9. Web monitoring

Overview

With Zabbix you can check several availability aspects of web sites.

To perform web monitoring Zabbix server must be initially configured with cURL (libcurl) support.

To activate web monitoring you need to define web scenarios. A web scenario consists of one or several HTTP requests or "steps". The steps are periodically executed by Zabbix server in a pre-defined order. If a host is monitored by proxy, the steps are executed by the proxy.

Since Zabbix 2.2 web scenarios are attached to hosts/templates in the same way as items, triggers, etc. That means that web scenarios can also be created on a template level and then applied to multiple hosts in one move.

The following information is collected in any web scenario:

- average download speed per second for all steps of whole scenario
- number of the step that failed
- last error message

The following information is collected in any web scenario step:

- download speed per second
- response time
- response code

For more details, see web monitoring items.

Data collected from executing web scenarios is kept in the database. The data is automatically used for graphs, triggers and notifications.

Zabbix can also check if a retrieved HTML page contains a pre-defined string. It can execute a simulated login and follow a path of simulated mouse clicks on the page.

Zabbix web monitoring supports both HTTP and HTTPS. When running a web scenario, Zabbix will optionally follow redirects (see option *Follow redirects* below). Maximum number of redirects is hard-coded to 10 (using cURL option CURLOPT_MAXREDIRS). All cookies are preserved during the execution of a single scenario.

See also known issues for web monitoring using HTTPS protocol.

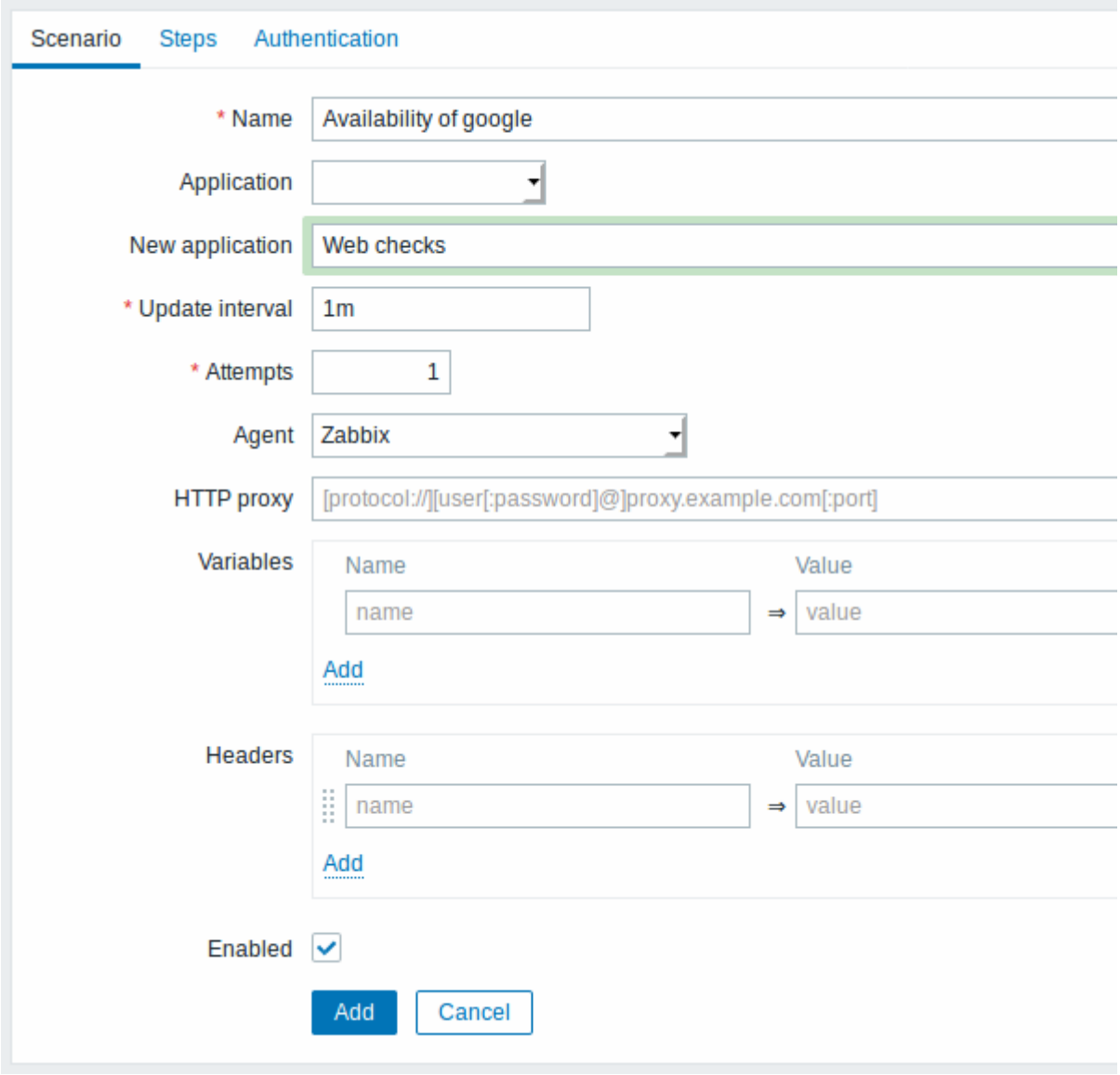
Configuring a web scenario

To configure a web scenario:

- Go to: *Configuration* → *Hosts* (or *Templates*)
- Click on *Web* in the row of the host/template

- Click on *Create scenario* to the right (or on the scenario name to edit an existing scenario)
- Enter parameters of the scenario in the form

The **Scenario** tab allows you to configure the general parameters of a web scenario.



All mandatory input fields are marked with a red asterisk.

Scenario parameters:

Parameter	Description
Host	Name of the host/template that the scenario belongs to.
Name	Unique scenario name. User macros and {HOST.*} macros are supported, since Zabbix 2.2.
Application	Select an application the scenario will belong to. Web scenario items will be grouped under the selected application in *Monitoring* → *Latest data*.
New application	Enter the name of a new application for the scenario.

Parameter	Description
Update interval	How often the scenario will be executed. Time suffixes are supported, e.g. 30s, 1m, 2h, 1d, since Zabbix 3.4.0. User macros are supported, since Zabbix 3.4.0. *Note* that if a user macro is used and its value is changed (e.g. 5m → 30s), the next check will be executed according to the previous value (farther in the future with the example values).
Attempts	The number of attempts for executing web scenario steps. In case of network problems (timeout, no connectivity, etc) Zabbix can repeat executing a step several times. The figure set will equally affect each step of the scenario. Up to 10 attempts can be specified, default value is 1. *Note*: Zabbix will not repeat a step because of a wrong response code or the mismatch of a required string. This parameter is supported starting with *Zabbix 2.2*.
Agent	Select a client agent. Zabbix will pretend to be the selected browser. This is useful when a website returns different content for different browsers. User macros can be used in this field, *starting with Zabbix 2.2*.
HTTP proxy	You can specify an HTTP proxy to use, using the format `[protocol://][username[:password]@]proxy.mycompany.com[:port]`. This sets the CURLOPT_PROXY cURL option. 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. User macros can be used in this field. This parameter is supported starting with *Zabbix 2.2*.
Variables	Variables that may be used in scenario steps (URL, post variables). They have the following format: **{macro1}**=value1 **{macro2}**=value2 **{macro3}**=regex:<regular expression> For example: {username}=Alexei {password}=kj3h5kJ34bd {hostid}=regex:hostid is ([0-9]+) The macros can then be referenced in the steps as {username}, {password} and {hostid}. Zabbix will automatically replace them with actual values. Note that variables with `regex:` need one step to get the value of the regular expression so the extracted value can only be applied to the step after. If the value part starts with `regex:` then the part after it is treated as a regular expression that searches the web page and, if found, stores the match in the variable. At least one subgroup must be present so that the matched value can be extracted. Regular expression match in variables is supported *since Zabbix 2.2*. User macros and {HOST.*} macros are supported, since Zabbix 2.2. Variables are automatically URL-encoded when used in query fields or form data for post variables, but must be URL-encoded manually when used in raw post or directly in URL.

Parameter	Description
Headers	Custom HTTP headers that will be sent when performing a request. Headers should be listed using the same syntax as they would appear in the HTTP protocol, optionally using some additional features supported by the CURLOPT_HTTPHEADER cURL option. For example: Accept-Charset=utf-8 Accept-Language=en-US Content-Type=application/xml; charset=utf-8 User macros and {HOST.*} macros are supported. Specifying custom headers is supported *starting with Zabbix 2.4*.
Enabled	The scenario is active if this box is checked, otherwise - disabled.

Note that when editing an existing scenario, two extra buttons are available in the form:

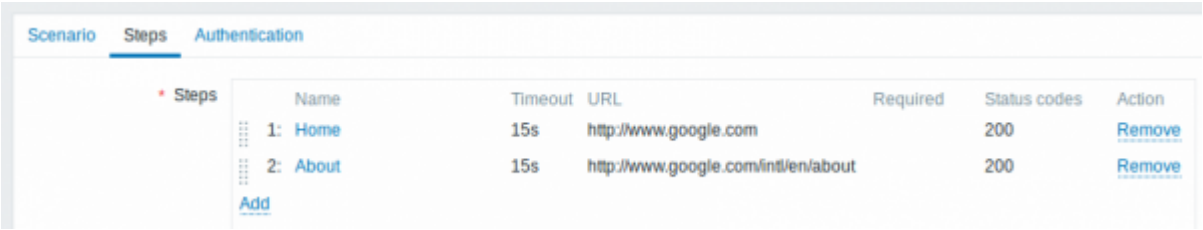
Clone	Create another scenario based on the properties of the existing one.
Clear history and trends	Delete history and trend data for the scenario. This will make the server perform the scenario immediately after deleting the data.

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

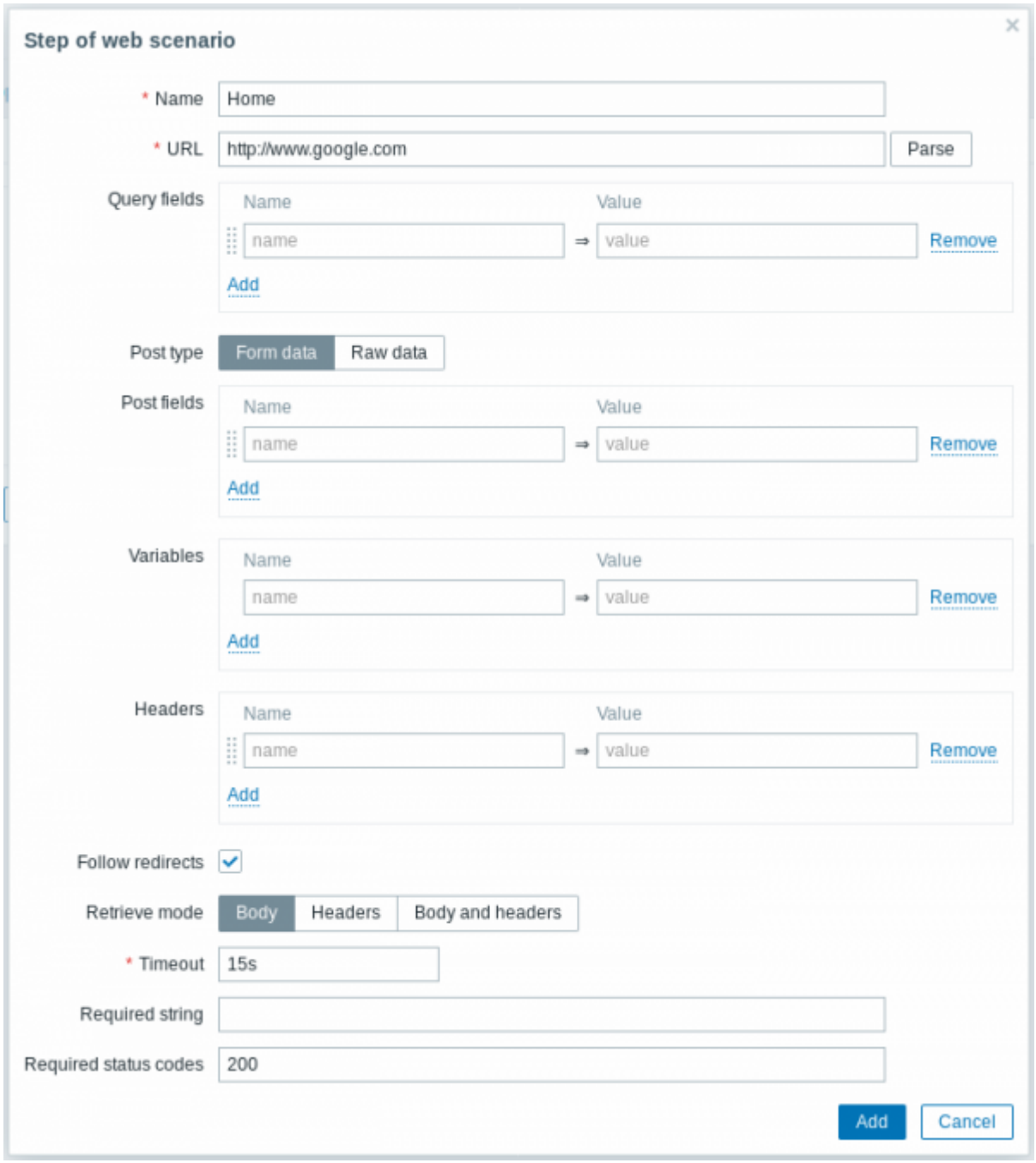
For HTTP checks - set the **http_proxy** environment variable for the Zabbix server user. For example, *http_proxy=http://proxy_ip:proxy_port*.

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

The **Steps** tab allows you to configure the web scenario steps. To add a web scenario step, click on *Add* in the *Steps* block.



Configuring steps



Step parameters:

Parameter	Description
Name	Unique step name. User macros and {HOST.*} macros are supported, since Zabbix 2.2.

Parameter	Description
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 since Zabbix 3.4. They are automatically punycode-converted to ASCII when executing the web scenario step. 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. Variables can be used in the URL, using the {macro} syntax. Variables can be URL-encoded manually using a {{macro}.urlencode()} syntax. User macros and {HOST.*} macros are supported, since Zabbix 2.2. Limited to 2048 characters *starting with Zabbix 2.4*.
Query fields	HTTP GET variables for the URL. Specified as attribute and value pairs. Values are URL-encoded automatically. Values from scenario variables, user macros or {HOST.*} macros are resolved and then URL-encoded automatically. Using a {{macro}.urlencode()} syntax will double URL-encode them. User macros and {HOST.*} macros are supported since Zabbix 2.2.
Post	HTTP POST variables. In **Form data** mode, specified as attribute and value pairs. Values are URL-encoded automatically. Values from scenario variables, user macros or {HOST.*} macros are resolved and then URL-encoded automatically. In **Raw data** mode, attributes/values are displayed on a single line and concatenated with a **&** symbol. Raw values can be URL-encoded/decoded manually using a {{macro}.urlencode()} or {{macro}.urldecode()} syntax. For example: id=2345&userid={user} If {user} is defined as a variable of the web scenario, it will be replaced by its value when the step is executed. If you wish to URL-encode the variable, substitute {user} with {{user}.urlencode()}. User macros and {HOST.*} macros are supported, since Zabbix 2.2.
Variables	Step-level variables that may be used for GET and POST functions. Specified as attribute and value pairs. Step-level variables override scenario-level variables or variables from the previous step. However, the value of a step-level variable only affects the step after (and not the current step). They have the following format: **{macro}**=value **{macro}**=regex:<regular expression> For more information see variable description on the scenario level. Having step-level variables is supported since Zabbix 2.2. Variables are automatically URL-encoded when used in query fields or form data for post variables, but must be URL-encoded manually when used in raw post or directly in URL.
Headers	Custom HTTP headers that will be sent when performing a request. Specified as attribute and value pairs. Headers on the step level will overwrite the headers specified for the scenario. For example, setting a 'User-Agent' attribute with no value will remove the User-Agent value set on scenario level. User macros and {HOST.*} macros are supported. This sets the CURLOPT_HTTPHEADER cURL option. Specifying custom headers is supported *starting with Zabbix 2.4*.

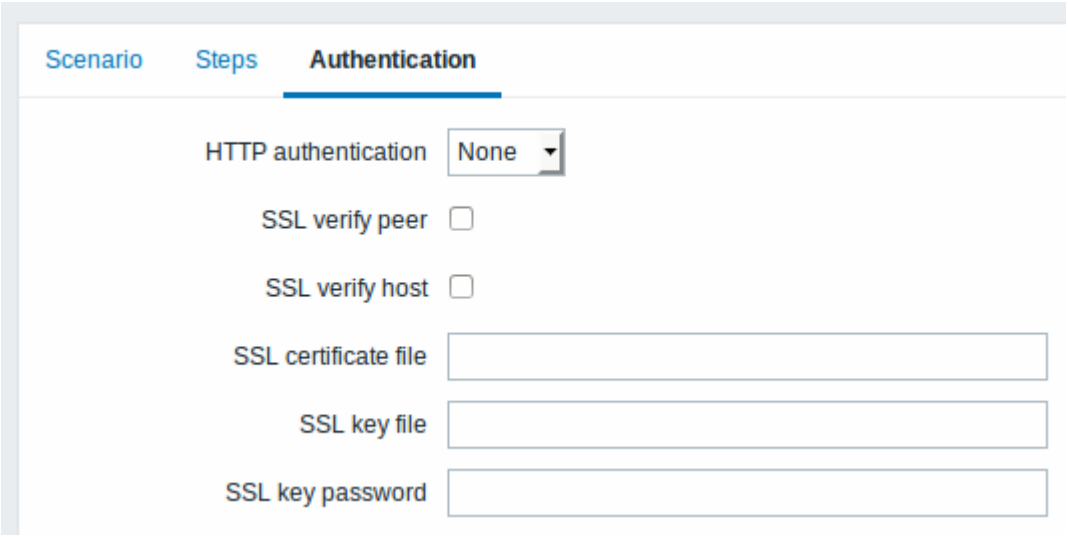
Parameter	Description
Follow redirects	Mark the checkbox to follow HTTP redirects. This sets the CURLOPT_FOLLOWLOCATION cURL option. This option is supported *starting with Zabbix 2.4*.
Retrieve mode	Select the retrieve mode: **Body** - retrieve only body from the HTTP response **Headers** - retrieve only headers from the HTTP response **Body and headers** - retrieve body and headers from the HTTP response This option is supported *since Zabbix 4.2*.
Timeout	Zabbix will not spend more than the set amount of time on processing the URL (maximum is 1 hour). Actually this parameter defines the maximum time for making connection to the URL and maximum time for performing an HTTP request. Therefore, Zabbix will not spend more than **2 x Timeout** seconds on the step. Time suffixes are supported, e.g. 30s, 1m, 1h. User macros are supported.
Required string	Required regular expression pattern. Unless retrieved content (HTML) matches the required pattern the step will fail. If empty, no check on required string is performed. For example: Homepage of Zabbix Welcome.*admin *Note*: Referencing regular expressions created in the Zabbix frontend is not supported in this field. User macros and {HOST.*} macros are supported, since Zabbix 2.2.
Required status codes	List of expected HTTP status codes. If Zabbix gets a code which is not in the list, the step will fail. If empty, no check on status codes is performed. For example: 200,201,210-299 User macros are supported since Zabbix 2.2.

Any changes in web scenario steps will only be saved when the whole scenario is saved.

See also a real-life example of how web monitoring steps can be configured.

Configuring authentication

The **Authentication** tab allows you to configure scenario authentication options.



Authentication parameters:

Parameter	Description
Authentication	Authentication options. **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. User macros can be used in user and password fields, *starting with Zabbix 2.2*.
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. This option is supported *starting with Zabbix 2.4*.
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. This option is supported *starting with Zabbix 2.4*.
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. `HOST.*` macros and user macros can be used in this field. This sets the CURLOPT_SSLCERT cURL option. This option is supported *starting with Zabbix 2.4*.
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. `HOST.*` macros and user macros can be used in this field. This sets the CURLOPT_SSLKEY cURL option. This option is supported *starting with Zabbix 2.4*.
SSL key password	SSL private key file password. User macros can be used in this field. This sets the CURLOPT_KEYPASSWD cURL option. This option is supported *starting with Zabbix 2.4*.

[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:

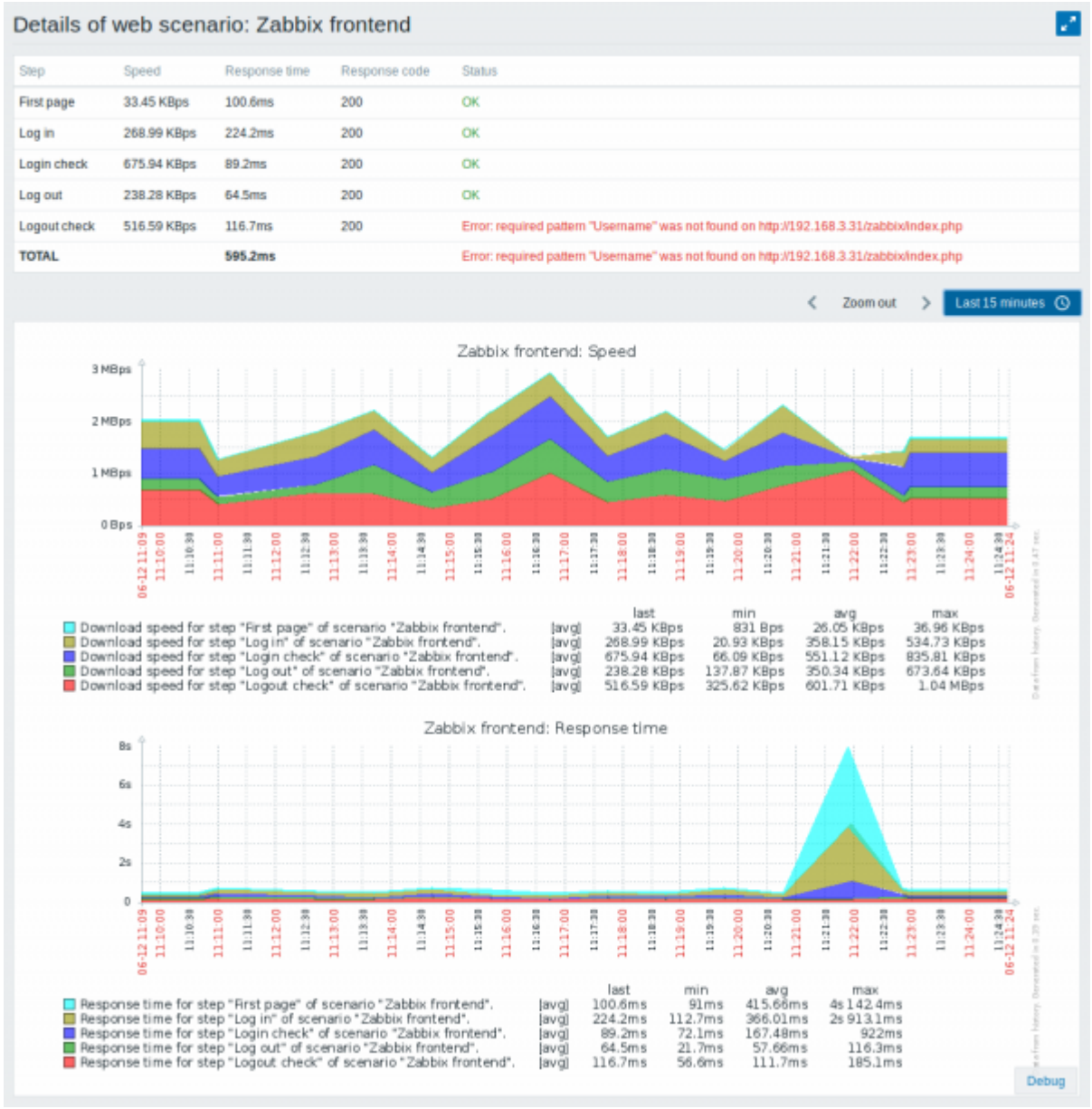
```
openssl pkcs12 -in ssl-cert.p12 -clcerts -nokeys -out ssl-cert.pem
openssl pkcs12 -in ssl-cert.p12 -nocerts -nodes  -out ssl-cert.key
```

Zabbix server picks up changes in certificates without a restart.
If you have client certificate and private key in a single file just specify it in a “SSL certificate file” field and leave “SSL key file” field empty. The certificate and key must still be in PEM format. Combining certificate and key is easy:

```
cat client.crt client.key > client.pem
```

Display

To view detailed data of defined web scenarios, go to *Monitoring* → *Web* or *Latest data*. Click on the scenario name to see more detailed statistics.



An overview of web monitoring scenarios can be viewed in *Monitoring* → *Dashboard*.

Extended monitoring

Sometimes it is necessary to log received HTML page content. This is especially useful if some web scenario step fails. Debug level 5 (trace) serves that purpose. This level can be set in server and proxy configuration files or using a runtime control option (`-R log_level_increase="http poller,N"`, where N is the process number). The following examples demonstrate how extended monitoring can be started provided debug level 4 is already set:

```
Increase log level of all http pollers:
shell> zabbix_server -R log_level_increase="http poller"
```

```
Increase log level of second http poller:
```

```
shell> zabbix_server -R log_level_increase="http poller,2"
```

If extended web monitoring is not required it can be stopped using the `-R log_level_decrease` option.