Summary of IBM MQ SSL TLS commands to connect clients and queue managers, using self-signed certificates, 2-way authentication

https://www.ibm.com/support/pages/node/7118737

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 ONLY a summary of the IBM MQ SSL TLS commands to connect clients and queue managers, using self-signed certificates, 2-way authentication

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

These are the chapters:

Chapter 1: Using SSL TLS in MQ 9.2 to connect a C-based client in Windows to a queue manager in Linux, using self-signed certificates, 2-way authentication

Chapter 2: Using SSL TLS in MQ 9.3 to connect a JMS client to a queue manager in Linux, using self-signed certificates, 2-way authentication

Chapter 3: Using SSL TLS to connect an IBM MQ 9.3 queue manager in Windows with another one in Linux, using self-signed certificates

+ 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).

+ 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.

+++ Chapter 1: Using SSL TLS in MQ 9.2 to connect a C-based client in Windows to a queue manager in Linux, using self-signed certificates, 2-way authentication

The commands were obtained from the following comprehensive tutorial:

https://www.ibm.com/support/pages/node/6470619
Using SSL TLS in MQ 9.2 to connect a C-based client in Windows to a queue manager in Linux, using self-signed certificates, 2-way authentication

The sections are:
- Usage notes
- Steps for 2-way authentication between a C-based client in Windows and a Linux queue manager
- Extreme summary: Commands for a Client in Windows
- Extreme summary: Queue manager in Linux
- Extreme summary: Commands for a Client in Linux

+ Configuration:

MQ Windows Client:
Hostname: finestra1
USERNAME=Administrator

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

+ Steps for 2-way authentication between a C-based client in Windows and a Linux queue manager

Step 1: Client (Windows): Create SSL client key database (CMS)

Step 2: Client (Windows) (Skip for 1-way): Create certificate

Step 3: Client (Windows) (Skip for 1-way): Extract the public SSL client certificate

Step 4: Client (Windows) (Skip for 1-way): 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.

Step 5: Server (Linux): Create SSL server key database

Step 6: Server (Linux): Create certificate

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

Step 8: Server (Linux): Copy Linux certificate to the SSL client 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: Server (Linux) (Skip for 1-way): Add the Windows certificate to Linux key database

Step 10: Server (Linux): Run MQSC commands for SSL server side queue manager

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

Step 12: Using sample amqssslc to test the sending of a message from Client (Windows) to Server (Linux)

Step 13: (Optional) Using CCDT file in JSON format and sample amqsputc to test the sending of a message from Client (Windows) to Server (Linux)

+ Extreme summary: Client in Windows

Step 1: Client (Windows): Create SSL client key database (CMS)

```
setmqenv -n Installation1
```

If necessary, you may need to specify the full path:

```
"C:\Program Files\IBM\MQ\bin\setmqenv" -n Installation1

set MQSSLKEYR=C:\ProgramData\IBM\MQ\ssl\clientkey

runmqckm -keydb -create  -db "C:\ProgramData\IBM\MQ\ssl\clientkey.kdb" -pw clientpass -
type cms  -stash
```

Step 2: Client (Windows) (Skip for 1-way): 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\ssl\clientkey.kdb" -stashed -label
ibmwebspheremqadministrator  -dn "CN=administrator,O=IBM,C=USA" -size 2048
```

Option 2: Specifying -pw passwordPhrase

```
runmqckm -cert  -create  -db "C:\ProgramData\IBM\MQ\ssl\clientkey.kdb" -pw clientpass -
label ibmwebspheremqadministrator  -dn "CN=administrator,O=IBM,C=USA" -size 2048

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

runmqckm -cert  -details -db "C:\ProgramData\IBM\MQ\ssl\clientkey.kdb" -stashed -label
ibmwebspheremqadministrator
```

Step 3: Client (Windows) (Skip for 1-way): Extract the public SSL client certificate

```
runmqckm -cert  -extract -db "C:\ProgramData\IBM\MQ\ssl\clientkey.kdb" -stashed -label
ibmwebspheremqadministrator  -target administrator.crt -format ascii
```

Step 4: Client (Windows) (Skip for 1-way): 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.

Exchange keys with the host that has the queue manager.
Perform steps the Server (Linux).

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

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

runmqckm -cert  -list    -db "C:\ProgramData\IBM\MQ\ssl\clientkey.kdb" -stashed
```

Step 12: Using sample amqssslc to test the sending of a message from Client (Windows) to Server (Linux)

Test (do NOT include the suffix .kdb in -k)

```
 amqssslc -m QMSTMTLS -c SSL.SVRCONN -x stmichel1.fyre.ibm.com(1419) -k
"C:\ProgramData\IBM\MQ\ssl\clientkey" -s TLS_AES_128_GCM_SHA256 -l
ibmwebspheremqadministrator
```

Step 13: (Optional) Using CCDT file in JSON format and sample amqsputc to test the sending of a message from Client (Windows) to Server (Linux)

Miscellaneous

Note regarding the deletion of a certificate (reuse the one for “-cert -details” and replace it with “-cert -delete)

```
runmqckm -cert  -delete -db "C:\ProgramData\IBM\MQ\ssl\clientkey.kdb" -stashed
-label ibmwebspheremqadministrator
```

+ Extreme summary: Queue manager in Linux

As user root: Facilitate the remote access from Windows user "Administrator", truncated to 12 characters in lowercase: "administrato"

```
  As user root: Add group for non-MQ administrators
  groupadd -g 1005 mqusers

  useradd -u 603 -g mqusers -s /bin/bash -d /home/administrato -m administrato
```

As user mqm:

```
. /opt/mqm/bin/setmqenv -n Installation1

setmqaut -m QMSTMTLS -t qmgr                               -g mqusers +connect +inq +dsp
setmqaut -m QMSTMTLS -t q -n SYSTEM.DEFAULT.MODEL.QUEUE    -g mqusers +inq +browse +get
+dsp
setmqaut -m QMSTMTLS -t q -n SYSTEM.ADMIN.COMMAND.QUEUE    -g mqusers +inq +put +dsp
setmqaut -m QMSTMTLS -t q -n SYSTEM.MQEXPLORER.REPLY.MODEL -g mqusers +inq +browse +get
+dsp +put
setmqaut -m QMSTMTLS -t q -n Q1                            -g mqusers +inq +browse +get
+dsp +put
```

Step 5: Server (Linux): Create SSL server key database

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

Step 6: Server (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

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

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

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

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

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

Exchange keys with the Client (Windows).

Step 9: Server (Linux) (Skip for 1-way): Add the Windows certificate to Linux key database

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

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

Step 10: Server (Linux): Run MQSC commands for SSL server side queue manager

```
## For 2-way

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

  DEFINE CHANNEL('SSL.SVRCONN') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
  SSLCIPH(TLS_AES_128_GCM_SHA256) +
  SSLCAUTH(REQUIRED) +
  SSLPEER('CN=administrator,O=IBM,C=USA') REPLACE

  REFRESH SECURITY TYPE(SSL)

  DEFINE QLOCAL(Q1) REPLACE

  END


## For 1-way

runmqsc QMSTMTLS

  DEFINE CHANNEL('SSL.SVRCONN') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
  SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256) +
  SSLCAUTH(OPTIONAL) +
  SSLPEER('') REPLACE

  REFRESH SECURITY TYPE(SSL)

  DEFINE QLOCAL(Q1) REPLACE

  END
```

Proceed with rest of steps from Client (Windows)

+ Extreme summary: Commands for a Linux Client

For completeness, this section provides the commands for a Linux Client.

User "mqm" creates directory to keep SSL related files:

```
mkdir /var/mqm/ssl
```

Setup the environment variables for MQ

```
.  /opt/mqm/bin/setmqenv -n Installation1

# Add the MQ directory with the samples into the PATH (not added by setmqenv)
export PATH=$PATH:$MQ_INSTALLATION_PATH/samp/bin
```

Step 1: Client (Linux): Create SSL client key database (CMS)

```
export MQSSLKEYR=/var/mqm/ssl/clientkey

runmqckm -keydb -create  -db /var/mqm/ssl/clientkey.kdb -pw clientpass -type cms -
stash
```

Step 2: Client (Linux) (Skip for 1-way): Create certificate

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

runmqckm -cert  -list    -db /var/mqm/ssl/clientkey.kdb -stashed

runmqckm -cert  -details -db /var/mqm/ssl/clientkey.kdb -stashed -label
ibmwebspheremqmqm
```

Step 3: Client (Linux) (Skip for 1-way): Extract the public SSL client certificate

```
runmqckm -cert -extract  -db /var/mqm/ssl/clientkey.kdb -stashed -label
ibmwebspheremqmqm -target /var/mqm/ssl/mqm-client.crt -format ascii
```

Step 4: Client (Linux) (Skip for 1-way): Copy Client Linus certificate to the SSL server side in Linux
Copy/transfer the public/signer SSL certificate administrator.crt in ASCII mode from the Client Linux host to the Server Linux host.

Exchange keys with the host that has the queue manager
Perform steps the Server (Linux).

Step 11: Client (Linux): Add the Server Linux certificate to the Client Linux key database

```
runmqckm -cert  -add     -db /var/mqm/ssl/clientkey.kdb -stashed -label
ibmwebspheremqqmstmtls        -file /var/mqm/ssl/QMSTMTLS.crt  -format ascii

runmqckm -cert  -list    -db /var/mqm/ssl/clientkey.kdb -stashed
```

Step 12: Using sample amqssslc to test the sending of a message from Client (Linux) to Server (Linux)

```
Test (do NOT include the suffix .kdb in -k)
 amqssslc -m QMSTMTLS -c SSL.SVRCONN -x 'stmichel1.fyre.ibm.com(1419)' -k
/var/mqm/ssl/clientkey -s TLS_AES_128_GCM_SHA256 -l ibmwebspheremqmqm
```

Step 13: (Optional) Using CCDT file in JSON format and sample amqsputc to test the sending of a message from Client (Linux) to Server (Linux)

<u>+++ Chapter 2: Using SSL TLS in MQ 9.3 to connect a JMS client to a queue manager in Linux, using self-signed certificates, 2-way authentication</u>

The commands were obtained from the following comprehensive tutorial:

https://www.ibm.com/support/pages/node/7142241
Using SSL TLS in MQ 9.3 to connect a JMS client to a queue manager in Linux, using self-signed certificates, 2-way authentication

<u>+ Overall sequence of steps</u>

Step 1: Client: Create SSL client key database, type pkcs12 (jks)
Step 2: Client (Skip for 1-way): Create personal certificate
Step 3: Client (Skip for 1-way): Extract the public SSL client certificate
Step 4: Client (Skip for 1-way): 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.

Step 5: Server (Linux): Create SSL server key database (type CMS)
Step 6: Server (Linux): Create certificate
Step 7: Server (Linux): Extract the public SSL server certificate
Step 8: Server (Linux): Copy Linux certificate to the SSL client 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: Server (Linux) (Skip for 1-way): Add the Windows certificate to Linux key database
Step 10: Server (Linux): Run MQSC commands for SSL server side queue manager

Step 11: Client: Add the Linux certificate to the Windows key database
Step 12: Using sample SSLSampleJMS to test the sending of a message from Client (Linux) to Server (Linux)

<u>+ Extreme summary: Client in Windows</u>

Step 1: Client (Windows): Create SSL client key database, type pkcs12 (jks)

Issue "setmqenv" to set the MQ environment variables, including the ones for running Java/JMS applications.

Create directory "ssl" (or "tls") such as:
```
 C:\ProgramData\IBM\MQ\ssl
```

Inside that directory create key database:

```
runmqckm -keydb -create  -db "C:\ProgramData\IBM\MQ\ssl\key.jks" -pw clientpass -type
pkcs12 -stash
```

Step 2: Client (Windows) (Skip for 1-way): Create personal certificate

```
runmqckm -cert  -create  -db "C:\ProgramData\IBM\MQ\ssl\key.jks" -stashed -label
ibmwebspheremqadministrator  -dn "CN=administrator,O=IBM,C=USA" -size 2048
```

List the newly created SSL certificate in Windows

```
runmqckm -cert  -list    -db "C:\ProgramData\IBM\MQ\ssl\key.jks" -stashed
```

List the details of the personal certificate:

```
runmqckm -cert  -details -db "C:\ProgramData\IBM\MQ\ssl\key.jks" -stashed -label
ibmwebspheremqadministrator
```

+ Deleting a certificate:
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\ssl\key.jks" -stashed
-label ibmwebspheremqadministrator
```

Then replace “details” for “delete”:

```
runmqckm -cert  -delete -db "C:\ProgramData\IBM\MQ\ssl\key.jks" -stashed
-label ibmwebspheremqadministrator
```

Step 3: Client (Windows) (Skip for 1-way): Extract the public SSL client certificate

This step is only needed when doing “2-way authentication”.
But it is not needed when doing “1-way authentication” (but it does not hurt if you do it).

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

The following command will create a file in the current directory.
For this tutorial this is the current directory so far:
C:\ProgramData\IBM\MQ\ssl

```
runmqckm -cert  -extract -db "C:\ProgramData\IBM\MQ\ssl\key.jks" -stashed -label
ibmwebspheremqadministrator  -target administrator.crt -format ascii
```

Step 4: Client (Windows) (Skip for 1-way): Copy Windows certificate to the SSL server side in Linux

This step is needed when doing “2-way authentication”.
It is not needed when doing “1-way authentication” (but it does not hurt if you do it).

Exchange keys with the host that has the queue manager.

Perform steps the Server (Linux).

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

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

List the certificates

```
C:\ProgramData\IBM\MQ\ssl>
runmqckm -cert  -list    -db "C:\ProgramData\IBM\MQ\ssl\key.jks" -stashed
```

Step 12: Using sample SSLSampleJMS to test the sending of a message from Client (Windows) to Server (Linux)

Go to the directory where the sample is stored.
For this tutorial is:

```
   cd c:\wintools
```

+ 1-way authentication: use channel JMSSSL1.SVRCONN

```
C:\WinTools> java -Dcom.ibm.mq.cfg.useIBMCipherMappings=false -
Djavax.net.ssl.trustStore="C:\ProgramData\IBM\MQ\ssl\key.jks" -
Djavax.net.ssl.trustStorePassword=clientpass SSLSampleJMS stmichel1.fyre.ibm.com 1419
JMSSSL1.SVRCONN QMSTMTLS TLS_AES_128_GCM_SHA256 C:\ProgramData\IBM\MQ\ssl\key.jks
clientpass
```

+ 2-way authentication: use channel JMSSSL2.SVRCONN

```
C:\WinTools> java -Dcom.ibm.mq.cfg.useIBMCipherMappings=false -
Djavax.net.ssl.trustStore="C:\ProgramData\IBM\MQ\ssl\key.jks" -
Djavax.net.ssl.trustStorePassword=clientpass SSLSampleJMS stmichel1.fyre.ibm.com 1419
JMSSSL2.SVRCONN QMSTMTLS TLS_AES_128_GCM_SHA256 C:\ProgramData\IBM\MQ\ssl\key.jks
clientpass
```

+ Extreme summary: Queue manager in Linux

As user root: Facilitate the remote access from Windows user "Administrator", truncated to 12 characters in lowercase: "administrato"

```
  As user root: Add group for non-MQ administrators
  groupadd -g 1005 mqusers

  useradd -u 603 -g mqusers -s /bin/bash -d /home/administrato -m administrato
```

As user mqm:

```
. /opt/mqm/bin/setmqenv -n Installation1

setmqaut -m QMSTMTLS -t qmgr                               -g mqusers +connect +inq +dsp
setmqaut -m QMSTMTLS -t q -n SYSTEM.DEFAULT.MODEL.QUEUE    -g mqusers +inq +browse +get
+dsp
setmqaut -m QMSTMTLS -t q -n SYSTEM.ADMIN.COMMAND.QUEUE    -g mqusers +inq +put +dsp
setmqaut -m QMSTMTLS -t q -n SYSTEM.MQEXPLORER.REPLY.MODEL -g mqusers +inq +browse +get
+dsp +put
setmqaut -m QMSTMTLS -t q -n Q1                            -g mqusers +inq +browse +get
+dsp +put
```

Step 5: Server (Linux): Create SSL server key database

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

Step 6: Server (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

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

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

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

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

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

Exchange keys with the Client (Windows).

Step 9: Server (Linux) (Skip for 1-way): Add the Windows certificate to Linux key database

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

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

Step 10: Server (Linux): Run MQSC commands for SSL server side queue manager

Note:
There are 2 channels for 2-way authentication used in this tutorial:
JMSSSL2.SVRCONN for the Windows client, userid "administrator"
JMSSSL2LNX.SVRCONN for the Linux client, userid "mqm"

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

DEFINE CHANNEL('JMSSSL2.SVRCONN') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
SSLCIPH(TLS_AES_128_GCM_SHA256) SSLCAUTH(REQUIRED) +
SSLPEER('CN=administrator,O=IBM,C=USA')

DEFINE CHANNEL('JMSSSL2LNX.SVRCONN') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
SSLCIPH(TLS_AES_128_GCM_SHA256) SSLCAUTH(REQUIRED) +
SSLPEER('CN=mqm,O=IBM,C=USA')

REFRESH SECURITY TYPE(SSL)

DEFINE QLOCAL(Q1) REPLACE

END
```

+ For “1-way authentication”:

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

DEFINE CHANNEL('JMSSSL1.SVRCONN') CHLTYPE(SVRCONN) TRPTYPE(TCP) +
SSLCIPH(TLS_AES_128_GCM_SHA256) SSLCAUTH(OPTIONAL) +
SSLPEER('')

REFRESH SECURITY TYPE(SSL)

DEFINE QLOCAL(Q1) REPLACE

END
```

Proceed with rest of steps from Client (Windows)

+ Extreme summary: Commands for a Linux Client

For completeness, this section provides the commands for a Linux Client.

User "mqm" creates directory to keep SSL related files:

```
mkdir /var/mqm/ssl
```

Setup the environment variables for MQ

```
.  /opt/mqm/bin/setmqenv -n Installation1
```

Step 1: Client (Linux): Create SSL client key database, type pkcs12 (jks)

Issue "setmqenv" to set the MQ environment variables, including the ones for running Java/JMS applications.

```
runmqckm -keydb -create  -db /var/mqm/ssl/key.jks -pw clientpass -type pkcs12 -stash
```

Step 2: Client (Linux) (Skip for 1-way): Create personal certificate

```
runmqckm -cert  -create  -db /var/mqm/ssl/key.jks -stashed -label ibmwebspheremqmqm -dn
"CN=mqm,O=IBM,C=USA" -size 2048

runmqckm -cert  -list    -db /var/mqm/ssl/key.jks -stashed

runmqckm -cert  -details -db /var/mqm/ssl/key.jks -stashed -label ibmwebspheremqmqm
```

Step 3: Client (Linux) (Skip for 1-way): Extract the public SSL client certificate

This step is only needed when doing “2-way authentication”.

```
cd /var/mqm/ssl
runmqckm -cert  -extract -db /var/mqm/ssl/key.jks  -stashed -label ibmwebspheremqmqm  -
target mqm.crt -format ascii
```

Step 4: Client (Linux) (Skip for 1-way): Copy Client Linus certificate to the SSL server side in Linux

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

Perform steps the Server (Linux).

Step 11: Client (Linux): Add the Server Linux certificate to the Client Linux key database

```
runmqckm -cert  -add     -db /var/mqm/ssl/key.jks -stashed -label ibmwebspheremqqmstmtls
-file /var/mqm/ssl/QMSTMTLS.crt  -format ascii

runmqckm -cert  -list    -db /var/mqm/ssl/key.jks.kdb -stashed
```

Step 12: Using sample SSLSampleJMS to test the sending of a message from Client (Linux) to

Server (Linux)

```
cd /home/mqm/ssl
```

1-way authentication:

```
$ java -Dcom.ibm.mq.cfg.useIBMCipherMappings=false -
Djavax.net.ssl.trustStore=/var/mqm/ssl/key.jks -
Djavax.net.ssl.trustStorePassword=clientpass SSLSampleJMS stmichel1.fyre.ibm.com 1419
JMSSSL1.SVRCONN QMSTMTLS TLS_AES_128_GCM_SHA256 /var/mqm/ssl/key.jks clientpass
```

2-way authentication:

```
$ java -Dcom.ibm.mq.cfg.useIBMCipherMappings=false -
Djavax.net.ssl.trustStore=/var/mqm/ssl/key.jks -
Djavax.net.ssl.trustStorePassword=clientpass SSLSampleJMS stmichel1.fyre.ibm.com 1419
JMSSSL2LNX.SVRCONN QMSTMTLS TLS_AES_128_GCM_SHA256 /var/mqm/ssl/key.jks
```

+++ Chapter 3: Using SSL TLS to connect an IBM MQ 9.3 queue manager in Windows with another one in Linux, using self-signed certificates

The commands were obtained from the following comprehensive tutorial:

https://www.ibm.com/support/pages/node/7121151
Using SSL TLS to connect an IBM MQ 9.3 queue manager in Windows with another one in Linux, using self-signed certificates

+ Configuration

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

+ Summary:

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
```

+++ end +++