Overview

The API Functional Monitoring solution enables developers and operators to perform consistent and constant testing of the functional behavior and performance of their APIs, throughout the entire API lifecycle, in both non-production and production environments.

API Functional Monitoring drives up the quality and reliability of publicly exposed APIs as they are being developed, tested and delivered into production. The monitor is not only specific to APIs that you might develop on Anypoint Platform, it can also be used to perform quality and reliability tests on any publicly exposed API.

In this Lab, we are going to create an API Functional Monitoring test that will call APIs on a regular interval and perform tests to ensure reliability and quality of a publicly exposed API.

Step 1: Test Endpoint in API Console

1. In the previous lab we reviewed Visualizer and the functionality available. Please return to Visualizer and select the node that you deployed in the previous lab.

2. Copy and paste the hostname of the application and paste it into a browser adding /console to the end of the url. This will open up the console for the API you deployed.
3. We want to select an endpoint to use in our functional monitor. To keep this simple let’s select the GET Product by id endpoint.

4. We need to give the endpoint an id to get a resource. We will use the value 1 for the id. In the product_id field on the console enter the value 1 and then click SEND
You should see the output for a product, possibly an ELMO toy.

5. Now the endpoint is working you should copy the url from the **request url** field and make a note of this for use in the functional monitor.
Step 2: Create a Functional Monitor

1. Return to Anypoint Platform and from the main menu in the top left, navigate to Monitoring and then click Functional Monitoring where you will be presented with a page that looks similar to this:
Monitor your APIs to validate the performance every day or every hour. Schedule a regular series of monitors to validate how it works across all environments.

Create monitor
Upload an existing monitor
2. We want to create a monitor, click **Create** You then be presented with a set of steps. First step is to give the monitor a name. Use your initials and the following **-omnichannel-monitor** For example, below Max Mule would look like this:

```
mm-omnichannel-monitor
```

3. Click next and you are onto step 2 of the process:
Here you are presented with a way to specify the endpoint & HTTP method that you will use to monitoring an API. There are two types of editor available, basic & advanced:

- **Basic** provides a way to call one endpoint using one method and also lets you specify the expected response code and headers.
- **Advanced** allows you to add multiple endpoints and create more complex assertions based on status codes, response bodies, and response headers.

4. We will build an advanced monitor with more than 1 assertion.

Click the **Advanced** option and you will be presented with the following:
5. Go back to the browser tab with the API console for the omni-channel-api and copy the request url for the query that was performed.

Then return to the browser tab with the functional monitor and paste in the request url.
6. Now we want to add some assertions to ensure we get the functional behaviour we are expecting from the API.

There are 2 assertions we are going to add:

a. Status code must equal 200
b. Response body id must equal 1

To add an additional assertion click the **add assertion** button when needed

7. You then have the choices for where you want your monitor to execute from and what time schedule you want the call to occur.

Leave the default Execution Location as us-east-1 and leave the monitor schedule at 15 minutes

Click **Next** to continue

8. Step 3 is where we can add reporters which feed 3rd party notification tools. You will add an email reporter as part of this step.
Click **Add Reporter** and 2 boxes will appear. On the first box open the drop down list and choose email

![Add Reporter](image)

Once Email is selected move to the next box, add your email address and press **enter** on the keyboard. You can enter multiple email addresses as needed.

**TIP** You need to press the **Enter** key to register the email address in the field.

9. The configuration of the monitor is now complete. The final step is to click the create button the bottom of the page.

Wait a few moments for the monitor to be created and deployed. The test will execute and start to give feedback on the executions.

**Step 3: Review Functional Monitor Output**

1. You can view the history of the executions by clicking on the timer symbol

![Monitor Output](image)

2. The history of the tests will show each execution at a high level
To review the test outputs click on **Detail** which shows open another view.

Click the down arrow on the right hand side to review further details on the execution including input HTTP and output HTTP headers and payload. If errors occur on the functional test you will get all the details on the input and output of the call in a downloadable file.

3. Return to the main functional monitoring menu by clicking on the item in the menu on the left
4. Select your functional monitor that you created.
You can get really advanced with functional monitors. Functional monitors can be written in Dataweave which gives you the ability to do much more with your functional monitoring activity.

5. Click the down arrow symbol to download the test file.

![Image of download button and functional monitor](image)

A zip file will be downloaded. Open the archive and then open the `main.dwl` file inside. You will see some Dataweave code that looks similar to the below:

```dataweave
import * from bat::BDD
import * from bat::Assertions
---
suite("mm-omnichannel-monitor1") in [  
  GET 'http://mm-mon-omni-channel-api.us-e2.cloudhub.io/api/products/product/1' with {  
    "headers": {}  
  }  
  assert [
    $.response.status mustEqual 200,
    $.response.body.id mustEqual 1
  ]
]
```

As an extended exercise in your own time, feel free to edit these scripts and upload them back into the platform to create more advanced functional monitors. You can find all the reference material [here](reference).

### Summary

In this lab, you completed the following steps:

- Tested the endpoint using the API Console
- Created and exercised a functional monitor
- Reviewed the output of the functional monitor executions and seen how the functional monitors are built up in Dataweave code.

Go Further:
• See the Functional Monitor doc for more information.

Congratulations! You have completed Lab 3, the final lab of Module 6

Take me to the TOP