MQ 9.1 REST API, remote administration via a gateway queue manager

https://www.ibm.com/support/pages/node/6209110

Date last updated: 13-May-2020

Angel Rivera – rivera@us.ibm.com
IBM MQ Support

+++ Objective

Once you have a MQ 9.1 Web Server in a given Installation (such as Installation1) in a server, you can use the MQ Web Console to administer local queue managers in the SAME installation as the Console (that is, in Installation1, but not in Installation2)
The MQ REST API, however, can work with both local and remote queue managers.

The objective of this tutorial is to demonstrate how to setup a gateway queue manager that is in the same Installation of the MQ Web Server on a host that has MQ 9.1, to administer via the MQ REST API a set of remote queue managers.

A big benefit is that you do not need to install the MQ Web Server on every remote server that has queue managers that you want to administer via the MQ REST API.

The remote queue managers need to have direct connectivity with the gateway queue manager by means of server and receiver channels.

LIMITATION: The remote queue managers must be MQ 8.0 or later.

The focus for this tutorial is to interact with the MQ REST API with No Security, which might be suitable for Testing environments, but not suitable for Production ones.
The rationale is explained in the 2 main tutorials referenced later on regarding the Configuration of the MQ Web Server and Using the MQ REST API.

The chapters are:

++ Main reference

https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.adm.doc/q131070_.htm
IBM MQ 9.1.x / IBM MQ / Administering / Administration using the REST API /
Remote administration using the REST API

- begin excerpt

[V9.1.0 Jul 2018]

You can use the REST API to administer remote queue managers, and the IBM® MQ objects that are associated with those queue managers.

This remote administration includes queue managers that are on the same system, but not in the same IBM MQ installation as the mqweb server.

Therefore, you can use the REST API to administer your entire IBM MQ network with only one installation that runs the mqweb server.

To administer remote queue managers, you must configure the administrative REST API gateway so that at least one queue manager in the same installation as the mqweb server acts as a gateway queue manager.

Then, you can specify the remote queue manager in the REST API resource URL to perform the specified administrative action.

+ end excerpt

++ Other references

+ See the following blog:

https://developer.ibm.com/messaging/2017/11/14/rest-api-gateway-now-can-manage-queue-managers-rest/
The REST API Gateway – now you can manage all your queue managers with REST!
gwydiontudur
Published on 14/11/2017 / Updated on 30/07/2018
=> It is focused on MQ 9.0.4 CD.

++ Configuration

+ Host-1: orizaba1.fyre.ibm.com
Linux RHEL 7.6

InstName: Installation1
InstPath: /opt/mqm
Version: 9.1.5.0
Queue manager: QMORI915 Port: 1415 = = > This is going to be the GATEWAY
This is shown as QM1 in the diagrams included in this tutorial.

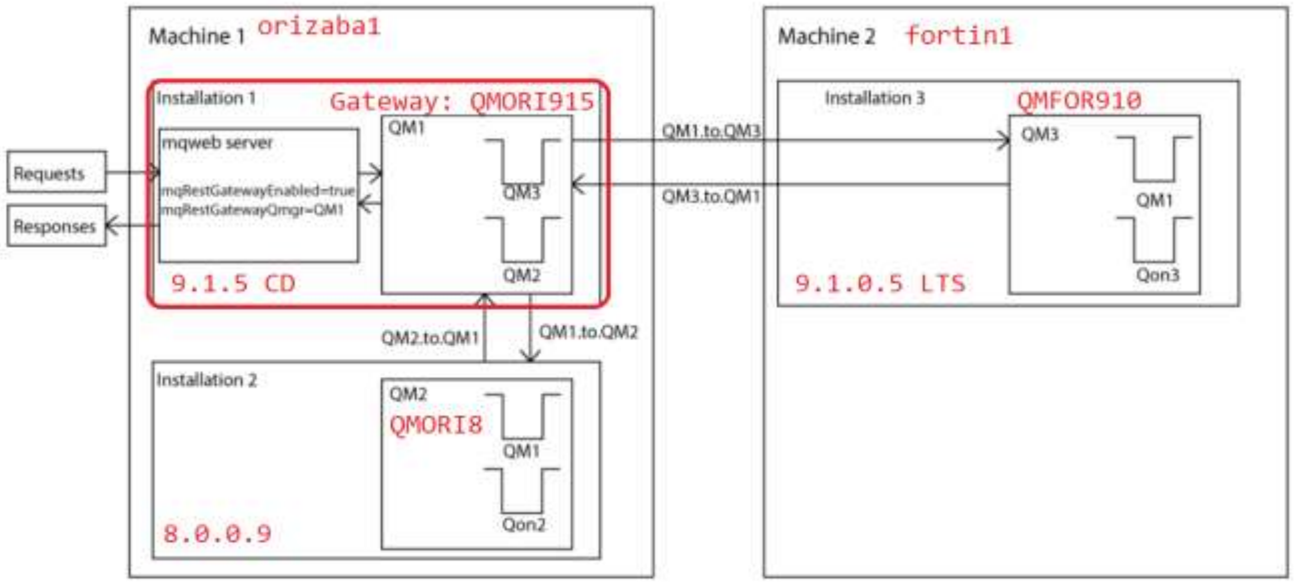
InstName: Installation2
InstPath: /opt/mqm80
Version: 8.0.0.9
Queue manager: QMORI8 Port: 1416
This is shown as QM2 in the diagrams included in this tutorial.

+ Host-2 : fortin1.fyre.ibm.com

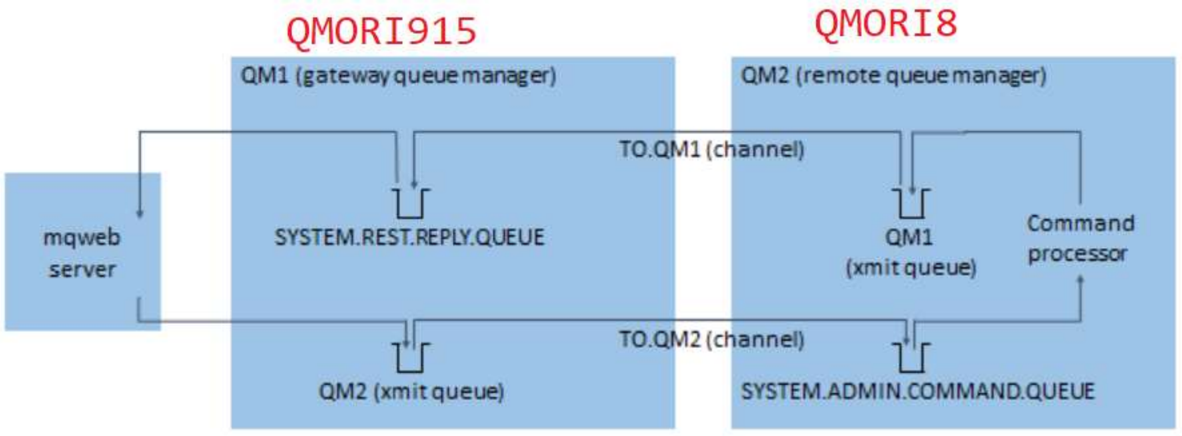
InstName: Installation2
InstPath: /opt/mqm910
Version: 9.1.0.5
Queue manager: QMFOR910 Port: 1420
This is shown as QM3 in the diagrams included in this tutorial.

The Main Reference has a useful diagram and it maps well for this tutorial:

Figure 1. Diagram of example configuration for remote administration by using the REST API.



The referenced Blog has another useful diagram that illustrates how an MQ REST API command that is issued by the MQ Web Server gets routed to a Gateway queue manager which in turn routes it to the remote queue manager (lower flow) and how is the response returned (upper flow).



++
+++ Chapter 1: Setup of MQ Console, MQ REST API and establish Gateway
++

+ See the following tutorials

https://www.ibm.com/support/pages/node/6118000
Configuring MQ 9.1 Web Server in Linux and in Windows with No Security (for Testing the MQ Web Console)

https://www.ibm.com/support/pages/node/6208006
Using the MQ 9.1.5 CD REST API in Linux with no security (for Testing environments)

+ Enable the Gateway for the MQ Web Server

It is necessary to stop the MQ Web Server:

```
  endmqweb
```

Enable the administrative REST API gateway by using the following command:

```
  setmqweb properties -k mqRestGatewayEnabled -v enabled
```

Configure which queue manager is used as the default gateway queue manager:

```
  setmqweb properties -k mqRestGatewayQmgr -v QMORI915
```

Restart the MQ Web Server

```
  strmqweb
```

Display the ports:

```
  dspmqweb
  MQWB1124I: Server 'mqweb' is running.
  URLS:
    https://orizaba1.fyre.ibm.com:9443/ibmmq/console/
    http://orizaba1.fyre.ibm.com:9080/ibmmq/console/
    https://orizaba1.fyre.ibm.com:9443/ibmmq/rest/
    http://orizaba1.fyre.ibm.com:9080/ibmmq/rest/
```

View the current configuration of the administrative REST API gateway by using the following command:

```
  dspmqweb properties -a
```

```
name="httpHost" value="*"
name="httpPort" value="9080"
name="httpsPort" value="9443"
name="ltpaCookieName" value="LtpaToken2_${env.MQWEB_LTPA_SUFFIX}"
name="ltpaExpiration" value="120"
name="managementMode" value="standard"
name="maxMsgTraceFileSize" value="200"
name="maxMsgTraceFiles" value="5"
name="maxTraceFileSize" value="20"
name="maxTraceFiles" value="2"
name="mqConsoleAutostart" value="true"
name="mqConsoleEarName" value="com.ibm.mq.webconsole"
name="mqConsoleFrameAncestors" value=""
name="mqRestAutostart" value="true"
name="mqRestCorsAllowedOrigins" value="*"
name="mqRestCorsMaxAgeInSeconds" value="0"
name="mqRestCsrfValidation" value="true"
name="mqRestGatewayEnabled" value="enabled"
name="mqRestGatewayQmgr" value="QMORI915"
name="mqRestMessagingEnabled" value="true"
name="mqRestMessagingFullPoolBehavior" value="overflow"
name="mqRestMessagingMaxPoolSize" value="20"
name="mqRestMftCommandQmgr" value=""
name="mqRestMftCoordinationQmgr" value=""
name="mqRestMftEnabled" value="false"
name="mqRestMftReconnectTimeoutInMinutes" value="30"
name="mqRestRequestTimeout" value="30"
name="secureLtpa" value="true"
name="traceSpec" value="*=info"
MQWB1100I: The 'dspmqweb' command completed successfully.
```

The mqRestGatewayEnabled field shows whether the gateway is enabled, and the mqRestGatewayQmgr field shows the name of the default gateway queue manager:

```
  dspmqweb properties -a | grep mqRestGateway
name="mqRestGatewayEnabled" value="enabled"
name="mqRestGatewayQmgr" value="QMORI915"
```

++
+++ Chapter 2: Full connectivity between the Gateway and the other queue managers
++

The steps in this chapter are based on the following page from the online manual:

https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.adm.doc/q021120_.htm
IBM MQ 9.1.x / IBM MQ / Administering / Administering remote IBM MQ objects /
Configuring queue managers for remote administration

Before you can remotely administer a queue manager from a local queue manager, you must create a sender and receiver channel, a listener, and a transmission queue on each queue manager. These channels and queues enable the commands to be sent to the remote queue manager and the responses to be received on the local queue manager.

You can use the following article which includes a Windows Batch file and a Unix shell script that can be used to generate the desired commands.

http://www-01.ibm.com/support/docview.wss?uid=swg21470997
Commands to setup both ways communication between 2 queue managers via Sender and Receiver channels

+ Review of the configuration:

Host-1: orizaba1
InstName: Installation1
Queue manager: QMORI915 Port: 1415 = = > This is going to be the GATEWAY

InstName: Installation2
Queue manager: QMORI8 Port: 1416

+ Host-2 : fortin1.fyre.ibm.com

InstName: Installation2
Queue manager: QMFOR910 Port: 1420

The queue managers need to be running, because runmqsc will be used to create the proper objects:

```
mqm@orizaba1.fyre.ibm.com: /home/mqm
$ dspmq -o installation -s
QMNAME(QMORI915)                                   STATUS(Running) INSTNAME(Installation1)
INSTPATH(/opt/mqm) INSTVER(9.1.5.0)
QMNAME(QMORI8)                                     STATUS(Running) INSTNAME(Installation2)
```

```
INSTPATH(/opt/mqm80) INSTVER(8.0.0.9)

mqm@orizaba1.fyre.ibm.com: /home/mqm
 $ ps -ef | grep runmqls
mqm      22874 22809  0 07:15 ?        00:00:00 /opt/mqm/bin/runmqlsr -r -m QMORI915 -t
TCP -p 1415
mqm      25528 25477  0 May08 ?        00:00:06 /opt/mqm80/bin/runmqlsr -r -m QMORI8 -t
TCP -p 1416
```

+ From Host-1, orizaba1.fyre.ibm.com, generate the connectivity files and feed them to the appropriate runmqsc

```
$ cd /home/mqm
$ . setmqenv -n Installation1

$ gen-mqsc-2-qmgrs   QMORI915 orizaba1.fyre.ibm.com 1415  QMORI8   orizaba1.fyre.ibm.com 1416
Create File: QMORI915.QMORI8.mqsc
Create File: QMORI8.QMORI915.mqsc

$ runmqsc QMORI915 < QMORI915.QMORI8.mqsc

$ gen-mqsc-2-qmgrs   QMORI915 orizaba1.fyre.ibm.com 1415  QMFOR910 fortin1.fyre.ibm.com  1420
Create File: QMORI915.QMFOR910.mqsc
Create File: QMFOR910.QMORI915.mqsc

$ runmqsc QMORI915 < QMORI915.QMFOR910.mqsc

$ . setmqenv -n Installation2
$ gen-mqsc-2-qmgrs   QMORI8   orizaba1.fyre.ibm.com 1416  QMORI915 orizaba1.fyre.ibm.com 1415
Create File: QMORI8.QMORI915.mqsc
Create File: QMORI915.QMORI8.mqsc

$ runmqsc QMORI8   < QMORI8.QMORI915.mqsc
```

+ Now from Host-2, fortin1.fyre.ibm.com:

```
$ cd /home/mqm

$ . setmqenv -n Installation2

$ gen-mqsc-2-qmgrs   QMFOR910 fortin1.fyre.ibm.com  1420  QMORI915 orizaba1.fyre.ibm.com 1415
Create File: QMFOR910.QMORI915.mqsc
Create File: QMORI915.QMFOR910.mqsc

$ runmqsc QMFOR910 < QMFOR910.QMORI915.mqsc
```

+ Now doe a remote test via runmqsc from orizaba1:
  runmqsc -w 30 -m GatewayQmgr   RemoteQmgr

```
mqm@orizaba1.fyre.ibm.com: /home/mqm
$ . setmqenv -n Installation1

$ runmqsc -w 30 -m QMORI915  QMORI8
Starting MQSC for queue manager QMORI8.
display queue(Q5) curdepth
     1 : display queue(Q5) curdepth
AMQ8409: Display Queue details.
   QUEUE(Q5)                               TYPE(QLOCAL)
   CURDEPTH(0)
end

$ runmqsc -w 30 -m QMORI915  QMFOR910
Starting MQSC for queue manager QMFOR910.
display queue(Q5) curdepth
     1 : display queue(Q5) curdepth
AMQ8409I: Display Queue details.
   QUEUE(Q5)                               TYPE(QLOCAL)
   CURDEPTH(0)
end

mqm@orizaba1.fyre.ibm.com: /home/mqm
$ . setmqenv -n Installation2
$ runmqsc -w 30 -m QMORI8  QMORI915
Starting MQSC for queue manager QMORI915
display queue(Q5) curdepth
AMQ8409I: Display Queue details.
   QUEUE(Q5)                               TYPE(QLOCAL)
   CURDEPTH(0)
end
```

+ Now do the corresponding test from fortin1:
  runmqsc -w 30 -m GatewayQmgr   RemoteQmgr

```
mqm@fortin1.fyre.ibm.com: /home/mqm
$ . setmqenv -n Installation2
$ runmqsc -w 30 -m QMFOR910 QMORI915
Starting MQSC for queue manager QMORI915.
display queue(Q5) curdepth
     1 : display queue(Q5) curdepth
AMQ8409I: Display Queue details.
```

```
   QUEUE(Q5)                               TYPE(QLOCAL)
   CURDEPTH(0)
end
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++ Chapter 3: Issuing REST API commands via the Gateway queue manager
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Remember that the gateway is QMORI915 in orizaba1:

+ Asking information to queue manager QMORI8 (using MQ 8.0) which is a local queue manager but in another installation to the Gateway queue manager

```
$ curl -s -k "https://orizaba1.fyre.ibm.com:9443/ibmmq/rest/v1/admin/qmgr/QMORI8" -
X GET
{"qmgr": [
  {"name": "QMORI8"}
]}

$ curl -s -k
"https://orizaba1.fyre.ibm.com:9443/ibmmq/rest/v1/admin/qmgr/QMORI8/queue?attrib
utes=general.isTransmissionQueue" -X GET
{"queue": [
  {
    "general": {"isTransmissionQueue": false},
    "name": "SYSTEM.ADMIN.STATISTICS.QUEUE",
    "type": "local"
  },
...
  {
    "general": {"isTransmissionQueue": true},
    "name": "SYSTEM.CLUSTER.TRANSMIT.QUEUE",
    "type": "local"
  },
...
  {
    "general": {"isTransmissionQueue": false},
    "name": "Q5",
    "type": "local"
  },
...
  {
    "name": "Q6_QMORI915",
    "remote": {
      "qmgrName": "QMORI915",
      "queueName": "Q6"
    },
    "type": "remote"
  },
```

+ Asking information to queue manager QMFOR910 (using MQ 9.1.0) which is a remote queue manager in host fortin1.

```
$ curl -s -k
"https://orizaba1.fyre.ibm.com:9443/ibmmq/rest/v1/admin/qmgr/QMFOR910" -X GET
{"qmgr": [{"name": "QMFOR910"}]}
```

```
$ curl -s -k
"https://orizaba1.fyre.ibm.com:9443/ibmmq/rest/v1/admin/qmgr/QMFOR910/queue/Q5
?status=status.currentDepth" -X GET
{"queue": [{
  "name": "Q5",
  "status": {"currentDepth": 0},
  "type": "local"
}]}
```

+++ end