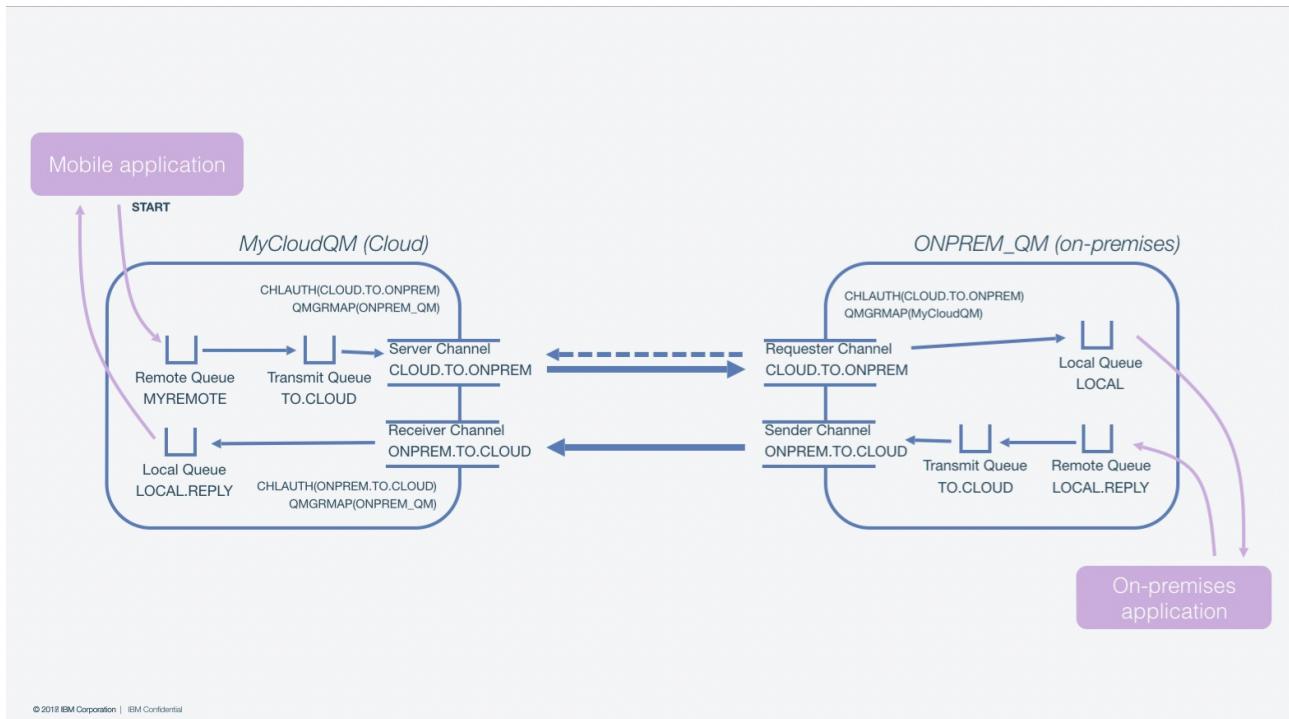


Connecting an IBM MQ On Cloud queue manager directly to an on-premises queue manager.



1. Table Of Contents

Connecting an IBM MQ On Cloud queue manager directly to an on-premises queue manager.	1
1. Table Of Contents	2
2. Connect To An On-Premise Queue Manager	3
2.1. Overview	3
2.2. Initial Setup	4
2.2.1. Obtain the connection details for your cloud queue manager	5
2.2.2. Connect to your on-premises queue manager	5
2.3. Configure Channels Between the Queue Managers	6
2.3.1. Start the (non-TLS) channels	12
2.3.2. Test sending a message from Cloud to on-premises	14
2.3.3. Test sending a message from on-premise to Cloud	16
2.4. Apply Server-Authentication TLS Security To the Channels	17
2.4.1. Re-check the channel status and send sample messages	22
2.5. Enable Mutual-Authentication TLS Security	23
2.5.1. Re-check the channel status and send sample messages	26
2.5.2. What to do if you cannot start the channels.	28

2. Connect To An On-Premise Queue Manager

2.1. Overview

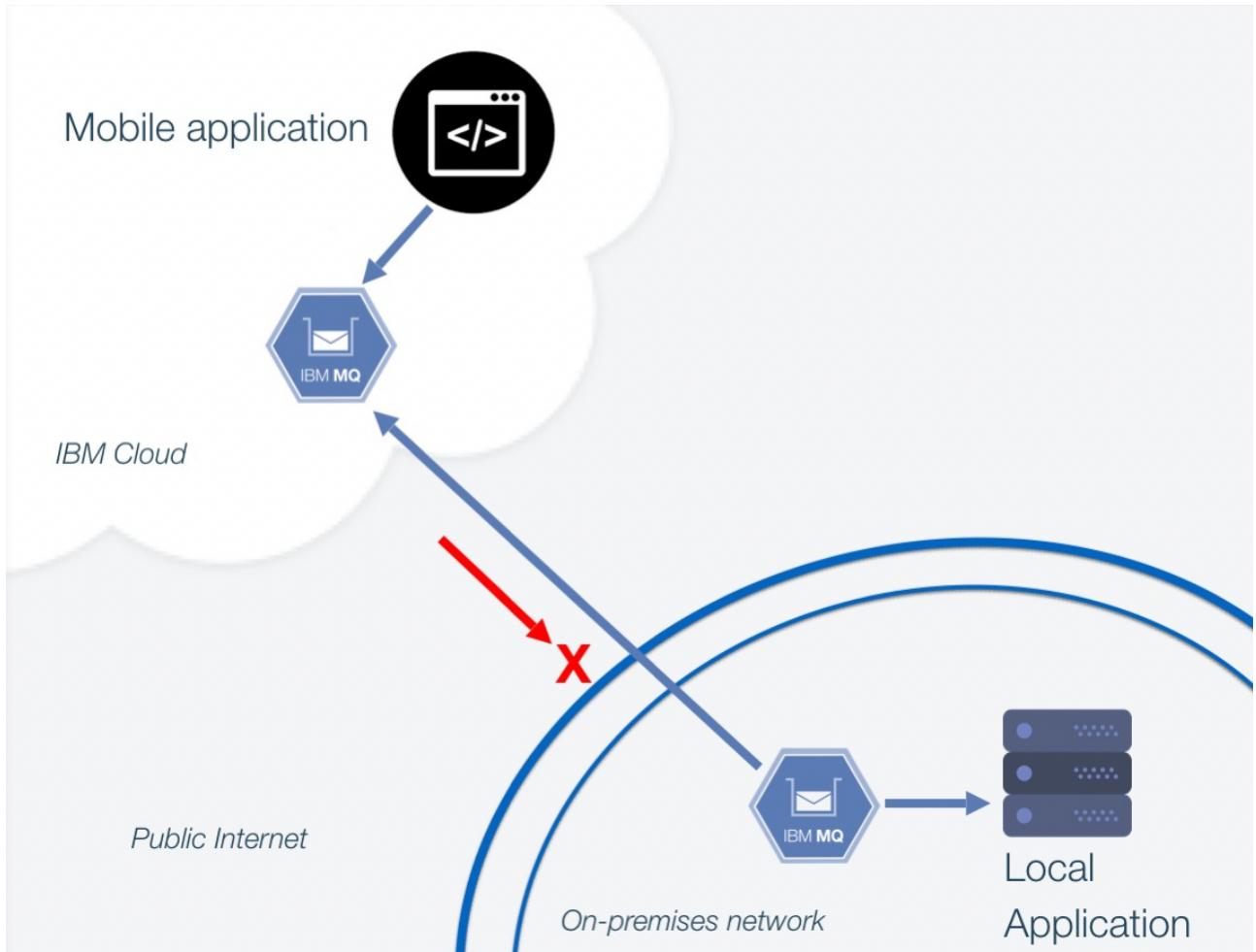
This document describes how to securely transfer messages between an application deployed in IBM Cloud and an application running in an on-premises data centre, using IBM MQ to provide the reliable secure connection between the two deployments.

The business scenario described is a mobile application being built and deployed using IBM Cloud. The source of data for this application is an existing on-premises application that is accessed by sending a message to an IBM MQ input queue, and the on-premises application returns a reply message to the MQ Cloud queue corresponding to the mobile application.

The mobile application will send its request message to a queue manager running in the MQ service in the IBM Cloud because it is close to the server on which the application logic runs. Once the request message is accepted by the queue manager, the application has done its part and can rely on IBM MQ to securely and reliably transmit the request message to the on-premises queue manager to which the existing on-premise application connects.

The on-premises queue manager has network connectivity that means it can call out to the internet in order to establish the connection to the Cloud queue manager, but the on-premises queue manager is not located in the DMZ and so cannot be contacted by incoming requests from the internet.

Figure 1 - Secure reliable connectivity to on-premises applications using IBM MQ



2.2. Initial Setup

There are two components of this system, which you will need to have installed :

1. A cloud-hosted queue manager deployed using the MQ service in IBM Cloud. In the following pages this is referred to as “MyCloudQM”.
2. An “on-premises” queue manager that will be connected to the cloud queue manager. In the following pages this is referred to as “ONPREM_QM”.

2.2.1. Obtain the connection details for your cloud queue manager

First obtain connection details for later use as follows:

Log in to your service instance in the IBM Cloud service console

Click on the row in the list that represents your queue manager, in order to view its details

Click the “Connection Information” button and select “Plain text” from the resulting dialog box

Save the text file for later use – the content is of the following form;

Platform: IBM MQ on Cloud

Queue manager name: MyCloudQM

Hostname: mycloudqm-1111.qm.us-south.mqcloud.ibm.com Listener port: 34567

Application channel name: CLOUD.APP.SVRCONN Administration channel name: CLOUD.ADMIN.SVRCONN

Deployment location: bmx-us-south

MQ Web Console login: <https://mycloudqm-1111.qm.us-south.mqcloud.ibm.com/ibmmq/console>

In particular it is important that you make note of the Queue Manager Name, the Hostname, the Listener Port and the Administration Channel Name attributes from the downloaded connection details as you will use them in the following sections.

You will also need a username/apikey pair to remotely connect to the queue manager. If you do not already have one, then click on “Administration” and select “runmqsc”. Here you will see instructions for remotely connecting, and also a “Create IBM Cloud API key” button. Use this to create and safely store an API key.

2.2.2. Connect to your on-premises queue manager

You will need administrative access to your on-premises queue manager later, so connect to it now with your choice of administration tool. The rest of the document describes using a console and mqsc commands for this work.

2.3. Configure Channels Between the Queue Managers

In this section you will configure the IBM MQ channels between the on-premises and cloud queue managers to enable messages to flow between the two zones. The channels you configure in this section will not be TLS encrypted but will allow you to demonstrate the successful transfer of messages in both directions. In a later section you will apply the TLS configuration to protect the data as it flows between the two queue managers.

A brief introduction to IBM MQ channels:

Connections between two IBM MQ queue managers are defined using a “channel”. A channel is a one-directional pipe through which messages can flow and when configuring connections between two queue managers (as in this scenario) there is a matching channel on each queue manager – for example a “Sender” channel on one queue manager has a matching “Receiver” channel on the other queue manager.

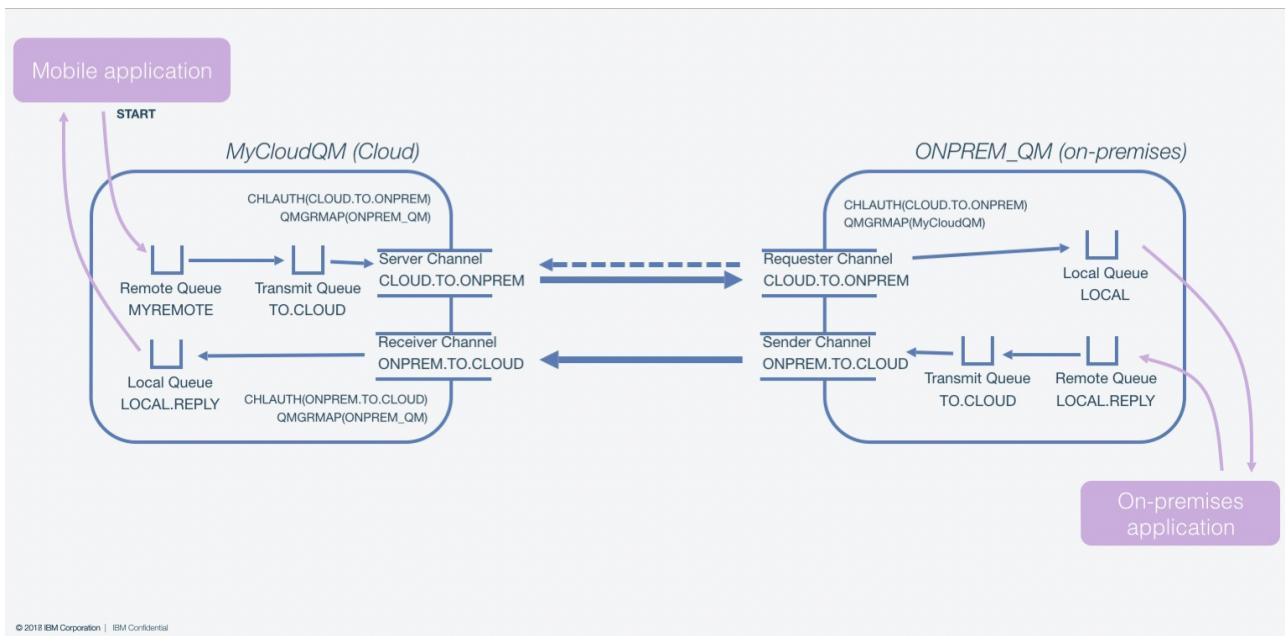
Typically, two sets of channels are configured so as to enable messages to flow in both directions, for example as required when a request message flowing in one direction might result in a reply message flowing back to the originator.

A channel such as a Sender channel that is responsible for sending messages to a different queue manager is also associated with a “transmit queue” (often shortened to XMIT queue). The name of the transmit queue is configured as an attribute in the definition of the channel. If the target queue manager is unavailable the outbound messages will be held on this transmit queue until connectivity can be re-established. This is often referred to as “store and forward” in that the source queue manager will store the outbound messages until they have been successfully forwarded to the target queue manager.

It is also possible to apply “channel authentication records” (CHLAUTH) to restrict the behaviour of the channel – for example to define that only a named queue manager is able to connect via a given channel. The name of the channel authentication record must match the name of the channel to which it is being applied, either as a literal match (they have exactly the same name) or by defining the name of the channel authentication record with a wildcard (such as “MYCHLS.*”).

A “remote queue definition” is used to provide a pointer to a queue which is running on a different queue manager. Messages that are sent to the remote queue are routed by IBM MQ to the defined queue on the remote queue manager.

The following diagram shows the type or configuration objects that you will need to create.



© 2018 IBM Corporation | IBM Confidential

Establishing an inbound connection to an on-premises queue manager using Requester-Server channels

It is important to note that the on-premises queue manager cannot be contacted directly by the Cloud queue manager (MyCloudQM) because it has no publicly reachable IP address, so you will replace the traditional Sender-Receiver channels in the top half of the diagram with Requester- Server channels.

In the Requester-Server approach the on-premises Requester channel will call out from the on- premises network to establish a connection to the Cloud queue manager, and then the Server channel will use the established connection to transmit messages back to the on-premises queue manager. This reverse-initiation means that you also need an extra channel authentication rule on the cloud side to grant permission for the on-premises queue manager to establish the initial connection.

You can configure all the objects required using the “runmqsc” command-line (CLI) tool as it enables you to concisely describe the objects that need to be created. You will be using one command-prompt session to configure the on-premises queue manager, and a second command-prompt session to configure the Cloud queue manager – be careful to ensure you are using the correct one at each step, as indicated by the instructions.

If you prefer you can also carry out the equivalent commands using the graphical MQ Explorer or MQ Console user interfaces.

To get started configuring the on-premises queue manager (right hand side of the picture above), open a Command Prompt on your local queue manager machine and execute the commands below.

Notes:

- The comment lines (starting “//”) are for information purposes and must not be entered into your command prompt.

- You will need to substitute some values to match your specific setup before executing the commands – in particular the value of the both CONNAME attributes must be substituted for the hostname and port of your cloud queue manager (and also the QMNAME and RQMNAME if you used a different name for your cloud queue manager)

```
// Starting with a new command prompt
// Connect to on-prem queue manager (using a local connection)

runmqsc QMNAME

// Confirm you are successfully connected by executing a command

PING QMGR

// Define the local queue that the on-premises application read

DEFINE QLOCAL(LOCAL)

// Define transmit queue that will be used for the on-prem sender channel

DEFINE QLOCAL(TO.CLOUD) USAGE(XMITQ)

// Create remote queue definition to route reply messages back to the cloud

DEFINE QREMOTE(LOCAL.REPLY) RNAME(LOCAL.REPLY) RQMNAME('MyCloudQM')
XMITQ(TO.CLOUD)

// Define the on-premises sender channel that will connect to the cloud

DEFINE CHANNEL(ON_PREM.TO.CLOUD) CHLTYPE(SDR) +
CONNAME('mycloudqm-1111.qm.us-south.mqcloud.ibm.com(34567') XMITQ(TO.CLOUD)
TRPTYPE(TCP)

// Define requester channel that will call out from on-prem queue manager

DEFINE CHANNEL(CLOUD.TO.ON_PREM) CHLTYPE(RQSTR) +
CONNAME('mycloudqm-1111.qm.us-south.mqcloud.ibm.com(34567') TRPTYPE(TCP)

// Create the authentication record to allow incoming connections from the
cloud QM

SET CHLAUTH(CLOUD.TO.ON_PREM) TYPE(QMGRMAP) QMNAME('MyCloudQM') ACTION(ADD)
USERSRC(CHANNEL)
```

After each command the runmqsc terminal will confirm that the command executed successfully, with output messages as follows;

5724-H72 (C) Copyright IBM Corp. 1994, 2017. Starting MQSC for queue manager ONPREM_QM.

AMQ8415I: Ping IBM MQ Queue Manager command complete.

AMQ8006I: IBM MQ queue created.

AMQ8006I: IBM MQ queue created.

AMQ8006I: IBM MQ queue created.

AMQ8014I: IBM MQ channel created.

AMQ8014I: IBM MQ channel created.

AMQ8877I: IBM MQ channel authentication record set.

Now that you have successfully configured the on-premises queue manager you must carry out the associated configuration on the cloud queue manager (left hand side of the picture above). You will again do this using the “runmqsc” CLI executed from a Command Prompt, but this time you will be connecting remotely as a client to the cloud queue manager.

Start a new command prompt to configure the cloud queue manager using runmqsc as shown below.

Notes:

- You will authenticate to the cloud queue manager using your MQ username and a platform API key for your user from the Identity and Access service in IBM Cloud. You will need to substitute your MQ username in place of the string “myusername” in the runmqsc command below, and enter your platform API key when prompted for the password
 - The comment lines (starting “//”) are for information purposes and must not be entered into your command prompt
 - You will need to substitute some values to match your specific setup before executing the commands – in particular the hostname and port values in the MQSERVER environment variable (and also the QMNAME and RQMNAME if you used a different name for your on-premises queue manager)
-

```
// Set this environment variable to tell runmqsc the hostname/port where the  
cloud queue manager is running
```

```
set MQSERVER=CLOUD.ADMIN.SVRCONN/TCP/mycloudqm-1111.qm.us-  
south.mqcloud.ibm.com(34567)
```

```

// Connect to cloud queue manager, using your MQ username and platform API
key

runmqsc -u myusername -c MyCloudQM

// Confirm you are successfully connected by executing a command

PING QMGR

// Define the transmit queue that will be used for the cloud sender channel

DEFINE QLOCAL(TO.ON_PREM) USAGE(XMITQ)

// Create a remote queue definition - this allows messages to be sent to the
on-premises queue

DEFINE QREMOTE(LOCAL) RNAME(LOCAL) RQMNAME(ONPREM_QM) XMITQ(TO.ON_PREM)

// Define the local queue that will be used to hold reply messages from the
on-premises Stock application

DEFINE QLOCAL(LOCAL.REPLY)

// Define the server channel that will respond to the on-prem requester
channel

DEFINE CHANNEL(CLOUD.TO.ON_PREM) CHLTYPE(SVR) XMITQ(TO.ON_PREM) TRPTYPE(TCP)

// Define receiver channel to accept connections from on-prem sender channel

DEFINE CHANNEL(ON_PREM.TO.CLOUD) CHLTYPE(RCVR) TRPTYPE(TCP)

// Create the channel authentication record to allow incoming connections
from the on-prem QM sender channel

SET CHLAUTH(ON_PREM.TO.CLOUD) TYPE(QMGRMAP) QMNAME(ONPREM_QM) ACTION(ADD)
USERSRC(CHANNEL)

// Create the channel authentication record to allow incoming connections
from the on-prem QM requester channel

SET CHLAUTH(CLOUD.TO.ON_PREM) TYPE(QMGRMAP) QMNAME(ONPREM_QM) ACTION(ADD)
USERSRC(CHANNEL)

```

After each command the runmqsc terminal will confirm that the command executed successfully, with output messages as follows;

5724-H72 (C) Copyright IBM Corp. 1994, 2017.

Starting MQSC for queue manager MYCLOUDQM.

AMQ8415I: Ping IBM MQ Queue Manager command complete.

AMQ8006I: IBM MQ queue created.

AMQ8006I: IBM MQ queue created.

AMQ8006I: IBM MQ queue created.

AMQ8014I: IBM MQ channel created.

AMQ8014I: IBM MQ channel created.

AMQ8877I: IBM MQ channel authentication record set.

AMQ8877I: IBM MQ channel authentication record set.

You have now configured all the necessary objects on both the cloud and the on-premises queue managers and are ready to start the communication channels.

2.3.1. Start the (non-TLS) channels

In this section you will start the channels you have defined, confirm that they are running successfully, and send a test message in each direction to prove the flow of messages.

Since the on-premises queue manager has to initiate both its outbound connection to the cloud and also trigger the inbound connection from the cloud (via the on-premises Requester channel) you must start both channels from the on-premises queue manager side.

Return to the runmqsc session for your on-premise queue manager (or create a new one if you have closed the original).

```
// Check you are attached to the on-premise QM - it should show "ONPREM_QM"  
DISPLAY QMGR QMNAME  
  
// Start the sender channel from on-premises to cloud  
  
START CHANNEL(ON_PREM.TO.CLOUD)  
  
// Wait for 15 seconds and check that the sender channel shows  
STATUS(RUNNING)  
  
DISPLAY CHSTATUS(ON_PREM.TO.CLOUD)  
  
// Start the requester channel on the on-premises queue manager START  
CHANNEL(CLOUD.TO.ON_PREM)  
  
// Wait for 15 seconds and check that the sender channel shows  
STATUS(RUNNING) DISPLAY CHSTATUS(CLOUD.TO.ON_PREM)
```

You will see the following output when the commands are executed;

```
AMQ8018I: Start IBM MQ channel accepted.  
AMQ8417I: Display Channel Status details.  
CHANNEL(ON_PREM.TO.CLOUD) CHLTYPE(SDR)  
CONNNAME(169.60.49.174(34567)) CURRENT  
RQMNAME(MyCloudQM) STATUS(RUNNING)
```

SUBSTATE (MQGET) XMITQ (TO.CLOUD)

AMQ8018I: Start IBM MQ channel accepted.

AMQ8417I: Display Channel Status details.

CHANNEL (CLOUD.TO.ON_PREM) CHLTYPE (RQSTR)

CONNNAME (169.60.49.174 (34567)) CURRENT

RQMNAME (MyCloudQM) **STATUS(RUNNING)**

SUBSTATE (RECEIVE)

2.3.2. Test sending a message from Cloud to on-premises

Using the runmqsc command prompt for your on-premises queue manager check that there are no messages currently on the on-premises “LOCAL” queue.

```
// Check you are attached to the on-premise QM - it should show "ONPREM_QM"  
DISPLAY QMGR QMNAME  
  
// Check the current queue depth of the local "LOCAL" queue is zero, ready  
for us to test the transmission of messages from cloud to on-premise; should  
show CURDEPTH(0)  
  
DISPLAY QSTATUS(LOCAL) CURDEPTH
```

You should already have an API key to use as a password for the MQ console. If you have not, then go back to the IBM Cloud Queue Manager list, select your queue manager and choose “Administer” then “runmqsc”. You will find a button labelled “Create IBM Cloud API Key” - use this to create your API key and save it.

Note: In later versions of the IBM MQ on Cloud service, single-signon is enabled so you may be redirected after following the instructions below straight to the MQ Console without entering a username/password.

Click the ‘Launch MQ Console’ button at the bottom of the administration page and the MQ Console will be opened in a new tab in your browser. Enter your MQ username and the platform API key that you just generated and downloaded for your user, then click Login.

You are now logged in to the MQ Console and can see the default queues, topics and other objects that have been created by the system in your new queue manager.

In the Queues widget, select the remote queue LOCAL and click the  icon to send a test message. Click the “Put” button to send your test message.

Now return to the runmqsc window for the ONPREM_QM, and re-run the command to display the queue depth of the local LOCAL queue. You will see that the queue depth has increased to 1, indicating that the message was successfully transmitted from the cloud queue manager to the on-premises queue manager.

```
// Check you are attached to the on-prem QM - it should show "ONPREM_QM"  
DISPLAY QMGR QMNAME  
  
// Check the current queue depth of the local "LOCAL" queue is now 1,  
indicating the message has been successfully received; should show  
CURDEPTH(1)  
  
DISPLAY QSTATUS(LOCAL) CURDEPTH
```

If you wish you can view the content of that message by opening up the MQ Explorer for the ONPREM_QM, right clicking on the LOCAL queue and selecting “Browse messages” then double-clicking on the message entry in the table and selecting the “Data” tab

2.3.3. Test sending a message from on-premise to Cloud

You will use the “amqspput” command-line utility to connect to the ONPREM_QM and send a test message which will be transmitted to the cloud queue manager.

First, use the MQ Console for the cloud queue manager to show that there are zero messages on the LOCAL.REPLY queue

Now open a new command prompt on your “on-premises” virtual machine and enter the following “amqspput” command to connect to the queue, then type some simple text for your message and press Enter twice (once to finish the message, the second to quit).

```
amqspput LOCAL.REPLY ONPREM_QM
Hello from on-premises
<enter>
<enter>
```

Now go back to the MQ Console for your cloud queue manager and click the Refresh icon in the top right corner (two arrows in a circle), to observe that the queue depth for LOCAL.REPLY is now 1 rather than 0.

You can browse the content of the message on the LOCAL.REPLY queue by selecting that row in the table, then pressing on the folder icon to Browse Messages

You have successfully demonstrated messages being transmitted over non-TLS channels from both Cloud to on-premises, and on-premises to Cloud.

The next section describes how to apply TLS security to the channels to protect the message data as it transmits across the Internet.

2.4. Apply Server-Authentication TLS Security To the Channels

In this section you will take the non-TLS channel connections that you defined and tested above and convert them to TLS channels. This ensures that the transmitted data is protected within a TLS session as it is transferred across the Internet between the on-premises queue manager and the cloud queue manager.

As discussed earlier, you are initiating both inbound and outbound channels from the on-premises queue manager, so you need to configure the on-premises queue manager to “trust” the server certificate that will be presented by the cloud queue manager when it makes the initial TCP connection.

The cloud queue manager is already configured with a default server certificate (a wildcard certificate issued by DigiCert) so you need to download that server certificate to the on-premises machine and configure the queue manager to trust it.

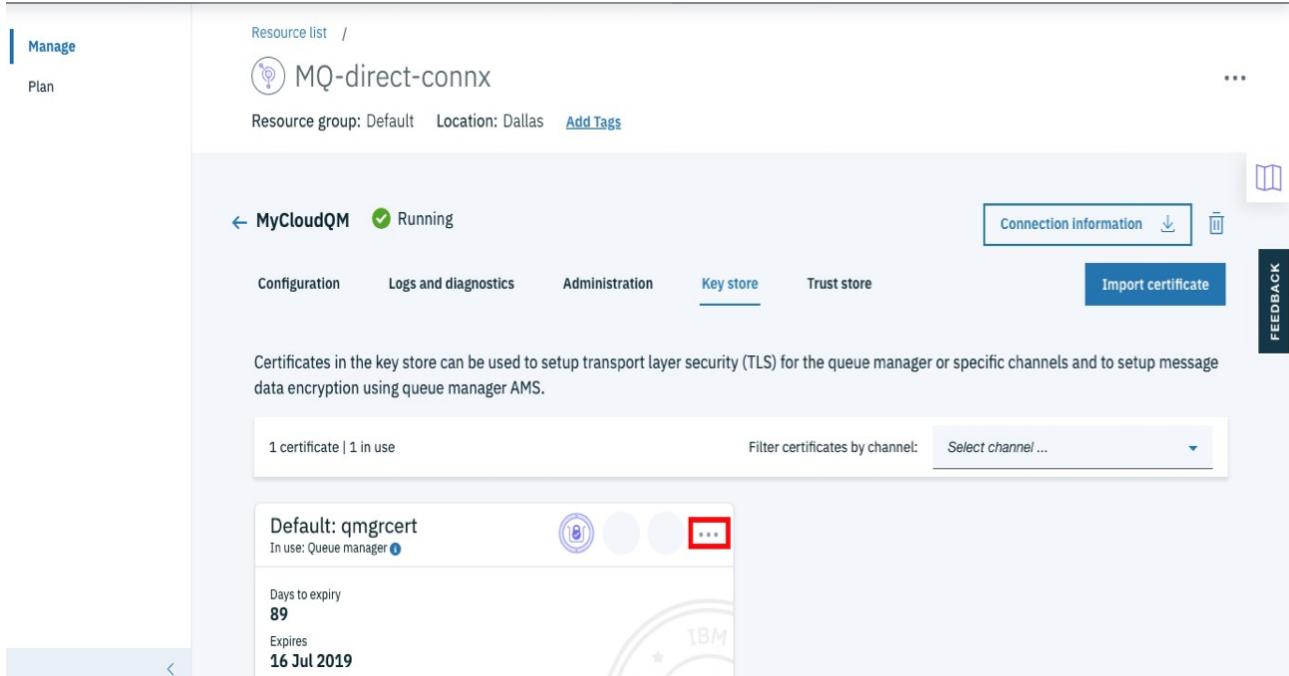
The second part of the configuration process is to set the cipher specification on the channels so that they are able to agree which cipher to use to establish the TLS session.

The easiest way to obtain a copy of the cloud queue manager’s server certificate is via the IBM Service Console.

From the on-premises machine on the IBM Cloud service console, with the list of queue managers showing, click the row containing your queue manager (MyCloudQM). Click on the ‘Key store’ tab to access the required certificate:

The screenshot shows the IBM Cloud Service Console interface for managing queue managers. The top navigation bar includes 'Manage' and 'Plan'. The main area displays a 'Resource list' for 'MQ-direct-connx'. A specific queue manager, 'MyCloudQM', is selected and shown as 'Running'. Below the status, there are tabs: 'Configuration' (selected), 'Logs and diagnostics', 'Administration', 'Key store' (which is highlighted with a red border), and 'Trust store'. Under the 'Key store' tab, it shows 'MQ version 9.1.2', 'Revision 1', and 'Last updated 16 Apr 2019'. A message box states 'Good news! This queue manager is running the latest version'. On the right side, there's a 'Small' plan summary with a price of '\$1.30 per hour' and '1 x VPC-hour price'. It also lists 'Non-persistent messages per second 1500' and 'Concurrent connections'. A 'FEEDBACK' button is located on the far right.

The easiest way to obtain a copy of the cloud queue manager's server certificate is via the IBM Service Console. Select your queue manager and click on "Keystore". The list of key store certificates will now be showing, click on the three dots icon within the "Default: qmrcert" box:



Resource list /

MQ-direc-connx

Resource group: Default Location: Dallas Add Tags

MyCloudQM Running

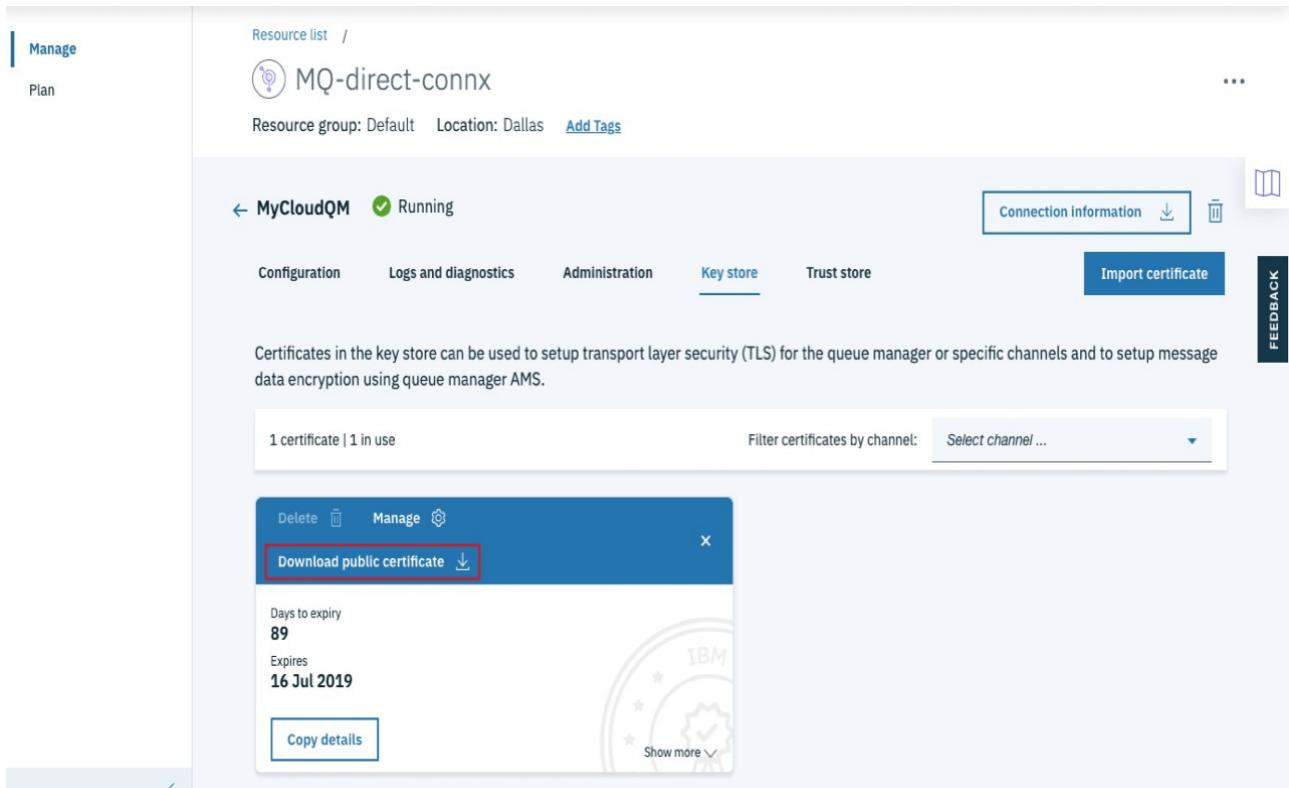
Configuration Logs and diagnostics Administration Key store Trust store Import certificate FEEDBACK

Certificates in the key store can be used to setup transport layer security (TLS) for the queue manager or specific channels and to setup message data encryption using queue manager AMS.

1 certificate | 1 in use Filter certificates by channel: Select channel ...

Default: qmrcert
In use: Queue manager ⓘ Days to expiry: 89 Expires: 16 Jul 2019

Click on "Download public certificate" and then save the certificate (qmrcert.pem).



Resource list /

MQ-direc-connx

Resource group: Default Location: Dallas Add Tags

MyCloudQM Running

Configuration Logs and diagnostics Administration Key store Trust store Import certificate FEEDBACK

Certificates in the key store can be used to setup transport layer security (TLS) for the queue manager or specific channels and to setup message data encryption using queue manager AMS.

1 certificate | 1 in use Filter certificates by channel: Select channel ...

Default: qmrcert
In use: Queue manager ⓘ Days to expiry: 89 Expires: 16 Jul 2019

Download public certificate ↴

Copy details Show more ↴

Make a note of the path to the downloaded certificate as you will need this for the next steps.

Now you can apply the downloaded certificate to the on-premises queue manager. Right-click on the “Command Prompt” icon on the desktop and choose “Run as administrator” to open a privileged command prompt.

Change directory into the ‘ssl’ directory for your queue manager; For example, on a Windows machine this would generally be in ‘c:\ProgramData\IBM\MQ\qmgrs\<qmname>\ssl’. On a Linux system, it would usually be found in ‘/var/mqm/qmgrs/<qmname>/ssl’.

Note: The directory in which your queue manager looks for the key store is defined by the variable SSLKEYR. If in doubt, runqmssc and DISPLAY QMGR to check that variable.

If you already have a ‘key.kdb’ file (key/trust store) in this directory, you can skip the next step. Otherwise, you will now use the “runmqakm” command to create a key store:

```
runmqakm -keydb -create -db key.kdb -pw mySecretPassword -stash
```

Next, add the cloud public certificate you downloaded earlier. The label should be ‘ibmwebspheremq<qmname>’ - where <qmname> is the name of your cloud queue manager (in lower case).

```
runmqakm -cert -add -db key.kdb -stashed -label ibmwebspheremqmycloudqm -file <path_to_cert>/qmgrcert.pem
```

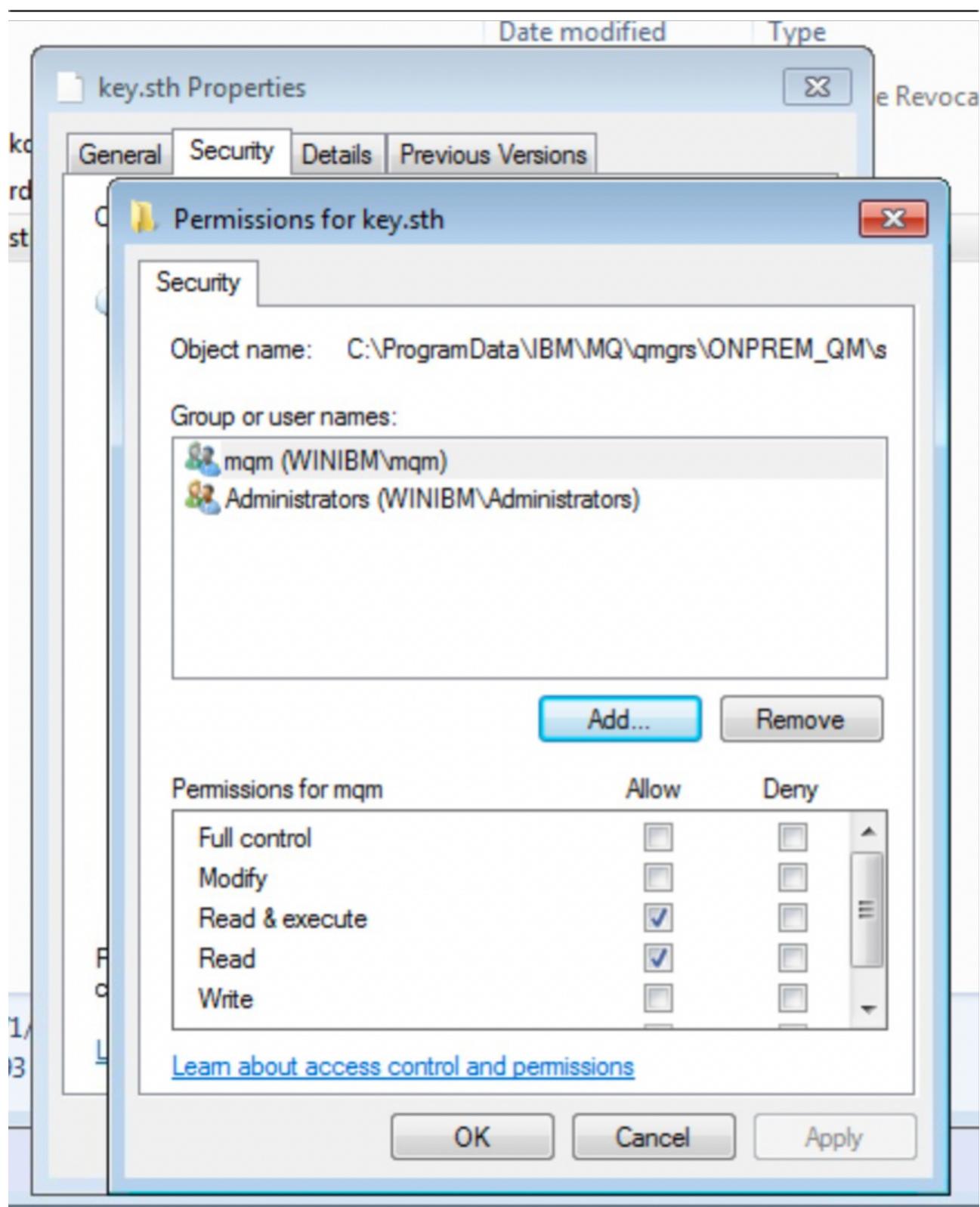
// Verify that your certificate is present by running this command:

```
runmqakm -cert -list -db key.kdb -stashed
```

Note: The default certificate “qmrcert” is signed by DigiCert so you must have the complete chain of certificates up to the digicert root CA in your key.kdb. If you are using any other certificate then it must be self-signed, or have its complete certificate chain upto its CA in the key.kdb.

If on a Windows system, you must add an extra permission to the Stash file so that it can be accessed by the MQ queue manager. Open a Windows Explorer and navigate to “C:\ProgramData\IBM\MQ\qmgrs\ONPREM_QM\ssl” (note that ProgramData is a hidden folder).

Right click on the “key.sth” file, click “Properties”. If prompted, click “Continue” to allow permissions to view the properties. Select the “Security” tab then click “Continue” again to view the security settings. Click “Add”, Then enter “mqm” in the “object names” and click “OK”. The resulting configuration should look as follows:



Click “OK” twice to close the file properties dialog.

You can now apply an agreed cipher spec to each of the four channels you defined earlier to instruct them to communicate over TLS. For the purposes of this example you can use ‘ANY_TLS12’ as the cipher spec. You will

also indicate that the channels which accept incoming connections should require the other end of the connection to provide a client certificate to identify itself by setting ‘SSLAUTH’ to ‘OPTIONAL’.

Return to the runmqsc command prompt that you opened to configure your cloud queue manager (or open a new prompt if you closed it).

```
// Check that you are connected to the correct queue manager (MyCloudQM)
DISPLAY QMGR QMNAME

// Apply the SSLCIPH and SSLCAUTH settings to the existing channels
ALTER CHL('ON_PREM.TO.CLOUD') CHLTYPE(RCVR) SSLCIPH(ANY_TLS12)
SSLCAUTH(OPTIONAL) CERTLBL('qmgrcert')

ALTER CHL('CLOUD.TO.ON_PREM') CHLTYPE(SVR) SSLCIPH(ANY_TLS12)
SSLCAUTH(OPTIONAL) CERTLBL('qmgrcert')

REFRESH SECURITY TYPE(SSL)
```

Now you need to do the equivalent steps on the on-premises queue manager. Return to the original command prompt for your on-premises queue manager (or open a new one if you have closed it).

```
// Check that you are connected to the correct queue manager (ONPREM_QM)
DISPLAY QMGR QMNAME
ALTER CHL('ON_PREM.TO.CLOUD') CHLTYPE(SDR) SSLCIPH(ANY_TLS12)
CERTLBL('ibmwebspheremqonprem_qm')

ALTER CHL('CLOUD.TO.ON_PREM') CHLTYPE(RQSTR) SSLCIPH(ANY_TLS12)
SSLCAUTH(OPTIONAL) CERTLBL('ibmwebspheremqonprem_qm')

REFRESH SECURITY TYPE(SSL)
```

You have now finished configuring the channels for TLS.

2.4.1. Re-check the channel status and send sample messages

Now that you have configured the TLS setup you can confirm the channels are running successfully and send some test messages again to prove the communication. The channels should have been stopped by the security refresh, but we will stop them to make sure.

```
// Check that you are connected to the correct queue manager  
(ONPREM_QM)  
DISPLAY QMGR QMNAME  
  
// Start the channels again in case they stopped while you were making  
the changes. // If the channel is already running you may see a  
message saying it is "in use".  
  
STOP CHANNEL(ON_PREM.TO.CLOUD)  
  
STOP CHANNEL(CLOUD.TO.ON_PREM)  
  
START CHANNEL(ON_PREM.TO.CLOUD)  
  
START CHANNEL(CLOUD.TO.ON_PREM)  
  
// Wait for 15 seconds and check the status of the channels to confirm  
all the changes // have taken effect. Both channels should show  
STATUS(RUNNING)  
  
DISPLAY CHSTATUS(ON_PREM.TO.CLOUD)  
  
DISPLAY CHSTATUS(CLOUD.TO.ON_PREM)
```

To test the transfer of messages over the TLS-enabled channels please repeat the steps defined above where you used the MQ Console and amqsput command respectively to demonstrate the successful flow of messages from the cloud queue manager to on-premises and vice versa.

2.5. Enable Mutual-Authentication TLS Security

In this section, you will take the server-authentication TLS enabled channels that you have defined and tested above and alter them to enable mutual authentication between the two queue managers. In order to achieve this, you will need to configure the on-cloud queue manager to “trust” a server certificate that will be presented by the on-premises queue manager and vice-versa.

Firstly, use the following commands to create a self-signed personal certificate on the on-premises machine by using runmqakm. Note that the command should all be on one line and the label should be ‘ibmwebspheremq<qmname>’ - where <qmname> is the name of your on-premises queue manager in lower case):

```
runmqakm -cert -create -db key.kdb -stashed -dn "CN=onprem_qm,O=IBM,C=GB"  
-label ibmwebspheremqonprem_qm
```

Now use the following command to extract the public part of the self-signed personal certificate on the on-premises machine by using runmqakm (Note, the command should all be on one line):

```
runmqakm -cert -extract -db key.kdb -stashed -label ibmwebspheremqonprem_qm  
-target <path_to_download_directory>/onpremcert.pem -format ascii -fips
```

You now need to import the “onpremcert.pem” file into the trusted key store of your on-cloud queue manager.

Access the IBM Cloud service console and from the list of queue managers, click the row containing your queue manager (MyCloudQM) and then click on the ‘Trust store’ tab.

Resource list /

MQ-direct-connx

Resource group: Default Location: Dallas Add Tags

MyCloudQM Running

Configuration Logs and diagnostics Administration Key store Trust store

MQ version 9.1.2 Revision 1 Last updated 16 Apr 2019 Good news! This queue manager is running the latest version

Details

Display name Enter a display name for your queue manager

Queue manager name MyCloudQM

Small

\$1.30 per hour
1 x VPC-hour price

Non-persistent messages per second
1500

Concurrent connections

Click on “Import certificate”, then on “Browse files” and navigate to the onpremcert.pem file you downloaded in the previous steps.

Then click “open” and then “next”. Specify the label to give to the certificate (such as “ibmwebspheremqonprem_qm”) and then click “Save”.

The newly imported certificate should now appear in the list of certificates in the trust store and it should show as being trusted:

Resource list /

MQ-direct-connx

Resource group: Default Location: Dallas Add Tags

MyCloudQM2 Running

Configuration Logs and diagnostics Administration Key store Trust store Import certificate

Label	Days to expiry	Trusted
□ CN=DigiCert SHA2 Secure Server CA, O=DigiCert Inc, C=US	1402	<input checked="" type="checkbox"/>
□ DigiCertRootCA	4570	<input checked="" type="checkbox"/>
□ ibmwebspheremqonprem_qm	365	<input checked="" type="checkbox"/>

Return to the runmqsc command prompt that you opened to configure your cloud queue manager (or open a new prompt if you closed it).

```
DISPLAY QMGR QMNAME
// Apply the SSLCIPH and SSLCAUTH settings to the existing channels

ALTER CHL('ON_PREM.TO.CLOUD') CHLTYPE(RCVR) SSLCIPH(ANY_TLS12)
SSLCAUTH(REQUIRED) CERTLBL('qmgrcert')

ALTER CHL('CLOUD.TO.ON_PREM') CHLTYPE(SVR) SSLCIPH(ANY_TLS12)
SSLCAUTH(REQUIRED) CERTLBL('qmgrcert')

// Refresh SSL security
REFRESH SECURITY TYPE(SSL)
```

Now you need to do the equivalent steps on the on-premises queue manager. Return to the original command prompt for your on-premises queue manager (or open a new one if you have closed it).

```
// Check that you are connected to the correct queue manager (ONPREM_QM)
DISPLAY QMGR QMNAME
ALTER CHL('CLOUD.TO.ON_PREM') CHLTYPE(RQSTR) SSLCIPH(ANY_TLS12)
SSLCAUTH(REQUIRED) CERTLBL('ibmwebspheremqonprem_qm')

// Refresh SSL security

REFRESH SECURITY TYPE(SSL)
```

You have now finished configuring the channels for mutual authentication TLS.

2.5.1. Re-check the channel status and send sample messages

Now that you have configured mutual authentication TLS you can confirm the channels are running successfully and send some test messages again to prove the communication.

Return to the runmqsc command prompt that you opened to configure your on-premises queue manager (or open a new prompt if you closed it).

```
// Check that you are connected to the correct queue manager (ONPREM_QM)  
DISPLAY QMGR QMNAME  
  
// Start the channels again in case they stopped while you were making the  
changes. // If the channel is already running you may see a message saying it  
is "in use".  
  
START CHANNEL(ON_PREM.TO.CLOUD) START CHANNEL(CLOUD.TO.ON_PREM)  
  
// Wait for 15 seconds and check the status of the channels to confirm all  
the changes  
// have taken effect. Both channels should show STATUS(RUNNING) and in each  
case, the value // of SSLPEER should match  
  
DISPLAY CHSTATUS(ON_PREM.TO.CLOUD) SSLPEER  
  
DISPLAY CHSTATUS(CLOUD.TO.ON_PREM) SSLPEER
```

Next, return to the runmqsc command prompt that you opened to configure your cloud queue manager (or open a new prompt if you closed it).

```
// Check that you are connected to the correct queue manager (MYCLOUDQM)  
DISPLAY QMGR QMNAME  
  
// Check the status of the channels to confirm all the changes  
// have taken effect. Both channels should show STATUS(RUNNING) and in each  
case, the value // of SSLPEER should match
```

```
DISPLAY CHSTATUS(ON_PREM.TO.CLOUD) SSLPEER
```

```
DISPLAY CHSTATUS(CLOUD.TO.ON_PREM) SSLPEER
```

To test the transfer of messages over the mutual authenticationTLS-enabled channels please repeat the steps defined earlier where you used the MQ Console and amqsput command respectively to demonstrate the successful flow of messages from the cloud queue manager to on-premises and vice versa.

2.5.2. What to do if you cannot start the channels.

If the channels do not start, then the certificate exchange has probably failed. There are some good tools to help you locate the issues which might have caused this.

First, examine your certificate pem files using openssl:

```
openssl x509 -in <pemfile> -text -noout
```

Check that the certificates are all readable, and that you have a full chain of certificates available. If you have a root certificate, make sure that it has a “Basic Constraints: CA:TRUE”.

Check the certificate chain in the key.kdb. Is there a complete chain from the certificates up to a self-signed ca certificate? Check that for each certificate which has an Issuer, the issuing certificate exists in the kdb. Check that the final certificate in the chain has an Issuer which is the same as its Subject, that that it has Basic Constraints: CA:TRUE.

```
runmqakm -cert -list -db key.kdb -stashed
```

There is a tool to validate certificates called mqcertck - the documentation is here:

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.1.0/com.ibm.mq.ref.adm.doc/q120895_.htm

There is also improved diagnostics for certificate management documented here:

<https://www.ibm.com/support/pages/node/1081227?lang=en>

There may also be information in the error logs for your queue manager. You can download the logs by selecting the “Logs and Diagnostics” tab in the queue manager details view.

The logs for your local qm will be located in /var/mqm/qmgrs/<qm>/errors/AMQ____.LOG - look for errors such as 575010 from above documentation.

Examine the channels using DISPLAY CHANNEL (name) and check the CERTLBL field is the name of your certificate, and that the SSLCIPH fiels is the right cipher spec.

Examine the queue manager using DISPLAY QMGR ALL and check that the SSLKEYR field has the correct location of the key.kdb file.