JMETER

TABLE OF CONTENTS

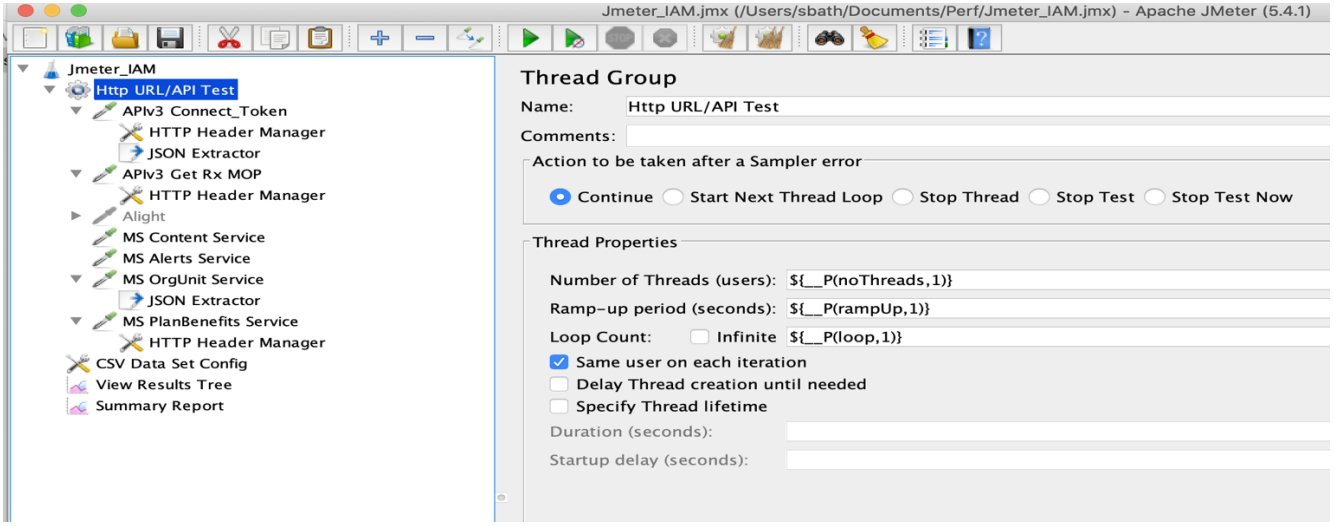
1. GETTING STARTED – INSTALLATION AND SETUP

JMeter is an open-source software that can perform stress test, load test, performance-oriented functional test, regression test, etc. In order to get started:

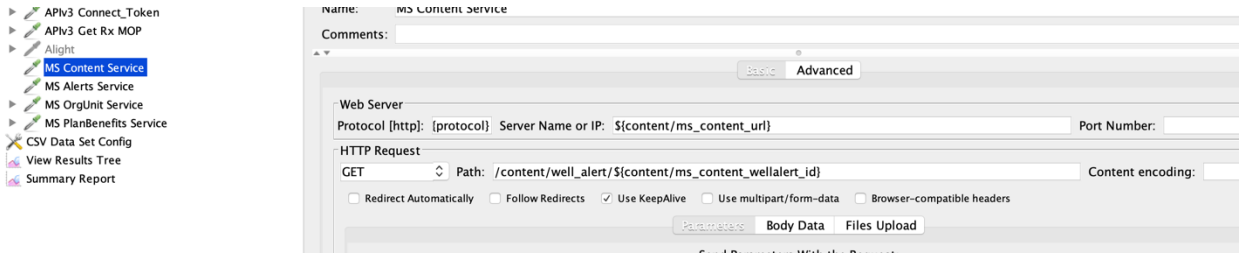
- Download Apache JMeter from https://jmeter.apache.org/download_jmeter.cgi, into the directory where you want JMeter to be installed.
- Open the terminal/ command line and navigate to the bin folder where JMeter is installed.
- Run the jmeter.bat (for Windows) or jmeter.sh (for Linux) file.
- This should launch the JMeter GUI

2. API TESTS USING HTTP REQUEST THREAD GROUP

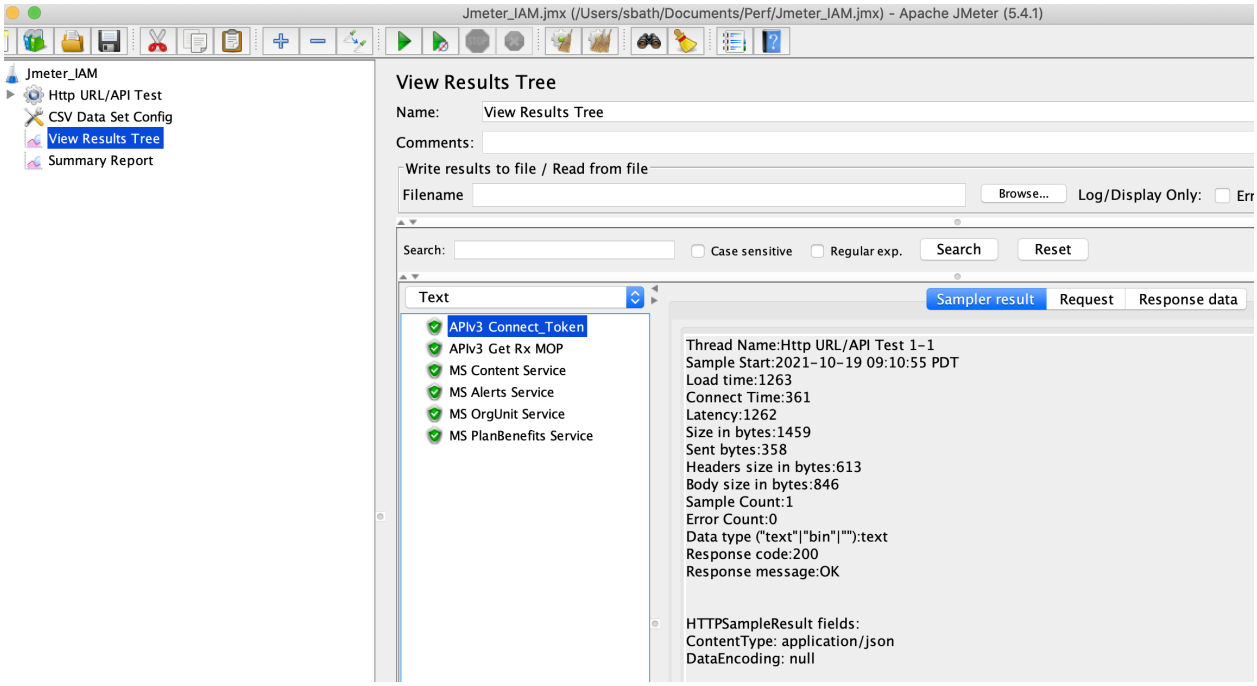
1. Create a new Test Plan from the GUI.
2. Add a Thread Group inside the Test Plan and keep the number of users, ramp up and the loop counters to be configurable and default to 1. The syntax example is: ${__P(noThreads,1)}
3. See screenshot below for other thread properties:



4. Right click the thread group and Add > Sampler > HTTP Request.
5. Inside the HTTP Request provide the Server name, HTTP Request type and the Path on which we are going to perform the API testing. Example – MS Content Service in the Jmeter_IAM test plan.



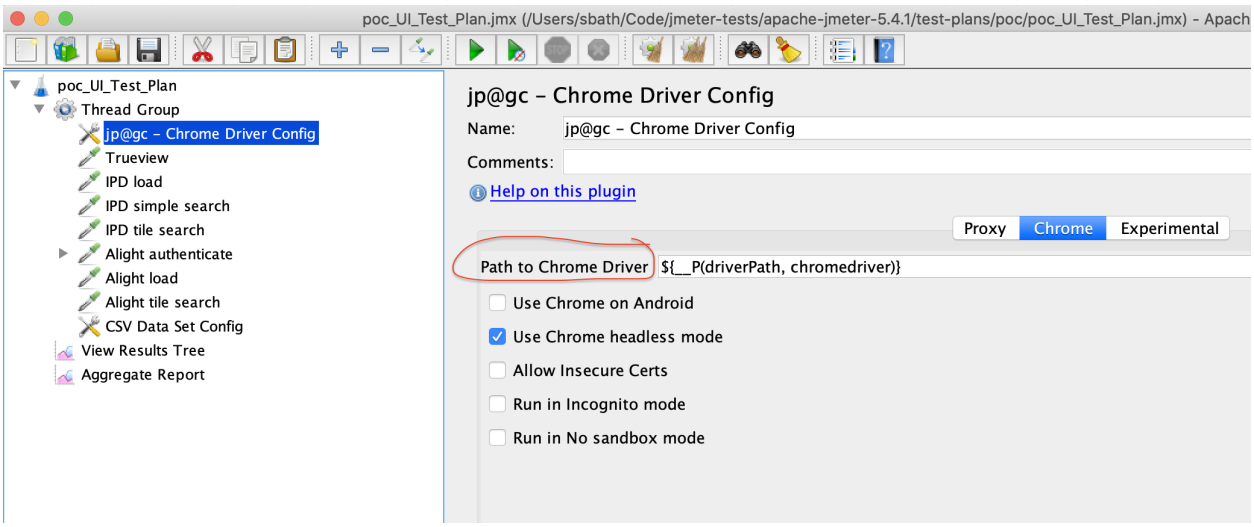
6. Provide any parameters or body data if expected.
7. Right click the test plan and Add > Config Element > CSV Data Set Config and provide the path to the csv file with the test data.
8. Right click the test plan and Add > Listener > View Results Tree. Similarly, there are other report listeners that can be added (ex: Aggregate Report)
9. For debug purposes, it is useful to Right click the test plan and Add > Sampler > Debug Sampler
10. Save the test plan and it should get saved with ".jmx" extension.
11. After saving the file, run the script by clicking the green button on the UI and results will be displayed.
12. Click on the View Results tree to view the test run.
13. If the response code is 200, it means that test was ran successfully.



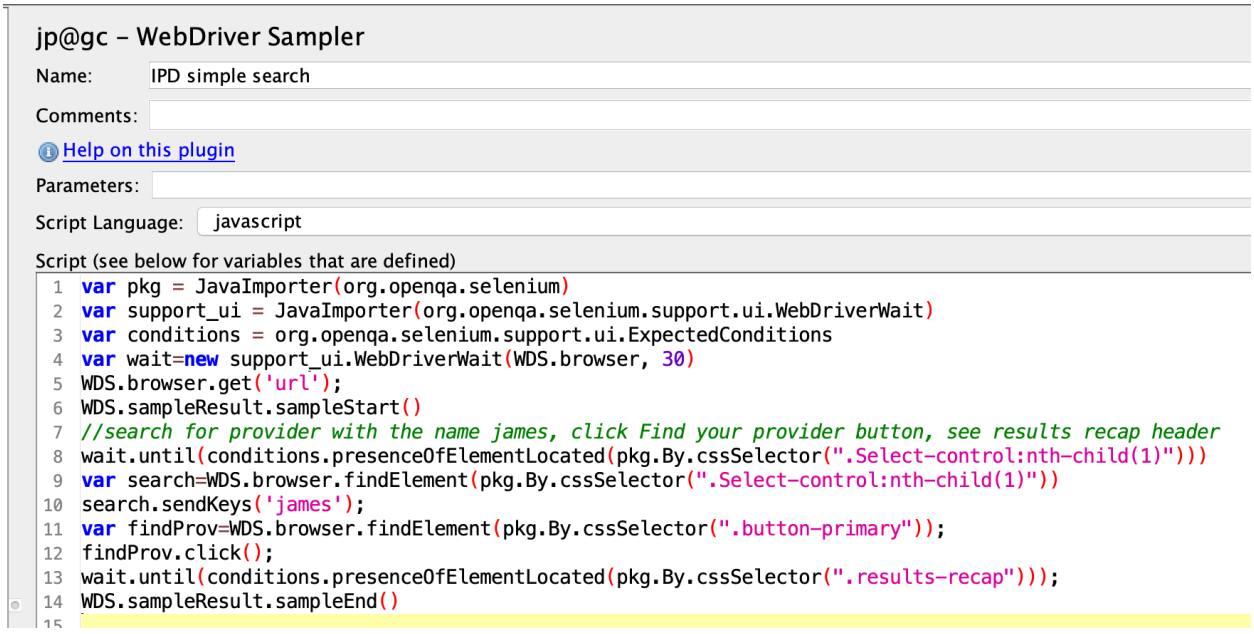
3. UI TESTS USING WEBDRIVER SAMPLER

PREREQUISITE: DOWNLOAD **PLUGINS-MANAGER.JAR** AND PUT IT INTO LIB/EXT DIRECTORY, THEN RESTART JMETER.

1. Create a new Test Plan from the GUI.
2. Add a Thread Group inside the Test Plan and keep the number of users, ramp up and the loop counters to be configurable and default to 1.
3. Right click the thread group and Add > Config Element > Chrome Driver Config. Make sure to save the chrome driver in the bin folder. Path to chrome driver is configurable. For windows it requires the extension.exe in the path



4. To add a test plan, right click the thread group and Add > Sampler > Web Driver Sampler
5. Check the poc_ui_test_plan.jmx in the gitlab project on how to create a UI test plan.



6. Right click the test plan and Add > Config Element > CSV Data Set Config and provide the path to the csv file with the test data.
7. Right click the test plan and Add > Listener > View Results Tree. Similarly, there are other report listeners that can be added (ex: Aggregate Report).
8. For debug purposes, it is useful to Right click the test plan and Add > Sampler > Debug Sampler.
9. Save the test plan and it should get saved with ".jmx" extension.
10. After saving the file, run the script by clicking the green button on the UI and results will be displayed.
11. Click on the View Results tree to view the test run.

4. RUNNING TESTS FROM COMMAND LINE:

1. API TESTS

```
bin % sh jmeter -JnoThreads=2 -JrampUp=1 -Jloop=2 -Jfilename=/local jmeter repo path/jmeter-tests/apache-jmeter-5.4.1/test-data/data_dev.csv -n -t /local jmeter repo path/jmeter-tests/apache-jmeter-5.4.1/test-plans/poc/poc_Jmeter_IAM.jmx -l /local jmeter repo path/jmeter-tests/apache-jmeter-5.4.1/test-reports/api/csv/$(date +"%m-%d-%Y-%H%M%S")_results.csv -e -o /local jmeter repo path/jmeter-tests/apache-jmeter-5.4.1/test-reports/api/html/$(date +"%m-%d-%Y-%H%M%S")_results.html
```

2. UI TESTS

```
bin % sh jmeter -JnoThreads=2 -JrampUp=1 -Jloop=2 -Jfilename=/local jmeter repo path/jmeter-tests/apache-jmeter-5.4.1/test-data/staging_ui.csv -JdriverPath=chromedriver -n -t /local jmeter repo path/jmeter-tests/apache-jmeter-5.4.1/test-plans/poc/poc_UI_Test_Plan.jmx -f -l /local jmeter repo path/jmeter-tests/apache-jmeter-5.4.1/test-reports/ui/csv/$(date +"%m-%d-%Y-%H%M%S")_results.csv -e -o /local jmeter repo path/jmeter-tests/apache-jmeter-5.4.1/test-reports/ui/html/$(date +"%m-%d-%Y-%H%M%S")_results.html
```

-JnoThreads: represents number of users

-JrampUp: time take to ramp-up to the full number of threads chosen

-Jloop: number of iterations

-Jfilename: csv input test data file location

-JdriverPath: browser driver path on the disk

-n: indicator to run my Jmeter script in Non GUI mode

-t: This will pick the Jmeter .jmx file and execute it

-f: force delete existing results file

-l: path to save the executed results into .csv format

-e: convert the .csv results into HTML format

-o: This command saves the html results into given output folder

5. JMETER RESULTS ANALYSIS

Summary Statistics - one of the useful summaries in JMeter Dashboard Report HTML. Below is the sample screenshot:

Requests	Executions			Response Times (ms)							Throughput	Network (KB/sec)	
Label	#Samples	FAIL	Error %	Average	Min	Max	Median	90th pct	95th pct	99th pct	Transactions/s	Received	Sent
Total	**142**	**22**	**15.49%**	**474.96**	**0**	**9519**	**141.00**	**893.10**	**1978.45**	**9104.91**	**2.10**	**9.90**	**8.67**
Accounts	1	1	100.00%	159.00	159	159	159.00	159.00	159.00	159.00	6.29	7.71	2.27
Accounts Ruby Service: Get Account Balance	1	0	0.00%	5.00	5	5	5.00	5.00	5.00	5.00	200.00	68.95	85.55
Accounts Ruby Service: Healthcheck	1	0	0.00%	23.00	23	23	23.00	23.00	23.00	23.00	43.48	8.49	7.56
Accounts Ruby Service: POST Account Balance	1	0	0.00%	18.00	18	18	18.00	18.00	18.00	18.00	55.56	19.69	32.34
Accounts Service: Get Account Balance	1	1	100.00%	141.00	141	141	141.00	141.00	141.00	141.00	7.09	4.11	1.40
Accounts Service: Ping Test	1	0	0.00%	18.00	18	18	18.00	18.00	18.00	18.00	55.56	35.92	9.06
AccountsRuby	1	0	0.00%	46.00	46	46	46.00	46.00	46.00	46.00	21.74	19.45	25.73
Alerts	1	0	0.00%	723.00	723	723	723.00	723.00	723.00	723.00	1.38	1.59	0.98
Alerts Service: Get Completed Participant Alerts Test	1	0	0.00%	230.00	230	230	230.00	230.00	230.00	230.00	4.35	1.61	0.86

- "**Request**" Label/name of the request microservices/item executed.
- "**Executions**" - Compose of number of samples versus errors ~ calculated and an error percentage. This is done for all requests in the test, as well as for each specific request. You might select an acceptable error percentage (e.g., 0.1% or 0.01% depending on the system being tested) as one of your test's acceptance criteria.
- "**Response Time**" is the overall average response as well as individual (microservices) request response time (in milliseconds). Percentiles also being reported in this table ~ 90th, 95th, and 99th percentiles.
- "**Throughput**" are the request transactions were completed in total and for each type of request.
- **Error Analysis** - this is where the error details ~ debug your load test or looking for the cause of a problem.

Type of error	Number of errors	% in errors	% in all samples
Non HTTP response code: java.net.UnknownHostException/Non HTTP response message: ches-qa-apiv3-service.qa.ches.awsnonprod.healthcareit.net	20	90.91%	14.08%
403/Forbidden	1	4.55%	0.70%
Non HTTP response code: java.net.UnknownHostException/Non HTTP response message: ches-qa-apiv3-service.qa.ches.awsnonprod.healthcareit.net: Name or service not known	1	4.55%	0.70%