Using SSL TLS to connect an IBM MQ 9.3 queue manager in Windows with another one in Linux, using self-signed certificates

https://www.ibm.com/support/pages/node/7121151

Date last updated: 15-Mar-2024

Angel Rivera
IBM MQ Support
https://www.ibm.com/products/mq/support
Find all the support you need for IBM MQ

+++ Objective

The objective of this document is to provide step-by-step details to:
- connect an MQ 9.3 queue manager in Windows
- to a single-instance queue manager running MQ 9.3 in Linux,
- using self-signed certificates (ok for Test queue managers, but not for Production, because no Certification Authority is involved, no root certificates).
- 2-way authentication (each queue manager authenticates the other queue manager).
- using the runmqckm (iKeycmd) from the command line, that is, not using the strmqikm (iKeyman) GUI.

The main scenario is to put a message in the remote queue definition in the Windows queue manager QMFINTLS and then the TLS enabled "sender" channel will transfer the message to the TLS enabled "receiver" channel in the Linux queue manager QMSTMTLS.

For illustration purposes the following protocol will be used:
TLS 1.3 compliant: TLS_AES_128_GCM_SHA256

If you want to use a TLS 1.2 compliant protocol, a typical one is:
TLS 1.2 compliant: TLS_RSA_WITH_AES_128_CBC_SHA256

In this document, these terms are used interchangeably: SSL and TLS.
TLS is the successor to SSL, but the term “SSL” is used for historical reasons and the MQ tooling refers to SSL, even though it applies also to TLS.

The configuration for SSL requires many steps and at different locations.
One common source of confusion when doing the setup for SSL/TLS is:
In which side of the connection a certain step/command needs to be taken?
Is “Step A” done in host-1 or in host-2?
This tutorial tries to be very explicit in this respect, and hopefully the confusion could be avoided.

This tutorial also provides an “extreme summary” with all the commands, without explanation. The idea is that you can copy these commands into a text editor, edit them to suit your needs and then copy/paste into the command prompts for Windows and Linux.

+ Why using "self-signed certificates"?

Because for novice users this is the easiest approach: it requires the least amount of effort and steps.

In which conditions would be ok to use them?
Self-signed TLS certificates are suitable for personal use or for applications that are used internally within an organization.
They are usually used for testing environments or low-risk internal networks only.

In which conditions would NOT be ok to use them?
Self-signed certificates are NOT suitable for software used by external users or high-risk or Production environments because the certificates cannot be verified with a Certification Authority (CA).

+ Auxiliary article for troubleshooting

During the preparation of this article, some errors were encountered and eventually were addressed. The following document was created to capture the symptoms and the actions that were taken to fix the problems.

https://www.ibm.com/support/pages/6955529
Exploring some troubleshooting scenarios for SSL/TLS in IBM MQ

++ Requisites

You need to have 2 queue managers that have non-SSL sender-receiver channels between them and that you can put a message in the remote queue definition in one queue manager, then the Sender channel will get the message from the corresponding Transmission queue and send it to the Receiver channel and the receiving queue manager puts the message in the destination queue.

The following tutorial can be helpful:
https://www.ibm.com/support/pages/node/152555
MQ Commands to setup two way communication between two queue managers on Sender and Receiver channels

++ Recommendation: Divide and conquer!

Many new MQ administrators try to accomplish all the configuration tasks (non SSL and SSL) at the same time. This would be fine if there were few steps and straightforward.
But there are many steps and it is very frustrating that after you do the configuration then your test fails, you get vague error messages and you do not know how to begin the troubleshooting.

Phase 1: Start without SSL/TLS - ensure that you configure and test the sender-receiver channels and all the necessary objects and do NOT use SSL at this stage.
It is extremely important that you debug any connectivity / configuration problems that are NOT related to SSL, such as missing remote queue definition, transmission queue, etc.

Phase 2: AFTER the non SSL test is successful, then you can add the SSL stuff and modify the channel definitions to use SSL, and restart the channels.

For Phase 1, the following tutorial does not use SSL, but it will be used to create all the necessary objects for sending a message from one queue manager to another.

++ Clarification of “extract”/“add” versus “export”/“import”

SSL uses public/private keys to provide a flexible encryption scheme that can be set-up at the time of the secure transaction.
When a certificate is created, it contains both the public and private keys.

The "extract" and "add" functions deal with ONLY the public keys.
That is, the "extract" gets the public key of a certificate from a database and the "add" puts the public key into a database.
The "extract" does NOT get the private key.
No passwords are required because the private key is not obtained.

The "export" and "import" functions deal with BOTH the public and private keys for a certificate.
Passwords are required due to the private key.

+++ Extreme summary

MQ Windows queue manager:
Hostname: finestra1
Queue manager name: QMFINTLS Port: 1423
User: Administrator

MQ Linux queue manager:
Hostname: stmichel1
Queue manager name: QMSTMTLS Port: 1419
User: mqm

+ Overall sequence of steps

Step 1: Queue Manager (Windows): Create SSL server key database (CMS)
Step 2: Queue Manager (Windows): Create certificate
Step 3: Queue Manager (Windows): Extract the public SSL server certificate
Step 4: Queue Manager (Windows): Copy Windows certificate to the SSL server side in Linux
Step 5: Queue Manager (Linux): Create SSL server key database
Step 6: Queue Manager (Linux): Create certificate
Step 7: Queue Manager (Linux): Extract the public SSL server certificate
Step 8: Queue Manager (Linux): Copy Linux certificate to the SSL server side in Windows
Step 9: Queue Manager (Linux): Add the Windows certificate to Linux key database
Step 10: Queue Manager (Linux): Run MQSC commands for SSL sender and receiver
Step 11: Queue Manager (Windows): Add the Linux certificate to the Windows key database
Step 12: Queue Manager (Windows): Run MQSC commands for SSL sender and receiver, refresh SSL and start channels
Step 13: Queue Manager (Linux): Refresh SSL and start sender channels
Step 14: Queue Manager (Windows): Start Sender channel
Step 15: Test the channels and remote queue definitions

+ Usage notes

It is recommended that you copy all the commands in this section and paste them in a plain text editor such as Notepad, which uses monospace font and no wrap around the lines.

Keep in mind that some commands are very long and even though are shown in several logical lines in this document, they need to be performed in one single long line.

Then make global replacements for the items such as key database, passwords, user name, queue manager, etc.

Finally, you can copy from Notepad each command and execute it in a command prompt for Window or for the remote host in Linux.

Step 1: Queue Manager (Windows): Create SSL server key database (CMS)

Issue the command to specify the MQ environment variables:
C:\> setmqenv -n Installation1

If necessary, you may need to specify the full path:
C:\> "C:\Program Files\IBM\MQ\bin\setmqenv" -n Installation1

runmqckm -keydb -create -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -pw winpass -type cms -stash

Step 2: Queue Manager (Windows): Create certificate

Note:
After stashing the kdb password, it is better to use the -stashed command line option than to specify the -pw option.

Option 1: Specifying -stashed
runmqckm -cert -create -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed -label ibmwebspheremqqmfintls -dn "CN=QMFINTLS,O=IBM,C=USA" -size 2048

Option 2: Specifying -pw passwordPhrase
runmqckm -cert -create -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -pw winpass -label ibmwebspheremqqmfintls -dn "CN=QMFINTLS,O=IBM,C=USA" -size 2048

The rest of the commands use -stashed

runmqckm -cert -list -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed

runmqckm -cert -details -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed -label ibmwebspheremqqmfintls

Step 3: Queue Manager (Windows): Extract the public SSL server certificate

runmqakm -cert -extract -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed -label ibmwebspheremqqmfintls -target QMFINTLS.crt -format ascii

Step 4: Queue Manager (Windows): Copy Windows certificate to the SSL server side in Linux

Copy/transfer the public/signer SSL certificate administrator.crt in ASCII mode from the Windows host to the Linux host.

In Linux, ensure that the file permissions and ownership are as follows (using example in which the uploaded crt from one host to another is located in /tmp)
ls -l /tmp/QMFINTLS.crt
-rw-r--r-- 1 mqm mqm 1126 Apr 10 08:07 /tmp/QMFINTLS.crt

Step 5: Queue Manager (Linux): Create SSL server key database

As user mqm, setup the environment variables for MQ:
. /opt/mqm/bin/setmqenv -n Installation1

Go into the directory:
cd /var/mqm/qmgrs/QMSTMTLS/ssl

Create the key database:
runmqckm -keydb -create -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -pw linuxpass -type cms -stash

Step 6: Queue Manager (Linux): Create certificate

runmqckm -cert -create -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb"
-stashed -label ibmwebspheremqqmstmtls -dn "CN=QMSTMTLS,O=IBM,C=USA"
-size 2048

List newly created SSL certificate in Windows
runmqckm -cert -list -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed

List the details of the certificate:
runmqckm -cert -details -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed -label ibmwebspheremqqmstmtls

Step 7: Queue Manager (Linux): Extract the public SSL server certificate

runmqckm -cert -extract -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed -label ibmwebspheremqqmstmtls -target ibmwebspheremqqmstmtls.crt -format ascii

Step 8: Queue Manager (Linux): Copy Linux certificate to the SSL server side in Windows
Copy/transfer the public/signer SSL certificate QMSTMTLS.crt in ASCII mode from the Linux host to the Windows host.

Step 9: Queue Manager (Linux): Add the Windows certificate to Linux key database

runmqckm -cert -add -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed -label ibmwebspheremqqmfintls -file QMFINTLS.crt -format ascii

runmqckm -cert -list -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed

Step 10: Queue Manager (Linux): Run MQSC commands for SSL sender and receiver

```
runmqsc QMSTMTLS
```

Just in case, let's stop the sender channel

```
STOP  CHANNEL(QMSTMTLS.QMFINTLS) STATUS(INACTIVE)
```

Now, it is time to specify the cipher:

```
ALTER CHANNEL(QMSTMTLS.QMFINTLS) CHLTYPE(SDR) SSLCIPH(TLS_AES_128_GCM_SHA256)

ALTER CHANNEL(QMFINTLS.QMSTMTLS) CHLTYPE(RCVR) SSLCIPH(TLS_AES_128_GCM_SHA256)

At this time, do not issue REFRESH yet.
Let's delay it until we have done the changes in Windows.
Do not exit runmqsc yet.
```

Step 11: Queue Manager (Windows): Add the Linux certificate to the Windows key database

```
runmqckm -cert  -add -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed -
label ibmwebspheremqqmstmtls -file ibmwebspheremqqmstmtls.crt  -format ascii

List the certificates
runmqckm -cert  -list  -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed
```

Step 12: Queue Manager (Windows): Run MQSC commands for SSL sender and receiver

```
runmqsc QMFINTLS
```

Just in case, let's stop the sender channel

```
STOP  CHANNEL(QMFINTLS.QMSTMTLS) STATUS(INACTIVE)
```

It is time to specify the cipher:

```
ALTER CHANNEL(QMFINTLS.QMSTMTLS) CHLTYPE(SDR) SSLCIPH(TLS_AES_128_GCM_SHA256)

ALTER CHANNEL(QMSTMTLS.QMFINTLS) CHLTYPE(RCVR) SSLCIPH(TLS_AES_128_GCM_SHA256)
```

Refresh the security and start the sender channel:

```
REFRESH SECURITY TYPE(SSL)
```

Let's delay further action until we have done the REFRESH security in Linux.

Do not exit runmqsc yet.

Step 13: Queue Manager (Linux): Refresh SSL and start channels

From the runmqsc QMSTMTLS for the Linux queue manager …

```
REFRESH SECURITY TYPE(SSL)

START CHANNEL(QMSTMTLS.QMFINTLS)
```

Check the status of both channels:

```
display CHSTATUS(QMSTMTLS.QMFINTLS)

display CHSTATUS(QMSTMTLS.QMFINTLS)
```

Step 14: Queue Manager (Windows): Start Sender channel

```
START CHANNEL(QMFINTLS.QMSTMTLS)
```

Check the status of both channels:

```
display CHSTATUS(QMSTMTLS.QMFINTLS)

display chstatus(QMFINTLS.QMSTMTLS)
```

Step 15: Test the channels and remote queue definitions

Put a message in the remote queue definition in one queue manager and verify that the message arrives to the destination queue in the other queue manager.

Miscellaneous:

If you need to delete the certificate, you can reuse the command that shows the “details” and replace “details” with “delete”:

For example, you can use as the base:

```
runmqckm -cert -details -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed
-label ibmwebspheremqqmfintls
```

Then replace “details” for “delete”:

```
runmqckm -cert -delete  -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed
-label ibmwebspheremqqmfintls
```

+++ Configuration

a) Windows:

MQ 9.3.0.15 LTS running in Windows
Hostname: finestra1
Queue manager name: QMFINTLS
Port: 1423
User: Administrator

MQ administrative command to create it:
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE -p 1423 QMFINTLS

In this technote, the default location for the MQ Data is used:
MQ_DATA_PATH=C:\ProgramData\IBM\MQ\

b) Linux

Single-instance queue manager running MQ 9.3.0.15 LTS in Linux RHEL Intel 64-bit
Hostname: stmichel1
Queue manager name: QMSTMTLS
Port: 1419
User: mqm

MQ administrative command to create it:
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE -p 1419 QMSTMTLS

+ Hint to disable channel authentication records and requiring passwords when using testing systems (not recommended for Production)

For testing purposes, disable connauth and chlauth to make things easier regarding channel authentication records and requiring passwords during connect.

Then do a 'refresh security type(connauth)'

runmqsc QMSTMTLS

alter qmgr chlauth(disabled) connauth('')
AMQ8005I: IBM MQ queue manager changed.

refresh security type(connauth)
AMQ8560I: IBM MQ security cache refreshed.

end

+ Note about SSL stanza in qm.ini of new queue managers in QM 9.2 or later

When creating new queue managers under MQ 9.2 or later, the qm.ini configuration file will have the default SSL stanza:

```
SSL:
   AllowTLSV13=Yes
   MinimumRSAKeySize=1
```

The attribute “AllowTLSV13=Yes” enables TLS 1.3 which allows the queue manager to use the TLS 1.3 CipherSpecs.
Note: The queue manager will be able to still use TLS 1.2 and other CipherSpecs!

The location for the SSL related files will be in a sub-directory "ssl", which is automatically created by crtmqm.

Linux:
```
  /var/mqm/qmgrs/QMGRNAME/ssl
```

Windows:
```
  C:\ProgramData\IBM\MQ\qmgrs\QMGRNAME\ssl
```

++ Step 1: Queue Manager (Windows): Create SSL server key database (CMS)

Issue the command to specify the MQ environment variables:
C:\> **setmqenv -n Installation1**

If necessary, you may need to specify the full path:
C:\> **"C:\Program Files\IBM\MQ\bin\setmqenv" -n Installation1**

Go to the sub-directory "ssl" of the queue manager, which is created automatically by crtmqm.

cd C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl

You can name the key repository whatever you would like, however, **it MUST HAVE a .kdb extension.**

Even though the instances of some commands are shown in 2 lines, it is only a single long line.
You must ensure that if you copy/paste the command, you use one single long line:

runmqckm -keydb -create -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -pw winpass -type cms -stash

After stashing the password (via the -stash option), for later commands is better to use the **-stashed** option, rathen than to specify again the -pw option.

You can add the -expire option to allow the key repository password to expire after a number of days.

Notice that 3 new files are created:

```
C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl> dir
02/16/2024  12:34 PM               144 key.kdb
02/16/2024  12:34 PM                80 key.rdb
02/16/2024  12:34 PM               193 key.sth
```

++ Step 2: Queue Manager (Windows): Create certificate

+ Create digital certificate.

Starting with MQ 8.0, the queue manager's certificate does not need to be as in MQ 7.x:
  ibmwebspheremq + qmgrname
… But we still recommend using that convention.

If the queue manager's label name is set to something else, the CERTLABL property of the queue manager must be set to the correct certificate labelname.
For more details see:
https://www.ibm.com/docs/en/ibm-mq/9.3?topic=attributes-channel-mqsc-keywords-c
IBM MQ / 9.3
Channel attributes for MQSC keywords (C)
CERTLABL (Certificate label)

If you follow the above convention, then you MUST USE the queue manager name in LOWERCASE for the label!
  -label ibmwebspheremqqmfintls

However, it is a common convention that for the CN, use the uppercase name.
  -dn "CN=QMFINTLS

runmqckm -cert -create -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed -label ibmwebspheremqqmfintls -dn "CN=QMFINTLS,O=IBM,C=USA" -size 2048

Where:
-label is the label name:
      Ibmwebspheremqqmfintls
   It is required to be the concatenation of:
      ibmwebspheremq + queue manager name in lower case
   In this case:
      ibmwebspheremq + qmfintls
-dn is the "Distinguished Name" in uppercase QMFINTLS
-size The recommended size is 2048 bits. The certificates with a size of 1024 are no longer recommended.

The key.db is updated (notice that the size is bigger, because it has now 1 certificate):

02/16/2024 12:42 PM **5,144 key.kdb**
02/16/2024 12:42 PM 80 key.rdb
02/16/2024 12:34 PM 193 key.sth

+ List newly created SSL certificate in Windows

runmqckm -cert -list -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed

```
Certificates in database C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb:
   ibmwebspheremqqmfintls
```

+ List the details of the certificate:

runmqckm -cert -details -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed -label ibmwebspheremqqmfintls

<begin certificate – ignore this line>

```
Label: ibmwebspheremqqmfintls
Key Size: 2048
Version: X509 V3
Serial Number: 65 CF C8 C6
Issued by: CN=QMFINTLS, O=IBM, C=USA
Subject: CN=QMFINTLS, O=IBM, C=USA
Valid: From: Friday, February 16, 2024 12:42:46 PM PST To: Saturday, February 15, 2025 12:42:46 PM
PST
Fingerprint:
    SHA1: 3C:1D:54:BE:D3:97:07:45:BE:E3:64:40:42:26:79:52:30:B8:22:00
    SHA256:
20:E0:A4:70:BB:C2:74:3F:EB:B2:E3:E0:0E:F5:60:E8:42:17:FF:DF:1F:B3:3D:C1:BC:A4:8B:B6:DF:5E:3B:20
    HPKP: NP9R9AG4M1im92wBJjYMd1eCrcxZNVGnI5pOu6vOBSc=


Extensions:
  - AuthorityKeyIdentifier: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: fa 27 60 cc c5 65 16 29  4d cb 30 7f 5a 87 7a e9  .....e..M.0.Z.z.
0010: 2e b0 61 ae                                        ..a.
]
]

  - SubjectKeyIdentifier: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: fa 27 60 cc c5 65 16 29  4d cb 30 7f 5a 87 7a e9  .....e..M.0.Z.z.
0010: 2e b0 61 ae                                        ..a.
]
]

Signature Algorithm: SHA256withRSA (1.2.840.113549.1.1.11)
Trust Status: enabled
```

<end certificate – ignore this line>

+ REFERENCE for deleting a certificate:

The following command is just for completeness, you do NOT need to issue it for this tutorial.

If you need to delete the certificate, you can reuse the command that shows the “details” and replace “details” with “delete”:

For example, you can use as the base:
runmqckm -cert -details -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed -label ibmwebspheremqqmfintls

Then replace “details” for “delete”:
runmqckm -cert -delete -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed -label ibmwebspheremqqmfintls

++ Step 3: Queue Manager (Windows): Extract the public SSL server certificate

The "extract" action gets the public key of a certificate from the database, but does NOT extract the private key.

The following command will create a file in the current directory.

C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl>
runmqakm -cert -extract -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed -label ibmwebspheremqqmfintls -target QMFINTLS.crt -format ascii
5724-H72 (C) Copyright IBM Corp. 1994, 2023.

Notice that in the current directory a new file will be created, which is the extracted certificate:

C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl> **dir**
02/16/2024 12:42 PM 5,144 key.kdb
02/16/2024 12:42 PM 80 key.rdb
02/16/2024 12:34 PM 193 key.sth
02/16/2024 12:52 PM 1,162 QMFINTLS.crt

This certificate file looks like this (only showing some lines though):

C:\ProgramData\IBM\MQ\ssl> **type QMFINTLS.crt**

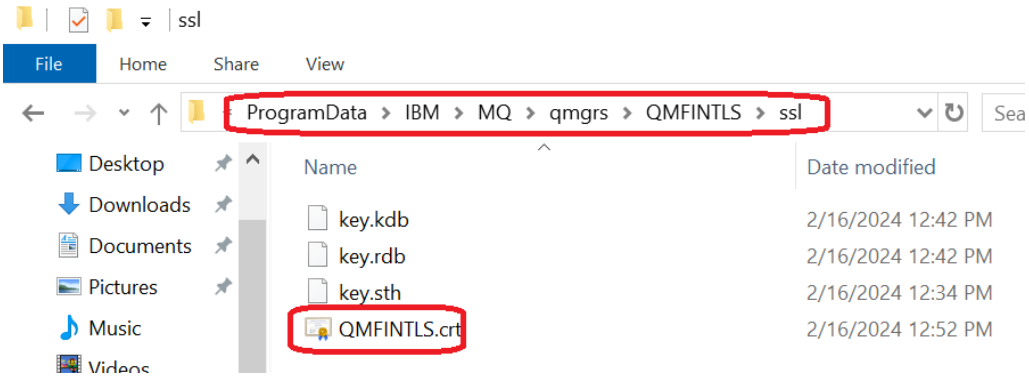
```
-----BEGIN CERTIFICATE-----
MIIDBDCCAeygAwIBAgIEZDQiNzANBgkqhkiG9w0BAQsFADAuMQwwCgYDVQQGEwNV
U0ExDDAKBgNVBAoTA0lCTTEQMA4GA1UEAxMHcW05M3dpbjAeFw0yMzA0MTAxNDUw
MzFaFw0yNDA0MDkxNDUwMzFaMC4xDDAKBgNVBAYTA1VTQTEMMAoGA1UEChMDSUJN
…
NaYmFcTt3dYaaDVVJZKiz/GMbeUk4mlMfZdqP106GTpkgC+N3fypmMgSIm/mZGac
1aHS8NwSQiU=
-----END CERTIFICATE-----
```

+ Using Windows Explorer to browse the extracted certificate

You can open the Windows Explorer and display the files from the "ssl" directory where the certificate *.crt file was extracted.

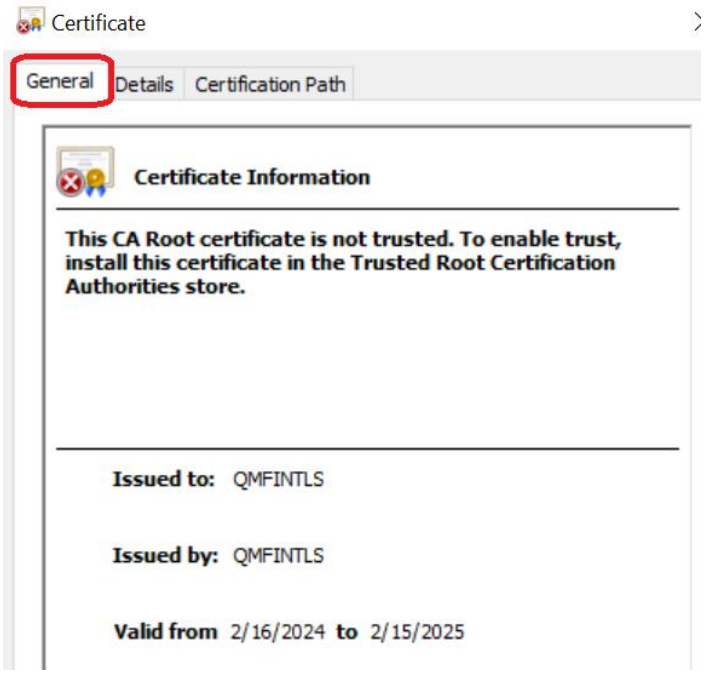
Notice that the suffix ".crt" is recognized.

Double click on the file:
QMFINTLS.crt

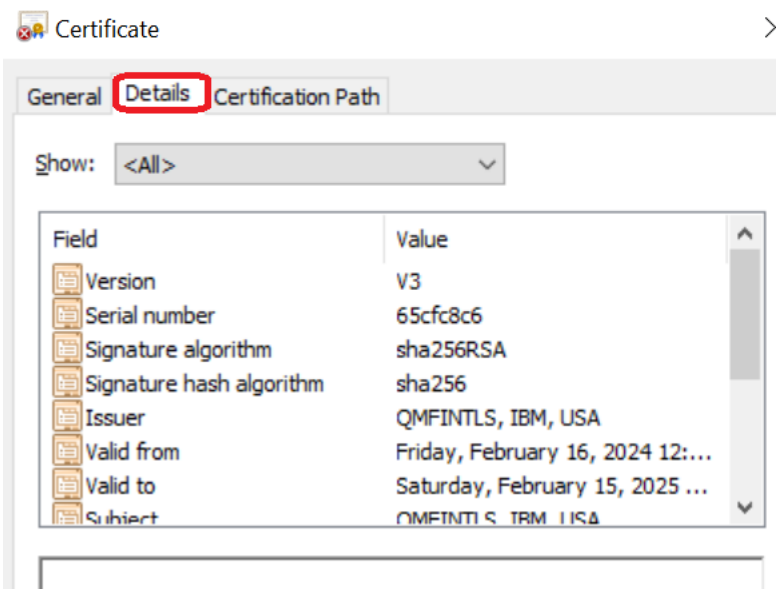


You will see a dialog box with 3 tabs:

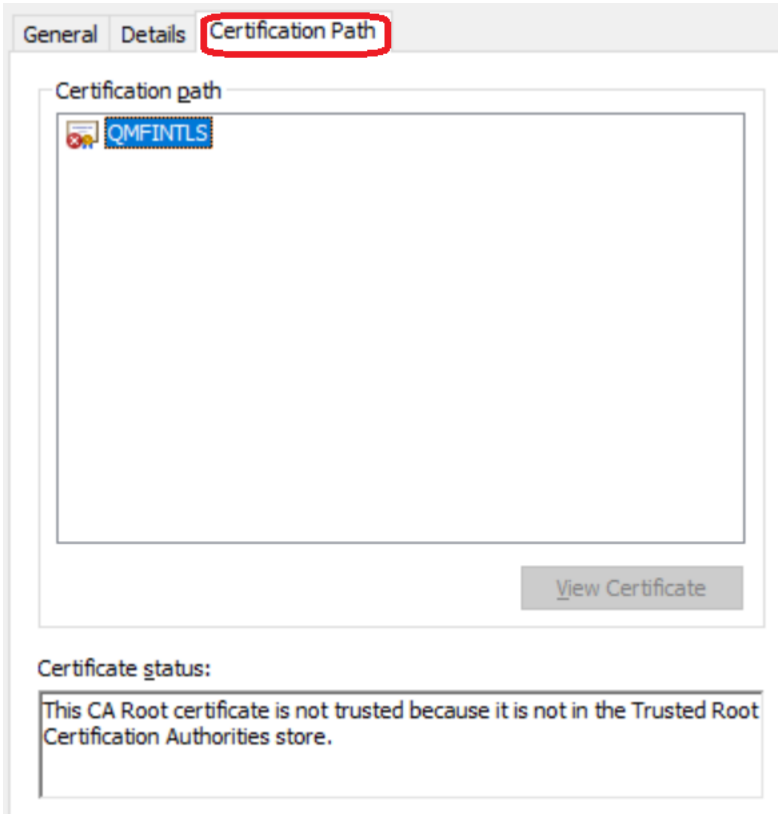
The tab "General" looks like this:



The tab "Details" looks like this:



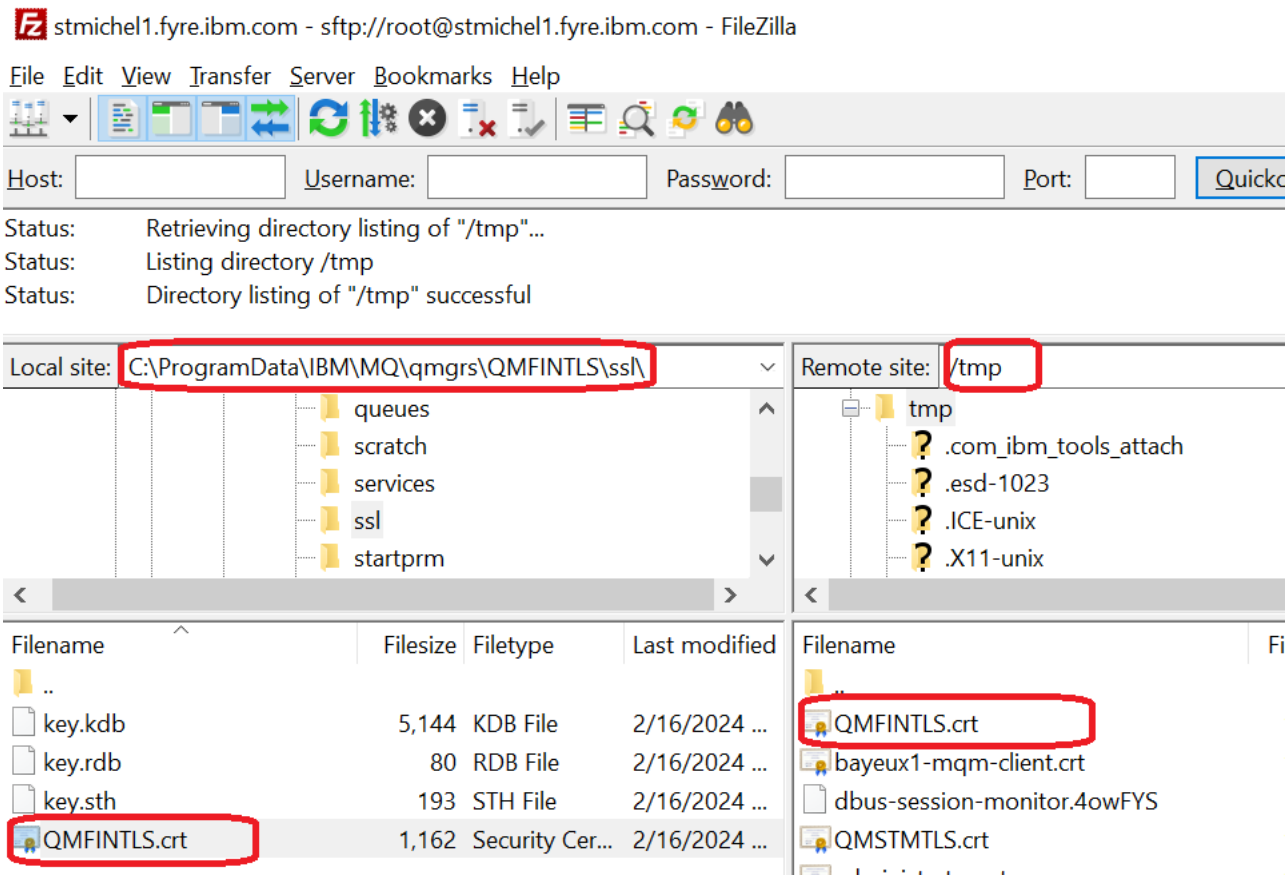
The tab "Certification Path" looks like this:



++ Step 4: Queue Manager (Windows): Copy Windows certificate to the SSL server side in Linux

Copy/transfer the public/signer SSL certificate administrator.crt in **ASCII mode** from the Windows host to the Linux host.

For this tutorial the utility “Filezilla” was used to copy the crt file into the Linux host at: **/tmp/QMFINTLS.crt**



In the Linux host stmichel1, ensure that the uploaded crt file has the ownership for user "mqm" and group "mqm"

ROOT stmichel1.fyre.ibm.com: /tmp
ls -l *.crt
-rw-r--r-- 1 root root 1162 Feb 16 13:49 QMFINTLS.crt

chown mqm:mqm QMFINTLS.crt

ls -l *.crt
-rw-r--r-- 1 mqm mqm 1162 Feb 16 13:49 QMFINTLS.crt

++ Step 5: Queue Manager (Linux): Create SSL server key database

As user mqm, setup the environment variables for MQ:
. /opt/mqm/bin/setmqenv -n Installation1

It is assumed that the queue manager QMSTMTLS has already being created and is using port 1419.

Notice that the subdirectory “ssl” is created by “crtmqm”.

mqm@stmichel1.fyre.ibm.com: /var/mqm/qmgrs/QMSTMTLS
$ **ls -ld ssl**
drwxrws--- 2 mqm mqm 170 Apr 6 12:34 ssl

Go into the directory:
cd /var/mqm/qmgrs/QMSTMTLS/ssl

Create the key database:

runmqckm -keydb -create -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -pw linuxpass -type cms -stash

Notice that 3 new files are created:

mqm@stmichel1.fyre.ibm.com: /var/mqm/qmgrs/QMSTMTLS/ssl
$ **ls -l**
-rw------- 1 mqm mqm 88 Apr 10 08:15 QMSTMTLS.kdb
-rw------- 1 mqm mqm 80 Apr 10 08:15 QMSTMTLS.rdb
-rw------- 1 mqm mqm 193 Apr 10 08:15 QMSTMTLS.sth

++ Step 6: Queue Manager (Linux): Create certificate

+ Create certificate

Starting with MQ 8.0, the queue manager's certificate does not need to be as in MQ 7.x:
ibmwebspheremq + qmgrname
But we still recommend using that convention (Note: qmgrname should be in lowercase)

If the queue manager's label name is set to something else, the CERTLABL property of the queue manager must be set to the correct certificate labelname.
For more details see:
https://www.ibm.com/docs/en/ibm-mq/9.3?topic=attributes-channel-mqsc-keywords-c
IBM MQ / 9.3
Channel attributes for MQSC keywords (C)
CERTLABL (Certificate label)

If you follow the above convention, then you MUST USE the queue manager name in LOWERCASE for the label, such as:
ibmwebspheremqqmstmtls

But for the CN, use the upper case name, such as:
CN=QMSTMTLS

runmqckm -cert -create -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed -label ibmwebspheremqqmstmtls -dn "CN=QMSTMTLS,O=IBM,C=USA" -size 2048

Notice that the key database is bigger, because it contains now 1 certificate:

-rw------- 1 mqm mqm **5088 Mar 14 09:50 QMSTMTLS.kdb**
-rw------- 1 mqm mqm 80 Mar 14 09:50 QMSTMTLS.rdb
-rw------- 1 mqm mqm 193 Mar 14 09:49 QMSTMTLS.sth

+ List newly created SSL certificate in Linux

runmqckm -cert -list -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed

```
Certificates in database /var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb:
   ibmwebspheremqqmstmtls
```

+ List the details of the certificate.

runmqckm -cert -details -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed -label ibmwebspheremqqmstmtls

Excerpt:

```
Label: ibmwebspheremqqmstmtls
Key Size: 2048
Version: X509 V3
Serial Number: 62 2F 72 60
Issued by: CN=QMSTMTLS, O=IBM, C=USA
Subject: CN=QMSTMTLS, O=IBM, C=USA
Valid: From: Monday, March 14, 2022 9:50:40 AM PDT To: Tuesday, March 14, 2023 9:50:40 AM PDT
…
Trust Status: enabled
```

++ Step 7: Queue Manager (Linux): Extract the public SSL server certificate

The "extract" action gets the public key of a certificate from the database, but does NOT extract the private key.

The following command will create a file in the current directory.
In this scenario is:
/var/mqm/qmgrs/QMSTMTLS/ssl

runmqckm -cert -extract -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed -label ibmwebspheremqqmstmtls -target QMSTMTLS.crt -format ascii

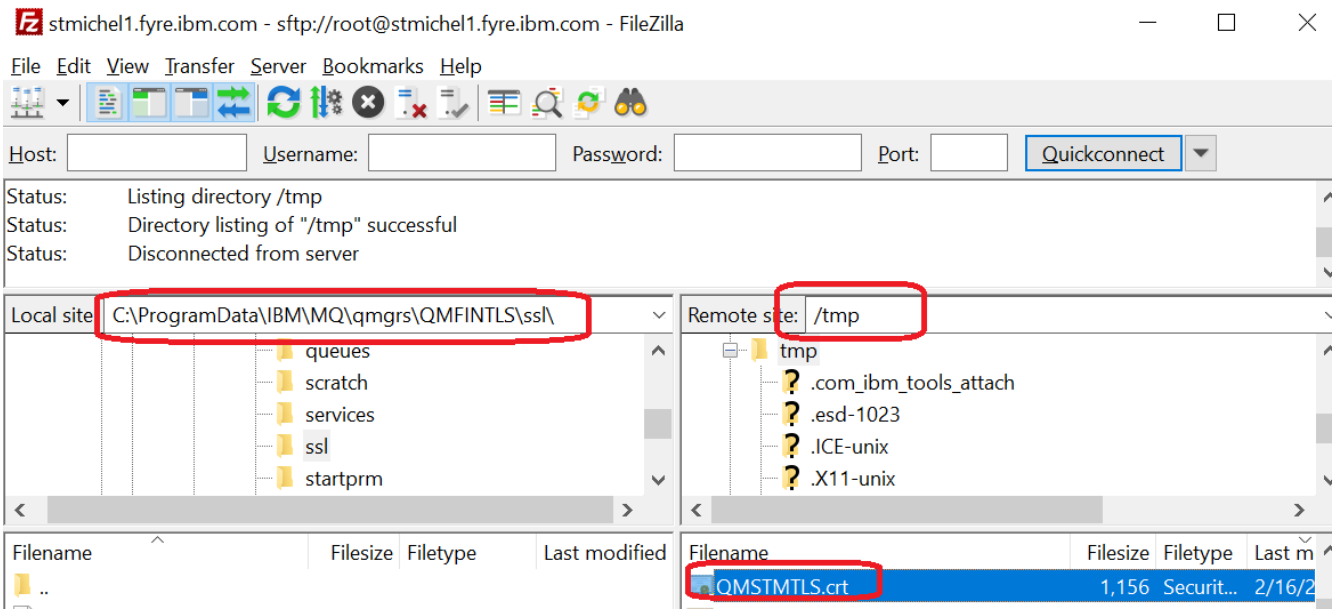
Notice that there is a new file, which has the certificate for the queue manager.

-rw------- 1 mqm mqm 1118 Mar 14 09:53 **QMSTMTLS.crt**
-rw------- 1 mqm mqm 5088 Mar 14 09:50 QMSTMTLS.kdb
-rw------- 1 mqm mqm 80 Mar 14 09:50 QMSTMTLS.rdb
-rw------- 1 mqm mqm 193 Mar 14 09:49 QMSTMTLS.sth

++ Step 8: Queue Manager (Linux): Copy Linux certificate to the SSL server side in Windows

Copy/transfer the public/signer SSL certificate QMSTMTLS.crt in ASCII mode from the Linux host to the Windows host.

In this tutorial, the *.crt from the ssl directory was copied into /tmp and transferred to Windows.



The corresponding *.crt from Windows was uploaded into /tmp and then copied to the ssl directory.

The end result is that the crt files from both queue managers were exchanged.

Linux:
mqm@stmichel1.fyre.ibm.com: /var/mqm/qmgrs/QMSTMTLS/ssl
$ cp /tmp/QMFINTLS.crt .
mqm@stmichel1.fyre.ibm.com: /var/mqm/qmgrs/QMSTMTLS/ssl
(715) ls -l *.crt
-rw-r--r-- 1 mqm mqm 1162 Feb 16 13:59 QMFINTLS.crt
-rw------- 1 mqm mqm 1156 Feb 16 09:39 QMSTMTLS.crt

Windows:
C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl>dir *.crt
02/16/2024 12:52 PM 1,162 QMFINTLS.crt
02/16/2024 02:00 PM 1,156 QMSTMTLS.crt

++ Step 9: Queue Manager (Linux): Add the Windows certificate to Linux key database

+ Add the public/signer certificate from Windows

mqm@stmichel1.fyre.ibm.com: /var/mqm/qmgrs/QMSTMTLS/ssl
$ **runmqckm -cert -add -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed -label ibmwebspheremqqmfintls -file QMFINTLS.crt -format ascii**

Notice that the size of the kdb became larger (it has now 2 certificates)

$ ls -l
mqm@stmichel1.fyre.ibm.com: /var/mqm/qmgrs/QMSTMTLS/ssl
-rw-r--r-- 1 mqm mqm 1162 Feb 16 13:59 QMFINTLS.crt
-rw------- 1 mqm mqm 1156 Feb 16 09:39 QMSTMTLS.crt
-rw------- 1 mqm mqm 20144 Feb 16 14:14 QMSTMTLS.kdb
-rw------- 1 mqm mqm 80 Feb 16 14:14 QMSTMTLS.rdb
-rw------- 1 mqm mqm 193 Feb 16 09:31 QMSTMTLS.sth

+ List the certificates.

$ runmqckm -cert -list -db "/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb" -stashed

```
Certificates in database /var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS.kdb:
   ibmwebspheremqqmstmtls
   ibmwebspheremqqmfintls
```

++ Step 10: Queue Manager (Linux): Run MQSC commands for SSL sender and receiver

Notice that in the attribute SSLKEYR, the SUFFIX .kdb MUST NOT BE SPECIFIED!
The default value is:

```
runmqsc QMSTMTLS

  DISPLAY QMGR SSLKEYR
AMQ8408I: Display Queue Manager details.
   QMNAME(QMSTMTLS)
   SSLKEYR(/var/mqm/qmgrs/QMSTMTLS/ssl/key)
```

But if you created the database with a different name, such as "QMSTMTLS.kdb", then you need to update SSLKEYR:

```
  ALTER QMGR SSLKEYR('/var/mqm/qmgrs/QMSTMTLS/ssl/QMSTMTLS')
```

It was recommended that you should have created and tested a pair of Sender-Receiver channels without security.

```
runmqsc QMSTMTLS

display CHANNEL(QMSTMTLS.QMFINTLS) CONNAME SSLCIPH
AMQ8414I: Display Channel details.
   CHANNEL(QMSTMTLS.QMFINTLS)                CHLTYPE(SDR)
   CONNAME(finestra1.fyre.ibm.com(1423))       SSLCIPH( )

display CHANNEL(QMFINTLS.QMSTMTLS) SSLCIPH
AMQ8414I: Display Channel details.
   CHANNEL(QMFINTLS.QMSTMTLS)                CHLTYPE(RCVR)
   SSLCIPH( )
```

Just in case, let's stop the sender channel

```
STOP  CHANNEL(QMSTMTLS.QMFINTLS) STATUS(INACTIVE)
```

Now, it is time to specify the cipher:

```
ALTER CHANNEL(QMSTMTLS.QMFINTLS) CHLTYPE(SDR) SSLCIPH(TLS_AES_128_GCM_SHA256)

ALTER CHANNEL(QMFINTLS.QMSTMTLS) CHLTYPE(RCVR) SSLCIPH(TLS_AES_128_GCM_SHA256)
```

At this time, do not issue REFRESH yet.

Let's delay it until we have done the changes in Windows.

Do not exit runmqsc yet.

++ Step 11: Queue Manager (Windows): Add the Linux certificate to the Windows key database

+ Add the public/signer certificate from QMSTMTLS

C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl>
runmqckm -cert -add -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed -label ibmwebspheremqqmstmtls -file QMSTMTLS.crt -format ascii

Notice that the file key.kdb is larger (it has now 2 certificates)

C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl>dir
02/16/2024 02:24 PM 10,144 key.kdb
02/16/2024 02:24 PM 80 key.rdb
02/16/2024 12:34 PM 193 key.sth
02/16/2024 12:52 PM 1,162 QMFINTLS.crt
02/16/2024 02:00 PM 1,156 QMSTMTLS.crt

+ List the certificates

runmqckm -cert -list -db "C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb" -stashed

```
Certificates in database C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key.kdb:
   ibmwebspheremqqmfintls
   ibmwebspheremqqmstmtls
```

++ Step 12: Queue Manager (Windows): Run MQSC commands for SSL sender and receiver, refresh SSL and start channels

The default name for the key.kdb and thus the default SSLKEYR attribute for the QMGR is shown below:

```
runmqsc QMFINTLS

display qmgr sslkeyr
AMQ8408I: Display Queue Manager details.
   QMNAME(QMFINTLS)
   SSLKEYR(C:\ProgramData\IBM\MQ\qmgrs\QMFINTLS\ssl\key)
```

It was recommended that you should have created and tested a pair of Sender-Receiver channels without security.

```
display CHANNEL(QMFINTLS.QMSTMTLS) CONNAME SSLCIPH
AMQ8414I: Display Channel details.
   CHANNEL(QMFINTLS.QMSTMTLS)               CHLTYPE(SDR)
   CONNAME(stmichel1.fyre.ibm.com(1419))   SSLCIPH( )

display CHANNEL(QMSTMTLS.QMFINTLS) SSLCIPH
AMQ8414I: Display Channel details.
   CHANNEL(QMSTMTLS.QMFINTLS)               CHLTYPE(RCVR)
   SSLCIPH( )
```

Just in case, let's stop the sender channel

```
STOP  CHANNEL(QMFINTLS.QMSTMTLS) STATUS(INACTIVE)
```

Now, it is time to specify the cipher:

```
ALTER CHANNEL(QMFINTLS.QMSTMTLS) CHLTYPE(SDR) SSLCIPH(TLS_AES_128_GCM_SHA256)

ALTER CHANNEL(QMSTMTLS.QMFINTLS) CHLTYPE(RCVR) SSLCIPH(TLS_AES_128_GCM_SHA256)
```

Now refresh the security:

```
REFRESH SECURITY TYPE(SSL)
```

Let's delay further action until we have done the REFRESH security in Linux.

Do not exit runmqsc yet.

++ Step 13: Queue Manager (Linux): Refresh SSL and start sender channels

From the runmqsc QMSTMTLS for the Linux queue manager …

Just in case, let's stop the sender channel

```
STOP  CHANNEL(QMSTMTLS.QMFINTLS) STATUS(INACTIVE)
```

Then REFRESH security and start the channel

```
REFRESH SECURITY TYPE(SSL)

START CHANNEL(QMSTMTLS.QMFINTLS)
```

Check the status of both channels:

```
display CHSTATUS(QMSTMTLS.QMFINTLS)
AMQ8417I: Display Channel Status details.
   CHANNEL(QMSTMTLS.QMFINTLS)              CHLTYPE(SDR)
   CONNAME(9.211.109.44(1414))             CURRENT
   RQMNAME(QMFINTLS)                       STATUS(RUNNING)
   SUBSTATE(MQGET)                         XMITQ(QMFINTLS)
```

The following is not an error because the SENDER channel in Windows has not been started yet.

```
display chstatus(QMFINTLS.QMSTMTLS)
AMQ8420I: Channel Status not found.
```

++ Step 14: Queue Manager (Windows): Start Sender channel

From the runmqsc QMFINTLS for the Windows queue manager …

```
START CHANNEL(QMFINTLS.QMSTMTLS)

Now check the status:

display chstatus(QMFINTLS.QMSTMTLS)
AMQ8417I: Display Channel Status details.
   CHANNEL(QMFINTLS.QMSTMTLS)              CHLTYPE(SDR)
   CONNAME(9.46.98.124(1419))              CURRENT
   RQMNAME(QMSTMTLS)                       STATUS(RUNNING)
   SUBSTATE(MQGET)                         XMITQ(QMSTMTLS)

display chstatus(QMSTMTLS.QMFINTLS)
AMQ8417I: Display Channel Status details.
   CHANNEL(QMSTMTLS.QMFINTLS)              CHLTYPE(RCVR)
   CONNAME(9.46.98.124)                    CURRENT
   RQMNAME(QMSTMTLS)                       STATUS(RUNNING)
   SUBSTATE(RECEIVE)
```

++ Step 15: Test the channels and remote queue definitions

Put a message in the remote queue definition in one queue manager and verify that the message arrives to the destination queue in the other queue manager.

+++ end +++