

As verificações de item HTTP não requerem nenhum agente em execução em um host que está sendo monitorado.

O agente HTTP suporta HTTP e HTTPS. O Zabbix seguirá opcionalmente redirecionamentos (veja a opção Seguir redireciona abaixo). O número máximo de redirecionamentos é codificado para 10 (usando a opção cURL CURLOPT\_MAXREDIRS).

## Configuração

Para configurar um item HTTP:

- Vá para: Configuração → Hosts
- Clique nos itens na linha do host
- Clique em Criar item
- Insira os parâmetros do item no formulário

Item Preprocessing

* Name	HTTP agent item												
Type	HTTP agent												
* Key	http_value_search												
* URL	http://localhost:9200/_str/values/_search												
Query fields													
Name	Value												
scroll	10s												
Add													
Request type	POST												
Timeout	3s												
Request body type	Raw data JSON data XML data												
Request body	<pre>{   "query": {     "bool": {       "must": [         "match": {           "termid": 28275         }       ]     }   } }</pre>												
Headers	<table border="1"> <tr> <td>Name</td> <td>Value</td> </tr> <tr> <td>name</td> <td>value</td> </tr> <tr> <td colspan="2">Add</td> </tr> </table>	Name	Value	name	value	Add							
Name	Value												
name	value												
Add													
Required status codes	200												
Follow redirects	<input type="checkbox"/>												
Retrieve mode	Body Headers Body and headers												
Convert to JSON	<input type="checkbox"/>												
HTTP proxy	http://user[:password]@proxy.example.com[:port]												
HTTP authentication	None												
SSL verify peer	<input type="checkbox"/>												
SSL verify host	<input type="checkbox"/>												
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	<table border="1"> <thead> <tr> <th>Type</th> <th>Interval</th> <th>Period</th> <th>Action</th> </tr> </thead> <tbody> <tr> <td>Flexible</td> <td>Scheduling</td> <td>50s</td> <td>1-7.00.00-24.00</td> </tr> <tr> <td colspan="4">Add</td> </tr> </tbody> </table>	Type	Interval	Period	Action	Flexible	Scheduling	50s	1-7.00.00-24.00	Add			
Type	Interval	Period	Action										
Flexible	Scheduling	50s	1-7.00.00-24.00										
Add													
* History storage period	90d												
* Trend storage period	365d												
Show value	As Is												
Enable trapping	<input checked="" type="checkbox"/>												
Allowed hosts	104.24.103.152												
New application													

Todos os campos de entrada obrigatórios estão marcados com um asterisco vermelho.

## Exemplos

## Exemplo 1

Envie solicitações GET simples para recuperar dados do Prometheus.

Aqui temos um exemplo da chamada que será usada (alguns dados serão ocultos pois trata de um serviço em produção).

The screenshot shows the Zabbix 'Items' configuration screen. At the top, there are tabs for 'All hosts / Azure-API' and 'Enabled'. Below these are buttons for 'ZBX', 'SNMP', 'JMX', 'IPMI', 'Applications 2', 'Items 2' (which is selected), 'Triggers', 'Graphs', 'Discovery rules', and 'Web scenarios'. The main area is titled 'Item' and contains the following fields:

- \* Name: Prometheus
- Type: HTTP agent
- \* Key: Prometheus\_teste
- \* URL: https://prometheus:9090/api/v1/query\_range?quer [Parse]
- Query fields:

Name	Value
name	value

[Add](#) [Remove](#)
- Request type: GET
- Timeout: 60s

Para configurar devemos preencher os seguintes campos:

Name (Nome para o sensor)

Type (definir como HTTP agent)

Key (coloque um nome de chave, recomendo usar o nome do sensor + alguma referência para o mesmo)

URL (endereço que vai ser chamado API)

Request type (defina conforme necessidade da API)

Timeout (recomendo avaliar o tempo que é necessário para carregar os dados)

Headers	Name	Value	Action	
	<input type="text" value="secretkey"/>	<input type="text" value="67c9WfDmpXGzC7Raf103Lwvqy5m0c"/> <a href="#">Remove</a>		
	<a href="#">Add</a>			
Required status codes	<input type="text"/>			
Follow redirects	<input type="checkbox"/>			
Retrieve mode	<a href="#">Body</a>	<a href="#">Headers</a>	<a href="#">Body and headers</a>	
Convert to JSON	<input type="checkbox"/>			
HTTP proxy	<input type="text" value="http://[user[:password]@]proxy.example.com[:port]"/>			
HTTP authentication	<a href="#">None</a> <a href="#">▼</a>			
SSL verify peer	<input type="checkbox"/>			
SSL verify host	<input type="checkbox"/>			
SSL certificate file	<input type="text"/>			
SSL key file	<input type="text"/>			
SSL key password	<input type="text"/>			
* Host interface	<a href="#">0.0.0.0 : 10050</a> <a href="#">▼</a>			
Type of information	<a href="#">Text</a> <a href="#">▼</a>			
* Update interval	<input type="text" value="1m"/>			
Custom intervals	Type	Interval	Period	Action
	<a href="#">Flexible</a>	<a href="#">Scheduling</a>	<input type="text" value="50s"/>	<input type="text" value="1-7,00:00-24:00"/> <a href="#">Remove</a>
	<a href="#">Add</a>			

Headers (se necessário defina conforme especificação da API usada)

Type of information (Preencha com o tipo de retorno – texto ou número)

Update interval (Intervalo entre coletas)

Editando o Preprocessing

Aqui eu tenho o seguinte retorno a partir da API:

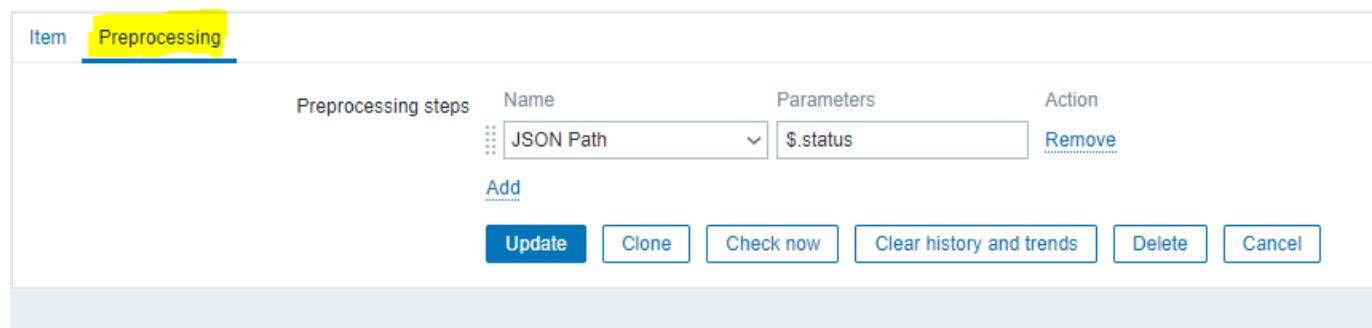


The screenshot shows a JSON editor interface with tabs for 'Pretty', 'Raw', 'Preview', and 'JSON'. The 'JSON' tab is selected. Below the tabs is a code editor area with line numbers 1 through 7. The JSON content is as follows:

```
1 [
2     "status": "success",
3     "data": {
4         "resultType": "matrix",
5         "result": []
6     }
7 ]
```

Sendo assim vou pegar o nó que identifica o status de sucesso da minha aplicação.

No Zabbix:



The screenshot shows the 'Preprocessing' tab of an item configuration in Zabbix. The tab has a yellow background. Below it, there is a table with columns: 'Preprocessing steps', 'Name', 'Parameters', and 'Action'. A single row is present with the following values:

Preprocessing steps	Name	Parameters	Action
JSON Path	\$.status		<a href="#">Remove</a>

Below the table are buttons for 'Add', 'Update', 'Clone', 'Check now', 'Clear history and trends', 'Delete', and 'Cancel'.

Acesse a opção (Guia) Preprocessing.

Preencha os steps

Name -> JSON Path

Parameters -> \$.status

Em parameters é que está a solução desse sensor, para cada nível abaixo devemos colocar da seguinte forma.

Exemplo:

`$.body.exemplo.exemplo1` (aqui faremos a leitura do valor em exemplo1)

Após colocar as informações necessárias clique em ADD.

Alguns minutos depois já pode conferir o resultado em “Latest Data”

Prometheus - teste (1 item)					
Prometheus <a href="#">Prometheus_teste</a>	1m	90d	HTTP agent	2018-11-27 11:24:25	success

Acima coloquei um exemplo real de uso (meu caso), mas como também quero que todos aprendam aqui vai um exemplo usando a API do <https://openweathermap.org>

Vamos lá.

Primeiro vamos configurar os campos:

* Name	<input type="text" value="Informacoes_tempo"/>								
Type	<input type="text" value="HTTP agent"/>								
* Key	<input type="text" value="Informacoes_tempo"/> <input type="button" value="Select"/>								
* URL	<input type="text" value="https://samples.openweathermap.org/data/2.5/weather?"/> <input type="button" value="Parse"/>								
Query fields	<table border="1"><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td>lat</td><td>35</td></tr><tr><td>lon</td><td>139</td></tr><tr><td>appid</td><td>b6907d289e10d714a6e88b30761fae2</td></tr></tbody></table>	Name	Value	lat	35	lon	139	appid	b6907d289e10d714a6e88b30761fae2
Name	Value								
lat	35								
lon	139								
appid	b6907d289e10d714a6e88b30761fae2								
Request type	<input type="text" value="GET"/>								
Timeout	<input type="text" value="60s"/>								

Devemos colocar o Name, Key, URL e aqui vamos definir o Query fields.

Em URL coloque: <https://samples.openweathermap.org/data/2.5/weather?>

Em Query fields: lat = 35

lon = 139

appid = b6907d289e10d714a6e88b30761fae22

Request type: GET

Type of information: Numeric (float)

Agora Ajustando o Preprocessing

The screenshot shows a user interface for configuring preprocessing steps. At the top, there are two tabs: 'Item' and 'Preprocessing'. The 'Preprocessing' tab is selected. Below the tabs, there is a table with four columns: 'Preprocessing steps', 'Name', 'Parameters', and 'Action'. Under 'Preprocessing steps', there is a dropdown menu set to 'JSON Path'. In the 'Name' column, the value '\$.main.temp' is entered. In the 'Parameters' column, there is a 'Remove' button. Below the table, there are several buttons: 'Add' (highlighted in blue), 'Update', 'Clone', 'Check now', 'Clear history and trends', 'Delete', and 'Cancel'.

O mapeamento deve ficar dessa forma: \$.main.temp

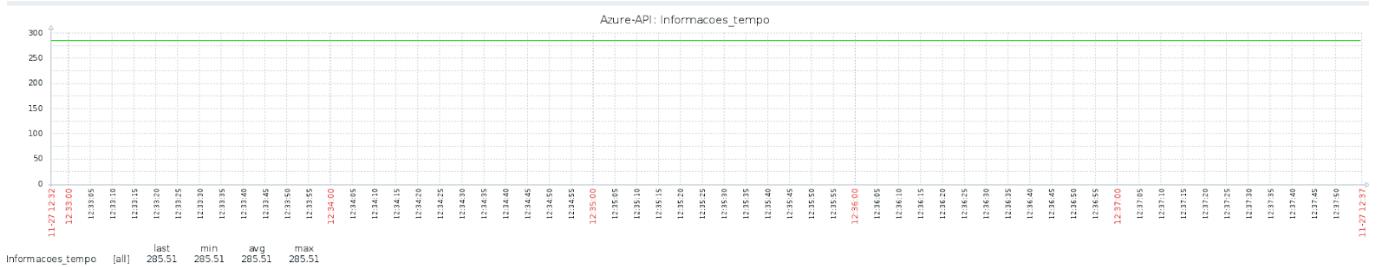
A informação capturada será:

A screenshot of a JSON object. The object has a 'base' key with the value 'stations'. The 'main' key contains a nested object. The 'temp' key in this nested object is highlighted with a yellow box. The entire JSON structure is as follows:

```
    ,
    "base": "stations",
    "main": {
        "temp": 285.514,
        "pressure": 1013.75,
        "humidity": 100,
        "temp_min": 285.514,
        "temp_max": 285.514,
        "sea_level": 1023.22,
        "grnd_level": 1013.75
    }
}
```

Após finalizado clique em Add.

Alguns minutos após pode conferir em “Latest Data” e você terá todos os dados, como estamos pegando um valor numérico o gráfico já ficará configurado conforme imagem abaixo.



Bem pessoal, espero que estas informações sejam úteis e possam ajudar a todos.

E como sempre qualquer dúvida fico a disposição.

Abraço a todos.