Testing GUI Apps with Sikulix & Dynatrace Synthetic

Contents

[Requirements 2](#_Toc85444854)

[ActiveGate settings 3](#_Toc85444855)

[Dynatrace bridge 3](#_Toc85444856)

[Sikulix requirements 3](#_Toc85444857)

[copy files 3](#_Toc85444858)

[Start program 3](#_Toc85444859)

[How it all works together 4](#_Toc85444860)

[GET /sikulix\_properties 4](#_Toc85444861)

[Bridge side 4](#_Toc85444862)

[Dynatrace side 5](#_Toc85444863)

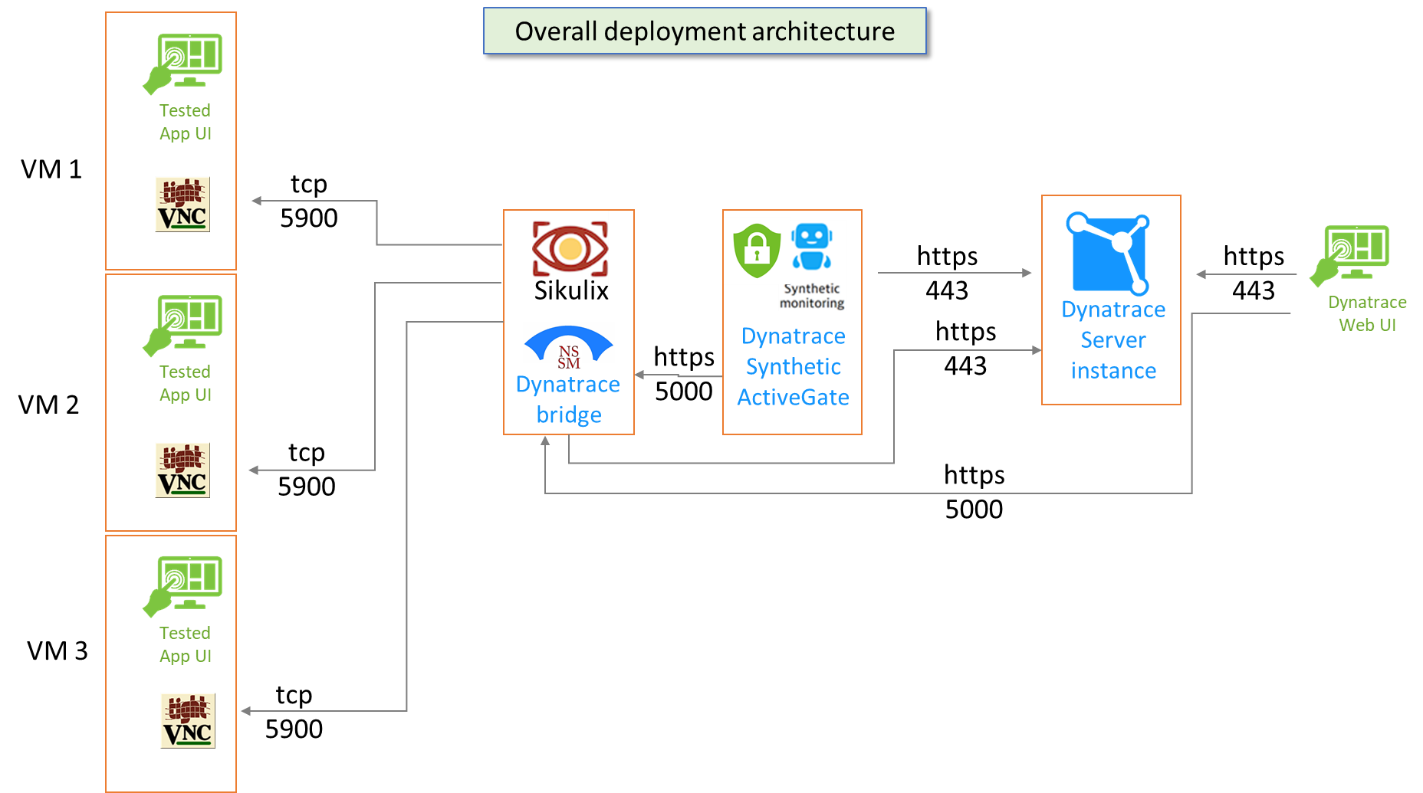
[POST /sikulix\_launcher 6](#_Toc85444864)

[Bridge side 6](#_Toc85444865)

[Dynatrace side 8](#_Toc85444866)

Author : [laurent.izac@dynatrace.com](mailto:laurent.izac@dynatrace.com)

Credits : [eric.maillard@dynatrace.com](mailto:eric.maillard@dynatrace.com)



# Requirements

Testing GUI Apps with Dynatrace Synthetic requires the following components :

* Dynatrace server instance : hosts the test settings and results ; will be accessed by end users through the Web UI.
* Dynatrace Synthetic ActiveGate : runs Synthetic tests using embedded Web browser and http client.
* Dynatrace Sikulix bridge :
  + acts as a proxy between Dynatrace and Sikulix.
  + Embeds Sikulix runtime to run GUI navigation scenarii against various client Apps.
  + Provides response times, errors and screenshots through a Web UI.
  + Sends test execution results as Third Party monitor in Dynatrace.
  + Can be installed as a service when used to test remote Desktop App accessed through VNC.
* TightVNC Server :
  + Allows remote connection to the Desktop session, with ability to open and close Desktop session as part of a SikuliX script.

Notes :

* Dynatrace Server and ActiveGate are the stock Dynatrace products.
* Dynatrace Sikulix bridge is an adhoc script written in Python.
* Sikulix is an [Opensource](http://sikulix.com/quickstart/) GUI testing tool.
* TightVNC Server is a [free](https://www.tightvnc.com/licensing-tvnserver.php) (GNU GPL) VNC Server available for Windows and Linux.

## ActiveGate settings

After having installed a Private location Synthetic ActiveGate, the following confguration must be changed to avoid exécution timeouts with Sikulix :

In file :

[Linux] /var/lib/dynatrace/synthetic/config/user.properties

[Windows] c:\ProgramData\dynatrace\synthetic\config\user.properties

Set the following :

[synthetic]

httpMonitorExecutionTimeoutInMs = 600000

Note : you must restart the « Dynatrace Synthetic » service afterwards, for the change to be effective.

## Dynatrace bridge

### Sikulix requirements

Sikulix requires a 64 bits Windows system, a 64 bits Java runtime (1.11+) and needs to have access to screen, mouse and keyboard to be able to execute tests.

Note : It also requires « Media Foundation » feature to be installed on the system :

Graphical user interface

Description automatically generated

Note : if installed on the same Windows machine as the Dynatrace Synthetic ActiveGate, it will be able to use the already installed jre

(found in « c:\Program Files\dynatrace\synthetic\jre\bin\java.exe »)

Note : The Sikulix runtime is a Jar file (sikulixide-2.0.5.jar) bundled with the Dynatrace Sikulix bridge.

### copy files

Unzip the [Dynatrace bridge files](https://dynatrace-my.sharepoint.com/:u:/p/laurent_izac/EQju0ni1IFpKi_sjjdLzJfMBIIujQl5e88OwnGcz6NJeDA?e=y5LmZ7) in a target folder…

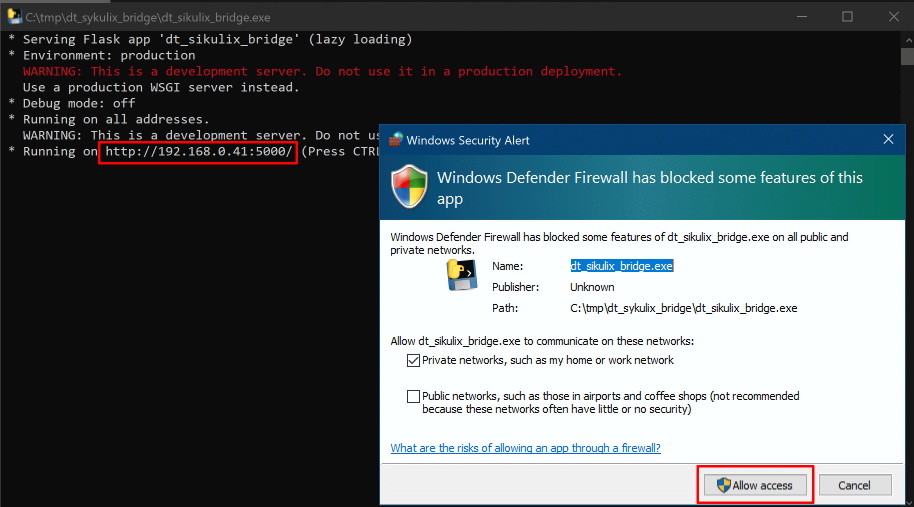
### Start program

From Windows Explorer, double click on « run.bat » to start the bridge.

Note : by default, the batch file is trying to use the Java runtime from the Synthetic ActiveGate.

Take note of the URL the server is running on.This will be used from Dyntrace Synthetic script to trigger Sikulix execution.

If you are presented with a security dialog box, Click on « Allow access » to let the program run.



Note : for the bridge to start automatically when the session (re-)opens, copy a shortcut for « run.bat » in «shell:startup ».

# How it all works together

The Bridge listens for http requests and expects 2 specific URLs :

* + GET « /testtool\_properties » allowing the caller to get a properties file;
  + POST « /testtool\_launcher » triggering a script execution with the following parameters in a json structure :
    - script\_name : pathname of the script to run ;
    - function\_to\_execute : name and arguments of function to execute in the script.
  + GET

## GET /testtool\_properties

### Bridge side

Below is an example json file containing properties that can be retrieved by calling the « /testtool\_properties » URL.

{

  "SIKULI\_SCRIPT" : "C:\\tmp\\Sikulix\\scripts\\office",

  "word\_doc\_1" : "MyTestWordDocument"

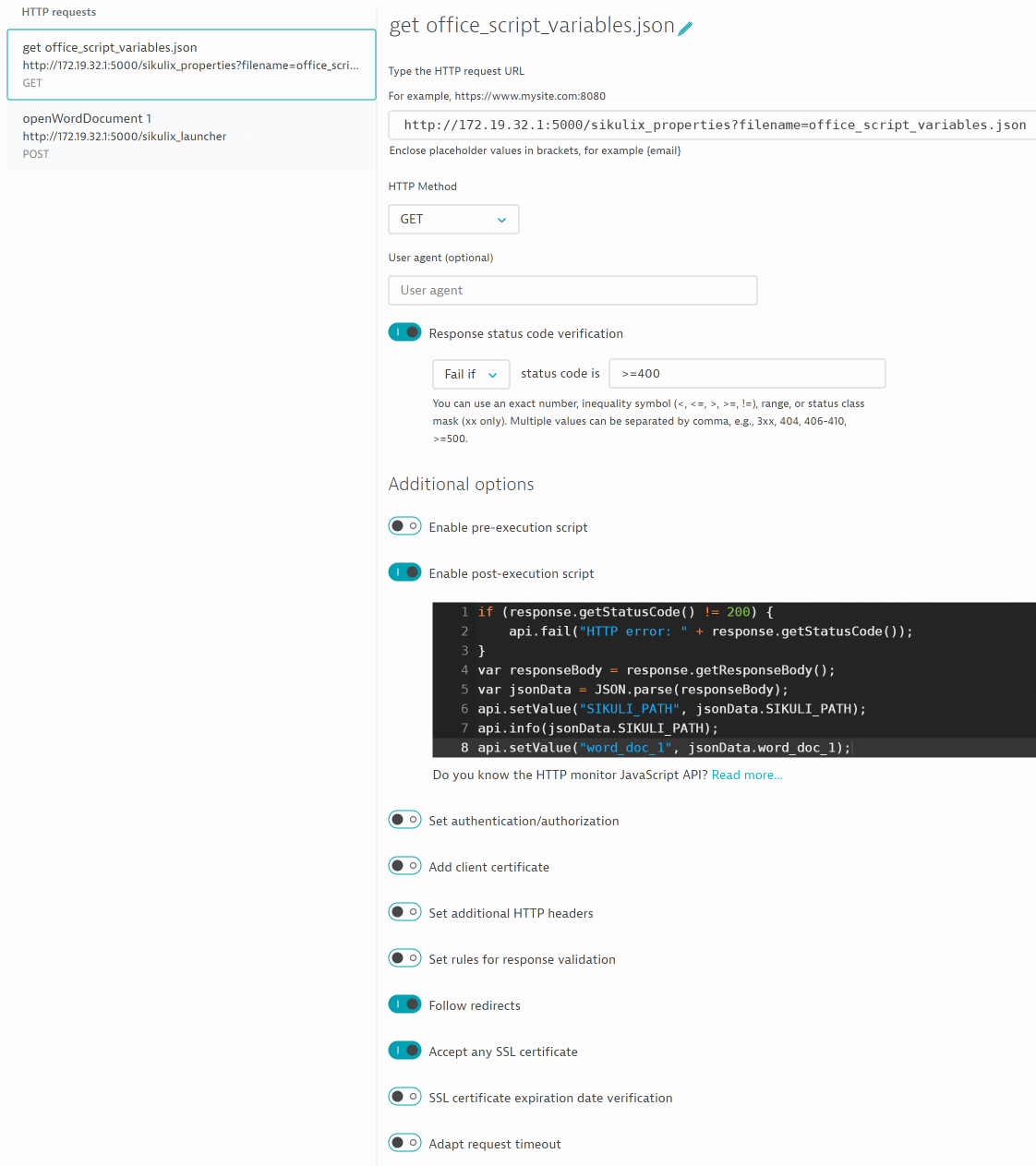
}

Note : the first key-value pair (SIKULI\_SCRIPT) is used to locate the Sikuli script to execute. All other key-value pairs will be adapted to your test context (set both keys and values according to your testing needs).

### Dynatrace side

Below is an example HTTP Monitor step that will retrieve the contents of these json properties :

A « GET » request is sent to the bridge, calling the « /sikulix\_properties » URL, with a « filename » parameter pointing to the pathname of the json file to retrieve :



A Post-execution script processes the json structure and extracts the contents to build 2 placeholders that will be used in the following steps :

if (response.getStatusCode() != 200) {

api.fail("HTTP error: " + response.getStatusCode());

}

var responseBody = response.getResponseBody();

var jsonData = JSON.parse(responseBody);

api.setValue("SIKULI\_PATH", jsonData.SIKULI\_PATH);

api.info(jsonData.SIKULI\_PATH);

api.setValue("word\_doc\_1", jsonData.word\_doc\_1);

## POST /testtool\_launcher

### Bridge side

Below is an example Sikulix script trying to open a Word document with parameters taken from a previously retrieved json properties content :

The POST body content is a json structure like the following :

{

  "script\_name" : " C:\\tmp\\Sikulix\\scripts\\office.sikuli",

  "function\_to\_execute" : "openWordDocument('MyTestWordDocument')",

}

From this content, the bridge calls the following « office.sikuli » script :

import sys

def openWordDocument(documentName):

click("1625821325583.png")

wait(1)

type("word")

click("1625821499357.png")

wait(1)

click("1625821549524.png")

click("1625821587111.png")

paste('C:\\my test folder\\my word documents')

click("1625821764224.png")

if (documentName == MyTestWordDocument'):

click("1625821716905.png")

elif (documentName == ' MyTestWordDocument 2'):

click("1625822469097.png")

elif (documentName == ' MyTestWordDocument 3'):

click("1625822510497.png")

else:

print('[error] word document does not exits : '+documentName)

click("1625821764224.png")

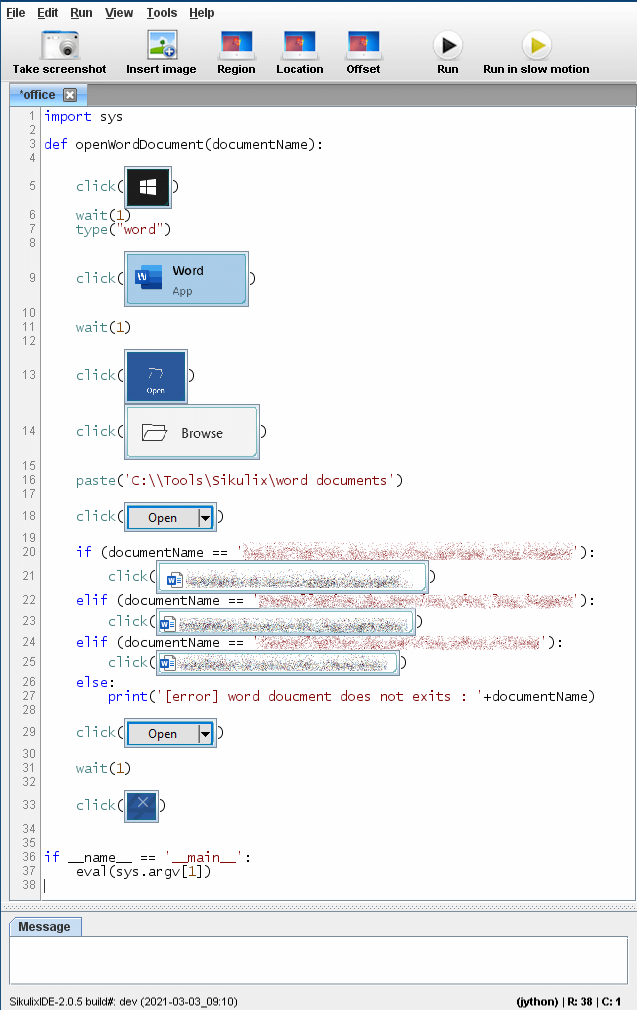
wait(1)

click("1625837585975.png")

if \_\_name\_\_ == '\_\_main\_\_':

    eval(sys.argv[1])

Here is what the script looks like when opened from the Sikulix IDE :

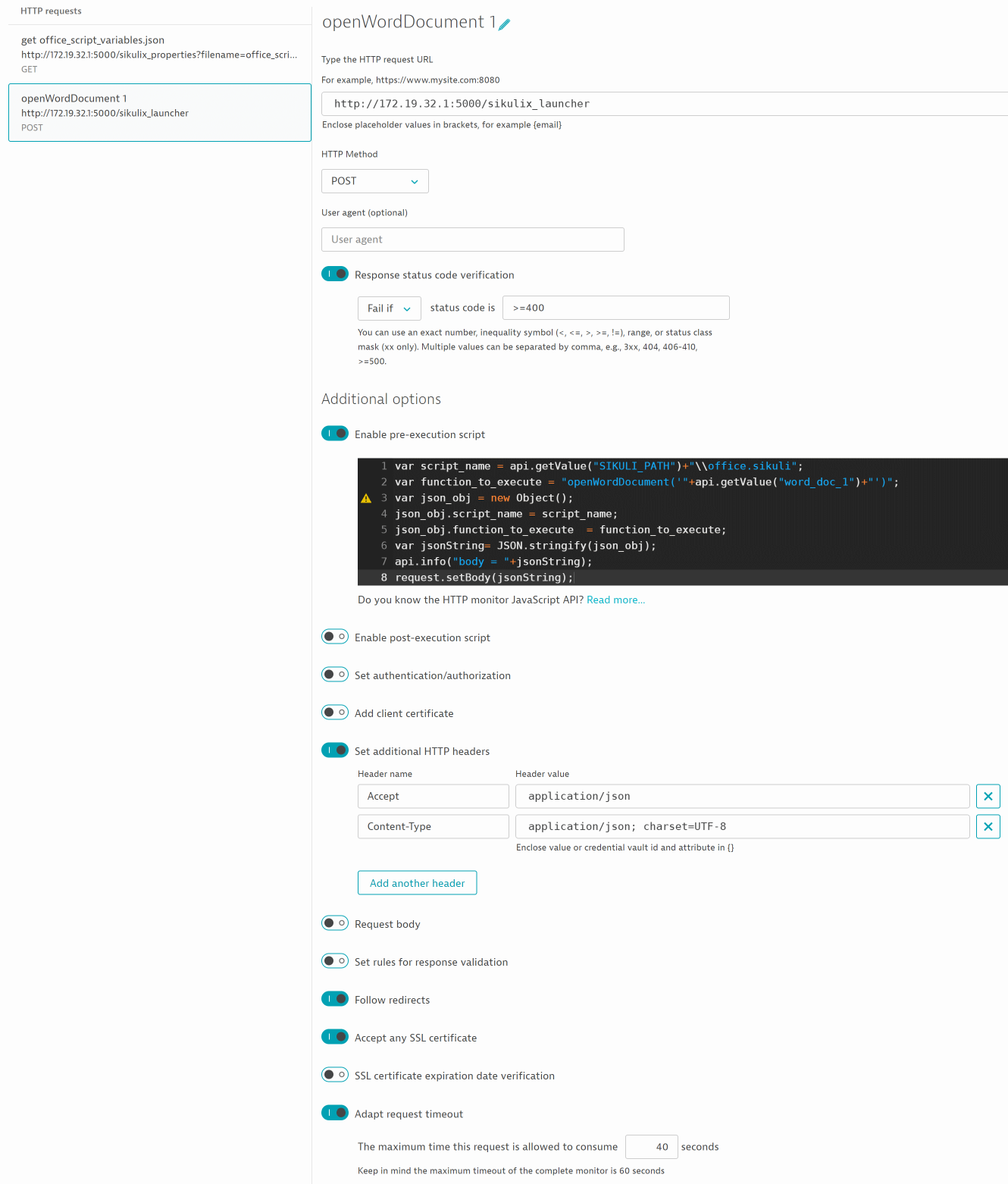


Note : the Sikulix IDE can be executed by double clicking on « sikulixide-2.0.5.jar », or by executing the following command :

* "c:\Program Files\dynatrace\synthetic\jre\bin\java.exe" -jar ./sikulixide-2.0.5.jar

### Dynatrace side

Below is an example HTTP Monitor step that will trigger the Sikulix script by calling the « /testtool\_launcher » POST URL :



A pre-execution srcript gets the placeholders from the previous step, builds the command line for Sikulix and sets the 2 POST parameters :

var script\_name = api.getValue("SIKULI\_PATH")+"\\office.sikuli";

var function\_to\_execute = "openWordDocument('"+api.getValue("word\_doc\_1")+"')";

var json\_obj = new Object();

json\_obj.script\_name = script\_name;

json\_obj.function\_to\_execute = function\_to\_execute;

var jsonString= JSON.stringify(json\_obj);

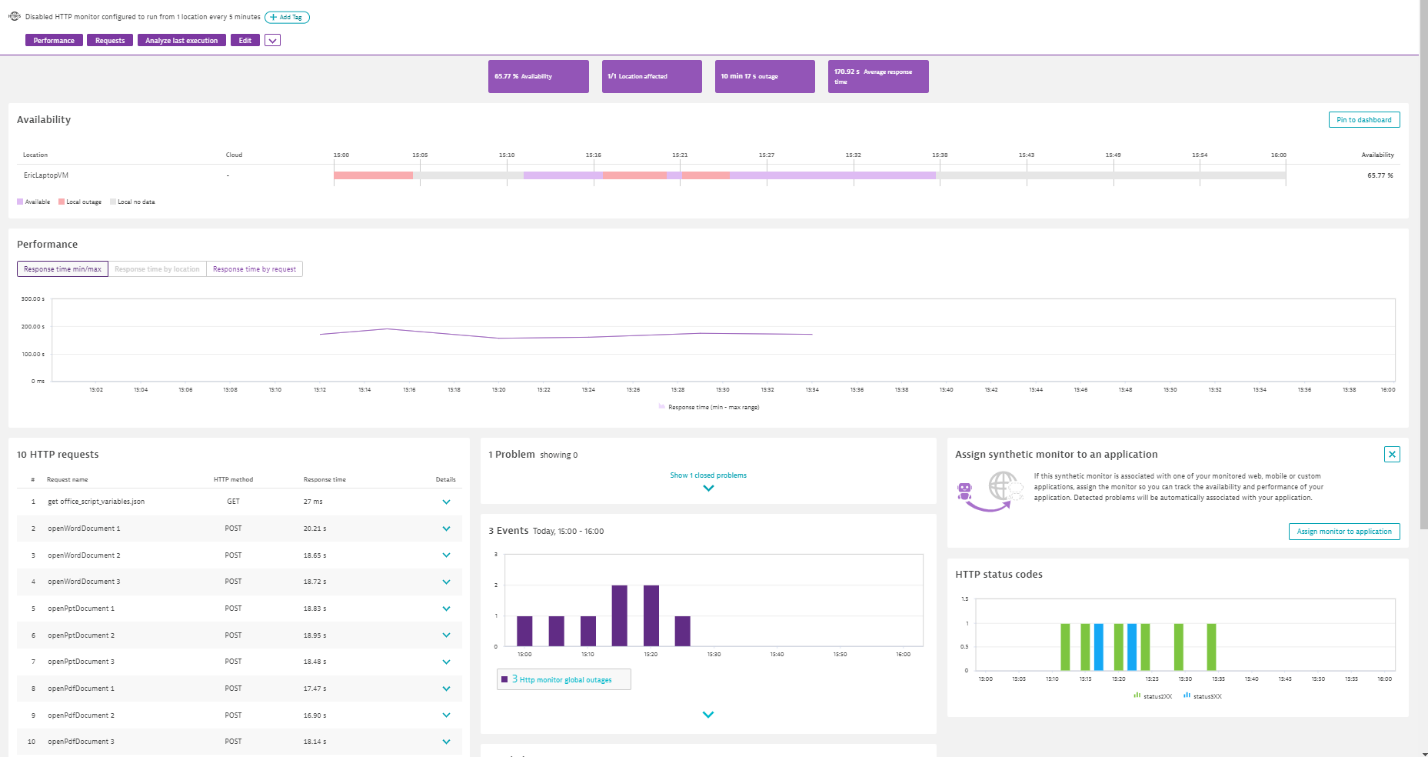
api.info("body = "+jsonString);

request.setBody(jsonString);

Note: as the POST content is sent as a json structure, make sure to set the appropriate request headers:

* Accept application/json
* Content-Type application/json; charset=UTF-8

Every time the Dynatrace http Monitor Synthetic script runs, say every 5 mn, it will report execution times and errors :



In case of error, the Sikulix error message is available in « Analyze last execution »  :



# Hints & Tips

## run a script in a minimized window :

1. Login to the computer from which you want to connect to the remote VM
2. Close all open remote desktop sessions
3. Click Start
4. Click Run
5. Type "regedit"
6. Press Enter
7. Go to "HKEY\_CURRENT\_USER\\Software\\Microsoft\\Terminal Server Client"
8. Create a DWORD value named "RemoteDesktop\_SuppressWhenMinimized"
9. Set value to 2
10. Close the Registry Editor.