

MQ Administration, the Web Console, & REST API

Sam Goulden
IBM MQ L3 Service
sgoulde4@uk.ibm.com

Agenda

■ Administration Overview

- ▶ Overview
- ▶ PCF
- ▶ MQ Explorer
- ▶ runmqsc

■ mqweb server

- ▶ What is it and how it's installed
- ▶ Timeline
- ▶ Configuration and Management

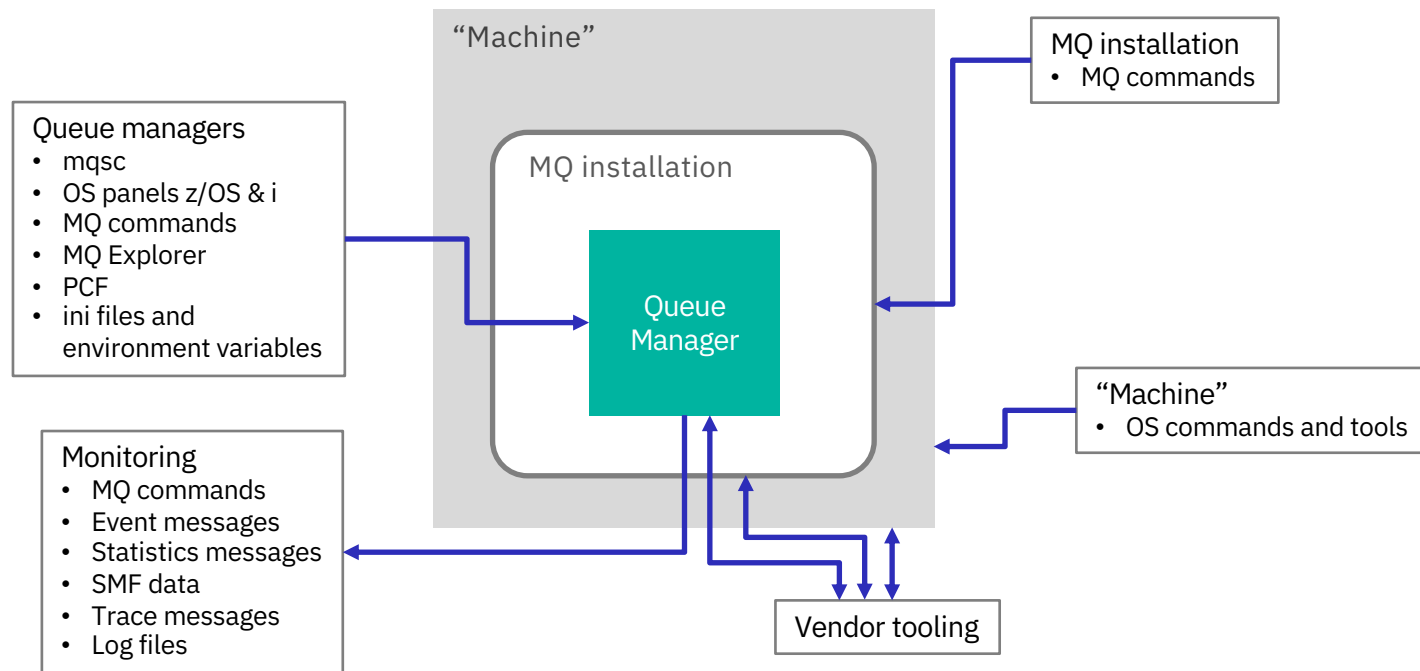
■ The MQ REST API

- ▶ Introduction and Examples

■ The MQ Console

- ▶ Introduction and Examples

Administering software MQ



From a tooling perspective PCF
is key

PCF – Programmable Command Format

■ Allows administration of...

- ▶ Resource management.
 - For example, queue creation and deletion.
- ▶ Performance monitoring.
 - For example, maximum queue depth or message rate.
- ▶ Control.
 - For example, tuning queue parameters such as maximum queue depth, maximum message length, and enabling and disabling queues.
- ▶ Message routing.
 - Definition of alternative routes through a network.

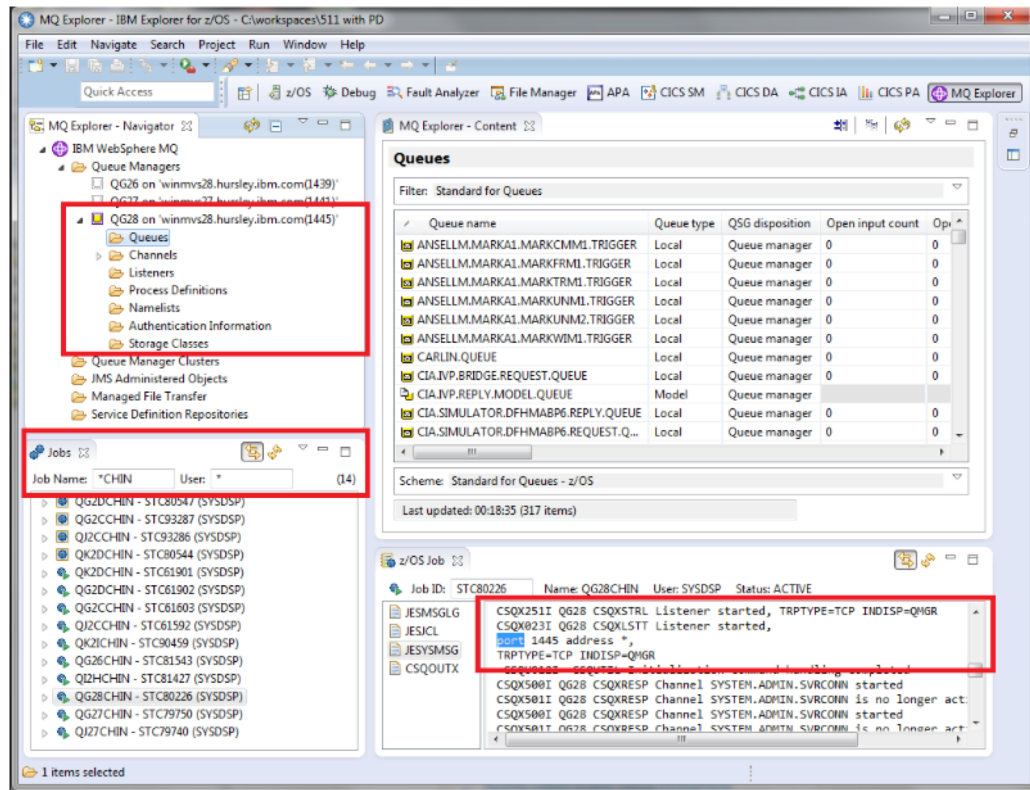
■ Works via a special message format

- ▶ The Queue Manager has a command server to service the command messages from the administration queue

■ Programmable Format

- ▶ Its possible to build applications to craft these, runmqsc uses them!

MQ Explorer



Graphical user interface in which you can administer and monitor IBM® MQ objects

Local and Remote QMs

Full Queue Manager administration including Clusters, MFT, MQTT support and more

Extensible via plug-ins

runmqsc

- Command line interface into QM administration
- Auto-complete feature as of version 8.0!
- PCF Commands under the covers

```
[goulden@goulden errors]$ runmqsc QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2016.
Starting MQSC for queue manager QM1.

define QLOCAL(Q1)
  1 : define QLOCAL(Q1)
AMQ8006: IBM MQ queue created.
DEFINE CHANNEL(NEW.CHANNEL) CHLTYPE(CLUSSDR) CONNAME(127.0.0.1) TRPTYPE(TCP)
  2 : DEFINE CHANNEL(NEW.CHANNEL) CHLTYPE(CLUSSDR) CONNAME(127.0.0.1) TRPTYPE(TCP)
AMQ8014: IBM MQ channel created.
DISPLAY CHANNEL(NEW.CHANNEL) CONNAME
  4 : DISPLAY CHANNEL(NEW.CHANNEL) CONNAME
AMQ8414: Display Channel details.
CHANNEL(NEW.CHANNEL)                                CHLTYPE(CLUSSDR)
CONNAME(127.0.0.1)
```

But for remote management, we need more

- **While PCF is very powerful, it is not that easy to use**
 - ▶ Requires an MQ client, and a supported programming language
 - ▶ Binary format
 - ▶ Multiple messages generated per request
 - ▶ There are tools to make this easier
- **There is a growing need for the ability to administer MQ from**
 - ▶ Any environment
 - ▶ Any programming language
 - ▶ By users who are not expert in MQ
- **Lots of customers are writing self-service web-portals for managing their infrastructure, including MQ**

**** Message ****

length - 724 of 724 bytes

```
00000000: 080A 4103 0000 0000 5744 5220 0200 0000 'A....WDR ....'
00000010: 8800 0000 6700 0000 514D 4752 315F 3230 '...g...QMGR1_20'
00000020: 3135 2D31 302D 3239 5F30 392E 3431 2E31 '15-10-29_09.41.1'
00000030: 3620 2020 2020 2020 2020 2020 2020 2020 '6'
00000040: 2020 2020 2020 2020 514D 4752 3120 2020 'QMGR1'
00000050: 2020 2020 2020 2020 2020 2020 2020 2020 ' '
00000060: 2020 2020 2020 2020 2020 2020 2020 2020 ' '
00000070: 2020 2020 2020 2020 0000 0000 0000 0000 '.....'
00000080: 58CA 0000 0000 0000 0000 0000 0000 0000 'X.....'
00000090: 644E 4656 2116 4656 3230 3135 2D31 302D 'dNFV!FV2015-10-'
000000A0: 3239 2020 0000 0000 3039 2E34 312E 3233 '29 ....09.41.23'
000000B0: 0100 0000 4D51 4D4D 0000 0000 3038 3030 '....MQMM....0800'
000000C0: 3030 3034 0000 0000 434C 5553 5445 5231 '0004....CLUSTER1'
000000D0: 2E51 4D47 5231 2020 2020 2020 0800 0000 'QMGR1 ....'
000000E0: 0800 0000 0200 0000 2020 2020 2020 2020 '.....'
000000F0: 2020 2020 2020 2020 2020 2020 2020 2020 '.....'
```



The mqweb server

IBM MQ Console Dashboard

Tab 1 + ↗

Local Queue Managers

+ - ▢ ▶ ■ More... Search...

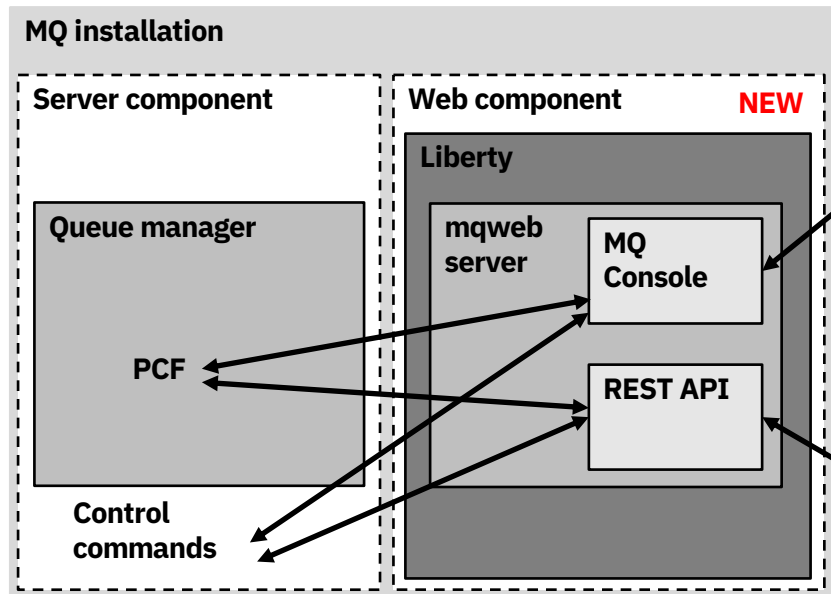
▲Name	Status
a	↑ Running
dave	↑ Running

Total: 2 Selected: 0 Updated: 8:02:28 AM



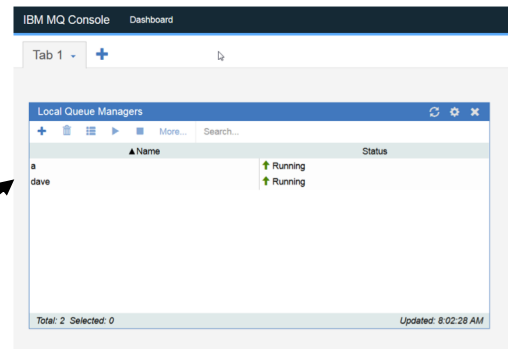
```
C:\Program Files\IBM\MQ_10>curl -k https://localhost:9443/ibmmq/rest/v1/qmgr/with%2Fslash
{"qmgr": [{
  "name": "with/slash",
  "status": "running"
}]}
```

The mqweb server



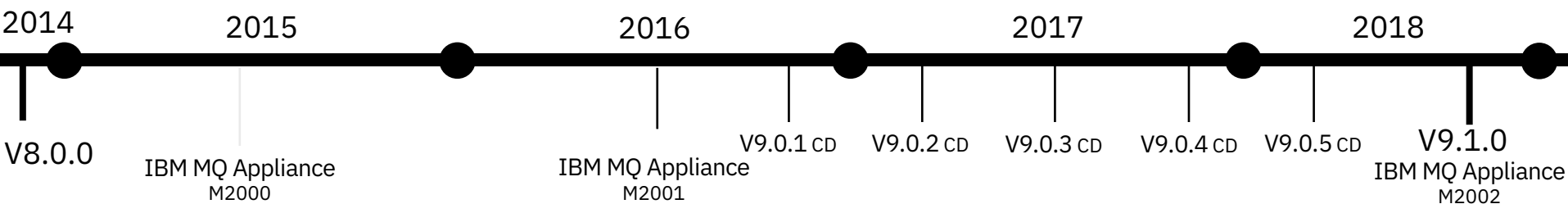
HTTP

HTTP



```
C:\Program Files\IBM\MQ_10>curl -k https://localhost:9443/ibmmq/rest/v1/qmgr/with%2Fslash
{"qmgr": [{"name": "with/slash", "status": "running"}]}
```

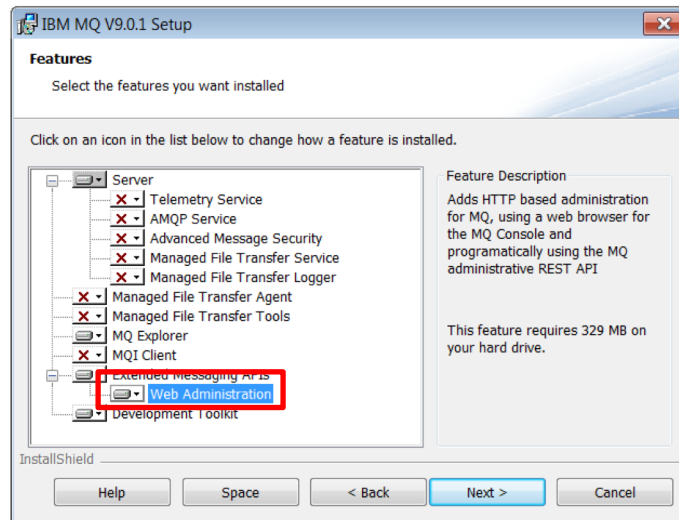
Overview



- **MQ 9.0.1 CD added support for a number of HTTP-based administration capabilities**
 - ▶ Focus on low barrier to entry and ease of use
 - ▶ MQ Console – a web-browser based graphical administration tool
 - ▶ MQ REST API – a programmatic administration API
 - Enhanced further during CD deliverables 9.0.2, 9.0.3, 9.0.4, and 9.0.5
- **As these are CD features these capabilities are supported on a subset of platforms**
 - ▶ Windows, Linux, AIX, z/OS and Appliance
- **LTS and CD currently in parity for the first time since we released it in 2016!**
 - ▶ Features are wrapped up into the current LTS release V9.1.0

Web component

- A new optional install component
- Contains the MQ Console, MQ administrative REST API plus prereqs
 - WebSphere Liberty Profile which runs the mqweb server
- New USS FMID on z/OS
 - JMS9016



The mqweb server

- The MQ Console and REST API are applications that run in a WebSphere Liberty Profile (WLP) server called mqweb

- WLP is provided as part of MQ install
- mqweb server definition provided out of the box when installing the web component

- **Once installed:**

- MQ Console is enabled
- REST API is enabled

CWWKE0001I: The server mqweb has been launched.

CWWKG0028A: Processing included configuration resource: C:\Program Files\IBM\Latest902\web\mq\etc\mqweb.xml

A CWWKG0028A: Processing included configuration resource: C:\Program Files (x86)\IBM\WebSphere MQ\web\installations\Latest902\servers\mqweb\mqwebuser.xml

CWWKE0002I: The kernel started after 2.493 seconds

CWWKF0007I: Feature update started.

CWWKO0219I: TCP Channel defaultHttpEndpoint-ssl has been started and is now listening for requests on host 127.0.0.1 (IPv4: 127.0.0.1) port 9443.

CWWKZ0018I: Starting application com.ibm.mq.rest.

CWWKZ0018I: Starting application com.ibm.mq.console.

SRVE0169I: Loading Web Module: com.ibm.mq.rest.v1.

SRVE0250I: Web Module com.ibm.mq.rest.v1 has been bound to default_host.

CWWKT0016I: Web application available (default_host): https://localhost:9443/ibmmq/rest/v1/

CWWKZ0001I: Application com.ibm.mq.rest started in 0.518 seconds.

SRVE0169I: Loading Web Module: mqconsole.

SRVE0250I: Web Module mqconsole has been bound to default_host.

CWWKT0016I: Web application available (default_host): https://localhost:9443/ibmmq/console/

SRVE0169I: Loading Web Module: com.ibm.mq.consoleinternal.

SRVE0250I: Web Module com.ibm.mq.consoleinternal has been bound to default_host.

CWWKT0016I: Web application available (default_host): https://localhost:9443/ibmmq/console/internal/

CWWKZ0001I: Application com.ibm.mq.console started in 0.525 seconds.

CWWKF0012I: The server installed the following features: [concurrent-1.0, jsp-2.2, servlet-3.1, ssl-1.0, jndi-1.0, basicAuthenticationMQ-1.0, websocket-1.0, json-1.0, localConnector-1.0, jaxrs-1.1].

CWWKF0008I: Feature update completed in 2.095 seconds.

CWWKF0011I: The server mqweb is ready to run a smarter planet.

REST023: MQ REST API level: p902-dfct-L170216.1

Configuring mqweb server

- Currently done by editing xml (standard WLP approach)
- File called **mqwebuser.xml** provided in MQ data directory
 - This is the only part of the WLP xml configuration that we support customers editing:
- From Version 9.0.4, use the **setmqweb** command e.g:
 - `setmqweb properties -k httpHost -v hostname`

```
<!--
Sample mqwebuser.xml file, included by server.xml, to contain user
configuration for the mqweb server.
-->
<server>
  <featureManager>
    <feature>appSecurity-2.0</feature>
  </featureManager>
</server>

<!--
Default MQ security configuration allows HTTPS TLS v1.2 ONLY and no user access,
refer to the IBM Knowledge Center section on "IBM MQ Console and REST API security"
for details of how to configure security.
-->
<sslDefault sslRef="mqDefaultSSLConfig"/>
<basicRegistry id="basic" realm="defaultRealm">
</basicRegistry>
</server>
```

Security enabled by default

No users defined

Configurable:

- Security
- Port number
- Logging
- Authentication token expiry
- and more

Managing mqweb server

- **Distributed: three new control commands**

- ▶ strmqweb, endmqweb, dspmqweb

```
C:\Program Files\IBM\Latest902\bin>strmqweb.bat
Starting server mqweb.
Server mqweb started.
```

```
C:\Program Files\IBM\Latest902\bin>dspmqweb.bat
Server mqweb is running.
```

URLs:

```
https://localhost:9443/ibmmq/console/
https://localhost:9443/ibmmq/rest/v1/
```

- **z/OS: Sample JCL – CSQ4WEBS – provided**

- ▶ Sets all necessary variables up and then starts up mqweb server

```
EDIT      USER.PROCLIB(MQWEBMWL) - 01.03
Command ==>
000058 //*****
000059 //*
000060 //      PROC
000061 //*
000062 //  SET  INSTDIR='/u/mleming/v902betamqm/web'
000063 //  SET  USERDIR='/u/mleming/mq902web'
000064 //*
000065 //STEP1  EXEC  PGM=BPXBATSL,REGION=0M,TIME=NOLIMIT,
000066 //  PARM='PGM &INSTDIR./lib/native/zos/s390x/bbgzsrv mqweb'
000067 //WLPUDIR DD  PATH='&USERDIR.'
000068 //STEPLIB DD  DSN=ANTZ.MQ.V900.DFCT.OUT.SCSQANLE,DISP=SHR
000069 //        DD  DSN=ANTZ.MQ.V900.DFCT.OUT.SCSQAUTH,DISP=SHR
000070 //STDOUT  DD  SYSOUT=*
000071 //STDERR  DD  SYSOUT=*
000072 //STDIN   DD  DUMMY
000073 //STDENV  DD  *
000074 JAVA_HOME=/java/java80_bit64_sr3_fp20/J8.0_64
000075 PATH=/u/mleming/v902betamqm/web/bin:/usr/sbin
000076 LIBPATH=/u/mleming/v902betamqm/java/lib
000077 //*
```

MQ REST API

What is REST?

- **REpresentational State Transfer**
- **HTTP is an example of a RESTful architecture**
- **HTTP defines resources (URL/URIs) and the operations (HTTP verbs) which can use them**
 - ▶ Originally used for serving web-pages
 - ▶ Work really well for APIs too
- **Generally light-weight and relatively simple to use, much simpler than SOAP web-services**
 - ▶ Have become incredibly common in recent years
- **However there are lots of interpretations of what it means to be RESTful**
 - ▶ MQ has taken the approach of following best-practice, and adherence to the various w3c standards when defining its REST API

MQ REST API

- An administrative API for managing MQ via REST
- Is much more intuitive to use than PCF and makes it easier to create MQ tooling, e.g. a self-service web-browser based MQ portal using JavaScript
 - ▶ No need for an MQ client!
 - ▶ Callable from any language which can invoke an HTTPS endpoint
 - ▶ Many languages now have built in, or easily added, support for REST
- **Payload format is JSON (JavaScript Object Notation)**
 - Human readable, not a binary format

Curly bracket denotes JSON object

Square bracket denotes JSON array

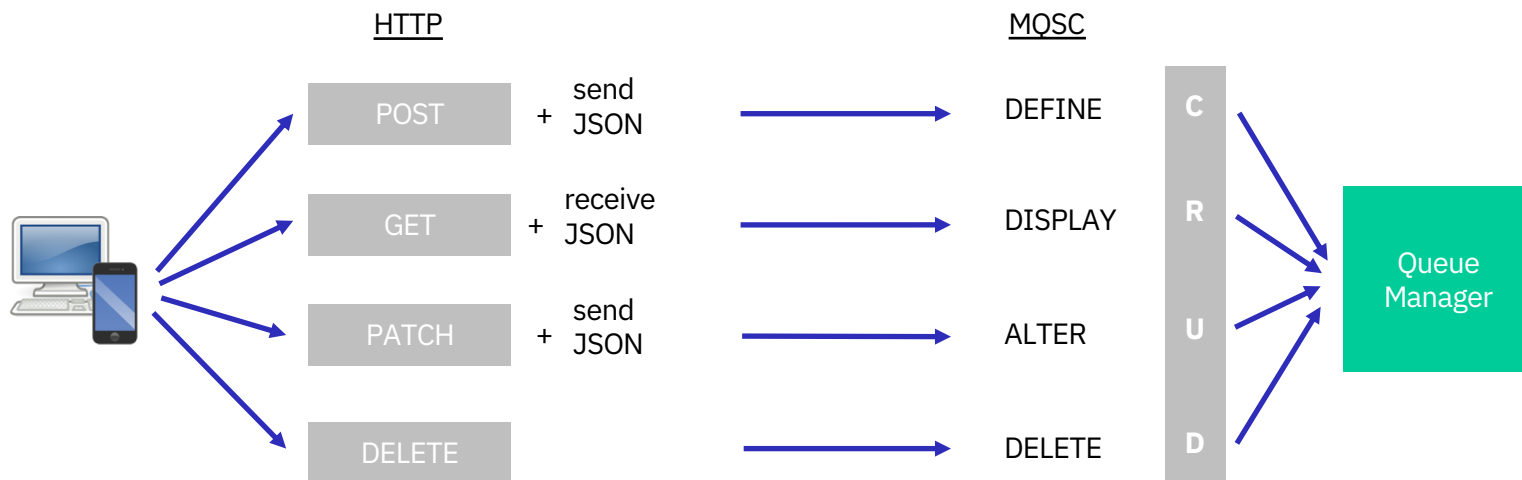
A nested unnamed object, in an array

Name, value pair. Where value is of type string

```
{  
  "qmgr": [  
    {  
      "name": "AQM1",  
      "state": "running"  
    }  
  ]  
}
```

MQ REST API

- Based off underlying MQ capabilities such as PCF and control commands, but adjusted to adhere to RESTful practices
- URL represent target object for command



Evolution of the MQ REST APIs

- **Iteratively developed in CD releases**

- ▶ 9.0.1:
 - REST API for administration introduced
 - Contains ability to list queue managers (dspmq) and their installation (dspmqver)
 - Not integrated into mqweb server/MQ security so disabled by default
- ▶ 9.0.2:
 - Integrated into mqweb server and MQ security, enabled by default
 - Contains CRUD for queues and the ability to display queue status
 - Supported on MQ Appliance
- ▶ 9.0.3:
 - Support for subset of DIS QMSTATUS on all platforms including z/OS
- ▶ 9.0.4
 - REST API for messaging introduced
 - Administration API further enhanced
 - Ability to run MQSC commands
 - Ability to display channels and subscriptions
 - REST API administration gateway introduced
- ▶ 9.0.5
 - REST API for MFT administration introduced



**This function all included in the 9.1
LTS release of MQ**

REST API Examples

GET /ibmmq/rest/v1/qmgr (dspmq)

- Ability to list queue managers associated with installation
- Example below uses curl to list all queue managers
 - -k flag tells it to ignore the fact that a self-signed certificate is being used on the mqweb server, you don't want to be doing this in production!

```
C:\temp>curl -k https://localhost:9443/ibmmq/rest/v1/qmgr
{"qmgr": [
  {
    "name": "AQM1",
    "state": "running"
  },
  {
    "name": "AQM2",
    "state": "endedImmediately"
  },
  {
    "name": "AQM3",
    "state": "endedImmediately"
  },
  {
    "name": "bob2",
    "state": "running"
  }
]}
```

GET /ibmmq/rest/v1/**qmgr** (dspmq)

- Can get information on just a specific queue manager
 - ▶ GET /ibmmq/rest/v1/**qmgr**/**{qmgrName}**
- Can request additional attributes too, or just a sub-set
 - ▶ GET /ibmmq/rest/v1/**qmgr**?attributes=*

```
C:\temp>curl -k https://localhost:9443/ibmmq/rest/v1/qmgr?attributes=*
{"qmgr": [
  {
    "extended": {
      "installationName": "Latest902",
      "isDefaultQmgr": false,
      "permitStandby": "notPermitted"
    },
    "name": "AQM1",
    "state": "running"
  },
  {
    "extended": {
      "installationName": "Latest902",
      "isDefaultQmgr": false,
      "permitStandby": "notApplicable"
    },
    "name": "AQM2",
    "state": "endedImmediately"
  }
],
```

GET /ibmmq/rest/v1/installation (dspmqver)

- Basic display

```
C:\temp>curl -k https://localhost:9443/ibmmq/rest/v1/installation
{"installation": [{
  "name": "Latest902",
  "platform": "windows",
  "version": "9.0.2.0"
}]}
```

- All attributes

```
C:\temp>curl -k https://localhost:9443/ibmmq/rest/v1/installation?attributes=*
{"installation": [{
  "extended": {
    "dataPath": "C:\\Program Files (x86)\\IBM\\WebSphere MQ",
    "description": "",
    "hostName": "9.20.230.214",
    "installationPath": "C:\\Program Files\\IBM\\Latest902",
    "level": "p902-dfct-L170216.1",
    "maximumCommandLevel": 902,
    "operatingSystem": "Windows 7 Professional x64 Edition, Build 7601: SP1",
    "primary": false
  },
  "name": "Latest902",
  "platform": "windows",
  "version": "9.0.2.0"
}]}
```

Queues...

■ DEFINE Q*

- ▶ POST to /ibmmq/rest/v1/qmgr/{qmgrName}/queue

`curl -k -X POST -H "Content-Type: application/json" -d '{"name": "Q1"}' https://localhost:9443/ibmmq/rest/v1/qmgr/bob2/queue`

Sending JSON payload

Queue manager name

Queue definition, very simple in this case

■ DISPLAY Q*

- ▶ GET to /ibmmq/rest/v1/qmgr/{qmgrName}/queue/{queueName}

```
C:\>curl -k "https://localhost:9443/ibmmq/rest/v1/qmgr/bob2/queue?name=Q1"
{"queue": [
  {
    "name": "Q.LOCAL1",
    "type": "local"
  },
  {
    "name": "Q.LOCAL",
    "type": "local"
  },
  {
    "name": "Q1",
    "type": "local"
  }
]}
```

Queues...

■ ALTER Q*

- ▶ PATCH to /ibmmq/rest/v1/qmgr/{qmgrName}/queue/{queueName}
- ▶ E.g: the following will PUT inhibit Q.LOCAL1

```
curl -k -X PATCH -H "Content-Type: application/json" -d "{\"general\":{\"inhibitPut\": true}}"
https://localhost:9443/ibmmq/rest/v1/qmgr/bob2/queue/Q.LOCAL1
```

■ DELETE Q*

- ▶ DELETE to /ibmmq/rest/v1/qmgr/{qmgrName}/queue/{queueName}

```
C:\>curl -k -X DELETE https://localhost:9443/ibmmq/rest/v1/qmgr/bob2/queue/Q.LOCAL1

C:\>curl -k https://localhost:9443/ibmmq/rest/v1/qmgr/bob2/queue/Q.LOCAL1
{"error": [{"action": "Resubmit the request with the name of an existing queue, or with no queue name to retrieve a list of queues.",
"explanation": "The MQ REST API was invoked specifying a queue name which cannot be located.",
"message": "REST037: Could not find the queue 'Q.LOCAL1' - the queue manager reason code is 2085 : 'MQRC_UNKNOWN_OBJECT_NAME'.",
"msgId": "REST037",
"type": "rest"}]}
```

Queues...

■ Also possible to issue DISPLAY QSTATUS

- ▶ GET to /ibmmq/rest/v1/qmgr/{qmgrName}/queue/{queueName}?status=*
- ▶ So you can get both the queue definition and its status at the same time!

```
C:\Program Files\IBM\Latest902\bin>curl -k "https://localhost:9443/ibmmq/rest/v1/qmgr/bob2/queue/Q.LOCAL?status=*"
{"queue": [{"name": "Q.LOCAL",
  "status": {
    "currentDepth": 0,
    "lastGet": "",
    "lastPut": "",
    "mediaRecoveryLogExtent": "",
    "monitoringRate": "off",
    "oldestMessageAge": -1,
    "onQueueTime": {
      "longSamplePeriod": -1,
      "shortSamplePeriod": -1
    },
    "openInputCount": 0,
    "openOutputCount": 0,
    "uncommittedMessages": 0
  },
  "type": "local"
}]}
```

MQSC for REST

Tailored RESTful support for individual MQ objects and actions are in the works...

However, to speed up full MQ admin support over REST we added the ability to submit arbitrary MQSC commands over REST

- ✓ Gives complete MQSC coverage quickly
- ✓ Simple to convert existing scripts
- ✗ Does not benefit from improved usability

HTTPS POST:

<https://host:port/ibmmq/v1/action/qmgr/QMGR1/mqsc>

```
{
  "type": "runCommand",
  "parameters": {
    "command": "STOP CHANNEL(CHANNEL.TEST)"
  }
}
```

```
{
  "commandResponse": [{
    "completionCode": 0,
    "reasonCode": 0,
    "text": ["AMQ8019: Stop IBM MQ channel accepted."]
  }],
  "overallCompletionCode": 0,
  "overallReasonCode": 0
}
```

Stopping a channel

REST API Discovery

API discovery

- Want to find out what is available in the MQ REST API, and don't want to read the KC?
- Then try out API discovery!
- Function in WLP that describes the MQ REST API using Swagger
- Makes it easier to see what is there, and try it out

Liberty REST APIs

Discover REST APIs available within Liberty

/ibm/api/explorer

API Discovery : APIs available from the API Discovery feature

Show/Hide | List Operations | Expand Operations

installation

Show/Hide | List Operations | Expand Operations

login

Show/Hide | List Operations | Expand Operations

qmgr

Show/Hide | List Operations | Expand Operations

queue

Show/Hide | List Operations | Expand Operations

DELETE	/ibmmq/rest/v1/qmgr/{qmgrName}/queue	Documented for completeness only - this operation will be rejected
GET	/ibmmq/rest/v1/qmgr/{qmgrName}/queue	Retrieves details of all queues
OPTIONS	/ibmmq/rest/v1/qmgr/{qmgrName}/queue	Defines available methods for queues at the queue manager level
PATCH	/ibmmq/rest/v1/qmgr/{qmgrName}/queue	Documented for completeness only - this operation will be rejected
POST	/ibmmq/rest/v1/qmgr/{qmgrName}/queue	Creates a queue

API discovery

GET

/ibmmq/rest/v1/qmgr/{qmgrName}/queue

Retrieves details of all queues

Implementation Notes

Retrieves details of all queues defined in the named Queue Manager, optionally specifying which attributes of the queues are to be retrieved

Response Class (Status 200)

A JSONArray containing a JSONObject describing the Queue

Model Example Value

```
{
  "namelist": "string",
  "qmgrId": "string",
  "name": "string",
  "qmgrName": "string",
  "queueType": "alias",
  "workloadQueueUse": "any",
  "workloadPriority": 0,
  "transmissionQueueForChannelName": "string",
  "workloadRank": 0
},
"timestamps": {
```

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
qmgrName	<input type="text" value="(required)"/>	Name of the Queue Manager containing the queue of interest	path	string

REST API Security

REST API security

Authorization

- **Role based access control. Need to be a member of at least one role**

- ▶ MQWebAdmin
- ▶ MQWebAdminRO
- ▶ MQWebUser

- **User and groups defined in a registry**

- ▶ Basic
- ▶ LDAP
- ▶ SAF (on z/OS)

- **REST is locked down by default, need to do some configuring**

- ▶ Samples provided to make this simpler

```
<!-- Roles for the MQ REST API -->
<enterpriseApplication id="com.ibm.mq.rest">
  <application-bnd>
    <security-role name="MQWebAdmin">
      <group name="MQWebUI" realm="defaultRealm"/>
    </security-role>
    <security-role name="MQWebAdminRO">
      <user name="mqreader" realm="defaultRealm"/>
    </security-role>
    <security-role name="MQWebUser">
      <special-subject type="ALL_AUTHENTICATED_USERS"/>
    </security-role>
  </application-bnd>
</enterpriseApplication>

<!-- Sample Basic Registry -->
<basicRegistry id="basic" realm="defaultRealm">
  <!-- This sample defines two users with unencoded passwords -->
  <!-- and a group, these are used by the role mappings above -->
  <user name="mqadmin" password="mqadmin"/>
  <user name="mqreader" password="mqreader"/>
  <group name="MQWebUI">
    <member name="mqadmin"/>
  </group>
</basicRegistry>

<!-- Example LDAP Registry -->
<ldapRegistry id="ldap"
  realm="MyOrganizationRealm"
  host="sso.example.com"
  port="389"
  ignoreCase="true"
  baseDN="o=example.com"
  certificateMapMode="EXACT_DN"
  ldapType="IBM Tivoli Directory Server"
  idsFilters="ibm_dir_server">
</ldapRegistry>
```

REST API authentication

■ Token based

- ▶ User logs in once with user id and password and then gets a cookie which is used for subsequent requests

```
curl -k -X POST -H "Content-Type: application/json"
```

```
-d '{"username":"mqadmin","password":"mqadmin"}'
```

```
https://localhost:9443/ibmmq/rest/v1/login -c c:\temp\cookiejar.txt
```

User id and password
provided as JSON payload

Cookie stored for use on next
request

■ Or HTTP basic authentication

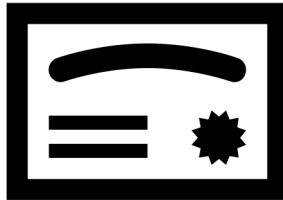
- ▶ User id and password provided as an encoded header, must be set for each request

```
C:\>curl -k -u mqadmin:mqadmin https://localhost:9443/ibmmq/rest/v1/qmgr/bob2/queue/LOCALQ1
{"queue": [{
  "name": "LOCALQ1",
  "type": "local"
}]}
```

REST API authentication

■ Or use a client certificate

- ▶ Must be provided with each call to the REST API
- ▶ Distinguished name from certificate is mapped to user in configured user registry



REST API Gateway

Enabling your whole estate for REST administration

Option 1

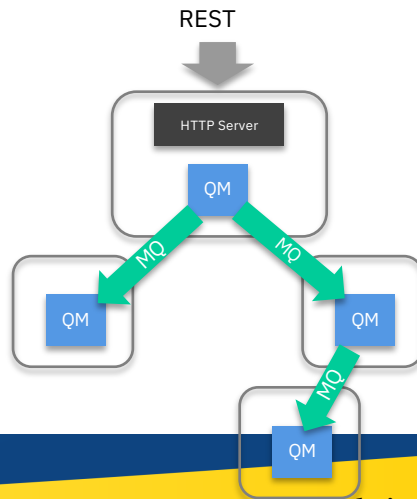
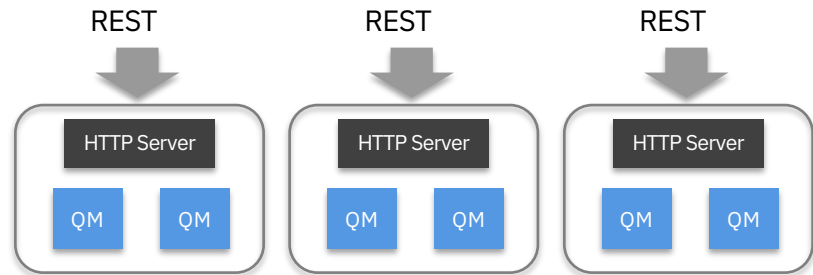
Administer each MQ installation separately, they must all be on the MQ 9.0.x CD release

Option 2

Manage a network of systems through gateway entry points

Not every queue manager will need to expose HTTPS endpoints

Pre-9.0.x queue managers are able to be administered through those 9.0.x gateways



MQ REST Gateway Queue Manager

- Default gateway queue manager defined

```
GET https://localhost:9443/ibmmq/rest/v1/admin/qmgr/MQZ1?status=*
```

```
ibm-mq-rest-gateway-qmgr: RESTQM0
```

```
{ "qmgr": [{  
  "name": "MQZ1",  
  "status": {  
    "channelInitiatorState": "running",  
    "connectionCount": 73,  
    "publishSubscribeState": "running",  
    "started": "2017-10-30T11:10:45.000Z"  
  }  
}] }
```

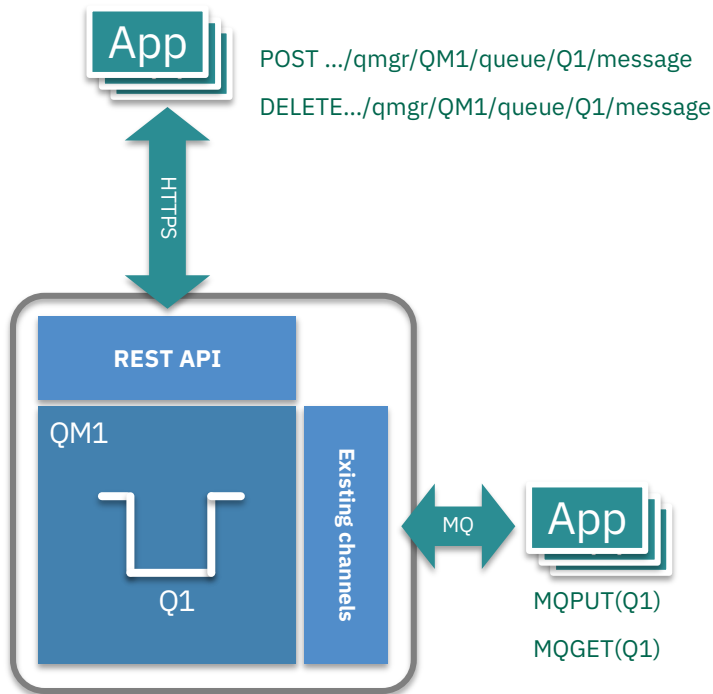
- If no default defined, set in HTTP header

- ▶ ibm-mq-rest-gateway-qmgr

REST API Messaging

REST Messaging

- The new HTTP server support in MQ 9.0.x provides the platform for a properly integrated REST API solution
- Allowing applications to put and get messages from a queue without installing any MQ software locally
- Ideal for environments with native REST support, such as common JavaScript libraries including NodeJS, and AngularJS
- Can only be used for point to point messaging
- For full functionality and resiliency an MQ client should still be used



The MQ Console

MQ Console

- **Browser based interface for administering and managing MQ**
 - No client side install needed
 - Originally available in MQ Appliance only
- **As of 9.0.1 a common capability across appliance and software MQ**
 - Re-engineered on AngularJS so different implementation than on 8.0.0.* appliance
 - Functional parity with MQ Console in 8.0.0.* appliance
- **Some capabilities not available on z/OS**
 - Can't create/delete/start/stop queue managers, etc
- **Can only interact with queue managers running in the same installation**
 - On z/OS all queue managers at the same CD level

MQ Console – log in

- **Point your web-browser at the MQ Console and log in**

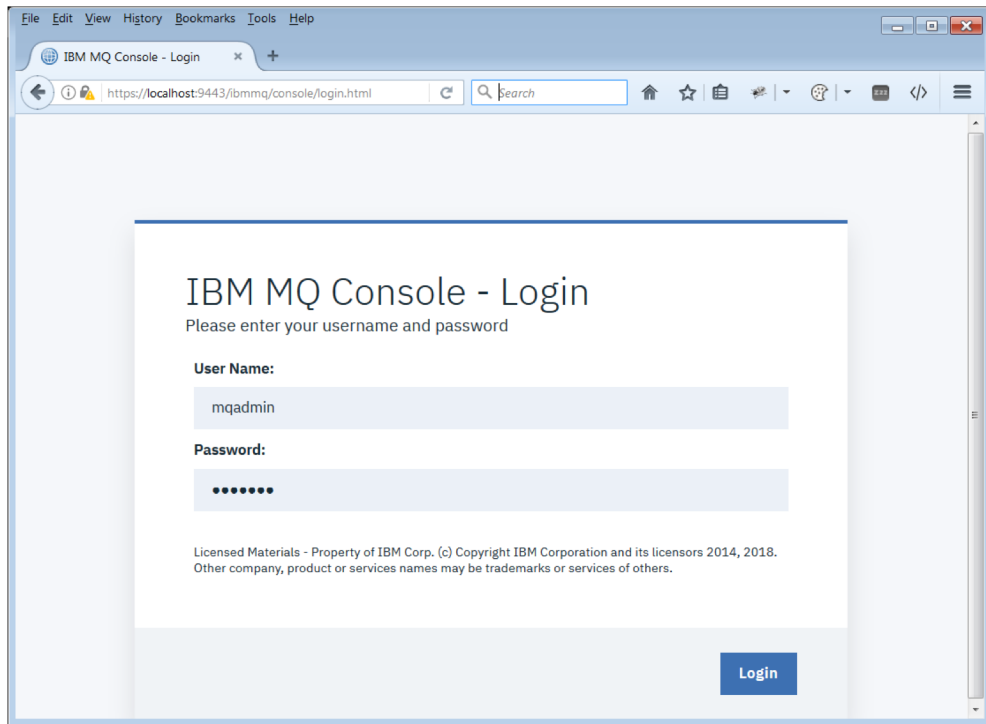
- ▶ With a user id and password
- ▶ With a client certificate

- **Log in credentials validated via user registry configured in the mqweb server**

- ▶ Like the REST API

- **Access determined by role**

- ▶ Same role names as REST API
- ▶ But in a different name space so REST users don't need to have same access as MQ Console users



The screenshot shows a web browser window with the title "IBM MQ Console - Login". The address bar displays "https://localhost:9443/ibmmq/console/login.html". The main content area has a light blue background and contains a white box with the following text:

IBM MQ Console - Login
Please enter your username and password

User Name:

Password:

Licensed Materials - Property of IBM Corp. (c) Copyright IBM Corporation and its licensors 2014, 2018.
Other company, product or services names may be trademarks or services of others.

Login

MQ Console – add widgets

- Console dashboard consists of a number of widgets, each widget shows information for a particular set of MQ objects: queue managers, queues, etc

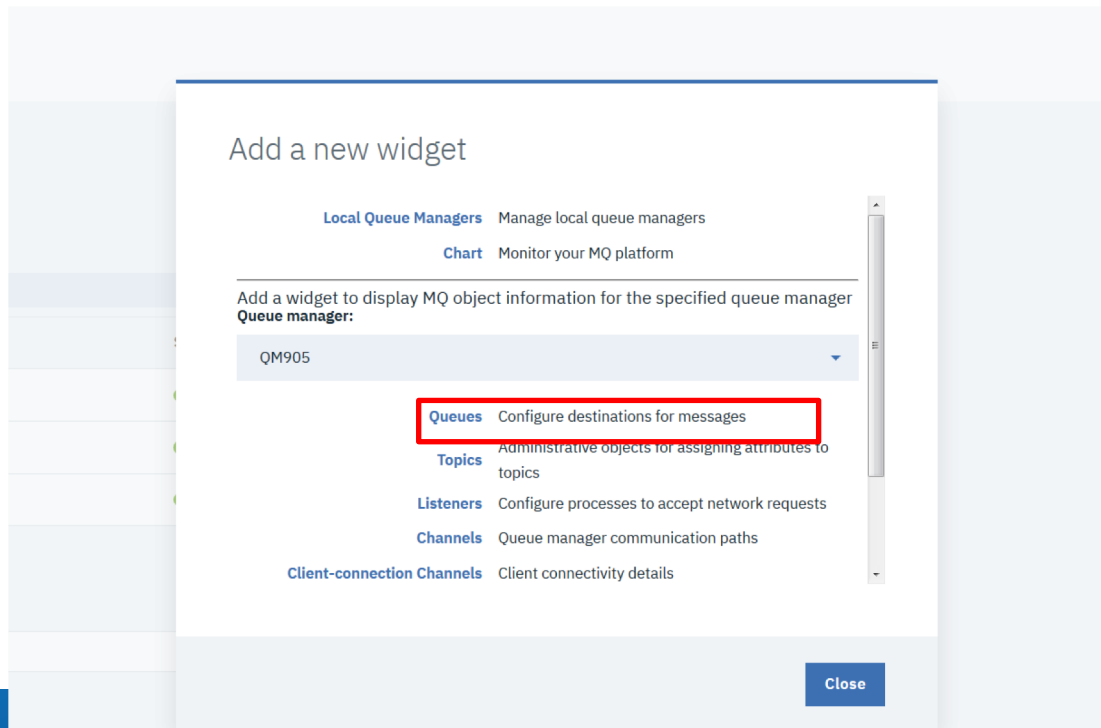
The screenshot shows the IBM MQ Console interface in a web browser. The browser's address bar displays `https://localhost:9443/ibmmq/console/`. The page title is "IBM MQ". Below the title, there is a tab labeled "Tab 1" and a "+" icon to add more tabs. On the right side of the page, there is a red-bordered button labeled "Add widget". The main content area displays the "Local Queue Managers" widget. This widget includes a search bar with a magnifying glass icon and the text "Search", and a "Create" button with a plus icon. Below these is a table with two columns: "Name" and "Status". The table lists three queue managers: QM905_3, QM905_2, and QM905, all with a status of "Running" indicated by a green dot. At the bottom of the table, it shows "Total: 3" and "Last updated: 8:19:52 AM".

Name	Status
QM905_3	Running
QM905_2	Running
QM905	Running

Total: 3
Last updated: 8:19:52 AM

MQ Console – add widgets

- Console dashboard consists of a number of widgets, each widget shows information for a particular set of MQ objects: queue managers, queues, etc



MQ Console – add widgets

- Console dashboard consists of a number of widgets, each widget shows information for a particular set of MQ objects: queue managers, queues, etc

The screenshot shows the IBM MQ Console interface. The browser address bar indicates the URL `https://localhost:9443/bmmq/console/`. The dashboard has a header with the 'IBM MQ' logo and a search bar. Below the header, there's a 'Tab 1' dropdown and a '+' button. The main content area contains two widgets. The 'Local Queue Managers' widget on the left shows a table with three entries, all with a status of 'Running'. The 'Queues on QM905' widget on the right, highlighted with a red border, shows a table with five entries, all with a 'Queue type' of 'Local' and a 'Queue depth' of 0 or 2. Both widgets have a 'Total' count and a 'Last updated' timestamp.

Local Queue Managers

Name	Status
QM905_3	Running
QM905_2	Running
QM905	Running

Total: 3 Last updated: 8:22:10 AM

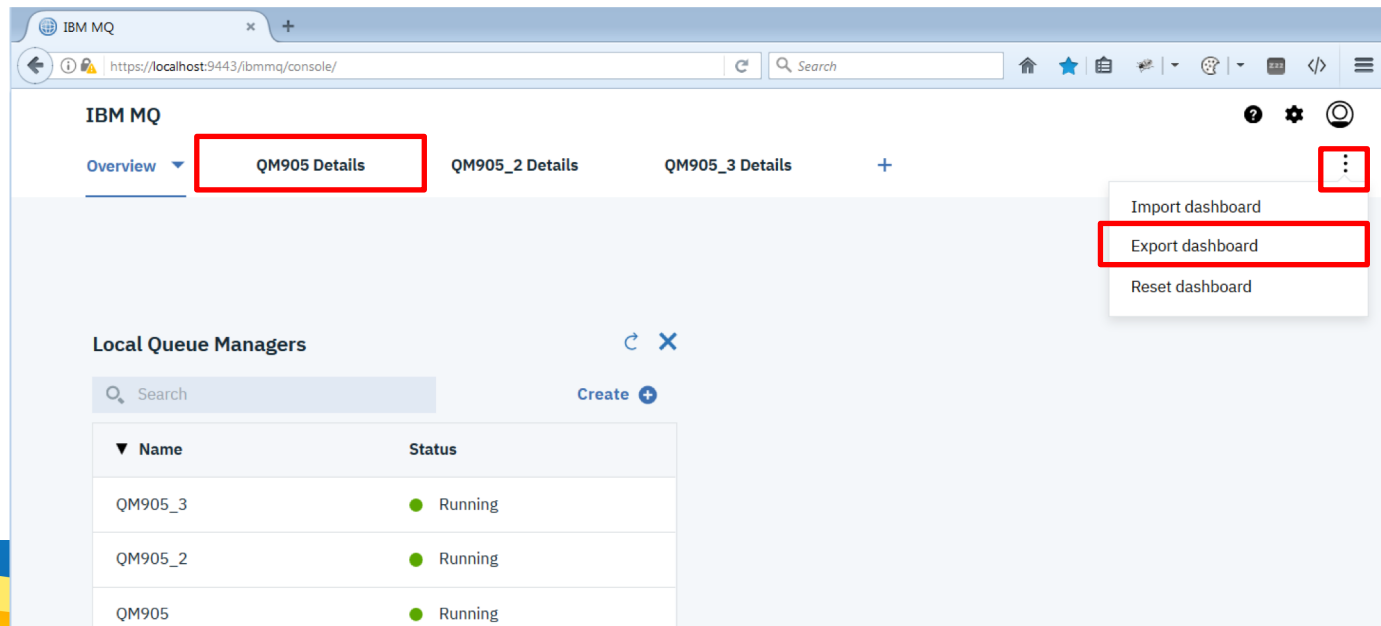
Queues on QM905

Name	Queue type	Queue depth
AQ	Local	2
MAXINSTQ	Local	30
NOTIFICATIONQ	Local	0
NOTificationQ	Local	0
Q1	Local	0

Total: 7 Last updated: 8:21:52 AM

MQ Console – layout

- Can use multiple tabs to help manage content
- Each user can lay out their dashboard according to their needs
- Can export dashboard to share layout with others



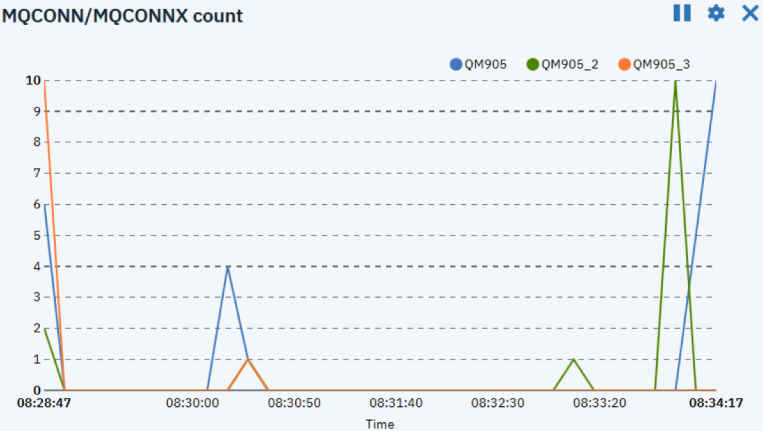
MQ Console - manage

- Monitor your MQ queue managers using charts generated from statistics information published to system topics
 - ▶ added in 9.0.0 on distributed platforms
- Display and alter objects using the properties editor
- Browse and send messages
- Provides a sub-set of MQ Explorer function

Properties for 'QM905_3'

General	Default transmission queue:	Channel auto definition:
Extended		Disabled
Cluster	Channel auto definition exit:	IP address version:
Repository		IPV4
Communication	Activity recording:	Trace-route recording:
Events	Message	Message
SSL	CHLAUTH records:	REVDNS lookup:
Statistics	Enabled	Enabled
Online monitoring		
Statistics monitoring		
Accounting monitoring		

MQCONN/MQCONN count



Subscribe to Topic

Topic string: *

/test/topic

Refresh

Search

▲ Date/Time	Message Body
May 18, 2018 8:37:11 AM	Hello world 1
May 18, 2018 8:37:14 AM	Hello world 2
May 18, 2018 8:37:17 AM	Hello world 3

