**How to use script to replicate MQ extension health rules**

There is a menu in the script. The menu options should be executed in sequential order, from 1 to 4. Run the script with

$python3 mq\_sammons.py

A screenshot of a computer

Description automatically generated

Use the bearer token configured named “MQ” in sammons or sammons-test controllers. Go to Gear Icon in top right of UI > Administration > API Clients –

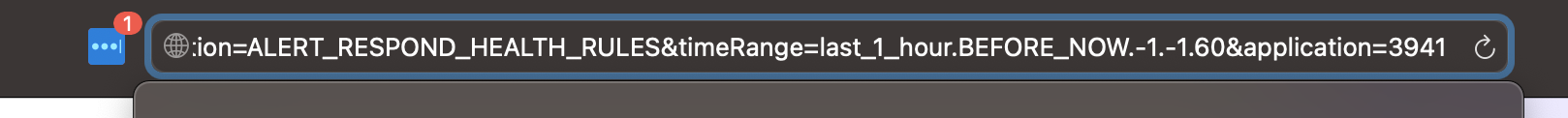
A screenshot of a computer

Description automatically generated

If you get an authentication error when running the script, regenerate the Secret using the “Generate Secret” button in the above screenshot, then configure this in the “api\_token\_input” field on line 9 of the script. Save the new secret using the save button on the above screenshot.

Option 1

1. Pick a health rule that you want to replicate from the IBM MQ extension. Take down the AppDynamics application name and health rule name. If using a health rule under the “Servers” tab, you will need to get the ID for “Servers” from the URL.   
     
   To get this ID, which can differ based on the AppDynamics controller, go to the “Servers” tab > health rules and look at the very end of the URL to get that ID.  
   A screenshot of a computer

   Description automatically generated  
     
   ID found at the end of the browser URL above -  
   
2. A screenshot of a computer program

   Description automatically generated  
     
     
   The name of the health rule is hard-coded to be “health\_rule.xml”  
   A screen shot of a computer

   Description automatically generated
3. Edit the exported health rule. You will replace the queue name occurrences with the string “PLACEHOLDER”. The next function run in the script will use that name to substitute in the queue names from a file and create XML health rule payloads. There are 5 lines where “PLACEHOLDER” is used, one for the name:  
     
   <name>MQ DEVMG01 - PLACEHOLDER - Queue Depth greater than 90 percent</name>  
     
     
     
     
   Two lines for each metric expression variable (4 total) -  
   <logical-metric-name>Application Infrastructure Performance|Root|Individual Nodes|VSR-D-MWMQ1|Custom Metrics|WebsphereMQ|DEVMG01|Queues|PLACEHOLDER|Current Queue Depth</logical-metric-name>  
     
   <metric-name>Custom Metrics|WebsphereMQ|DEVMG01|Queues|PLACEHOLDER|Current Queue Depth</metric-name>  
     
     
   <logical-metric-name>Application Infrastructure Performance|Root|Individual Nodes|VSR-D-MWMQ1|Custom Metrics|WebsphereMQ|DEVMG01|Queues|PLACEHOLDER|Max Queue Depth</logical-metric-name>  
     
   <metric-name>Custom Metrics|WebsphereMQ|DEVMG01|Queues|PLACEHOLDER|Max Queue Depth</metric-name>  
     
   The PLACEHOLDER is where new queue names will be substituted in the next script.0 percent</name>
4. Now it is time to run the next function in Menu Option 2 from the script.  
   get\_queue\_extension\_names(app)  
     
   The app name is used as the single parameter input, which is “3941”. This function will output the names of the queues, based on a hard-coded metric path. This metric path is   
   "Application Infrastructure Performance|Root|Individual Nodes|VSR-Q-MWMQ1|Custom Metrics|WebsphereMQ|QAMG01|Queues"  
     
   You can substitute any name from your metric browser for VSR-Q-MWMQ1.  
     
   A screenshot of a computer

   Description automatically generated  
     
   to run. A file will be created with the outputted queue names called “queue\_names.txt”  
     
   Remove any queues from this text file that you do NOT want to create health rules for, such as dead letter queues.

1. Now you can run the next function (Menu Option 3) to create the XML health rule payloads in your script’s home directory called “queue\_hr\_payloads”.  
   A screen shot of a computer

   Description automatically generated  
     
   Below you can see all of these payloadA screenshot of a black screen

   Description automatically generated
2. Now it is time to run the last function (Menu Option 4)  
   import\_health\_rules\_from\_dir  
     
   That takes the directory with the XML health rule payloads as the first input parameter, the controller URL as the second, and the app name/ID “3941” as parameters. Running this final function will create health rules for each of the payloads created in the previous step. You may want to validate a few of those payloads for correctness before the import, as this final step can create hundreds of rules.  
   A screen shot of a computer

   Description automatically generated

Verify your MQ health rules with correct configurations are in the controller –

A screenshot of a computer

Description automatically generated