API Tools Solution

URL-XI Basic Training

2021-10-22

Contents

1 Introduction

This document contains practical training on URL-XI, the new scripting tool offered by Apica System. It is a self-paced training that you can do at your own pace.

The purpose of training is that you should be able to understand the concept of URL-XI and create scripts that solve real use cases for API monitoring.

1.1 References

- The Axios Home page. URL-XI http requests are based on the axios library. See https://axios-http.com/docs/req_config
- Good sites for training with URL-XI
  - https://httpbin.org/
  - https://jsonplaceholder.typicode.com/guide/

2 URL-XI Introduction

2.1 Concept

- A solution from Apica for creating advanced API checks. You create an url-xi test case (script) in a json file. We call the file the test configuration file.

- URL-XI is a simplified version POSTMAN and using a similar concept but has better support for a flow of request and correlation of request. It can additionally return other metrics than response times of the http requests as the main return value of the test case.

- The tool has a model-based testing framework inspired by Mocha and Jest.

- A test case is built up in several steps. A step contains one or several API requests.

- Reporting is done on each step and all underlying requests.

- The tool is built for API monitoring from the bottom up.

  - Detailed timings are reported per request.

  - You can customize what should be reported as the result of a test case.

You can run a test case from the command line and as a check in Apica Synthetics (ASM).

2.2 Detailed information about url-xi

Can be found at: https://www.npmjs.com/package/@apica-io/url-xi

2.3 URL-XI in Apica Synthetic

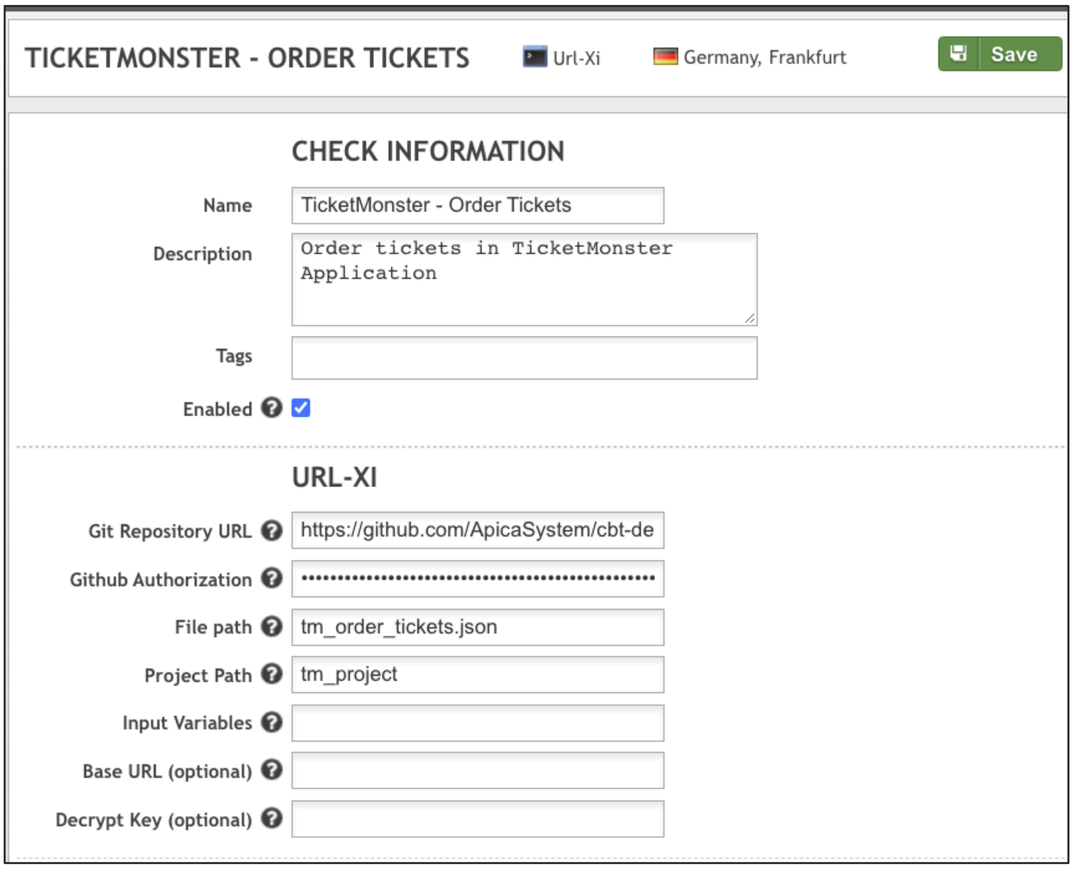
You can run URL-XI as check in ASM.

- When you deploy test cases to ASM they should be pushed to a repository like GitHub or Bitbucket.

- ASM will access and download the test configuration file and related project assets when the check is executed.

- You can view the results of a check execution as a nice UI report or as raw JSON result with all information included.

2.3.1 Check configuration in ASM



TICKETMONSTER - ORDER TICKETS   Url-Xi   Germany, Frankfurt   Save

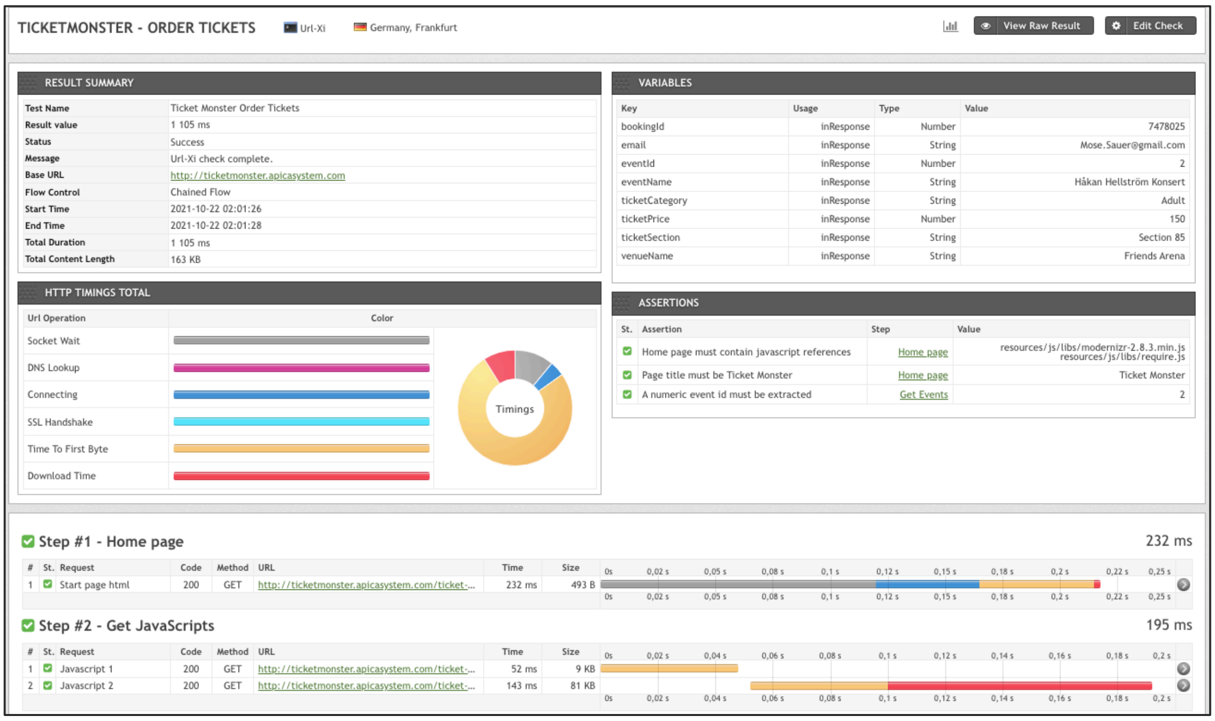
CHECK INFORMATION

Name	TicketMonster - Order Tickets
Description	Order tickets in TicketMonster Application
Tags	
Enabled	☑

URL-XI

Git Repository URL	https://github.com/ApicaSystem/cbt-de
Github Authorization	••
File path	tm_order_tickets.json
Project Path	tm_project
Input Variables	
Base URL (optional)	
Decrypt Key (optional)	

2.3.2 API Tools result page for URL-XI



TICKETMONSTER - ORDER TICKETS   Url-Xi   Germany, Frankfurt

RESULT SUMMARY

Test Name	Ticket Monster Order Tickets
Result value	1 105 ms
Status	Success
Message	Url-Xi check complete.
Base URL	http://ticketmonster.apicasystem.com
Flow Control	Chained Flow
Start Time	2021-10-22 02:01:26
End Time	2021-10-22 02:01:28
Total Duration	1 105 ms
Total Content Length	163 KB

VARIABLES

Key	Usage	Type	Value
bookingId	inResponse	Number	7478025
email	inResponse	String	Mose.Sauer@gmail.com
eventId	inResponse	Number	2
eventName	inResponse	String	Håkan Hellström Konsert
ticketCategory	inResponse	String	Adult
ticketPrice	inResponse	Number	150
ticketSection	inResponse	String	Section 85
venueName	inResponse	String	Friends Arena

HTTP TIMINGS TOTAL

Url Operation	Color
Socket Wait	
DNS Lookup	
Connecting	
SSL Handshake	
Time To First Byte	
Download Time	

ASSERTIONS

St.	Assertion	Step	Value
✓	Home page must contain javascript references	Home page	resources/js/libs/modernizr-2.8.3.min.js resources/js/libs/require.js
✓	Page title must be Ticket Monster	Home page	Ticket Monster
✓	A numeric event id must be extracted	Get Events	2

☑ Step #1 - Home page   232 ms

#	St.	Request	Code	Method	URL	Time	Size
1	✓	Start page html	200	GET	http://ticketmonster.apicasystem.com/ticket-...	232 ms	493 B

0s 0,02 s 0,05 s 0,08 s 0,1 s 0,12 s 0,15 s 0,18 s 0,2 s 0,22 s 0,25 s

☑ Step #2 - Get JavaScripts   195 ms

#	St.	Request	Code	Method	URL	Time	Size
1	✓	Javascript 1	200	GET	http://ticketmonster.apicasystem.com/ticket-...	52 ms	9 KB
2	✓	Javascript 2	200	GET	http://ticketmonster.apicasystem.com/ticket-...	143 ms	81 KB

0s 0,02 s 0,04 s 0,06 s 0,08 s 0,1 s 0,12 s 0,14 s 0,16 s 0,18 s 0,2 s

3 Lab 1 – Install URL-XI and setup environment

3.1 Purpose

Get a little bit familiar with the tool by running url-xi from command line with a couple of switches.

3.2 The training setup with directories

You should setup the training by cloning the GitHub repository you have got from the instructor. The directory you have cloned will be the working directory for the training.

```
jostgren:url-xi janostgren$ ls -l
total 24
-rw-r--r--   1 janostgren  staff   42 18 Okt 15:58 README.md
drwxr-xr-x   2 janostgren  staff   64 19 Okt 07:49 api-project
-rw-r--r--   1 janostgren  staff  318 18 Okt 15:59 package.json
-rw-r--r--   1 janostgren  staff  577 13 Okt 16:04 redirect_sample.json
drwxr-xr-x  11 janostgren  staff  352 20 Okt 13:17 results
drwxr-xr-x  12 janostgren  staff  384 20 Okt 12:45 solutions
```

- The *solutions* directory contains solutions to all labs. You should newer change anything in that directory. The repository content can be updated.
- You should try to solve all labs without looking at the solutions.
- The *results* directory is a directory for storing results from executions of the url-xi command.
- The *api-project* directory is project directory for ASM API labs.
- The *redirect_sample.json* file is a simple url-xi test case that you use in the first labs.

3.3 Install URL-XI from NPM

1) Install Node JS on your workstation. Information about how to install Node are found here https://nodejs.org/en/download/.

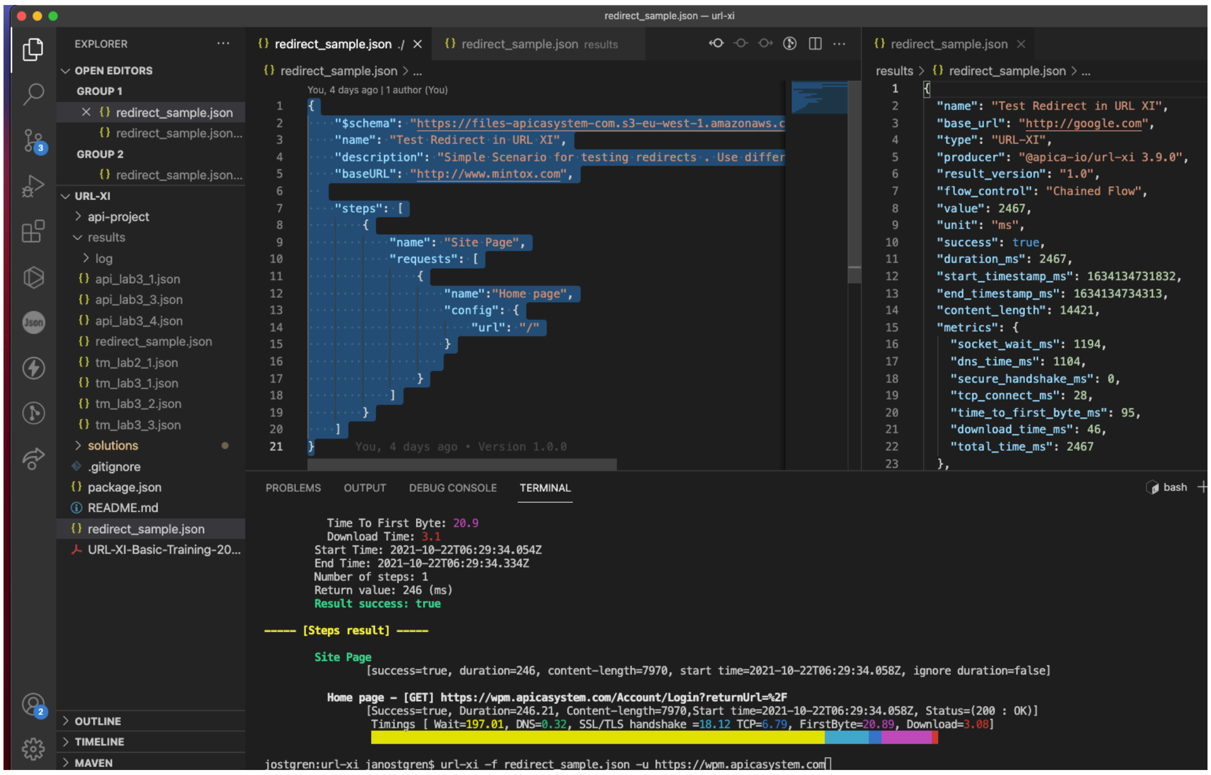
2) Install latest version of url-xi with the following command
npm install -g @apica-io/url-xi.

3) Verify your installation by enter url-xi from the command shell.

3.4 Install and setup Visual Studio Code (Optional)

We recommend you install Microsoft Visual Studio code. It will make it easier to create url-xi scripts, due to the strong support for JSON.

3.4.1 Working in Visual Studio



redirect_sample.json > ...
You, 4 days ago | 1 author (You)

```
{
    "$schema": "https://files-apicasystem-com.s3-eu-west-1.amazonaws.c
    "name": "Test Redirect in URL XI",
    "description": "Simple Scenario for testing redirects . Use differ
    "baseURL": "http://www.mintox.com",

    "steps": [
        {
            "name": "Site Page",
            "requests": [
                {
                    "name":"Home page",
                    "config": {
                        "url": "/"
                    }

                }
            ]
        }
    ]
}       You, 4 days ago • Version 1.0.0
```

results > redirect_sample.json > ...

```
{
  "name": "Test Redirect in URL XI",
  "base_url": "http://google.com",
  "type": "URL-XI",
  "producer": "@apica-io/url-xi 3.9.0",
  "result_version": "1.0",
  "flow_control": "Chained Flow",
  "value": 2467,
  "unit": "ms",
  "success": true,
  "duration_ms": 2467,
  "start_timestamp_ms": 1634134731832,
  "end_timestamp_ms": 1634134734313,
  "content_length": 14421,
  "metrics": {
    "socket_wait_ms": 1194,
    "dns_time_ms": 1104,
    "secure_handshake_ms": 0,
    "tcp_connect_ms": 28,
    "time_to_first_byte_ms": 95,
    "download_time_ms": 46,
    "total_time_ms": 2467
  },
```

```
          Time To First Byte: 20.9
          Download Time: 3.1
        Start Time: 2021-10-22T06:29:34.054Z
        End Time: 2021-10-22T06:29:34.334Z
        Number of steps: 1
        Return value: 246 (ms)
        Result success: true

----- [Steps result] -----

        Site Page
                [success=true, duration=246, content-length=7970, start time=2021-10-22T06:29:34.058Z, ignore duration=false]

          Home page - [GET] https://wpm.apicasystem.com/Account/Login?returnUrl=%2F
                [Success=true, Duration=246.21, Content-length=7970,Start time=2021-10-22T06:29:34.058Z, Status=(200 : OK)]
                 Timings [ Wait=197.01, DNS=0.32, SSL/TLS handshake =18.12 TCP=6.79, FirstByte=20.89, Download=3.08]

jostgren:url-xi janostgren$ url-xi -f redirect_sample.json -u https://wpm.apicasystem.com
```

- Working with development of test configuration (left tab)
- View results (right tab). You use this view to create extractors and assertions from the result.
- Running test cases in terminal window (bottom)

3.5 Create first project for url-xi

You only need to do this if you have not download the content from GitHub respository.

Create a directory to work in in. We will call it the project directory.

1) Create a sub-directory called results in project directory.
2) Copy a simple script form the samples directory in your global url-xi installation. This directory *is /usr/local/lib/node_modules/@apica-io/url-xi/samples* on Mac and Linux. Try the following command to show loction and globally installed packages.

```
npm list -g --depth 0
```

3) Copy the *redirect_sample.json* file from the sample directory.

```
{
    "$schema": "https://files-apicasystem-com.s3-eu-west-
1.amazonaws.com/schemas/url-xi-schema-v1-0.json",
    "name": "Test Redirect in URL XI",
    "description": "Simple Scenario for testing redirects . Use different base
urls",
    "baseURL": "http://www.mintox.com",

    "steps": [
        {
            "name": "Site Page",
            "requests": [
                {
                    "name":"Home page",
                    "config": {
                        "url": "/"
                    }

                }
            ]
        }
    ]
}
```

The sample script will look like this. You will use it as a template when creating new scripts.
It is important that you always use the schema reference on the first line in the json file, it will help you editing the file in Visual Studio Code.

3.6 Run the sample with different parameters settings

1) Check syntax of the url-xi command

```
url-xi -h
Usage: url-xi [options]

Options:
  -V, --version                      output the version number
  -f, --file <test_file>             The test configuration file
  -r, --results <result_dir>         The result directory
  -xh, --xheaders <headers>          extra headers (default: "{}")
  -i, --inputs [inputs...]           input variables. Format name=value format
  -u, --url <url>                    base url
  -l, --log_level <log_level>        log level (default: "info")
  -nd, --nodata                      no response/request data in report
  -m, --mask                         Mask sensitive data from report
  -po, --parse_only                  parse json only. No not run (default: false)
  -prod, --production                Production mode. Minimal logging and content in
results (default: false)
  --no-keep_alive                    No keep alive of http connections
  -rn, --result_name <result_name>   name of the result
  -s, --server                       start as server
  -p, --port <port>                  server port (default: "8070")
  -tc, --time_calc <time_calc>       Custom request-time calculation (default:
"totalTime")
  -of, --out_format <out_format>     Output format in json result. CRS or Default
(default: "default")
  -proj --project <project>          Project should be a directory or a zip file
  -ts, --table_server <table_server>  Apica Table Server (Experimental testing)
  -dk, --decryptKey <decryptKey>     Cryptify Decrypt key
  --proxy <proxy>                    Http(s) proxy (default: "")
  -h, --help                         display help for command
```

2) Run url-xi with no optional parameters.

```
url-xi -f redirect_sample.json
```

3) Run with -r switch and examine the result report located in results directory and change log level to debug

```
url-xi -f redirect_sample.json -r results -l debug
```

4) Examine the result file in the results directory and the log files. We encourage you to install the fx utility for look at json files.

```
npm install -g fx
```

5) Use other base urls for with -u option.

```
url-xi -f redirect_sample.json -r results -l debug -u http://google.com
```

6) Try the -of crs switch and -prod switch and look at the results. It is the switches used in ASM.

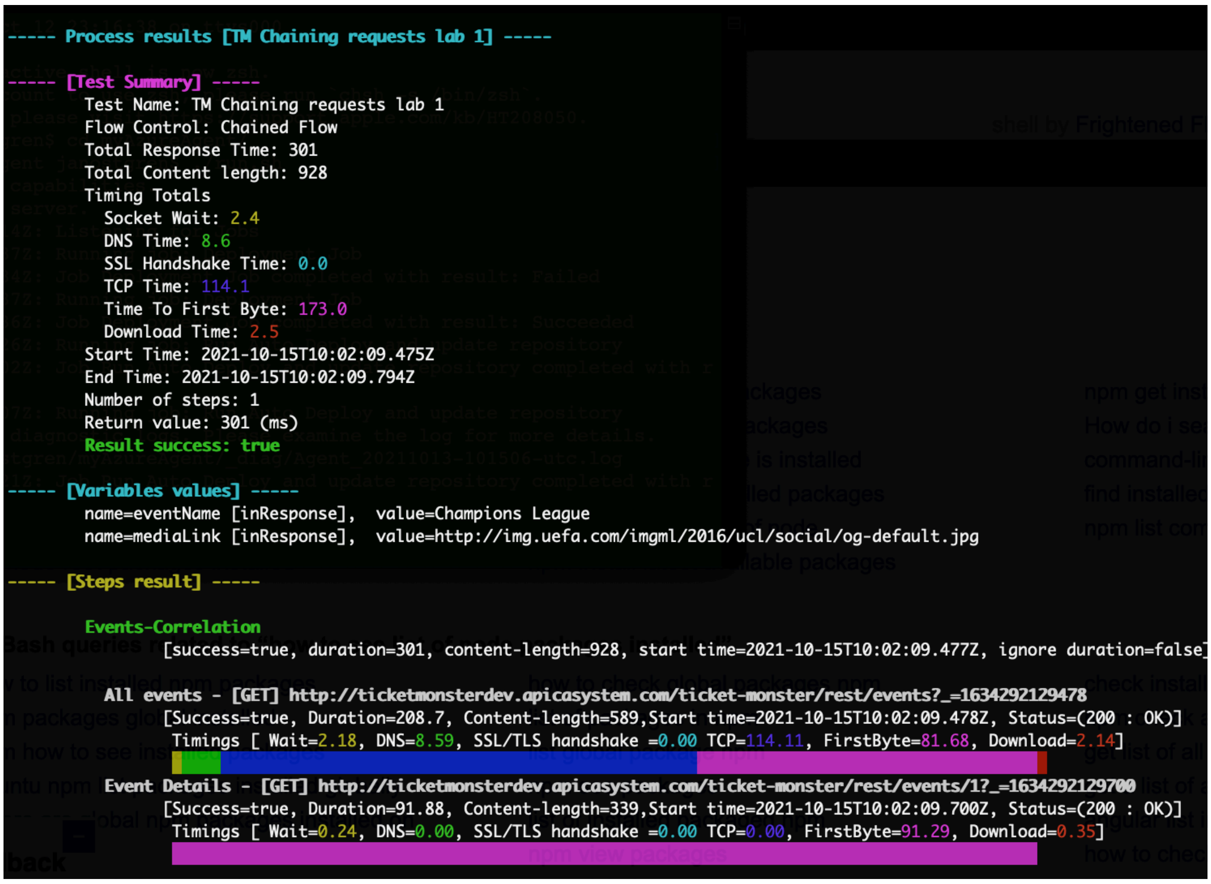
4 LAB 2 – Simple chain API example

4.1 Purpose

Learn the basics of chaining requests with extractors.
We will use the rest API for TicketMonster demo application for this lab.
http://ticketmonsterdev.apicasystem.com/ticket-monster/

4.2 Exercise 1 – Chain events to events detail request

4.2.1 Final result



```
----- Process results [TM Chaining requests lab 1] -----

----- [Test Summary] -----
        Test Name: TM Chaining requests lab 1
        Flow Control: Chained Flow
        Total Response Time: 301
        Total Content length: 928
        Timing Totals
          Socket Wait: 2.4
          DNS Time: 8.6
          SSL Handshake Time: 0.0
          TCP Time: 114.1
          Time To First Byte: 173.0
          Download Time: 2.5
        Start Time: 2021-10-15T10:02:09.475Z
        End Time: 2021-10-15T10:02:09.794Z
        Number of steps: 1
        Return value: 301 (ms)
        Result success: true

----- [Variables values] -----
        name=eventName [inResponse],  value=Champions League
        name=mediaLink [inResponse],  value=http://img.uefa.com/imgml/2016/ucl/social/og-default.jpg

----- [Steps result] -----

        Events-Correlation
                [success=true, duration=301, content-length=928, start time=2021-10-15T10:02:09.477Z, ignore duration=false]

          All events - [GET] http://ticketmonsterdev.apicasystem.com/ticket-monster/rest/events?_=1634292129478
                [Success=true, Duration=208.7, Content-length=589,Start time=2021-10-15T10:02:09.478Z, Status=(200 : OK)]
                 Timings [ Wait=2.18, DNS=8.59, SSL/TLS handshake =0.00 TCP=114.11, FirstByte=81.68, Download=2.14]

          Event Details - [GET] http://ticketmonsterdev.apicasystem.com/ticket-monster/rest/events/1?_=1634292129700
                [Success=true, Duration=91.88, Content-length=339,Start time=2021-10-15T10:02:09.700Z, Status=(200 : OK)]
                 Timings [ Wait=0.24, DNS=0.00, SSL/TLS handshake =0.00 TCP=0.00, FirstByte=91.29, Download=0.35]
```

4.2.2 Specification

Artifact	Name/Content	Comments
test name	tm_lab2_1.json	
baseURL	http://ticketmonsterdev.apicasystem.com/ticket-monster	
variables	*eventName* – In response *mediaLink* – In response, url	From extractions of request 2
Step 1	Name: Events-Correlation	
Request 1	url: '/rest/events' name: All Events	

	Extractors eventId – Random id , jsonpath	The value of *eventId* will be used in next request.
Request 2	url: /rest/events/{{eventId}} name: Event Details	
	Extractors eventName – Variable to display mediaLink – Variable to display	Use jsonpath for extraction

Hints:

- Start with just include the first request.
- Use -r switch
- Examine the result with the fx or in Visual Studio.
- Include the extractor in request 1.
- Run with debug option.

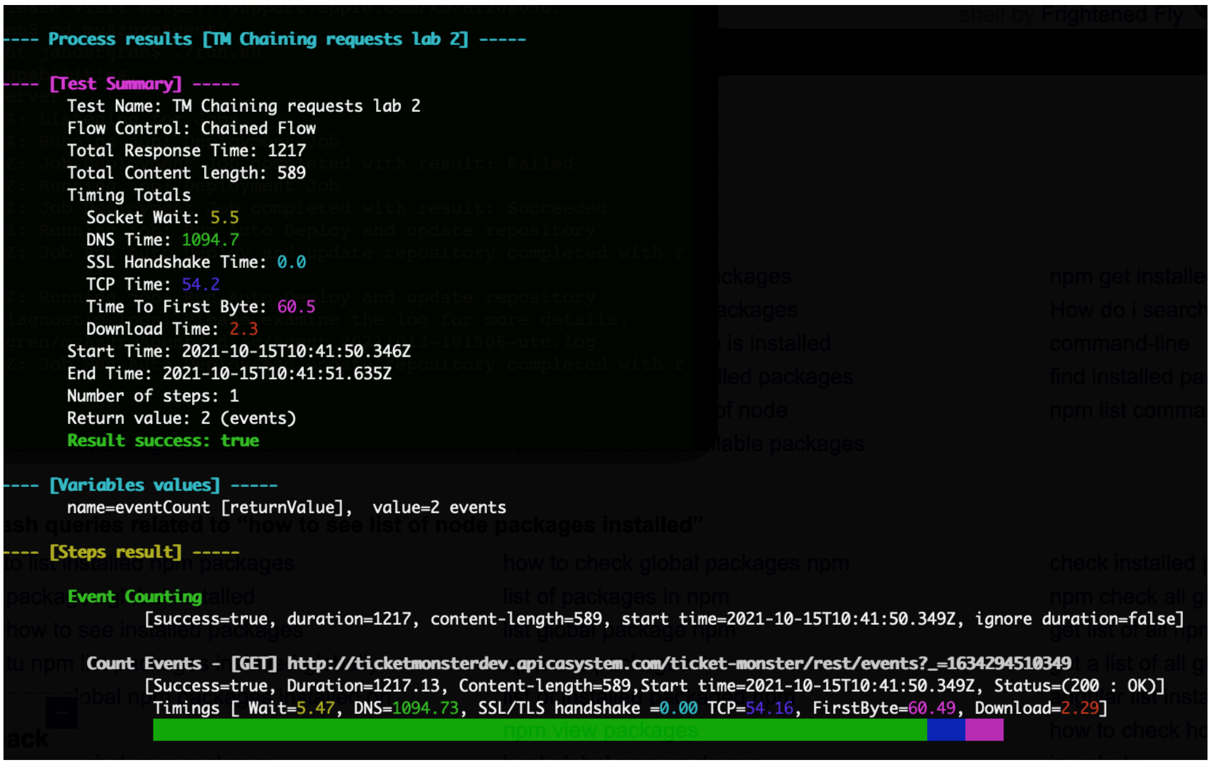
```
{
    "$schema": "https://files-apicasystem-com.s3-eu-west-
1.amazonaws.com/schemas/url-xi-schema-v1-0.json",
    "name": "TM Chaining requests lab 1",
    "description": "Chaining requests in TicketMonster. ",
    "baseURL": "http://ticketmonsterdev.apicasystem.com/ticket-monster",

    "steps": [
        {
            "name": "Events-Correlation",
            "requests": [
                {
                    "name":"All events",
                    "config": {
                        "url": "/rest/events",
                        "params": {"_":"{{$timestamp}}"}
                    } ,
                    "extractors": []
                }
            ]
        }
    ]
}
```

- This is a good start for your solution.
- Insert the extractor as next step.

4.3 Exercise 2 – Use a custom return value

In this exercise we will create a customized return value for the script.

4.3.1 Final result



```
---- Process results [TM Chaining requests lab 2] -----

---- [Test Summary] -----
        Test Name: TM Chaining requests lab 2
        Flow Control: Chained Flow
        Total Response Time: 1217
        Total Content length: 589
        Timing Totals
          Socket Wait: 5.5
          DNS Time: 1094.7
          SSL Handshake Time: 0.0
          TCP Time: 54.2
          Time To First Byte: 60.5
          Download Time: 2.3
        Start Time: 2021-10-15T10:41:50.346Z
        End Time: 2021-10-15T10:41:51.635Z
        Number of steps: 1
        Return value: 2 (events)
        Result success: true

---- [Variables values] -----
        name=eventCount [returnValue],  value=2 events

---- [Steps result] -----

        Event Counting
                  [success=true, duration=1217, content-length=589, start time=2021-10-15T10:41:50.349Z, ignore duration=false]

          Count Events - [GET] http://ticketmonsterdev.apicasystem.com/ticket-monster/rest/events?_=1634294510349
                  [Success=true, Duration=1217.13, Content-length=589,Start time=2021-10-15T10:41:50.349Z, Status=(200 : OK)]
                   Timings [ Wait=5.47, DNS=1094.73, SSL/TLS handshake =0.00 TCP=54.16, FirstByte=60.49, Download=2.29]
```

4.3.2 Lab specification

Artifact	Name/Content	Comments
test name	tm_lab2_2.json	
baseURL	*http://ticketmonsterdev.apicasystem.com/ticket-monster*	
variables	*eventCount* – returnValue	The customized return value
Step 1	Name: Event Counting	
Request 1	url: '/rest/events' name: Count Event	
	Extractors eventCount – All Ids , jsonpath, counter	The value of *eventCount* will be used as return value

5 LAB 3 – Example with ASM API

5.1 Purpose

In this lab section we will do exercis with ASM rest API.
See https://api-wpm2.apicasystem.com/v3/Help for information about the API.
This section covers more advanced topics of url-xi scripting and we will use a project for storing shared variables and JavaScript[1] files.

5.2 Create the project api-project

The directory *api-project* is our project directory.

5.2.1 Shared variables

The shared variables in the vars directory. Save as *api-project/vars/common_vars.json*

```
[
    {
        "key": "authTicket",
        "type": "string",
        "usage": "input",
        "hideValue": true,
        "value": "XXX-XXX"
    },
    {
        "key": "asmUrl",
        "type": "url",
        "usage": "input",
        "value": "https://wpm.apicasystem.com"
    }
]
```

- The authTicket does not contain a valid ticket. You edit it or use the -i authTicket=”my ticket” when running the labs.

5.2.2 Include section in all scripts

You should include an include section in all your scripts

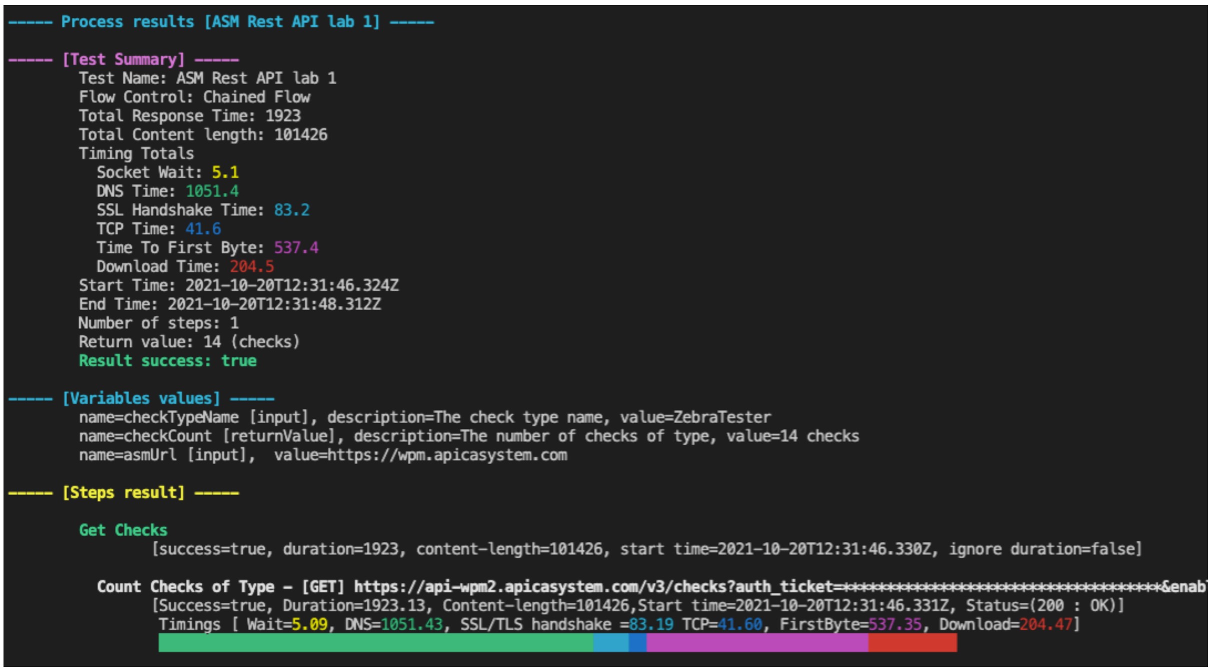
[1] It is possible to write JavaScript directly in the json file, but easier debug and view in an external file. Small JavaScript snippets should be done directly in the json file like an in-line script.

```
"includes": [
        {
            "name": "api-project",
            "scope": "project",
            "type": "vars",
            "src": "common_vars.json"
        }
    ],
```

- The include section should be placed be fore the steps section in the script

5.3 Exercise 1 – Count checks of specific type

5.3.1 Final result



```
----- Process results [ASM Rest API lab 1] -----

----- [Test Summary] -----
        Test Name: ASM Rest API lab 1
        Flow Control: Chained Flow
        Total Response Time: 1923
        Total Content length: 101426
        Timing Totals
          Socket Wait: 5.1
          DNS Time: 1051.4
          SSL Handshake Time: 83.2
          TCP Time: 41.6
          Time To First Byte: 537.4
          Download Time: 204.5
        Start Time: 2021-10-20T12:31:46.324Z
        End Time: 2021-10-20T12:31:48.312Z
        Number of steps: 1
        Return value: 14 (checks)
        Result success: true

----- [Variables values] -----
        name=checkTypeName [input], description=The check type name, value=ZebraTester
        name=checkCount [returnValue], description=The number of checks of type, value=14 checks
        name=asmUrl [input],  value=https://wpm.apicasystem.com

----- [Steps result] -----

        Get Checks
                [success=true, duration=1923, content-length=101426, start time=2021-10-20T12:31:46.330Z, ignore duration=false]

          Count Checks of Type - [GET] https://api-wpm2.apicasystem.com/v3/checks?auth_ticket=****************************************&enab
                [Success=true, Duration=1923.13, Content-length=101426,Start time=2021-10-20T12:31:46.331Z, Status=(200 : OK)]
                 Timings [ Wait=5.09, DNS=1051.43, SSL/TLS handshake =83.19 TCP=41.60, FirstByte=537.35, Download=204.47]
```

5.3.2 Lab specification

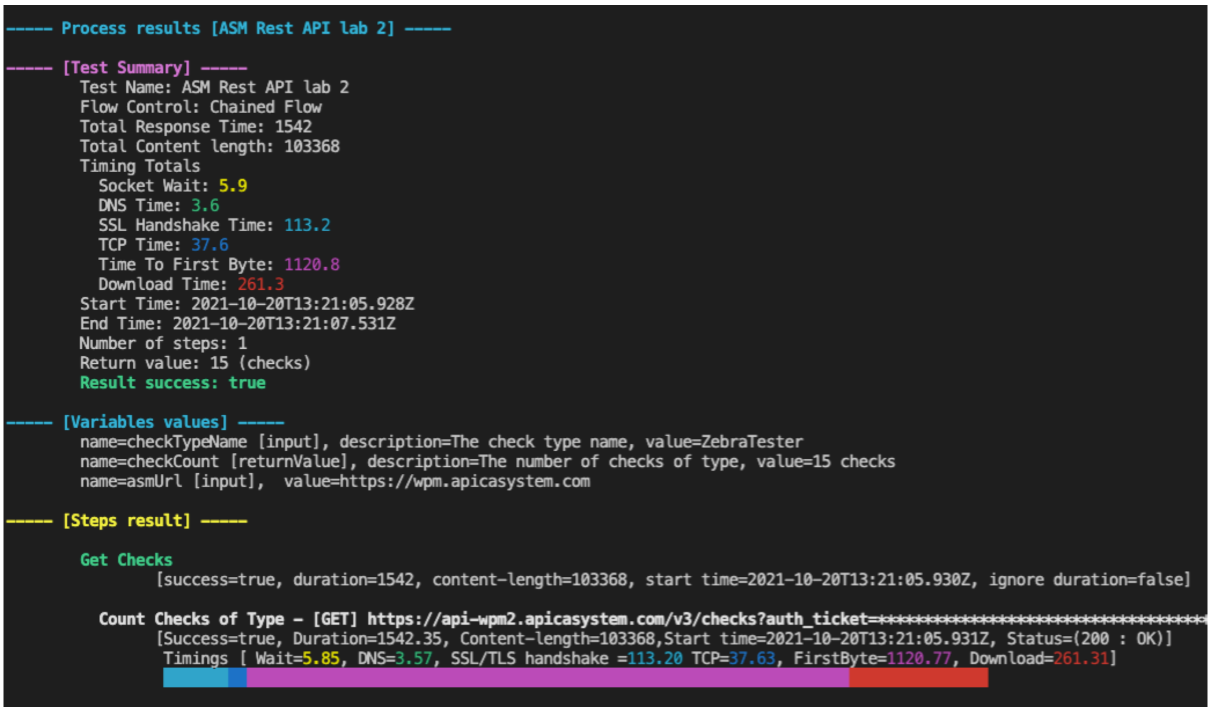
Count checks of specific type . The input parameter checkTypeName should match the json property *check_type_name* in the response array.

Artifact	Name/Content	Comments
test name	api_lab3_1.json	
baseURL	https://api-wpm2.apicasystem.com/v3/	
variables	*checkTypeName* *checkCount* – returnValue	The check type name The customized return value
Includes	*See above for the include*	
Step 1	Name: Get Checks	
Request 1	url: 'checks'	

	name: Count Checks of type	
	Parameters	auth_ticket : {{authTicket}} enabled: “true” severity: “I”
	Extractors checkCount – jsonpath, counter	The value of *checkCount* will be used as return value. You will need to do an advanced jsonpath which filter on the *checkTypeName* value. You can use the mustache expression {{checkTypeName}}

5.4 Exercise 2 – Count checks of specific type

5.4.1 Final result



```
----- Process results [ASM Rest API lab 2] -----

----- [Test Summary] -----
        Test Name: ASM Rest API lab 2
        Flow Control: Chained Flow
        Total Response Time: 1542
        Total Content length: 103368
        Timing Totals
          Socket Wait: 5.9
          DNS Time: 3.6
          SSL Handshake Time: 113.2
          TCP Time: 37.6
          Time To First Byte: 1120.8
          Download Time: 261.3
        Start Time: 2021-10-20T13:21:05.928Z
        End Time: 2021-10-20T13:21:07.531Z
        Number of steps: 1
        Return value: 15 (checks)
        Result success: true

----- [Variables values] -----
        name=checkTypeName [input], description=The check type name, value=ZebraTester
        name=checkCount [returnValue], description=The number of checks of type, value=15 checks
        name=asmUrl [input],  value=https://wpm.apicasystem.com

----- [Steps result] -----

        Get Checks
                [success=true, duration=1542, content-length=103368, start time=2021-10-20T13:21:05.930Z, ignore duration=false]

          Count Checks of Type - [GET] https://api-wpm2.apicasystem.com/v3/checks?auth_ticket=************************************
                [Success=true, Duration=1542.35, Content-length=103368,Start time=2021-10-20T13:21:05.931Z, Status=(200 : OK)]
                 Timings [ Wait=5.85, DNS=3.57, SSL/TLS handshake =113.20 TCP=37.63, FirstByte=1120.77, Download=261.31]
```

5.4.2 Lab specification

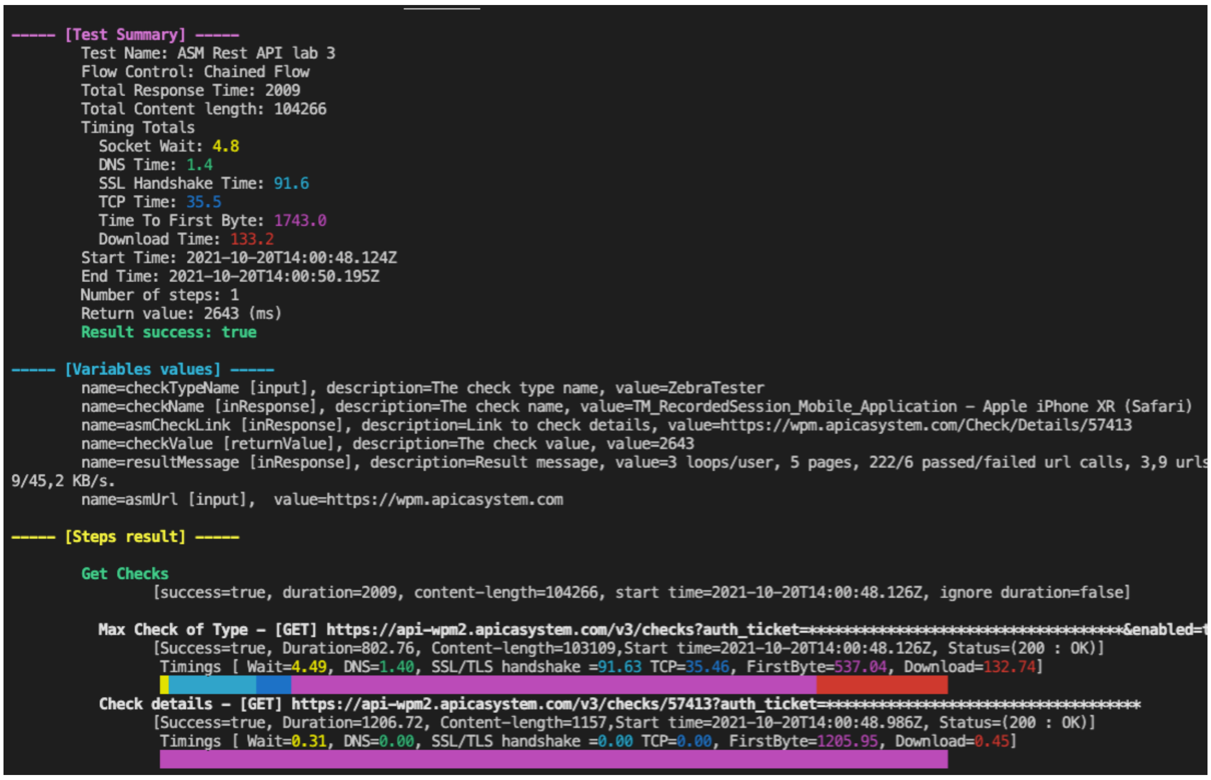
Same as lab 1 but using JavaScript to filter the checks.
See: https://www.npmjs.com/package/@apica-io/url-xi#javascript-support for the syntax of the JavaScript support.

See: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter for how to do a filter on an array.

Artifact	Name/Content	Comments
test name	api_lab3_2.json	
baseURL	https://api-wpm2.apicasystem.com/v3/	
variables	*checkTypeName* *checkCount* – returnValue	The check type name The customized return value
Includes	*See above for the include*	
Step 1	Name: Get Checks	
Request 1	url: ‘checks’ name: Count Checks of type	
	params	auth_ticket : {{authTicket}} enabled: “true” severity: “I”
	script scope: after name:”Count Checks” script:”countChecks.js”	Do a java script filter function on the response array.

5.5 Exercise 3 – Get details of check with max value

5.5.1 Final result



```
----- [Test Summary] -----
        Test Name: ASM Rest API lab 3
        Flow Control: Chained Flow
        Total Response Time: 2009
        Total Content length: 104266
        Timing Totals
          Socket Wait: 4.8
          DNS Time: 1.4
          SSL Handshake Time: 91.6
          TCP Time: 35.5
          Time To First Byte: 1743.0
          Download Time: 133.2
        Start Time: 2021-10-20T14:00:48.124Z
        End Time: 2021-10-20T14:00:50.195Z
        Number of steps: 1
        Return value: 2643 (ms)
        Result success: true

----- [Variables values] -----
        name=checkTypeName [input], description=The check type name, value=ZebraTester
        name=checkName [inResponse], description=The check name, value=TM_RecordedSession_Mobile_Application - Apple iPhone XR (Safari)
        name=asmCheckLink [inResponse], description=Link to check details, value=https://wpm.apicasystem.com/Check/Details/57413
        name=checkValue [returnValue], description=The check value, value=2643
        name=resultMessage [inResponse], description=Result message, value=3 loops/user, 5 pages, 222/6 passed/failed url calls, 3,9 urls
9/45,2 KB/s.
        name=asmUrl [input],  value=https://wpm.apicasystem.com

----- [Steps result] -----

        Get Checks
                [success=true, duration=2009, content-length=104266, start time=2021-10-20T14:00:48.126Z, ignore duration=false]

          Max Check of Type - [GET] https://api-wpm2.apicasystem.com/v3/checks?auth_ticket=************************************&enabled=
                [Success=true, Duration=802.76, Content-length=103109,Start time=2021-10-20T14:00:48.126Z, Status=(200 : OK)]
                 Timings [ Wait=4.49, DNS=1.40, SSL/TLS handshake =91.63 TCP=35.46, FirstByte=537.04, Download=132.74]

          Check details - [GET] https://api-wpm2.apicasystem.com/v3/checks/57413?auth_ticket=************************************
                [Success=true, Duration=1206.72, Content-length=1157,Start time=2021-10-20T14:00:48.986Z, Status=(200 : OK)]
                 Timings [ Wait=0.31, DNS=0.00, SSL/TLS handshake =0.00 TCP=0.00, FirstByte=1205.95, Download=0.45]
```

5.5.2 Lab specification

Return the detailed information of a check with max value for the specified type.

See: https://www.npmjs.com/package/@apica-io/url-xi#javascript-support for the syntax of the JavaScript support.

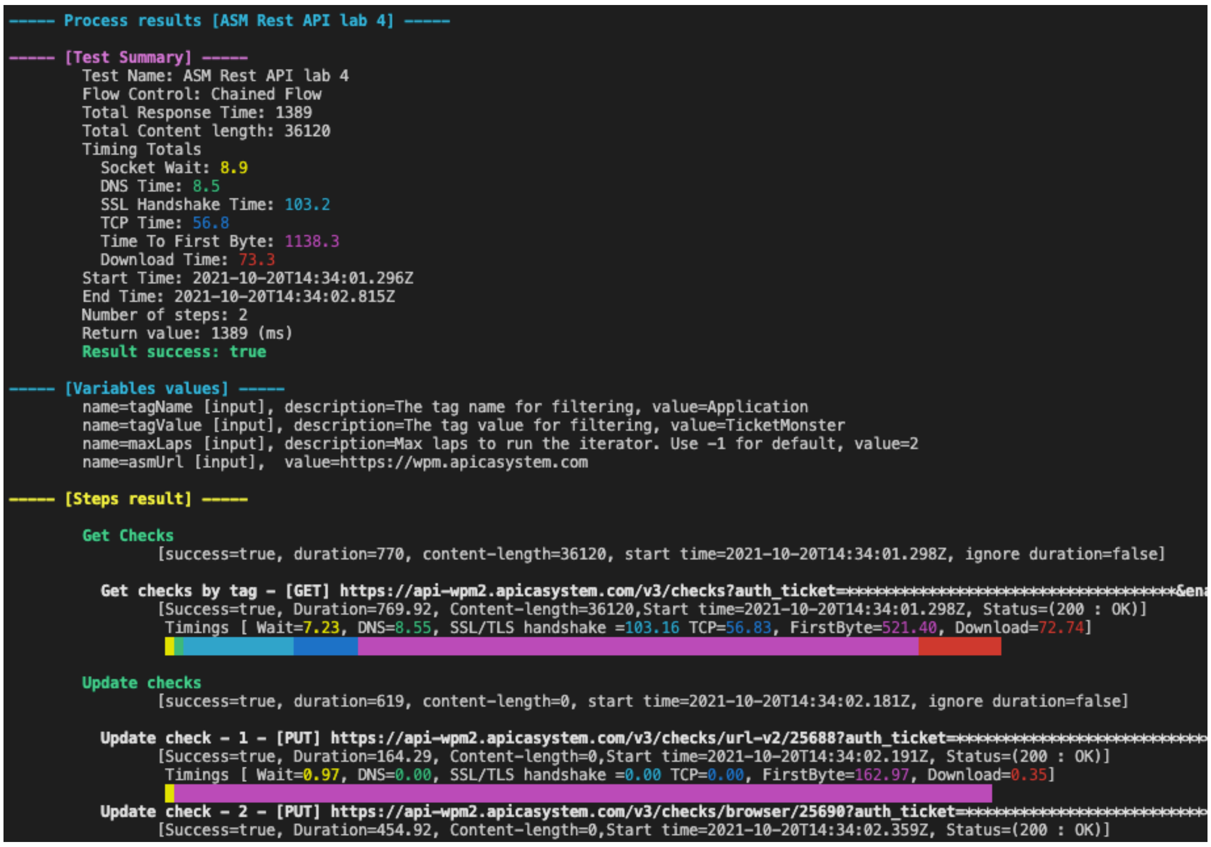
See: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort for how to do a sort of an array.

Artifact	Name/Content	Comments
test name	api_lab3_3.json	
baseURL	https://api-wpm2.apicasystem.com/v3/	
variables	*checkTypeName - input* *checkValue* – returnValue asmCheckLink – in response,url checkName – in response resultMessage – in response	The check type name The customized return value The link to the specific check
Includes	*See above for the include*	
Step 1	Name: Get Checks	
Request 1	url: 'checks' name: Count Checks of type	
	Params:	auth_ticket : {{authTicket}} enabled: "true" severity: "I"
	Extractors checks – jsonpath, array	Use same jsonpath as in lab 1.
	script scope: after name:"Get max Check" script:"getMaxCheck.js"	Do a java script max function on the response array. Return check.id of item 1 in sorted array as variable 'checkId'
Request 2	url: 'checks/{{checkId}}' name: Check details	
	params	auth_ticket : {{authTicket}}
	Extractors checkName – name checkValue – value resultMessage – Last result message	check value will be used as return value

	Tranformers type: “replace "target": "asmCheckLink" ”source”: <the work to do>	Do a transformation to store the link to the check in the *asmCheckLink* variable

5.6 Exercise 4 – Update SLA on all checks matching a tag

5.6.1 Final result



```
----- Process results [ASM Rest API lab 4] -----

----- [Test Summary] -----
        Test Name: ASM Rest API lab 4
        Flow Control: Chained Flow
        Total Response Time: 1389
        Total Content length: 36120
        Timing Totals
          Socket Wait: 8.9
          DNS Time: 8.5
          SSL Handshake Time: 103.2
          TCP Time: 56.8
          Time To First Byte: 1138.3
          Download Time: 73.3
        Start Time: 2021-10-20T14:34:01.296Z
        End Time: 2021-10-20T14:34:02.815Z
        Number of steps: 2
        Return value: 1389 (ms)
        Result success: true

----- [Variables values] -----
        name=tagName [input], description=The tag name for filtering, value=Application
        name=tagValue [input], description=The tag value for filtering, value=TicketMonster
        name=maxLaps [input], description=Max laps to run the iterator. Use -1 for default, value=2
        name=asmUrl [input],  value=https://wpm.apicasystem.com

----- [Steps result] -----

        Get Checks
                [success=true, duration=770, content-length=36120, start time=2021-10-20T14:34:01.298Z, ignore duration=false]

          Get checks by tag - [GET] https://api-wpm2.apicasystem.com/v3/checks?auth_ticket=************************************&ena
                [Success=true, Duration=769.92, Content-length=36120,Start time=2021-10-20T14:34:01.298Z, Status=(200 : OK)]
                 Timings [ Wait=7.23, DNS=8.55, SSL/TLS handshake =103.16 TCP=56.83, FirstByte=521.40, Download=72.74]

        Update checks
                [success=true, duration=619, content-length=0, start time=2021-10-20T14:34:02.181Z, ignore duration=false]

          Update check - 1 - [PUT] https://api-wpm2.apicasystem.com/v3/checks/url-v2/25688?auth_ticket=*************************
                [Success=true, Duration=164.29, Content-length=0,Start time=2021-10-20T14:34:02.191Z, Status=(200 : OK)]
                 Timings [ Wait=0.97, DNS=0.00, SSL/TLS handshake =0.00 TCP=0.00, FirstByte=162.97, Download=0.35]
          Update check - 2 - [PUT] https://api-wpm2.apicasystem.com/v3/checks/browser/25690?auth_ticket=**************************
                [Success=true, Duration=454.92, Content-length=0,Start time=2021-10-20T14:34:02.359Z, Status=(200 : OK)]
```

5.6.2 Lab specification

In this lab exercise we will update the SLA target for all checks matching a specified tag name and tag value.

See: https://www.npmjs.com/package/@apica-io/url-xi#javascript-support for the syntax of the JavaScript support.

Artifact	Name/Content	Comments
test name	api_lab3_4.json	
baseURL	https://api-wpm2.apicasystem.com/v3/	
variables	*tagName – input* *tagValue – input* *maxLaps – number -value =-1*	You can use maxLaps to limit the number of rows in the iterator
Includes	*See above for the include*	
Step 1	Name: Get Checks	
Request 1	url: 'checks' name: 'Get checks by tag'	
	Params	"auth_ticket":"{{authTicket}}" "enabled": "true" "severity": "I" "include_tag": "{{tagName}}~{{tagValue}}"
	Extractors checks, array	All checks
Step 2	Update checks	
	iterator: value: "{{checks}}" varName : "check" maxLaps: "{{maxLaps}}"	
Request 1	url: 'fixed in before script' name: 'Update check – {{$lapidx1}}' method: 'put' data {}	The configuration url data should be fixed in the before JavaScript to the request. The check.check_type_api are often part of the url. For some check types it needs to be changed.
	params	auth_ticket : {{authTicket}}
	scripts scope:"before" name:"PrepareUpdate" script:"prepareUpdate.js"	The script should create the url and populate data. Data is the request body. You can set sla_target and sla_target_2 to 99.5

6 Lab 4 – Use certificates and encryption

6.1 Purpose

The purpose of the labs in this section is to understand how you can use certificates for authentication and how you can encrypt files like certificates.

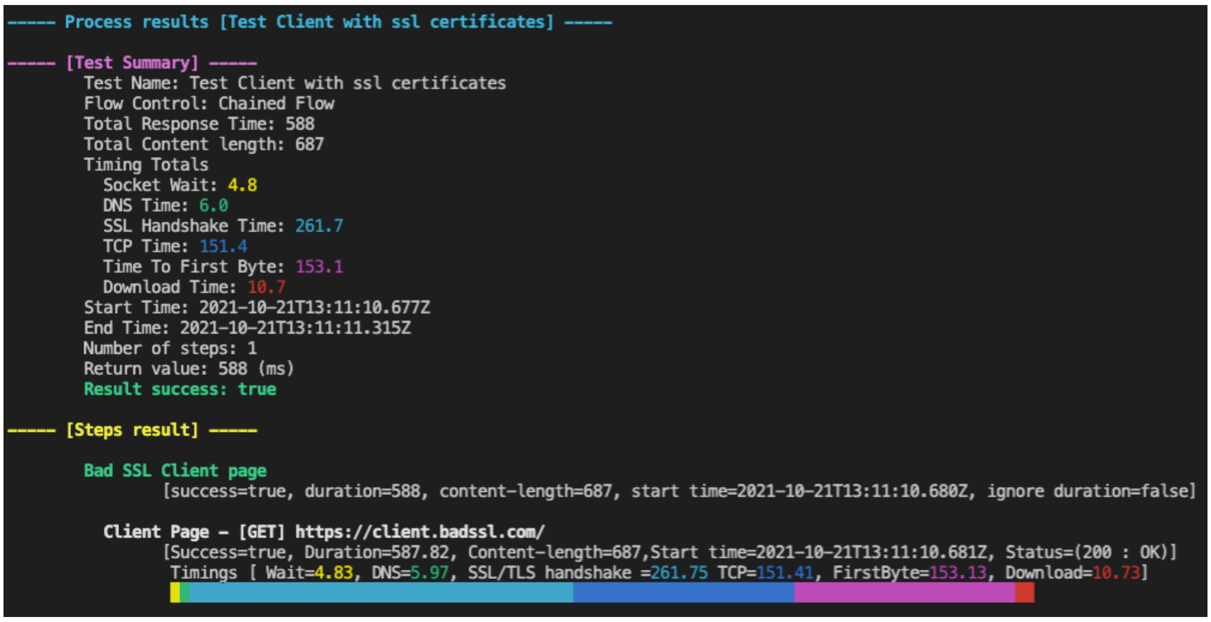
6.2 Instructions

We will not give detailed instructions for the labs in this section. You need figure out how to do it be reading the documentation in npm for url-xi.

6.3 Lab 1

Authenticate to https://client.badssl.com with the provided certifates we give you

6.3.1 Final results



```
----- Process results [Test Client with ssl certificates] -----

----- [Test Summary] -----
        Test Name: Test Client with ssl certificates
        Flow Control: Chained Flow
        Total Response Time: 588
        Total Content length: 687
        Timing Totals
          Socket Wait: 4.8
          DNS Time: 6.0
          SSL Handshake Time: 261.7
          TCP Time: 151.4
          Time To First Byte: 153.1
          Download Time: 10.7
        Start Time: 2021-10-21T13:11:10.677Z
        End Time: 2021-10-21T13:11:11.315Z
        Number of steps: 1
        Return value: 588 (ms)
        Result success: true

----- [Steps result] -----

        Bad SSL Client page
                [success=true, duration=588, content-length=687, start time=2021-10-21T13:11:10.680Z, ignore duration=false]

          Client Page - [GET] https://client.badssl.com/
                [Success=true, Duration=587.82, Content-length=687,Start time=2021-10-21T13:11:10.681Z, Status=(200 : OK)]
                 Timings [ Wait=4.83, DNS=5.97, SSL/TLS handshake =261.75 TCP=151.41, FirstByte=153.13, Download=10.73]
```

6.3.2 Hints

- Copy recursively the *badssl_certs* directory from the solution directory to the working directory.
- You can choose to use the pem based certificates or the p12 based certificate. Password and passphrase is badssl.com for p12(pfx) certificate.
- You need a httpAgent object and include section in the test configuration file.

6.4 Lab 2 – Test encryption

Use the same lab as in lab 1 with encryption.

6.4.1 Hints

- Copy the badssl_certs directory to ec_badssl_certs directory
- Install Crypitfy from npm
- Encrypt the certificates with cryptify
- Run with the -dk <key> option