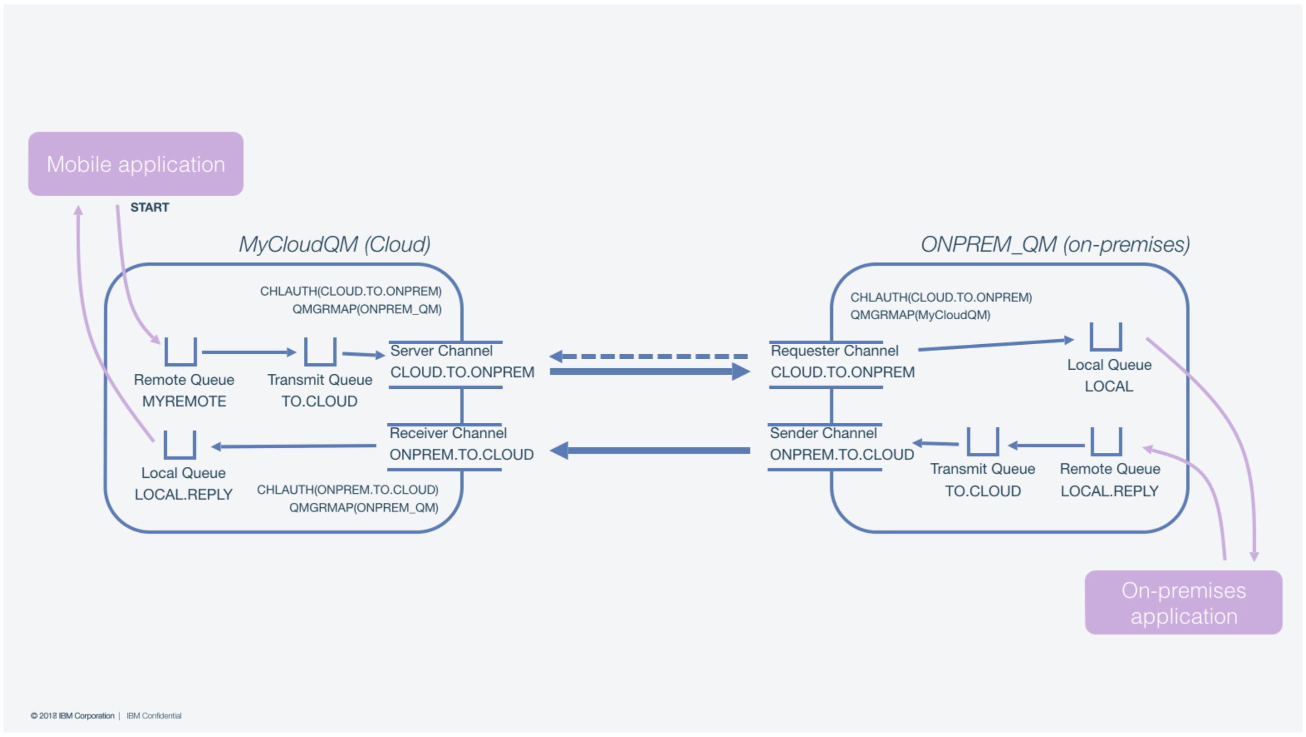
Connecting an IBM MQ On Cloud queue manager directly to an on-premises queue manager.



1. TABLE OF CONTENTS

2. CONNECTING TO AN ON-PREMISES QUEUE MANAGER

2.1. OVERVIEW

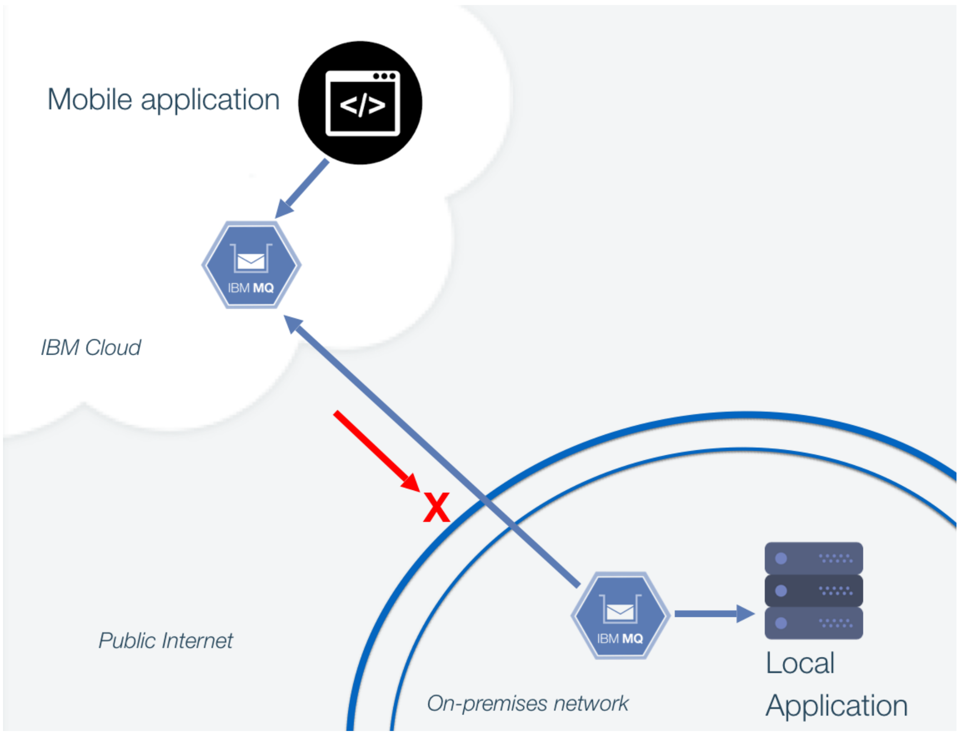
This document describes how to securely transfer messages between an application deployed in IBM Cloud and an application running in an on-premises data centre, using IBM MQ to provide the reliable secure connection between the two deployments.

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

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 for human use” 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.

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. CONFIGURING CHANNELS BETWEEN THE ON-PREMISES AND CLOUD 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.

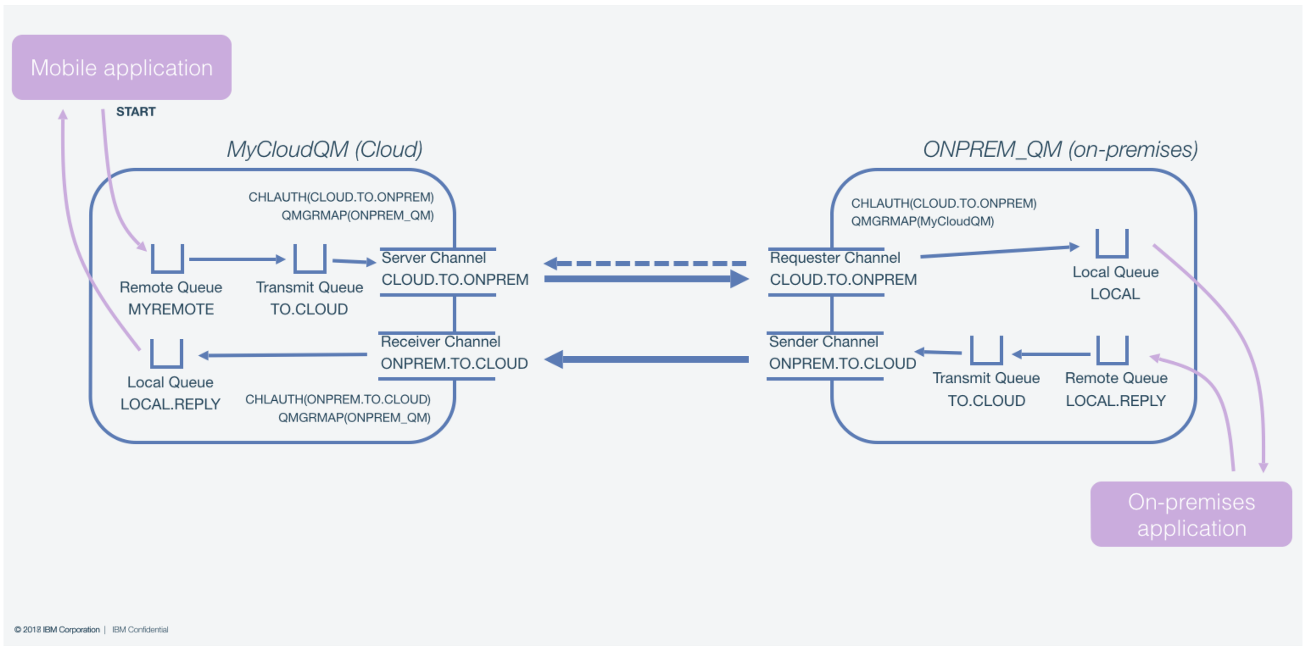
Typically, two sets of channels are configured so as to enable messages to flow in both directions, for example as required when a request message flowing in one direction might result in a reply message flowing back to the originator.

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 (as you will see below). 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.


Mobile application
START
MyCloudQM (Cloud)
CHLAUTH(CLOUD.TO.ONPREM) QMGRMAP(ONPREM_QM)
Remote Queue MYREMOTE
Transmit Queue TO.CLOUD
Server Channel CLOUD.TO.ONPREM
Local Queue LOCAL.REPLY
Receiver Channel ONPREM.TO.CLOUD
CHLAUTH(ONPREM.TO.CLOUD) QMGRMAP(ONPREM_QM)
ONPREM_QM (on-premises)
CHLAUTH(CLOUD.TO.ONPREM) QMGRMAP(MyCloudQM)
Requester Channel CLOUD.TO.ONPREM
Local Queue LOCAL
Sender Channel ONPREM.TO.CLOUD
Transmit Queue TO.CLOUD
Remote Queue LOCAL.REPLY
On-premises application
© 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 because it is on the same machine)
runmqsc ONPREM_QM

// Confirm you are successfully connected by executing a command
PING QMGR

// Define the local queue that the on-premises application would consume from
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(ONPREM.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.ONPREM) 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.ONPREM) 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.ONPREM) USAGE(XMITQ)

// Create a remote queue definition that allows messages to be addressed to the on-premises queue
DEFINE QREMOTE(LOCAL) RNAME(LOCAL) RQMNAME(ONPREM_QM) XMITQ(TO.ONPREM)

// 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.ONPREM) CHLTYPE(SVR) XMITQ(TO.ONPREM) TRPTYPE(TCP)

// Define receiver channel to accept connections from on-prem sender channel
DEFINE CHANNEL(ONPREM.TO.CLOUD) CHLTYPE(RCVR) TRPTYPE(TCP)

// Create the channel authentication record to allow incoming connections from the on-prem QM sender
channel
SET CHLAUTH(ONPREM.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.ONPREM) 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(ONPREM.TO.CLOUD)

// Wait for 15 seconds and check that the sender channel shows STATUS(RUNNING)
DISPLAY CHSTATUS(ONPREM.TO.CLOUD)

// Start the requester channel on the on-premises queue manager
START CHANNEL(CLOUD.TO.ONPREM)

// Wait for 15 seconds and check that the sender channel shows STATUS(RUNNING)
DISPLAY CHSTATUS(CLOUD.TO.ONPREM)
```

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

```
...
AMQ8018I: Start IBM MQ channel accepted.

...
AMQ8417I: Display Channel Status details.
   CHANNEL(ONPREM.TO.CLOUD)                CHLTYPE(SDR)
   CONNAME(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.ONPREM)                CHLTYPE(RQSTR)
   CONNAME(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
```

The following paragraphs explain how to log into the MQ administration console using your MQ username and platform API key .

In the IBM Cloud service console, with the list of queue managers showing, click the row containing your queue manager in order to open the details about that queue manager;

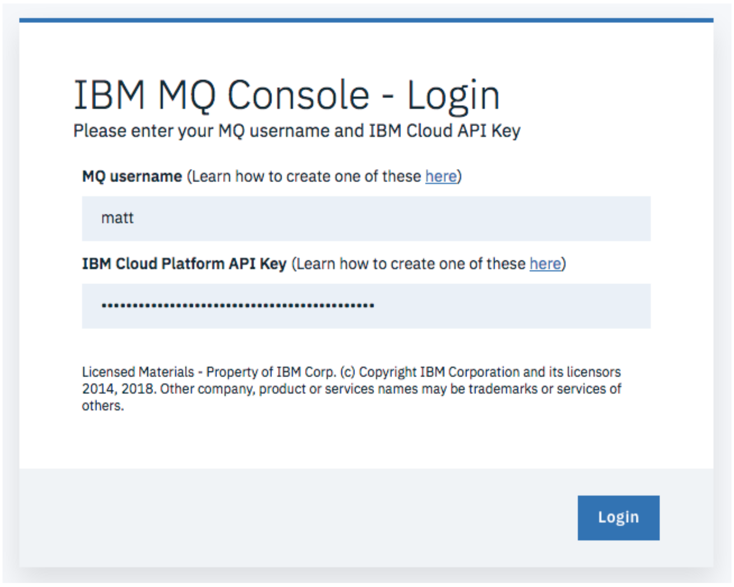
On the Configuration tab you can see details about the queue manager such as its name (MyCloudQM), hostname and port, MQ version, size (Trial) etc. Click the Administration tab for your queue manager;

Behind the scenes your user will be added as an administrator for queue managers in this service instance and an MQ username has been automatically generated for your user. You can modify your MQ username at any time through the "User permissions" tab if you wish to do so.

You will also see that you will need an IBM Cloud platform API key associated with your username in order to log in to the MQ administrative tools such as MQ Console, MQ Explorer and runmqsc. If you do not already have an IBM Cloud platform API key defined for your user then you will be invited to create one, or else you can reset the existing one you created through the MQ service if you have forgotten it.

Show and/or download the API key (platformApiKey.json) and store it somewhere safe then click Close. This platform API key is a password associated with your user in IBM Cloud, and forms the second part of the credentials needed to access the MQ Explorer.

Click the 'Launch MQ Console' button at the bottom of the 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.



IBM MQ Console - Login

Please enter your MQ username and IBM Cloud API Key

MQ username (Learn how to create one of these here)

matt

IBM Cloud Platform API Key (Learn how to create one of these here)

••

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.

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 “amqsput” 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 “amqsput” 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).

```
amqsput 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. APPLYING TLS SECURITY TO THE QUEUE MANAGER 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 (and vice versa).

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.

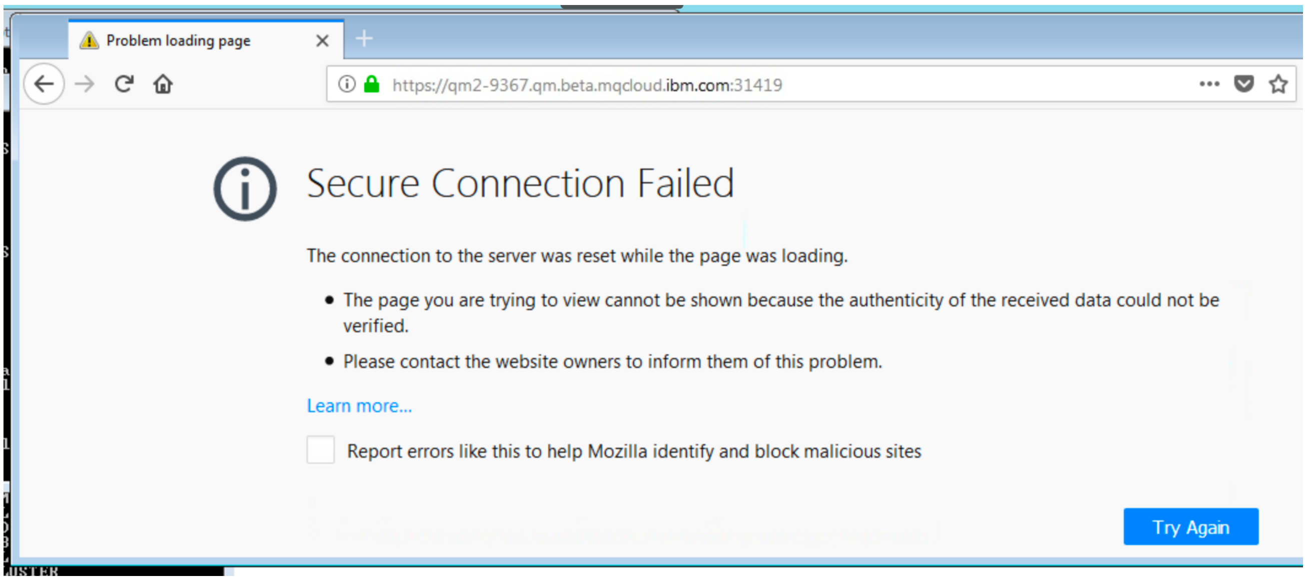
The second part of the configuration process is to set the cipher specification on the channels so that they are able to agree which cipher to use to establish the TLS session.

The easiest way to obtain a copy of the cloud queue manager's server certificate on a Windows machine is to attempt to connect to the same host/port in a browser and then use the browser to export the certificate that is presented by the queue manager. (Note that on Unix based operating systems this step is typically done using an "open_ssl" command)

Open Firefox and navigate the browser tab to the equivalent URL as this for your queue manager host and TCP listener port. (Note that you are connecting to the MQ listener port here – not the MQ Console endpoint)

https://mycloudqm-1111.qm.us-south.mqcloud.ibm.com:34567

The queue manager isn't expecting to receive HTTPS requests on this port, so after a minute or so you will get an error "Secure Connection Failed" as shown in the following screenshot. Importantly you can also see the green padlock icon next to the address bar indicating that the browser received a copy of the queue manager's server certificate (and that it trusts that certificate because it was issued by a trusted CA such as DigiCert and matches the specified hostname);


Problem loading page
https://qm2-9367.qm.beta.mqcloud.ibm.com:31419

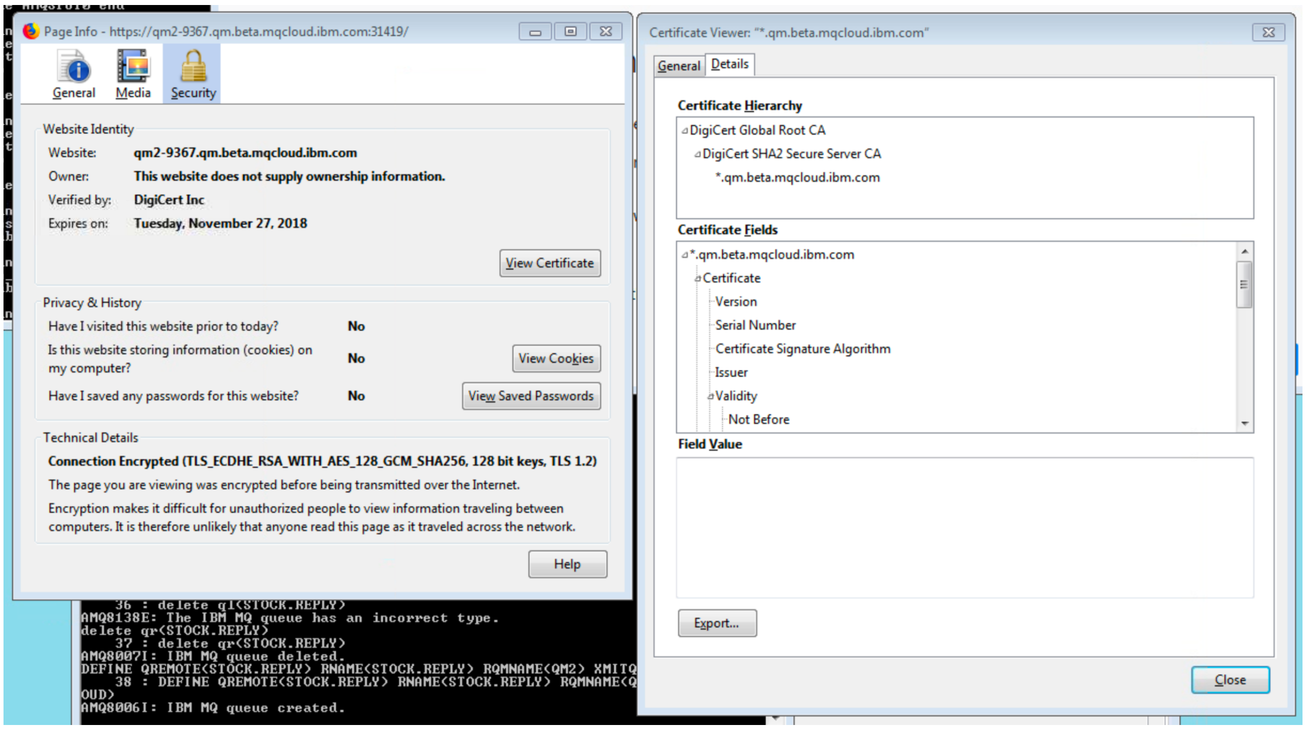
Secure Connection Failed

The connection to the server was reset while the page was loading.

- The page you are trying to view cannot be shown because the authenticity of the received data could not be verified.
- Please contact the website owners to inform them of this problem.

Learn more...
Report errors like this to help Mozilla identify and block malicious sites
Try Again

Click on that green padlock icon, then on the arrow next to the hostname, then on “More Information”. In the resulting dialog box go to the Security tab and click “View Certificate”, then click the Details tab and click the Export button as shown below;



Website Identity

Website:	qm2-9367.qm.beta.mqcloud.ibm.com
Owner:	This website does not supply ownership information.
Verified by:	DigiCert Inc
Expires on:	Tuesday, November 27, 2018

Privacy & History

Have I visited this website prior to today?	No	
Is this website storing information (cookies) on my computer?	No	View Cookies
Have I saved any passwords for this website?	No	View Saved Passwords

Technical Details

Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)

The page you are viewing was encrypted before being transmitted over the Internet.

Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

```
    36 : delete ql(STOCK.REPLY)
AMQ8138E: The IBM MQ queue has an incorrect type.
delete qr(STOCK.REPLY)
    37 : delete qr(STOCK.REPLY)
AMQ8007I: IBM MQ queue deleted.
DEFINE QREMOTE(STOCK.REPLY) RNAME(STOCK.REPLY) RQMNAME(QM2) XMITQ
    38 : DEFINE QREMOTE(STOCK.REPLY) RNAME(STOCK.REPLY) RQMNAME(Q
OUD)
AMQ8006I: IBM MQ queue created.
```

Certificate Hierarchy

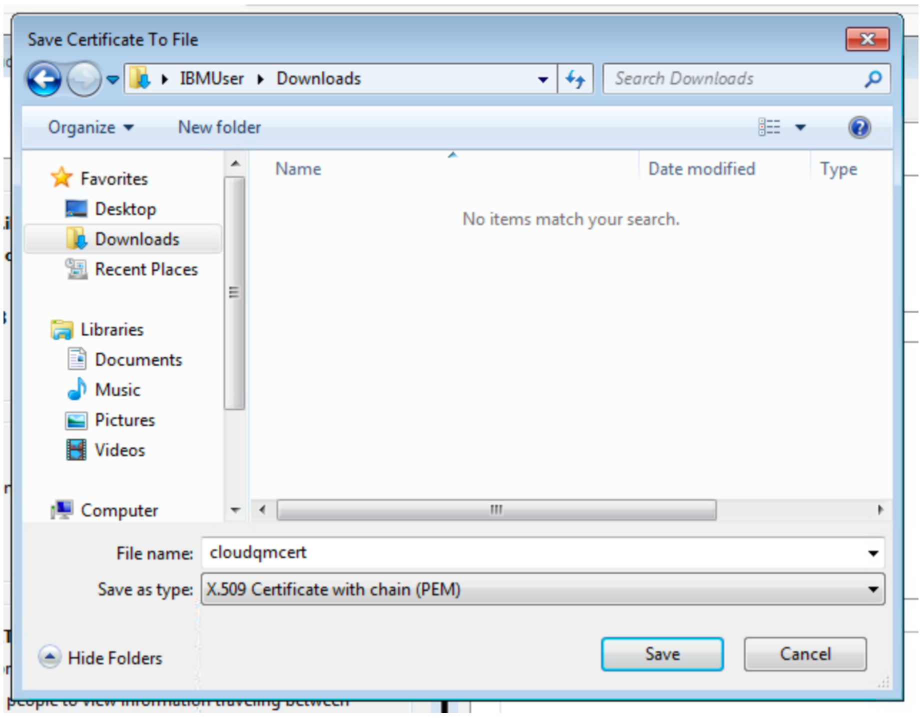
- DigiCert Global Root CA
  - DigiCert SHA2 Secure Server CA
    - *.qm.beta.mqcloud.ibm.com

Certificate Fields

- *.qm.beta.mqcloud.ibm.com
  - Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
    - Issuer
    - Validity
      - Not Before

Field Value

Pick a name such as “cloudqmcert”, select “X.509 Certificate **with chain** (PEM)” as the file type and save the certificate into a location you will remember, such as “C:\Users\IBMUser\Downloads\”



No items match your search.

File name: cloudqmcert

Save as type: X.509 Certificate with chain (PEM)

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 to the `\ssl` directory for your queue manager;

```
cd C:\ProgramData\IBM\MQ\qmgrs\ONPREM_QM\ssl
```

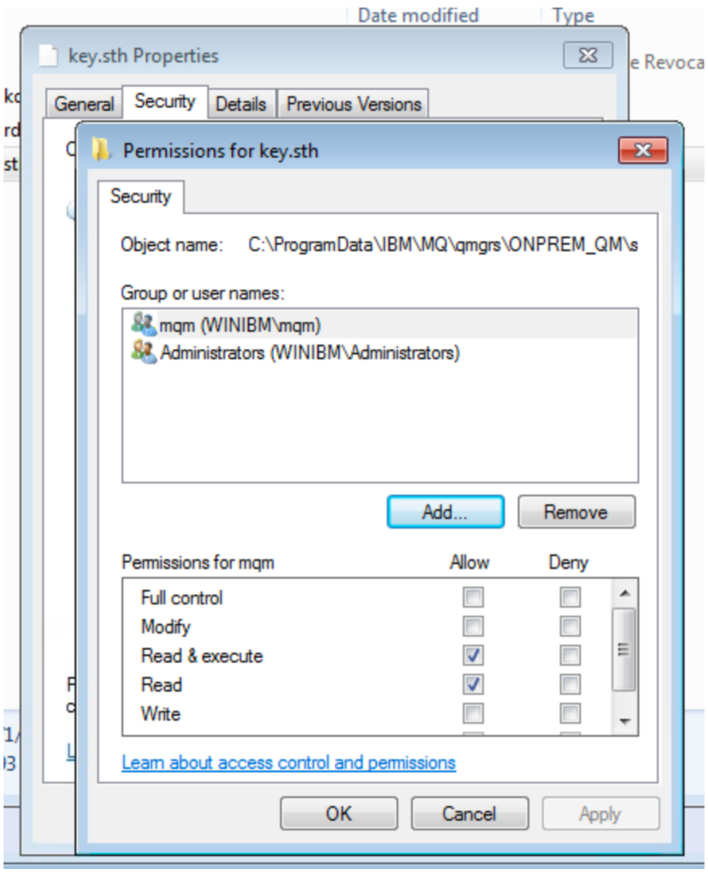
Run the "dir" command and confirm that there are not currently any files in that directory. You will now use the "runmqakm" command to create a key store and add the downloaded certificate to it. Be sure to update the second command to reflect the correct path to your downloaded certificate if you saved it somewhere else, and note that the second command must be executed as a single line.

```
runmqakm -keydb -create -db key.kdb -pw mySecretPassword -stash
runmqakm -cert -add -db key.kdb -stashed -label CloudQM -file C:
\Users\IBMUser\Downloads\cloudqmcert.crt

// Verify that your certificate is present by running this command
runmqakm -cert -list -db key.kdb -stashed
```

Now 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, so that the resulting configuration looks as follows;


Permissions for key.sth

Object name: C:\ProgramData\IBM\MQ\qmgrs\ONPREM_QM\s

Group or user names:
- mqm (WINIBM\mqm)
- Administrators (WINIBM\Administrators)

Permissions for mqm	Allow	Deny
Full control	☐	☐
Modify	☐	☐
Read & execute	✓	☐
Read	✓	☐
Write	☐	☐

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. The list of supported cipher specs is [described here], but for the purposes of this example you can use ECDHE_RSA_AES_128_CBC_SHA256. You will also indicate that the channels which accept incoming connections should not require the other end of the connection to provide a client certificate to identify itself by setting SSLCAUTH to OPTIONAL.

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

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

// Apply the SSLCIPH and SSLCAUTH settings to the existing channels
ALTER CHL('ONPREM.TO.CLOUD') CHLTYPE(RCVR) SSLCIPH(ECDHE_RSA_AES_128_CBC_SHA256) SSLCAUTH(OPTIONAL)
ALTER CHL('CLOUD.TO.ONPREM') CHLTYPE(SVR) SSLCIPH(ECDHE_RSA_AES_128_CBC_SHA256) SSLCAUTH(OPTIONAL)
```

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('ONPREM.TO.CLOUD') CHLTYPE(SDR) SSLCIPH(ECDHE_RSA_AES_128_CBC_SHA256)
ALTER CHL('CLOUD.TO.ONPREM') CHLTYPE(RQSTR) SSLCIPH(ECDHE_RSA_AES_128_CBC_SHA256) SSLCAUTH(OPTIONAL)
```

You have now finished configuring the channels for TLS and you can confirm that messages can be successfully sent through the TLS-encrypted channels.

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.

```
// 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(ONPREM.TO.CLOUD)
START CHANNEL(CLOUD.TO.ONPREM)

// 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(ONPREM.TO.CLOUD)
DISPLAY CHSTATUS(CLOUD.TO.ONPREM)
```

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.