Monitoring WEB

Go back to Pandora FMS documentation index

Classic Web Monitoring

Introduction

E In the Enterprise version, it is possible to monitor a Web using the WEB Server component(Goliat Server).

This feature comes from an old project of the founder of Pandora FMS: Goliat F.I.S.T. It was an open source project to perform dynamic load audits on web services. You can still find the source code (from 2002) whose updating stopped around 2010.

In Pandora FMS, it works as an independent server, similar to the Network server, WMI Server or remote plugin server. This system operates under the principle of *web transaction*, where each complete transaction against one or more WEB pages is defined by one or more consecutive steps, which must be successfully concluded in order to consider the transaction properly completed. The execution of a *web transaction* faithfully reproduces the complete browsing process that can include aspects such as authenticating yourself in a form, clicking on a menu option or filling in a form, verifying that each step returns a specific text string.

Any failure at any point in the process would result in a checking failure. The complete transaction includes the downloading of all resources (graphics, animations, etc.) that the actual navigation includes. In addition to performing response time and performance checks, it is possible to extract values from the web pages and then process them.

Goliat is able to monitor both HTTP and HTTPS in a transparent way for the user, supports session management through cookies, parameter passage, and of course, downloading the resources associated with each page. It also has **important limitations such as the dynamic management of javascript at runtime**. For more complex web transactions, Pandora FMS has another much more powerful (and complex) component called WUX monitoring).

Installation and Configuration

To be able to use Goliat, activate it in the Pandora FMS Enterprise server:

```
webserver 1
```

Depending on the number of requests you want to make, you may have to increase the number of threads and the default timeout:

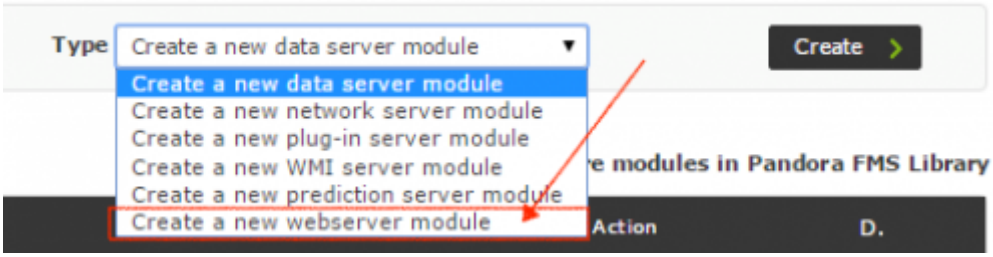
```
web_threads 1
web_timeout 60
```

There is an advanced configuration token that will allow you to change the type of library you use under Goliat, LWP or CURL. CURL is used by default, but you can change it at any time:

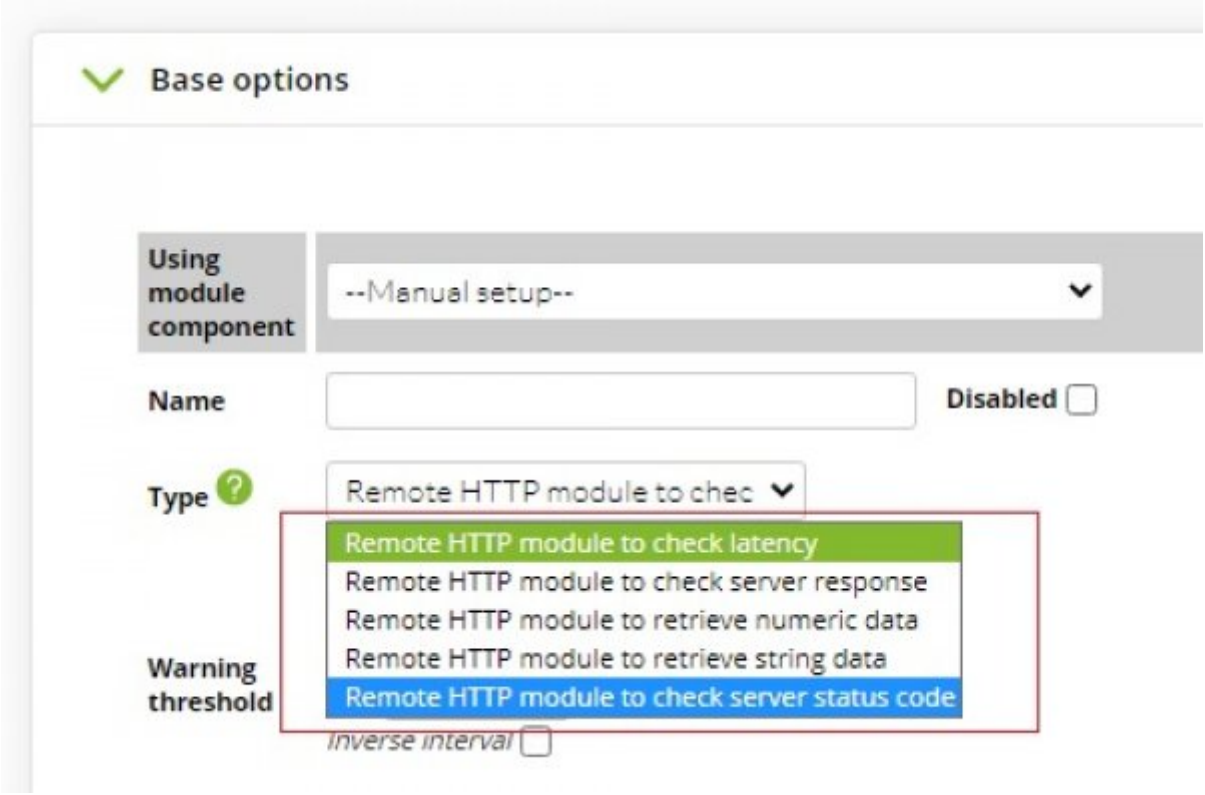
```
web_engine curl
```

Creating Web Modules

To monitor a web page remotely, once the agent is created, click on the **modules** tab. Then, select **Create a new webserver module** and click on **Create**:



Click on **Create** to see a form where to fill out the appropriate fields to monitor a web page.



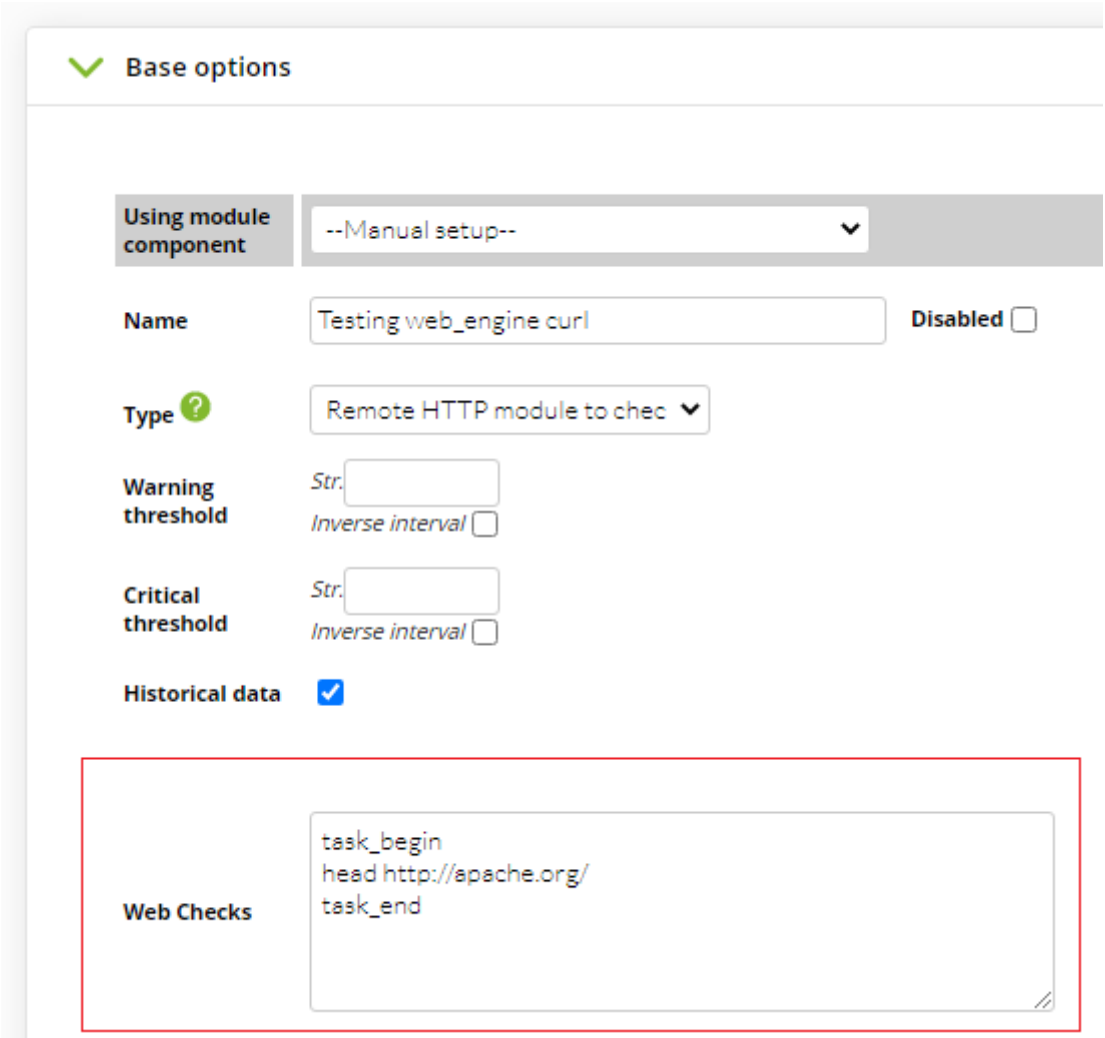
There are several check types to choose from:

- **Remote HTTP module to check latency**: It obtains the total time that elapses from the first request until the last one is checked (in a WEB test there are one or more intermediate requests that complete the transaction). If it is defined in the check definition that the transaction must be carried out more than once, the average time of each request is used.
- **Remote HTTP module to check server response**: It shows the values 1 (OK) or 0 (CRITICAL) as a result from checking the entire transaction. If there are several attempts, but at least one of them has failed, the test as a whole is also considered to have failed. Precisely the number of attempts is sometimes used to avoid false positives. In case you may want to perform the test several times in case of a failure, use the *retries* field in advanced fields.
- **Remote HTTP module to retrieve numeric data**: It retrieves a numeric value from an HTTP response using a regular expression.
- **Remote HTTP module to retrieve string data**: It retrieves a string from an HTTP response using a regular expression.
- **Remote HTTP module to check server status code:**: By means of the **curl** tool properly enabled with the `web_engine curl` configuration token, HTTP headers can be returned.

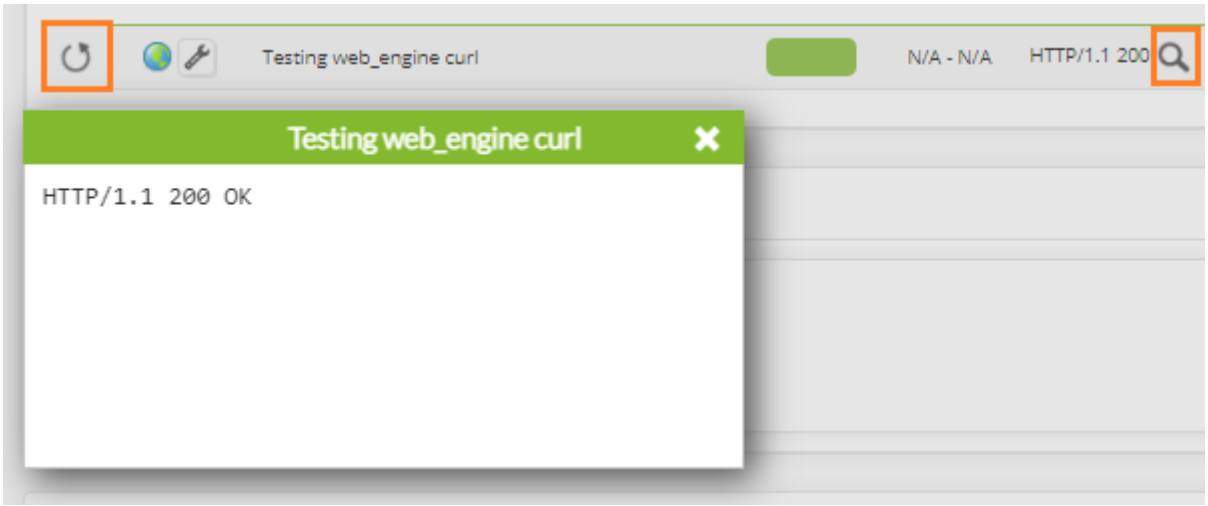
Web Checks: This essential field defines the WEB check to be performed. This is defined in one or more steps, or simple requests. These simple requests must be written in a special format in the Web checks field. Checks start with the tab page `task_begin` and end with the tab page `task_end`.

A complete example of a simple transaction would be the following:

```
task_begin
head http://apache.org/
task_end
```



After saving, you can force the execution of the Module and visualize its result:



Another example with more commands:

```
task_begin
get http://apache.org/
cookie 0
resource 0
```

```
check_string Apache Software Foundation
task_end
```

In this basic example, it is being checked whether there is a string in a web page. To that end, there is the variable `check_string`. This variable does not allow you to check HTML itself, it only searches for text sub-strings. If you look for “Apache Software Foundation” on the <http://apache.org> and if that text string exists, the check will return OK (if it is *Remote HTTP module to check server response*)

To ensure that a string does not exist on a web page, you can use the variable `check_not_string>`

```
check_not_string Section 3
```

The arguments taken by the `check_string` syntax are not normal text strings, they are “regular expressions”. That is to say, if you search for the string `Pandora FMS (4.0)` you will have to search for it with a regular expression, e.g. `Pandora FMS \(4.0\)`. This allows you to perform much more powerful researches, but you should be aware that any character other than a letter or number will have to be escaped with con `\`.

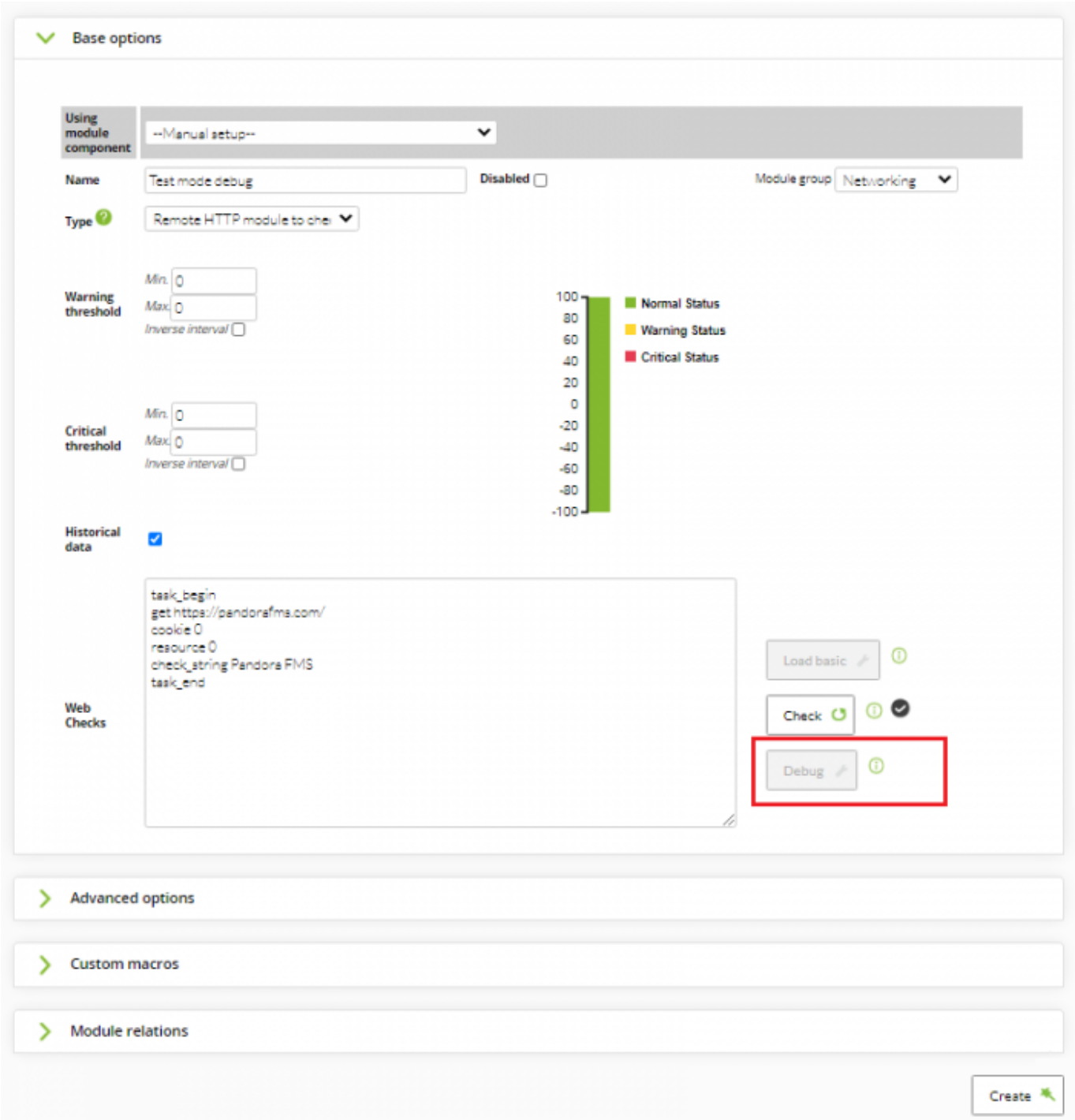
There are several extra variables to check forms:

- **resource (1 or 0)**: It downloads all the web resources (images, videos, etc).
- **cookie (1 or 0)**: It keeps a cookie or an open session for later checks.
- **variable_name**: The name of a variable in a form.
- **variable_value**: The value of the previous variable on the form.

By using these variables, it is possible to send data to forms and check whether they work appropriately or not.

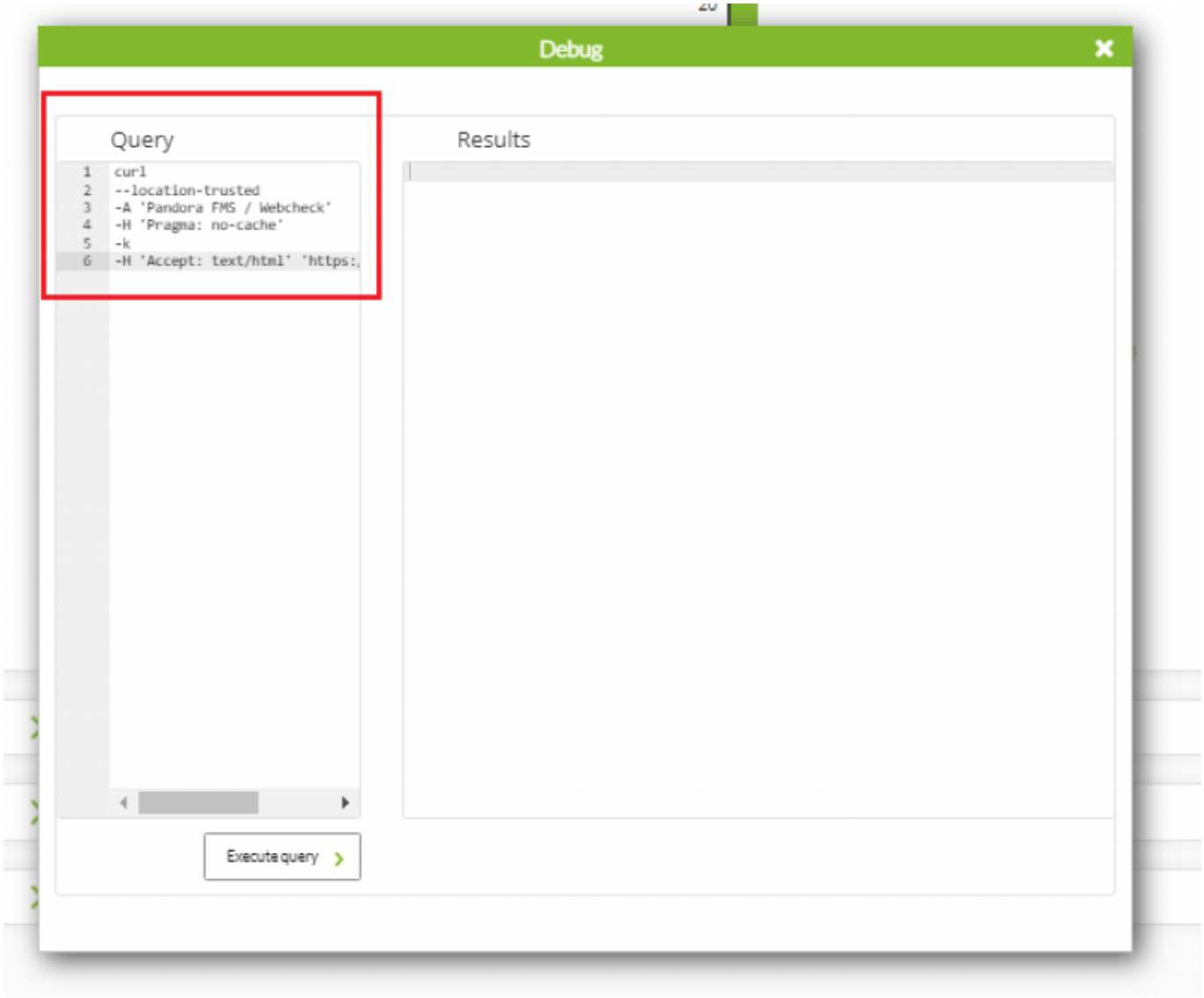
In some specific cases, the domain redirection may **not** work. To solve this problem, create a module that uses the final domain address after all redirections are completed.

For the previous case, the **curl** command has the parameter `-L` in its short version and `–location` in its long version, so when it receives HTTP 3XX redirections, it executes again against the redirected domain. **However, the flexibility of Pandora FMS** allows you to use the debug button.

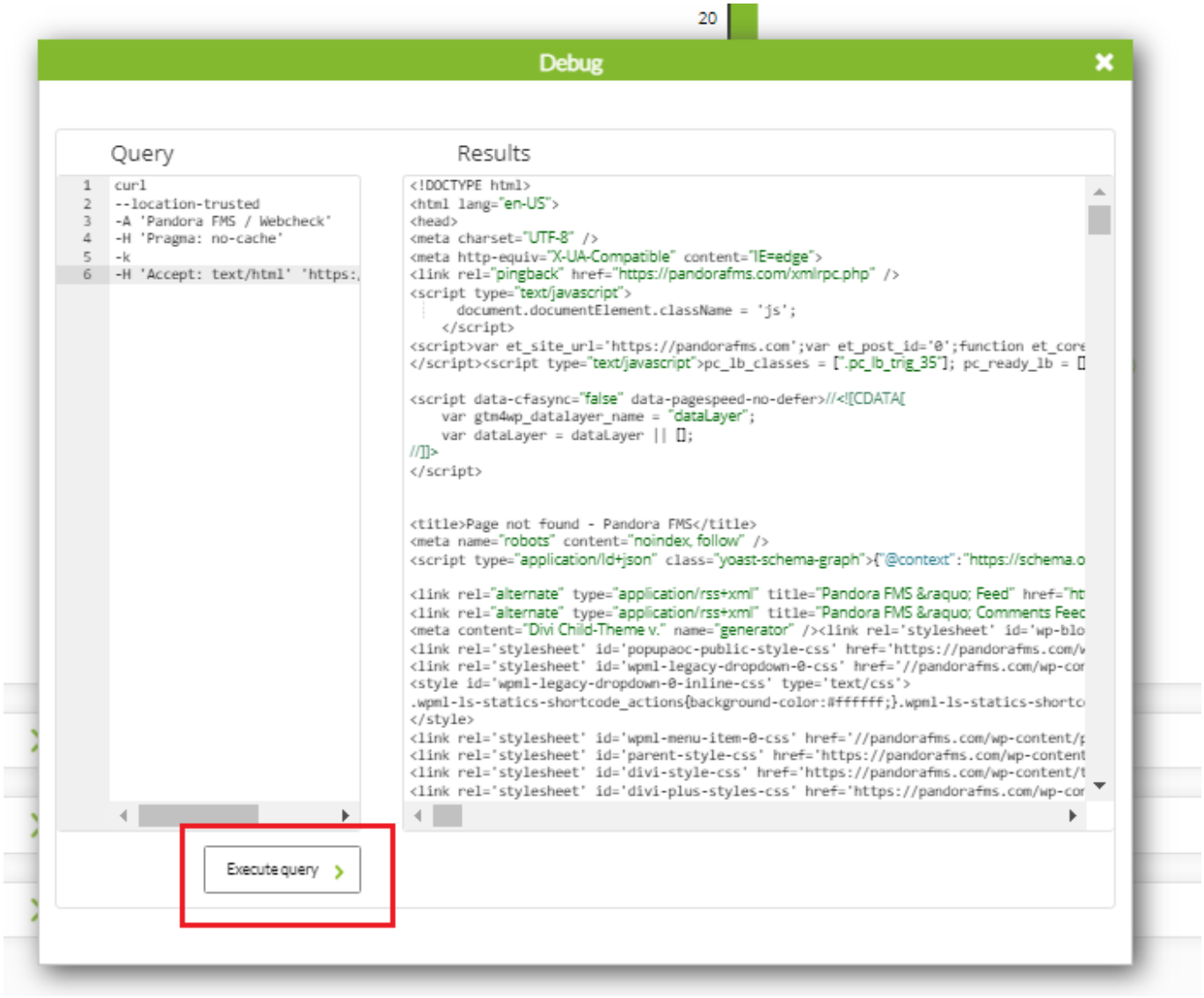


At the moment of creating the Module, it is inactive and you will be able to make use of it after you have done the first check, which you can force to run to save time.

When you modify this Module, click on the **Debug** button and you can enter debug mode to edit the **Query**:



You can execute the query that the module has with the button **Execute query** as well as modify and re-execute it with other values until you get the desired result.



Checking website loading time

If you want to check the latency of a website, select the module type named **Remote HTTP module to check latency**. If you want to find out the latency of the <https://pandorafms.com»

```
task_begin
get https://pandorafms.com
task_end
```

Add the `resource 1` configuration token so that the download time you calculate includes the download of all resources (javascript, CSS, images, etc.), thus calculating a more real data.

The download time of the website is NOT the time it takes to see a web site in a browser, as this usually depends on the loading time of Javascript, and Goliat downloads the javascript, but does not run it.

Website Checks through a Proxy

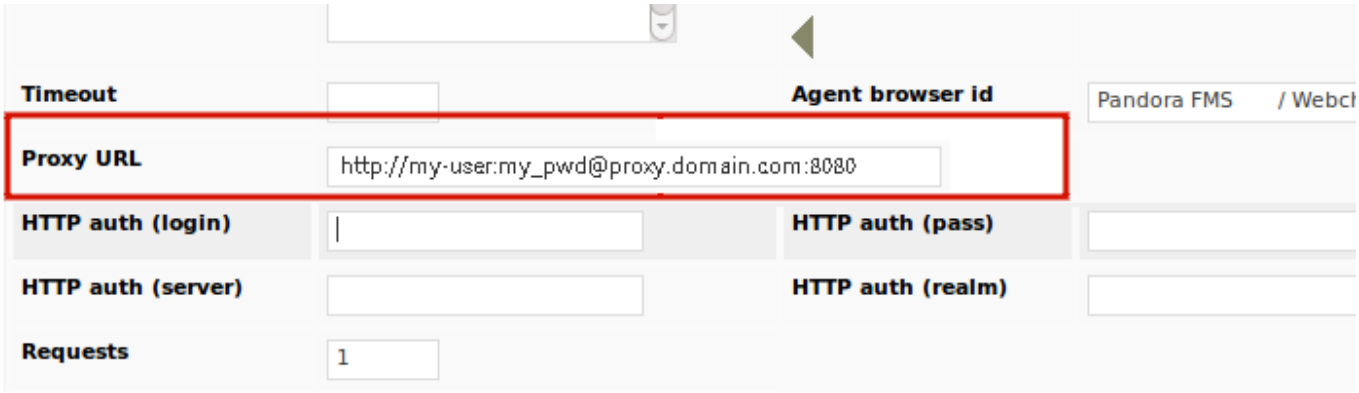
You can also carry out website checks by using a proxy. To configure the proxy, add the proxy URL in the 'Proxy URL' field which is located under **Advanced options**:

An example of the URL could be:

```
http://proxy.domain.com:8080
```

If the proxy requires an authentication, you may use the following where `my-user` is the username and `my_pwd` is the password:

```
http://my-user:my_pwd@proxy.domain.com:8080
```

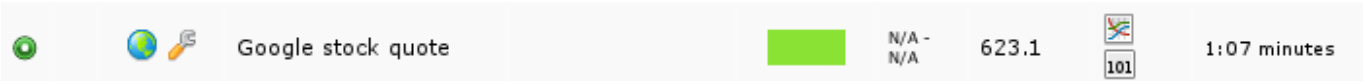


Retrieving data from a website

Sometimes monitoring does not consist of finding out whether a specific Web site is working or how long it takes, but to get a real time value, such as Google's stock market value. To that end, use a **Remote HTTP module to retrieve numeric data** with the appropriate regular expression:

```
task_begin
get http://finance.google.com/finance/info?client=ig&q=NASDAQ%3aGOOG
get_content \d+\.\d+
task_end
```

The output will look like this:



It is also possible to specify a more complex regular expression for collecting data from more complex HTTP responses with the `get_content_advanced` configuration token:

```
task_begin
get http://finance.yahoo.com/q?s = GOOG
get_content_advanced <span id ="yfs_l84_goog">([\d\.]+)</span>
task_end
```

The part of the regular expression defined in `get_content_advanced` must be enclosed in brackets.

To configure the thresholds that will trigger warning or critical status, use the module configuration to verify that the received string matches what is expected.

Website form checking

Web form checking is much more complex than simply checking a text on a website. This example check will use Pandora FMS Console, log in, and verify that it has been able to do it. It verifies a text in the **workspace** section where it shows the data of the user who has logged in. If it is a default console, the admin user contains the description “Admin Pandora”.

To be able to perform this type of checks, you must have the necessary credentials. In addition, go to the page and get the HTML code to be able to see the names of the variables. Then, it is necessary to have a basic knowledge of HTML to understand how Goliat works.

The ideal procedure when designing a WEB transactional test with several steps is to test it step by step, in case something was missed in one of the steps.

The example Console is:

```
http://192.168.70.116/pandora_console/
```

Analyzing the HTML code, it is observed that the variables of the login form are:

- `nick>` user name
- `pass>` user password

The variables `variable_name` and `variable_value` should be used together to validate the form. Pandora FMS Console has by default the `admin` and `pandora` values.

Firstly, access the form, send the user and password and authenticate (determining the success of that authentication will be seen in the following step).

```
task_begin
post http://192.168.70.116/pandora_console/index.php?
login =1
variable_name nick
variable_value admin
variable_name pass
variable_value pandora
```

```
cookie 1
resource 1
task_end
```

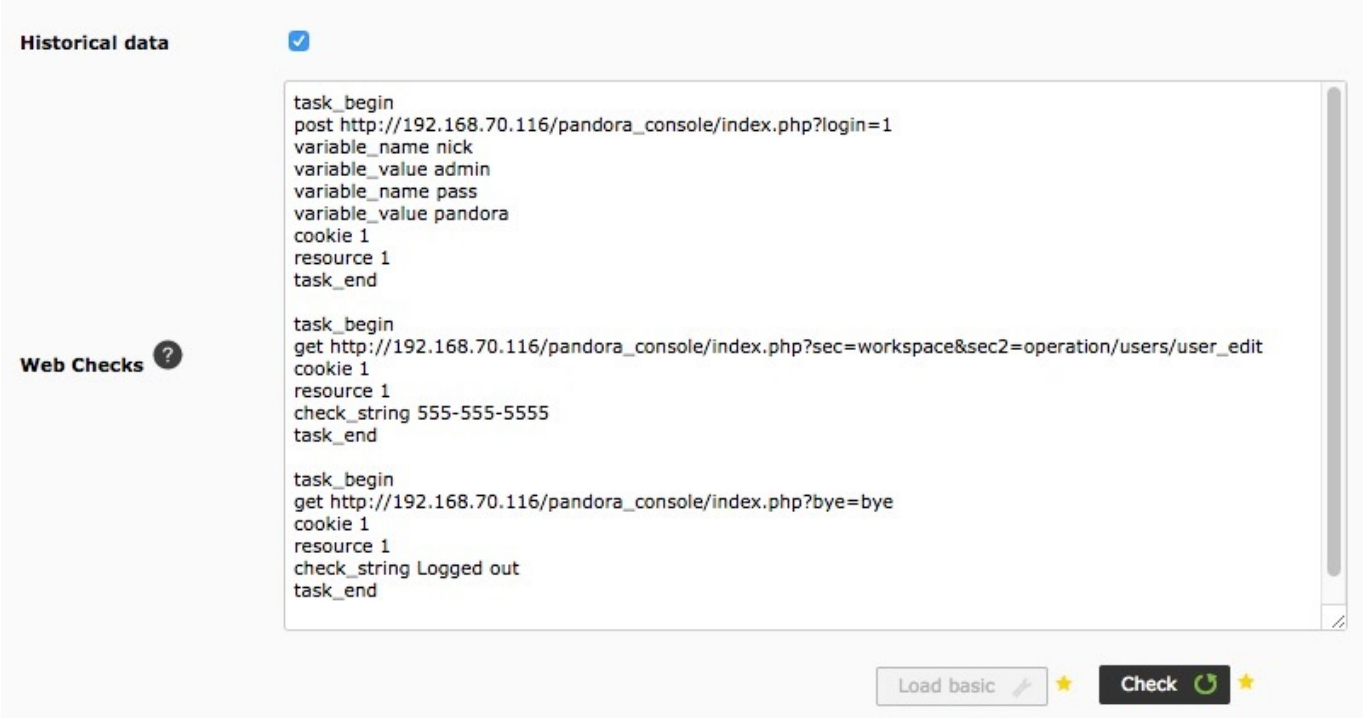
The *token* `cookie 1` is used to keep the persistence of values obtained in the previous step, after a successful authentication. Then go to the page of user details and look for the phone, which by default for the "admin" user is 555-555-555. If you can see it, it means that you have logged in correctly in the console (note the use of the `check_string` command):

```
task_begin
get http://192.168.70.116/pandora_console/index.php?
sec = workspace&sec2= operation/users/user_edit
cookie 1
resource 1
check_string 555-555-5555
task_end
```

And finally, log out from the Console and look for the logout message `Logged out>`

```
task_begin
get http://192.168.70.116/pandora_console/index.php? bye = bye
cookie 1
resource 1
check_string Logged out
task_end
```

Total *script* check:



WEB query performance

The fields of the advanced properties are similar to those of other types of modules, although there are some different and specific fields of WEB checks:

Timeout

This is the expiry time during the request. If it is exceeded, the request for verification will be discarded.

Agent browser id

This is the web browser identifier to be used, since certain pages only accept some web browsers (see https://www.zytrax.com/tech/web/browser_ids.htm for more information).

Requests

Pandora FMS will repeat the check the amount of times indicated in this parameter. If one of the checks fails, the check will be considered erroneous. Depending on the number of checks in the module, a certain number of pages will be obtained. That means that if the module consists of three checks, three pages will be downloaded, and if some value has been set in the **Requests** field, then the number of downloads will be multiplied by it. It is important to keep this in mind to know the total time it will take for the module to complete the operations.

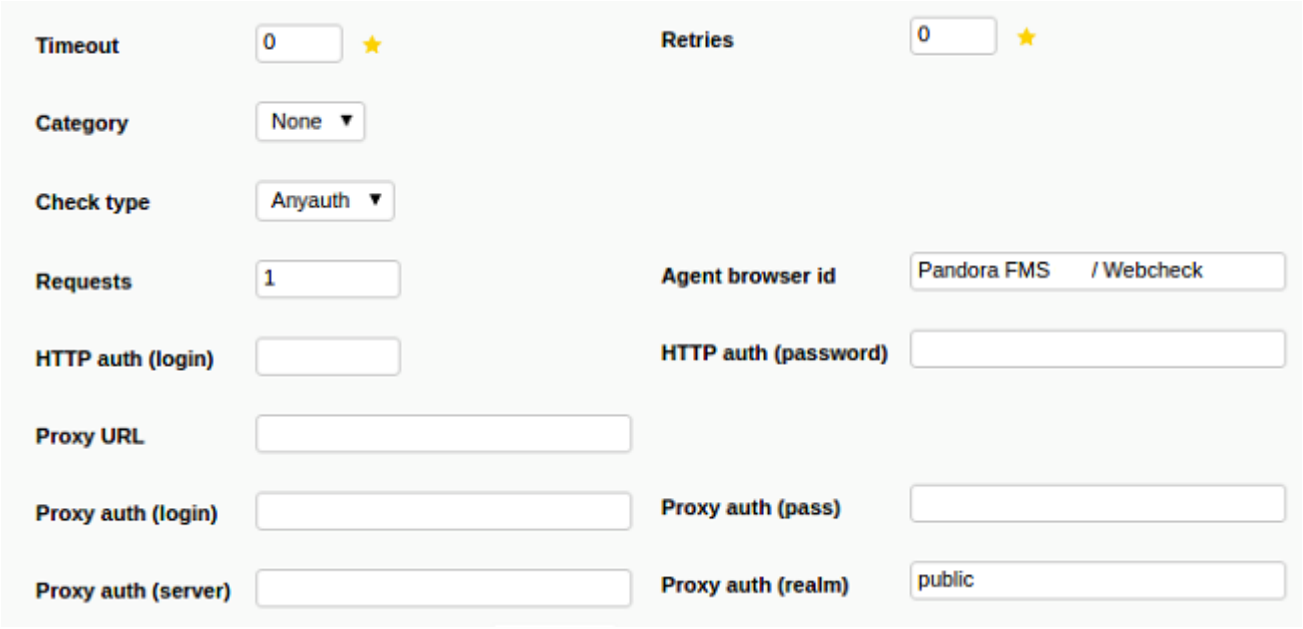
Retries

The number of times it does a **Request** until getting a successful result. Examples:

- retries = 2 and Requests = 1: If the first test fails, it will retry once more and if the second one works, the check is valid.
- Retries = 1 and Requests = 2: It performs two checks, but if any of them fails, it will result in a failed check.

Simple HTTP Authentication

Some websites might require simple HTTP authentication. Generaly it is used as a fast authentication, a “hi” of minimum security that allows accessing advanced security checks (encryption, data persistance, etc.)



It can be configured in the advanced check options (or directly in the WEB task definition) with the following configuration tokens:

Check type

HTTP server check type. ;http auth (login): HTTP user. ;http auth (password): Password. **Proxy auth realm**

Auth realm's name. ;Proxy auth (server): Domain and HTTP port on which to listen on. **Proxy URL**

Proxy server url. ;Proxy auth (login): Proxy connection user. ;Proxy auth (pass): Proxy connection password.

Full example:

task_begin get http://artica.es/pandoraupdate4/ui/ cookie 1 resource 1 check_string Pandora FMS Update Manager \(4.0\) http_auth_serverport artica.es:80 http_auth_realm Private area http_auth_user admin http_auth_pass xxxx task_end

Webservice and API Monitoring

With Pandora FMS and Goliat you may REST APIs, except for more complex APIs based on protocols such as SOAP or XML-RPC.

To check an API with this specific call which returns a number (from '0' to 'n') if it works properly; and if not, it returns nothing, (using `my_user` and `my_pass` as credentials):

task_begin get http://artica.es/integria/include/api.php?user = my_user&pass = my_pass&op = get_stats&params = opened,,1 check_string \n0-9+ task_end

This will return a reply similar to this one:

HTTP/1.1 200 OK Cache-Control: no-store, no-cache, must-revalidate, post-check =0, pre-check =0 Connection: close Date: Mon, 13 May 2013 15:39:27 GMT Pragma: no-cache Server: Apache Vary:

Accept-Encoding Content-Type: text/html Expires: Thu, 19 Nov 1981 08:52:00 GMT Client-Date: Mon, 13 May 2013 15:39:27 GMT Client-Peer: 64.90.57.215:80 Client-Response-Num: 1 Client-Transfer-Encoding: chunked Set-Cookie: a81d4c5e530ad73e256b7729246d3d2c = pcasWqI6pZzT2x2AuWo602; path =/

0

By checking the output with a regular expression, you may verify that everything works properly. For more complex answers, use other regular expressions accordingly.

More examples:

task_begin get https://swapi.co/api/planets/1/ get_content Tatooine task_end

In this case, the module created to show data must be **Remote HTTP module to retrieve string data (web_content_string)**.

task_begin get https://pokeapi.co/api/v2/pokemon/ditto/ get_content imposter task_end

Just like the previous module, the type of data defined needs to be 'Remote HTTP module to retrieve string data (web_content_string)' for the module to work properly.

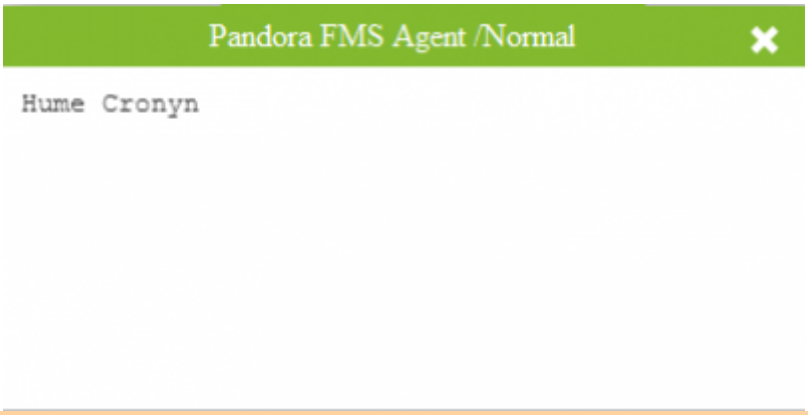
You may perform calls with **get_content_advanced**:

task_begin get https://api.hillbillysoftware.com/Awards/ByYear/1990 get_content_advanced “Nominee”:“(a-za-z+)”,“Year”:“1990” task_end

Result:

```
−<ArrayOf_Awards>
  −<_Awards>
      <Category>Outstanding Lead Actor In A Miniseries Or Special</Category>
      <Nominee>Hume Cronyn</Nominee>
      <Type>Emmy</Type>
      <Winner>1</Winner>
      <Year>1990</Year>
    </_Awards>
  −<_Awards>
      <Category>Outstanding Lead Actor In A Miniseries Or Special</Category>
      <Nominee>Michael Caine</Nominee>
      <Type>Emmy</Type>
      <Winner>0</Winner>
      <Year>1990</Year>
    </_Awards>
  −<_Awards>
      <Category>Outstanding Lead Actor In A Miniseries Or Special</Category>
      <Nominee>Tom Hulce</Nominee>
      <Type>Emmy</Type>
      <Winner>0</Winner>
      <Year>1990</Year>
    </_Awards>
  −<_Awards>
      <Category>Outstanding Lead Actor In A Miniseries Or Special</Category>
      <Nominee>Albert Finney</Nominee>
      <Type>Emmy</Type>
      <Winner>0</Winner>
      <Year>1990</Year>
    </_Awards>
  −<_Awards>
      <Category>Outstanding Lead Actor In A Miniseries Or Special</Category>
      <Nominee>Art Carney</Nominee>
      <Type>Emmy</Type>
      <Winner>0</Winner>
      <Year>1990</Year>
    </_Awards>
  −<_Awards>
    −<Category>
        Outstanding Lead Actress In A Miniseries Or Special
      </Category>
      <Nominee>BARBARA HERSHEY</Nominee>
      <Type>Emmy</Type>
      <Winner>1</Winner>
      <Year>1990</Year>
    </ Awards>
```

In Pandora FMS the result would be displayed as follows:



It is important to properly define the capture groups within the parentheses so that the call is executed correctly.

When creating API calls, it is important to know if the destination API has the right permissions to allow calls.

HTTPS Monitoring

Goliat is able to check both HTTP and HTTPS. To carry out checks on secured websites which use HTTPS, incorporate the protocol into its URL, e.g.:

0@@

Advanced Options

Modifying HTTP Headers

With the *header* option, you are able to modify HTTP headers or create your own. The example below changes the *Host* HTTP header:

task_begin get http://192.168.1.5/index.php header Host 192.168.1.1 task_end

Debugging Web checks

If you want to debug Web checks, add the `debug <log_file>` option. The files `log_file.req` and `log_file.res` will be created along with the contents of the HTTP request and response:

task_begin get http://192.168.1.5/index.php debug /tmp/request.log task_end

Using CURL instead of LWP

LWP library sometimes crashes when multiple threads issue HTTPS requests simultaneously (due to an OpenSSL constraint). The alternative is to use the **curl** tool To solve this problem, edit the file named `/etc/pandora/pandora_server.conf` and the following line:

web_engine curl

Restart Pandora FMS Server, and the CURL binary will be used to perform web checks instead of LWP.

Advanced Transactional Monitoring

In addition to the feature offered by Goliath, there are other ways to carry out web transactional monitoring.

- In a distributed way (UX), deployed as an “agent” in systems different to that of the server, even in unaccessible networks.
- In a centralized way (WUX)

Go back to Pandora FMS documentation index