Using SSL TLS to connect two MQ queue managers in MQ 7.5 and MQ 8.0 / MQ 9.0 using self-signed certificates

IBM Techdoc: 7048223
http://www.ibm.com/support/docview.wss?uid=swg27048223

Date last updated: 19-Aug-2016

Angel Rivera – rivera@us.ibm.com
Mike Cregger - mike_cregger@us.ibm.com
IBM MQ Support

+++ Objective

The objective of this document is to provide the step-by-step details for connecting a MQ queue manager 7.5 in one platform (Windows) to another queue manager 8.0 running on another platform (Linux), using SSL TLS self-signed certificates.

For illustration purposes the following protocol will be used, which is valid in MQ 7.5 and 8.0.
TLS_RSA_WITH_AES_128_CBC_SHA

Note for MQ 9.0:
The commands mentioned in this tutorial apply too to MQ 9.0 and they were verified in a test queue manager running in MQ 9.0.

+ Update on August-2016

An appendix was added to include a brief description of using the IBM Key Management GUI (iKeyman) for the corresponding runmqakm commands.

+++ Summary of steps using runmqakm:
Sender in Windows connecting to a Receiver in Linux

Step 1: Client (Windows): Create SSL client key database
cd C:\var\mqm\Qmgrs\QM75WIN\ssl
runmqakm -keydb -create -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw clientpass -type cms -expire 365 -stash

Step 2: Client (Windows): Create certificate

+ Create certificate: self-signed
runmqakm -cert -create -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw clientpass -label ibmwebspheremqqm75win -dn "CN=QM75WIN,O=IBM,C=USA" -expire 365 -sigalg sha1 -size 2048

+ List newly created SSL certificate in Windows
runmqakm -cert -list -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw client-pass

+ List the details of the certificate.
runmqakm -cert -details -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw clientpass -label ibmwebspheremqqm75win

+ REFERENCE: In case that you need to delete the certificate (we are providing this command just for completeness, you do NOT need to issue it for this tutorial)
runmqakm -cert -delete -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw clientpass -label <label>

Step 3: Client (Windows): Extract the public SSL client certificate
runmqakm -cert -extract -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw clientpass -label ibmwebspheremqqm75win -target QM75WIN.crt -format ascii

Step 4: Client (Windows): Copy Windows certificate to the SSL server side in Linux
Copy/transfer the public/signer SSL certificate QM75WIN.crt in ASCII mode from the Windows host to the Linux host.

Step 5: Server (Linux): Create SSL server key database
cd /var/mqm/qmgrs/QM80LNX/ssl
runmqakm -keydb -create -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw serverpass -type cms -expire 365 -stash

Step 6: Server (Linux): Create certificate

+ Create certificate: self-signed
runmqakm -cert -create -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw serverpass -label ibmwebspheremqqm80lnx -dn "CN=QM80LNX,O=IBM,C=USA" -expire 365 -sigalg sha1 -size 2048

+ List newly created SSL certificate in Linux
runmqakm -cert -list -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw server-pass

+ List the details of the certificate.
runmqakm -cert -details -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw serverpass -label ibmwebspheremqqm80lnx

+ REFERENCE: In case that you need to delete the certificate (we are providing this command just for completeness, you do NOT need to issue it for this tutorial)
runmqakm -cert -delete -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw serverpass -label <label>

Step 7: Server (Linux): Extract the public SSL server certificate
runmqakm -cert -extract -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw serverpass -label ibmwebspheremqqm80lnx -target QM80LNX.crt -format ascii

Step 8: Server (Linux): Copy Linux certificate to the SSL client side in Windows
Copy/transfer the public/signer SSL certificate QM80LNX.crt in ASCII mode from the Linux host to the Windows host.

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

+ Add the public/signer certificate
runmqakm -cert -add -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw server-pass -label ibmwebspheremqqm75win -file QM75WIN.crt -format ascii

+ List the certificates.
runmqakm -cert -list -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw server-pass

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

```
runmqsc QM80LNX
ALTER QMGR SSLKEYR('/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX')
DEFINE CHANNEL('QM75WIN.TO.QM80LNX') CHLTYPE(RCVR) TRPTYPE(TCP) +
 SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA) SSLCAUTH(REQUIRED) +
 SSLPEER('CN=QM75WIN,O=IBM,C=USA') REPLACE
REFRESH SECURITY TYPE(SSL)
end
```

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

```
+ Add the public/signer certificate
runmqakm -cert -add -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw client-
pass -label ibmwebspheremqqm80lnx -file QM80LNX.crt -format ascii
+ List the certificates.
runmqakm -cert -list -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw client-
pass
```

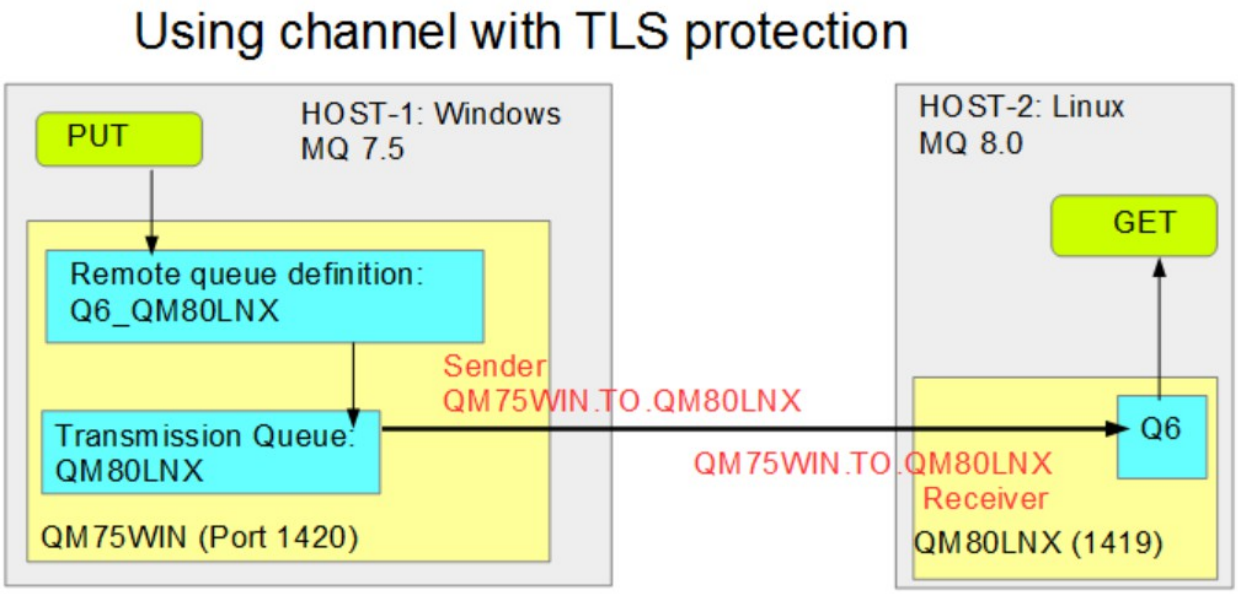
Step 12: Client (Windows): Run MQSC commands for SSL client side queue manager

```
runmqsc QM75WIN
ALTER QMGR SSLKEYR('C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN')
DEFINE CHANNEL('QM75WIN.TO.QM80LNX') CHLTYPE(SDR) TRPTYPE(TCP) +
  XMITQ('QM80LNX') CONNAME('9.30.145.117(1419)') +
  SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA) +
  SSLPEER('CN=QM80LNX,O=IBM,C=USA') REPLACE
DEFINE QL(QM80LNX) USAGE(XMITQ) REPLACE
REFRESH SECURITY TYPE(SSL)
START CHANNEL('QM75WIN.TO.QM80LNX')
DISPLAY CHSTATUS(QM75WIN.TO.QM80LNX)
DISPLAY CHSTATUS(QM75WIN.TO.QM80LNX) ALL
end
```

Step 13: Test of sending message from Client (Windows) to Server (Linux)

++ Scope

This tutorial provides all the steps to configure a TLS enabled "sender" channel in the Windows queue manager, and a TLS enabled "receiver" channel in the Linux queue manager. This will be used to Put a message in the Windows queue manager and send it to the Linux queue manager.



Conceptually, the queue manager that has the SENDER channel is a "client" and the queue manager that has the RECEIVER channel is a "server". These terms will be used in the tutorial, in addition to the distinction of Windows (MQ 7.5) and Linux (MQ 8.0).
Why?
One common source of confusion is: which side of the connection a certain step/command needs to be taken on.
Therefore, by being extremely explicit in this tutorial, hopefully this source of confusion will be eliminated.

++ 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.
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.

+++ Configuration

a) MQ 7.5.0.5 running in Windows 7
Queue Manager name: QM75WIN
Hostname: angelillo.x.com
Port: 1420
Data directory: C:\var\mqm\Qmgrs\QM75WIN
SSL directory: C:\var\mqm\Qmgrs\QM75WIN\ssl
Channel for administrative purposes (MQ Explorer): SYSTEM.ADMIN.SVRCONN
Channel with TLS enablement:
Sender channel: QM75WIN.TO.QM80LNX

b) MQ 8.0.0.4 running on Linux Intel 64-bit
Queue Manager name: QM80LNX
Hostname: Suse-host4 ip-9-30-145-117.svl.ibm.com
Port: 1419
Data directory: /var/mqm/qmgrs/QM80LNX
SSL directory: /var/mqm/qmgrs/QM80LNX/ssl
Channel for administrative purposes (MQ Explorer): SYSTEM.ADMIN.SVRCONN
Channel with TLS enablement:
Receiver channel: QM75WIN.TO.QM80LNX

c) The 2 queue managers involved in the scenario were available prior to the scenario described in this document.

In addition, because these queue managers are used for TESTING and not for PRODUCTION, some intrinsic security layers were relaxed, in order to focus on the SSL/TLS aspects of the configuration, which is the primary goal for this document.

Once the SSL/TLS function is established and working fine, then you could enable these other intrinsic security layers.

c.1) For MQ 7.5 and MQ 8.0:

Starting with MQ 7.1, a new security feature was introduced: channel authentication records.
The default behavior is that MQ administrators are blocked from doing a remote access.
For more information on the errors and the workarounds see:
http://www.ibm.com/support/docview.wss?uid=swg21577137
WMQ 7.1, 7.5, 8.0 queue manager RC 2035 MQRC_NOT_AUTHORIZED or AMQ4036 when using client connection as an MQ Administrator

Main workaround mentioned in the above technote:
For a testing queue manager:

```
  ## For MQ 7.1 and later: if desiring to exploit channel authentication records and allow remote connections by an MQ Administrator:
  set CHLAUTH(*) TYPE(BLOCKUSER) USERLIST('nobody','*MQADMIN')
  set CHLAUTH(SYSTEM.ADMIN.*) TYPE(BLOCKUSER) USERLIST('nobody')
```

c.2) For MQ 8.0:
The MQ Administrators are required to provide a password when doing a remote access.
http://www.ibm.com/support/docview.wss?uid=swg21680930
MQ 8.0: errors AMQ5540, AMQ5541 and AMQ5542, application did not supply a user ID and password, 2035 MQRC_NOT_AUTHORIZED

For a testing queue manager or if you want to have your queue manager with the same behavior as in MQ 7.x and not requiring passwords to be specified.
Issue the following 2 runmqsc commands to change the value of CHCKCLNT from REQDADM to OPTIONAL for the AUTHINFO shown below and this will allow users to not necessarily provide a userid/password.

```
  ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) +
    CHCKCLNT(OPTIONAL)
  REFRESH SECURITY TYPE(CONNAUTH)
```

d) TLS Protocol:

For illustration purposes I chose the following protocol, which is valid in MQ 7.5, 8.0 and 9.0.
TLS_RSA_WITH_AES_128_CBC_SHA

Here are the web pages regarding this protocol for the different versions of MQ:

http://www.ibm.com/support/knowledgecenter/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q014260_.htm?lang=en
MQ 7.5.0 > WebSphere MQ > Security > Confidentiality of messages >
Specifying CipherSpecs
.
CipherSpec name: TLS_RSA_WITH_AES_128_CBC_SHA
Protocol used: TLS 1.0
Data integrity: SHA-1
Encryption algorithm: AES
Encryption bits: 128
FIPS: Yes
Suite B 128 bit: No
Suite B 192 bit: No
Platforms: Available only on UNIX, Linux, and Windows platforms.

http://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.sec.doc/q014260_.htm?lang=en
MQ 8.0.0 > IBM MQ > Security > Confidentiality of messages >
Enabling CipherSpecs
.
Platform support: All
CipherSpec name: TLS_RSA_WITH_AES_128_CBC_SHA
Protocol used: TLS 1.0
Data integrity: SHA-1
Encryption algorithm: AES
Encryption bits: 128
FIPS: Yes
Suite B: No

http://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.sec.-doc/q014260_.htm
WebSphere MQ > WebSphere MQ 9.0.0 > IBM MQ > Security > Confidentiality of messages >
Enabling CipherSpecs

++ References

The following technote provides the steps for non SSL non TLS connections.
http://www-01.ibm.com/support/docview.wss?uid=swg21470997
Commands to setup both ways communication between 2 queue managers via Sender and Receiver channels

https://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.sce.doc/q014200_.htm
WebSphere MQ > WebSphere MQ 8.0.0 > IBM MQ > Scenarios > Security scenarios > Connecting two queue managers using SSL or TLS >
Using CA-signed certificates for mutual authentication

If you would like a shorter version of the instructions, please consult:
https://developer.ibm.com/answers/questions/250219/how-do-i-configure-ssl-between-2-mq-queue-managers.html
How do I configure SSL between 2 MQ queue managers (Sender/Receiver channels)?
By Mike Cregger, IBM MQ Support

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

Host: Windows

Login as an MQ administrator

runmqakm -keydb -create -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw clientpass -type cms -expire 365 -stash

Result: 4 files were created in the specified ssl directory:
C:\> dir C:\var\mqm\Qmgrs\QM75WIN\ssl\
05/26/2016 02:24 PM   88 QM75WIN.crl
05/26/2016 02:24 PM   88 QM75WIN.kdb
05/26/2016 02:24 PM   88 QM75WIN.rdb
05/26/2016 02:24 PM   129 QM75WIN.sth

++
Step 2: Client (Windows): Create certificate: self-signed
++

Host: Windows

+ Create the certificate: self-signed

runmqakm -cert -create -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw clientpass -label ibmwebspheremqqm75win -dn "CN=QM75WIN,O=IBM,C=USA" -expire 365 -sigalg sha1 -size 2048

Where:
-label is the label name:
    ibmwebspheremqqm75win
  It is required to be the concatenation of:
    ibmwebspheremq + queue manager in lower case
  In this case:
    ibmwebspheremq + qm75win
-dn is the "Distinguished Name"
-sigalg is the signature algorithm
-size The recommended size is 2048 bits. The certificates with a size of 1024 are no longer recommended.

For more details, see:
http://www.ibm.com/support/knowledgecenter/SSFKSJ_7.5.0/com.ibm.mq.sec.doc/q012680_.htm
WebSphere MQ > WebSphere MQ 7.5.0 > WebSphere MQ > Security > Setting up security > Working with SSL or TLS > Working with SSL or TLS on UNIX and Windows systems > Setting up a key repository on UNIX, Linux and Windows systems

Result: Notice that the size of QM75WIN.kdb was increased
C:\>dir C:\var\mqm\Qmgrs\QM75WIN\ssl\
05/26/2016 02:24 PM   88 QM75WIN.crl
05/26/2016 02:28 PM   **5,088 QM75WIN.kdb => size increase**
05/26/2016 02:24 PM   88 QM75WIN.rdb
05/26/2016 02:24 PM   129 QM75WIN.sth

+ List newly created SSL certificate in Windows
Host: Windows

Result: notice that a "personal" certificate was created.

```
runmqakm -cert -list -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb"
-pw clientpass

Certificates found
* default, - personal, ! trusted
-       ibmwebspheremqqm75win
```

+ List the details of the certificate.
For brevity, I am deleting some lines.

```
runmqakm -cert -details -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw
clientpass -label ibmwebspheremqqm75win

Label : ibmwebspheremqqm75win
Key Size : 2048
Version : X509 V3
Serial : 4dc6c31900fceeff
Issuer : CN=QM75WIN,O=IBM,C=USA
Subject : CN=QM75WIN,O=IBM,C=USA
Not Before : May 25, 2016 2:28:37 PM EDT
Not After : May 26, 2017 2:28:37 PM EDT
Public Key
   30 81 9F 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01
Public Key Type : RSA (1.2.840.113549.1.1.1)
Fingerprint : SHA1 :
   0A 4C 87 BF 3D 32 70 D1 1C 3A C7 B7 F8 78 2E BC
Fingerprint : MD5 :
   78 10 23 BD 31 52 17 38 AE 05 EA 1A 44 DB 7F 9E
Fingerprint : SHA256 :
   37 18 EA 73 B9 D9 D3 B6 3C 8C 26 F6 DE B9 7E 9A
   CD 4C 19 9B 4F 69 30 DD 2D 5C 0B 66 48 46 57 85
Signature Algorithm : SHA1WithRSASignature (1.2.840.113549.1.1.5)
Value
   55 15 FD 52 89 0E 92 C2 74 5A 8D A4 41 30 1C EC
   01 F2 55 F6 EF AB 68 28 95 EC 23 D8 29 BD CB F9
Trust Status : Enabled
```

+ Reference: deleting a certificate

We are including the following command just for completeness, you do NOT need to issue it for this tutorial.
If you need to delete the certificate, specify the proper label and issue:

runmqakm -cert -delete -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw clientpass -label <label>

++
Step 3: Client (Windows): Extract the public SSL client certificate
++

Host: Windows

Notes:
- The flag -extract deals only with the signer/public key of a certificate, and does NOT deal with the private key.
- The flag "-format ascii" means "Base64-encoded ASCII"

runmqakm -cert -extract -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw clientpass -label ibmwebspheremqqm75win -target QM75WIN.crt -format ascii

Notice a new file:
C:\var\mqm\Qmgrs\QM75WIN\ssl> dir
05/26/2016 02:24 PM 88 QM75WIN.crl
05/26/2016 02:38 PM **1072 QM75WIN.crt => new file**
05/26/2016 02:28 PM 5,088 QM75WIN.kdb
05/26/2016 02:24 PM 88 QM75WIN.rdb
05/26/2016 02:24 PM 129 QM75WIN.sth

The new file looks like this.
For brevity, I am showing only few lines and I am deleting others:

C:\var\mqm\Qmgrs\QM75WIN\ssl> type QM75WIN.crt
-----BEGIN CERTIFICATE-----
MIICJzCCAZCgAwIBAgIIQMjeXQWo9ocwDQYJKoZIhvcNAQELBQAwVjEMMAoGA1UE
ucxvCfw+QqOf8pubxzVzf78lJFPaYbuzXgi+
-----END CERTIFICATE-----

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

Host: Windows
Use FTP to transfer the SSL certificate QM75WIN.crt in ASCII mode from the Windows host to the Linux host.
The file can be placed in the directory:
/var/mqm/qmgrs/QM80LNX/ssl/

Host: Linux
mqm@SUSE-host4: /var/mqm/qmgrs/QM80LNX/ssl
$ ls -l
-rw-r--r-- 1 root mqm **1072 2016-05-26 11:41 QM75WIN.crt => New file**

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

Host: Linux

Log in as an MQ Administrator

cd /var/mqm/qmgrs/QM80LNX/ssl

runmqakm -keydb -create -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw serverpass -type cms -expire 365 -stash

Result: 4 files were created in the specified ssl directory:

```
mqm@SUSE-host4: /var/mqm/qmgrs/QM80LNX/ssl
$ ls -l
-rw-r--r-- 1 root mqm 712 2016-05-26 11:41 QM75WIN.crt
-rw------- 1 mqm  mqm  88 2016-05-26 11:48 QM80LNX.crl => new
-rw------- 1 mqm  mqm  88 2016-05-26 11:48 QM80LNX.kdb => new
-rw------- 1 mqm  mqm  88 2016-05-26 11:48 QM80LNX.rdb => new
-rw------- 1 mqm  mqm 129 2016-05-26 11:48 QM80LNX.sth => new
```

++
Step 6: Server (Linux): Create certificate: self-signed
++

Host: Linux

runmqakm -cert -create -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw serverpass -label ibmwebspheremqqm80lnx -dn "CN=QM80LNX,O=IBM,C=USA" -expire 365 -sigalg sha1 -size 2048

Note:
In 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:
http://www.ibm.com/support/knowledgecenter/SSFKSJ_9.0.0/com.ibm.mq.ref.con.-doc/q113280_.htm
WebSphere MQ > WebSphere MQ 9.0.0 > IBM MQ > Reference > Configuration reference > Channel attributes > Channel attributes in alphabetical order > Certificate label (CERTLABL)

Result: Notice that the size of QM80LNX.kdb was increased
mqm@SUSE-host4: /var/mqm/qmgrs/QM80LNX/ssl
$ ls -l
-rw-r--r-- 1 root mqm 1072 2016-05-26 11:41 QM75WIN.crt
-rw------- 1 mqm mqm   88 2016-05-26 11:48 QM80LNX.crl
-rw------- 1 mqm mqm **5088 2016-05-26 11:51 QM80LNX.kdb => size increase**
-rw------- 1 mqm mqm   88 2016-05-26 11:48 QM80LNX.rdb
-rw------- 1 mqm mqm 129 2016-05-26 11:48 QM80LNX.sth

+ List newly created SSL certificate in Linux

Result: notice that a "personal" certificate was created.

runmqakm -cert -list -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw serverpass

Certificates found
* default, - personal, ! trusted, # secret key
-       ibmwebspheremqqm80lnx

+ List the details of the certificate.
For brevity, I am deleting some lines.

```
runmqakm -cert -details -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw
serverpass -label ibmwebspheremqqm80lnx

Label : ibmwebspheremqqm80lnx
Key Size : 2048
Version : X509 V3
Serial : 1b8895dac7a7708f
Issuer : CN=QM80LNX,O=IBM,C=USA
Subject : CN=QM80LNX,O=IBM,C=USA
Not Before : May 25, 2016 11:51:50 AM PDT
Not After : May 26, 2017 11:51:50 AM PDT
Public Key
   30 81 9F 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01
Public Key Type : RSA (1.2.840.113549.1.1.1)
Fingerprint : SHA1 :
   CF 70 23 06 AB 25 29 35 EF 87 EB A3 49 2A 3B 78
Fingerprint : MD5 :
   9E 85 14 24 57 2E 83 EF 61 45 25 0A F4 31 5E B0
Fingerprint : SHA256 :
   74 E4 FF 26 1F BD 6B F9 CA 19 D8 35 7D 17 39 DA
Extensions
   SubjectKeyIdentifier
     keyIdentifier:
   04 93 F0 52 04 84 43 95 E2 96 1F 7C F7 AA 7F 87
   CB DA BE D2
   AuthorityKeyIdentifier
     keyIdentifier:
   04 93 F0 52 04 84 43 95 E2 96 1F 7C F7 AA 7F 87
   CB DA BE D2
     authorityIdentifier:
     authorityCertSerialNumber:
Signature Algorithm : SHA1WithRSASignature (1.2.840.113549.1.1.5)
Value
   7C 1D 3A 08 31 FC 9E DC D9 5A AA C7 E3 E7 D1 F6
Trust Status : Enabled
```

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

Host: Linux

Notes:
- The flag -extract deals only with the signer/public key of a certificate, and does NOT deal with the private key.
- The flag "-format ascii" means "Base64-encoded ASCII"

runmqakm -cert -extract -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw serverpass -label ibmwebspheremqqm80lnx -target QM80LNX.crt -format ascii

Notice the new file: QM80LNX.crt

$ ls -l
-rw-r--r-- 1 root mqm 1072 2016-05-26 11:41 QM75WIN.crt
-rw------- 1 mqm mqm   88 2016-05-26 11:48 QM80LNX.crl
-rw------- 1 mqm mqm **1143 2016-05-26 11:56 QM80LNX.crt => new file**
-rw------- 1 mqm mqm 5088 2016-05-26 11:51 QM80LNX.kdb
-rw------- 1 mqm mqm   88 2016-05-26 11:48 QM80LNX.rdb
-rw------- 1 mqm mqm 129 2016-05-26 11:48 QM80LNX.sth

The new file looks like this.
For brevity, I am showing only few lines and I am deleting others:

$ cat QM80LNX.crt
-----BEGIN CERTIFICATE-----
MIICbTCCAdagAwIBAgIIetAWwN2vaPQwDQYJKoZIhvcNAQELBQAwVzEMMAoGA1UE
hw==
-----END CERTIFICATE-----

++
Step 8: Server (Linux): Copy Linux certificate to the SSL client side in Windows
++

Host: Linux

Use FTP to transfer the SSL certificate QM80LNX.crt in ASCII mode from the Linux host to the Windows host.

The file can be placed in the directory:
C:\var\mqm\Qmgrs\QM75WIN\ssl

Host: Windows
Notice the new file:

```
05/26/2016  02:24 PM                88 QM75WIN.crl
05/26/2016  02:38 PM             1,072 QM75WIN.crt
05/26/2016  02:28 PM             5,088 QM75WIN.kdb
05/26/2016  02:24 PM                88 QM75WIN.rdb
05/26/2016  02:24 PM               129 QM75WIN.sth
05/26/2016  02:59 PM             1143 QM80LNX.crt => new file
```

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

Host: Linux

+ Add the public/signer certificate

Note: the flag -add deals only with the signer/public key of a certificate, and does NOT deal with the private key.

runmqakm -cert -add -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw server-pass -label ibmwebspheremqqm75win -file QM75WIN.crt -format ascii

Notice the size increase for QM80LNX.kdb
$ ls -l
-rw-r--r-- 1 root mqm 1072 2016-05-26 11:41 QM75WIN.crt
-rw------- 1 mqm mqm   88 2016-05-26 11:48 QM80LNX.crl
-rw------- 1 mqm mqm 1143 2016-05-26 11:56 QM80LNX.crt
-rw------- 1 mqm mqm **10088 2016-05-26 11:59 QM80LNX.kdb => size increase**
-rw------- 1 mqm mqm   88 2016-05-26 11:48 QM80LNX.rdb
-rw------- 1 mqm mqm 129 2016-05-26 11:48 QM80LNX.sth

+ List the certificates.
Notice the new one: ibmwebspheremqqm75win

runmqakm -cert -list -db "/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb" -pw server-pass

Certificates found
* default, - personal, ! trusted, # secret key
!   ibmwebspheremqqm75win
-      ibmwebspheremqqm80lnx

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

Host: Linux

NOTES regarding SSLPEER:
Even though the SSLPEER attribute for the queue manager is optional, it is a good practice to use it for extra security.
Notice that the SSLPEER needs to match the details from the Windows certificate (from Step 2: Client (Windows): Create certificate).
    Issuer : CN=QM75WIN,O=IBM,C=USA

NOTES regarding SSLKEYR:
GSKit is going to add the ".kdb" suffix for the key database. That is, do NOT add the kdb extension!
You MUST provide the value for SSLKEYR as follows: full path name of the key store MINUS the .kdb suffix (the .kdb is added at runtime):
Full path name with suffix: /var/mqm/qmgrs/QM80LNX/ssl/QM80LNX.kdb
Full path name minus suffix: /var/mqm/qmgrs/QM80LNX/ssl/QM80LNX

Run the following MQSC commands for creating a RECEIVER channel.

```
runmqsc QM80LNX

ALTER QMGR SSLKEYR('/var/mqm/qmgrs/QM80LNX/ssl/QM80LNX')

DEFINE CHANNEL('QM75WIN.TO.QM80LNX') CHLTYPE(RCVR) TRPTYPE(TCP) +
 SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA) SSLCAUTH(REQUIRED) +
 SSLPEER('CN=QM75WIN,O=IBM,C=USA') REPLACE

* The refresh for SSL ends all SSL channels, then re-loads the keystore and certifi-
cates, and restarts any SSL channels that were running

REFRESH SECURITY TYPE(SSL)

* Define local queue for testing

DEFINE QLOCAL(Q6)

end
```

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

Host: Windows

+ Add the public/signer certificate

Note: the flag -add deals only with the signer/public key of a certificate, and does NOT deal with the private key.

runmqakm -cert -add -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw client-pass -label ibmwebspheremqqm80lnx -file QM80LNX.crt -format ascii

Notice the size increase for QM75WIN.kdb

```
05/26/2016  02:24 PM                88 QM75WIN.crl
05/26/2016  02:38 PM             1,072 QM75WIN.crt
05/26/2016  03:08 PM            10,088 QM75WIN.kdb => size increase
05/26/2016  02:24 PM                88 QM75WIN.rdb
05/26/2016  02:24 PM               129 QM75WIN.sth
05/26/2016  02:59 PM             1,143 QM80LNX.crt
```

+ List the certificates.
Notice the new one: ibmwebspheremqqm80lnx

runmqakm -cert -list -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb" -pw client-pass

```
Certificates found
* default, - personal, ! trusted
!       ibmwebspheremqqm80lnx
-       ibmwebspheremqqm75win
```

++
Step 12: Client (Windows): Run MQSC commands for SSL client side queue manager
++

Host: Windows

NOTES regarding SSLPEER:
Even though the SSLPEER attribute for the queue manager is optional, it is a good practice to use it for extra security.
Notice that the SSLPEER needs to match the details from the Linux certificate (from Step 6: Server (Linux): Create certificate).
    Issuer : CN=QM80LNX,O=IBM,C=USA

NOTES regarding SSLKEYR:
GSKit is going to add the ".kdb" suffix for the key database. That is, do NOT add the kdb extension!
You MUST provide the value for SSLKEYR as follows: full path name of the key store MINUS the .kdb suffix (the .kdb is added at runtime):
Full path name with suffix: C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN.kdb
Full path name minus suffix: C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN

Run the following MQSC commands for creating a SENDER channel and auxiliary object (transmission queue).
In addition, create a remote queue definition in order to test the channel.

runmqsc QM75WIN

```
ALTER QMGR SSLKEYR('C:\var\mqm\Qmgrs\QM75WIN\ssl\QM75WIN')

DEFINE CHANNEL('QM75WIN.TO.QM80LNX') CHLTYPE(SDR) TRPTYPE(TCP) +
  XMITQ('QM80LNX') CONNAME('9.30.145.117(1419)') +
  SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA) +
  SSLPEER('CN=QM80LNX,O=IBM,C=USA') REPLACE

DEFINE QL(QM80LNX) USAGE(XMITQ) REPLACE

* The refresh for SSL ends all SSL channels, then re-loads the keystore and certifi-
cates, and restarts any SSL channels that were running

REFRESH SECURITY TYPE(SSL)

* Define remote queue definition for QM80LNX
```

```
DEFINE QREMOTE(Q6_QM80LNX) RNAME(Q6) RQMNAME(QM80LNX) XMITQ(QM80LNX)

* Start the channel and display the status
START CHANNEL('QM75WIN.TO.QM80LNX')
DISPLAY CHSTATUS(QM75WIN.TO.QM80LNX)
DISPLAY CHSTATUS(QM75WIN.TO.QM80LNX) ALL

end
```

For the DISPLAY CHSTATUS(QM75WIN.TO.QM80LNX)
you should see: STATUS(RUNNING)

```
AMQ8417: Display Channel Status details.
   CHANNEL(QM75WIN.TO.QM80LNX)           CHLTYPE(SDR)
   CONNAME(9.30.145.117(1419))           CURRENT
   RQMNAME(QM80LNX)                      STATUS(RUNNING)
   SUBSTATE(MQGET)                       XMITQ(QM80LNX)
```

For the DISPLAY CHSTATUS(QM75WIN.TO.QM80LNX) ALL
you should see:

```
  SSLCERTI(CN=QM80LNX,O=IBM,C=USA)
  SSLPEER(SERIALNUMBER=51:73:46:05:81:9B:AE:BA,CN=QM80LNX,O=IBM,C=USA)
  STATUS(RUNNING)
```

The SSLCERTI and SSLPEER attributes (remote signer and remote peer DN) help validate that the channel is using the certificates.

```
AMQ8417: Display Channel Status details.
   CHANNEL(QM75WIN.TO.QM80LNX)           CHLTYPE(SDR)
   BATCHES(0)                            BATCHSZ(50)
   BUFSRCVD(2)                           BUFSSENT(2)
   BYTSRCVD(472)                         BYTSSENT(472)
   CHSTADA(2016-06-14)                   CHSTATI(08.27.10)
   COMPHDR(NONE,NONE)                    COMPMSG(NONE,NONE)
   COMPRATE(0,0)                         COMPTIME(0,0)
   CONNAME(9.30.145.117(1419))           CURLUWID(05F85F5710000101)
   CURMSGS(0)                            CURRENT
   CURSEQNO(0)                           EXITTIME(0,0)
   HBINT(300)                            INDOUBT(NO)
   JOBNAME(0000185C000008B8)             LOCLADDR(9.76.138.138(58347))
   LONGRTS(999999999)                    LSTLUWID(0000000000000000)
   LSTMSGDA( )                           LSTMSGTI( )
```

LSTSEQNO(0) MCASTAT(RUNNING)
MONCHL(OFF) MSGS(0)
NETTIME(0,0) NPMSPEED(FAST)
RQMNAME(QM80LNX) SHORTRTS(10)
SSLCERTI(CN=QM80LNX,O=IBM,C=USA) SSLKEYDA( )
SSLKEYTI( )
SSLPEER(SERIALNUMBER=51:73:46:05:81:9B:AE:BA,CN=QM80LNX,O=IBM,C=USA)
SSLRKEYS(0) **STATUS(RUNNING)**
STOPREQ(NO) SUBSTATE(MQGET)
XBATCHSZ(0,0) XMITQ(QM80LNX)
XQTIME(0,0) RVERSION(08000004)
RPRODUCT(MQMM)

++
Step 13: Test of sending message from Client (Windows) to Server (Linux)
++

Host: Windows

Use sample amqsput to put a message into the remote queue definition Q6_QM80LNX

C:\var\mqm\Qmgrs\QM75WIN\ssl> amqsput Q6_QM80LNX QM75WIN
Sample AMQSPUT0 start
target queue is Q6_QM80LNX
Testing of SSL channel
Sample AMQSPUT0 end

Host: Linux

Use sample amqsget to get the message sent from Windows

mqm@SUSE-host4: /var/mqm/qmgrs/QM80LNX/ssl
$ amqsget Q6 QM80LNX
Sample AMQSGET0 start
message <Testing of SSL channel>
no more messages
Sample AMQSGET0 end

Result:
The message with contents:
  Testing of SSL channel
... was successfully sent from the Client queue manager QM75WIN in Windows, to the Server queue manager QM80LNX in Linux.

++ APPENDIX ++ Brief description of using the GUI: IBM Key Management (iKeyman)

This section shows briefly how to perform the equivalent tasks in the GUI for the actions performed with the runmqakm commands described in this techdoc.

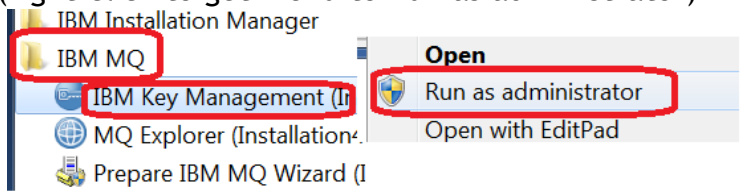
Note: The name of the queue manager and the name of the client key database are different in these examples (because it was convenient).

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

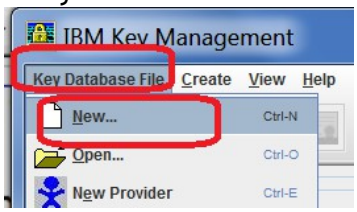
cd C:\var\mqm\Qmgrs\QM75W\ssl

runmqakm -keydb -create -db "C:\var\mqm\Qmgrs\QM75WIN\ssl\key.kdb"
  -pw clientpass -type cms -expire 365 -stash

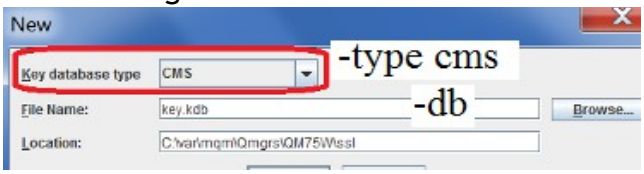
Start the GUI: (right-click to get menu to Run as administrator)



Create SSL client key database: Key Database File > New ...

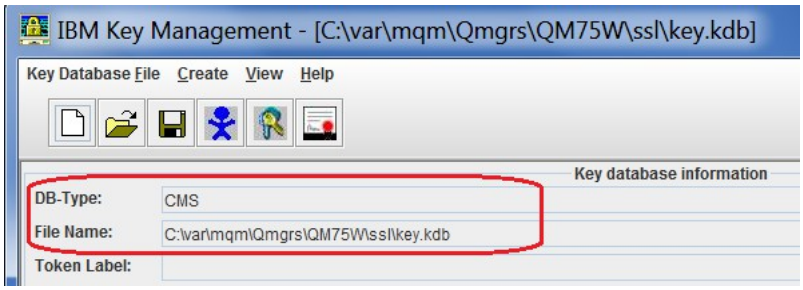


Specify: Key database type: CMS
... and other fields. Click OK to get next screen.



Then enable the checkboxes and press OK
(*) Expiration time ===> Enter the desired days, in this case: 365
(*) Stash password to a file

You will see the following data in the top section: Key database information
DB-Type: CMS
File Name: C:\var\mqm\Qmgrs\QM75WIN\ssl\key.kdb



The result is the creation of 3 files:

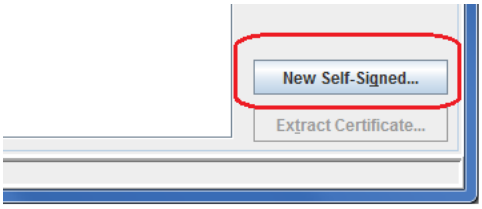
```
C:\var\mqm\Qmgrs\QM75W\ssl> dir
08/16/2016  03:22 PM                88 key.kdb
08/16/2016  03:22 PM                80 key.rdb
08/16/2016  03:22 PM               129 key.sth
```

Step 2: Client (Windows): Create certificate: self-signed

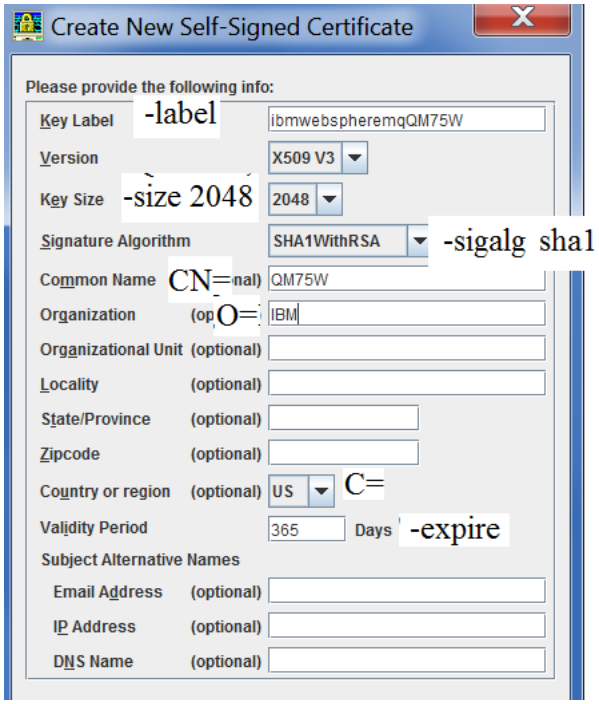
+ Create certificate

```
runmqakm -cert -create -db "C:\var\mqm\Qmgrs\QM75W\ssl\key.kdb" -pw clientpass
-label ibmwebspheremqQM75W -dn "CN=QM75W,O=IBM,C=USA" -expire 365  -sigalg
sha1 -size 2048
```

Click on "New Self-Signed..."

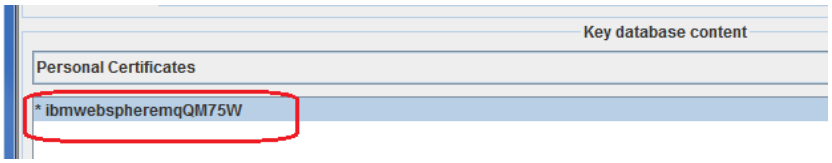


Provide the desired values:



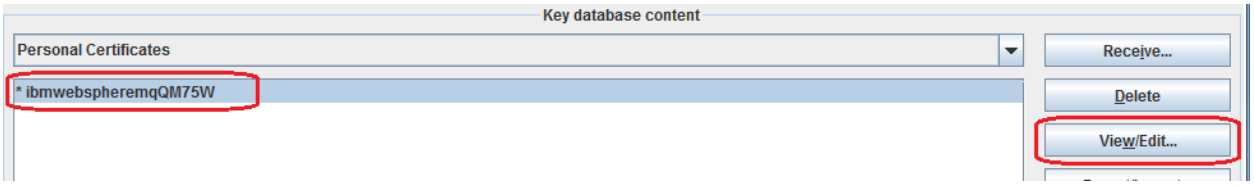
+ List newly created SSL certificate in Windows
runmqakm -cert -list -db "C:\var\mqm\Qmgrs\QM75W\ssl\key.kdb" -pw clientpass

After creating a personal, self-signed certificate, it will be show in the section "Key database content", under "Personal Certificates".

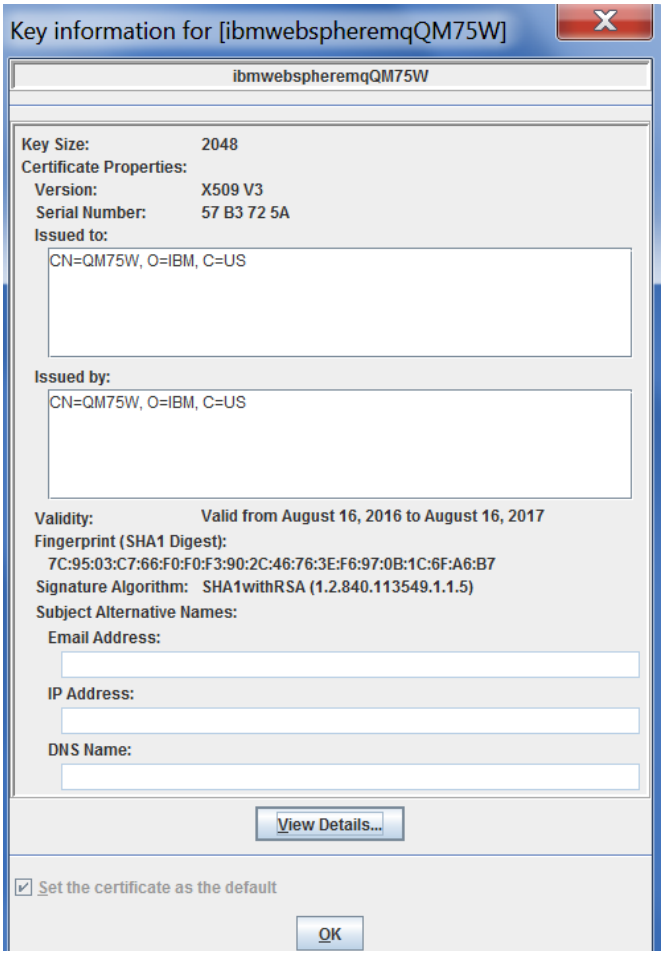


+ List the details of the certificate.
runmqakm -cert -details -db "C:\var\mqm\Qmgrs\QM75W\ssl\key.kdb" -pw clientpass -label ibmwebspheremqQM75W

Select the desired certificate and click the button "View/Edit..."

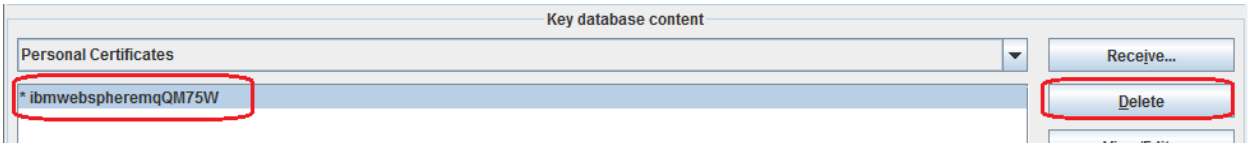


A window will be shown with the details:



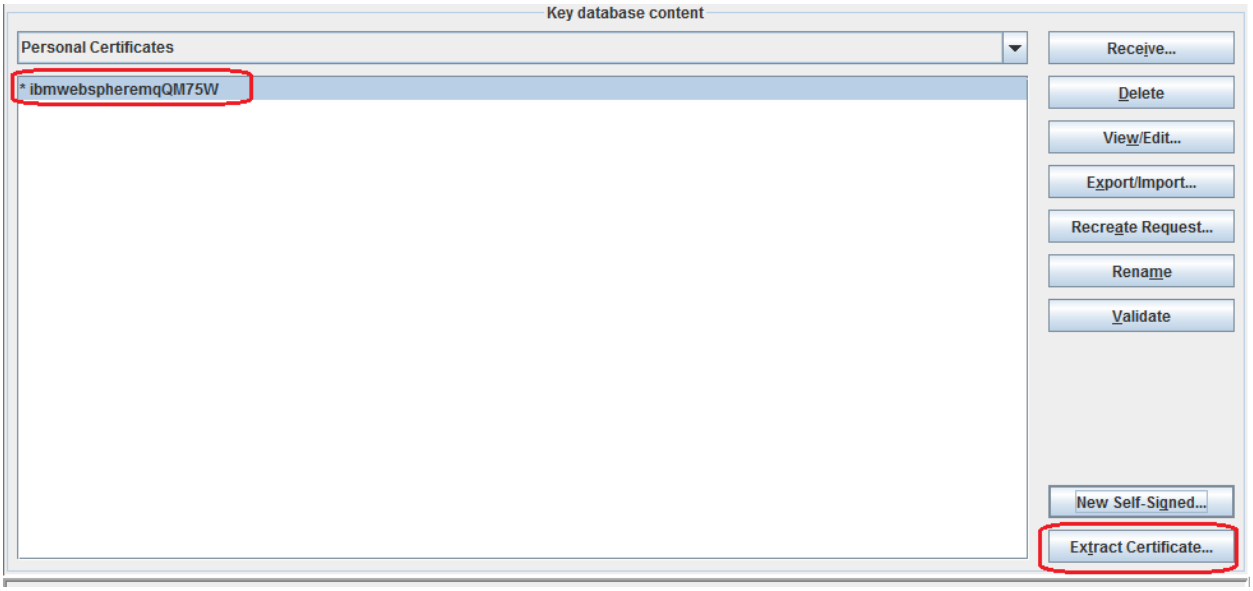
+ REFERENCE: In case that you need to delete the certificate (we are providing this command just for completeness, you do NOT need to issue it for this tutorial)
runmqakm -cert -delete -db "C:\var\mqm\Qmgrs\QM75W\ssl\key.kdb" -pw clientpass -label <label>

Select the desired certificate and click "Delete"

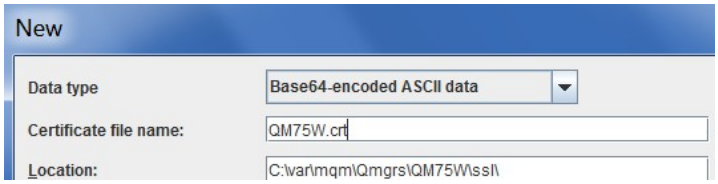


Step 3: Client (Windows): Extract the public SSL client certificate
runmqakm -cert -extract -db "C:\var\mqm\Qmgrs\QM75W\ssl\key.kdb" -pw clientpass -label ibmwebspheremqQM75W -target QM75W.crt -format ascii

Select the desired certificate and click "Extract Certificate..."



Then enter the desired Certificate file name.
Note that the "Data type" of "Base64-enconded ASCII data" corresponds to the flag "-format ascii".

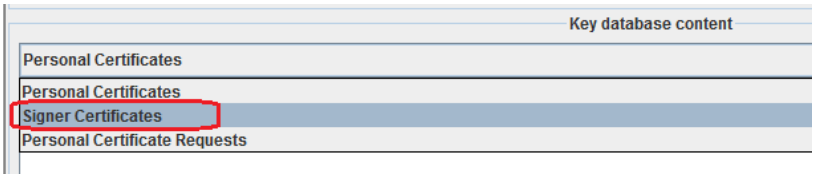


Step 11: Client (Windows): Add the Linux certificate to the Windows key database

+ Add the public/signer certificate
runmqakm -cert -add -db "C:\var\mqm\Qmgrs\QM75W\ssl\key.kdb" -pw clientpass -label ibmwebspheremqqm80lnx -file QM80LNX.crt -format ascii

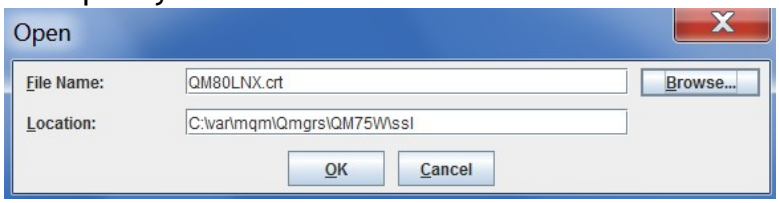
Hum... there is no button "Add" !

Indeed! Under the "Personal Certificates" view of the Key database, there is no "Add".



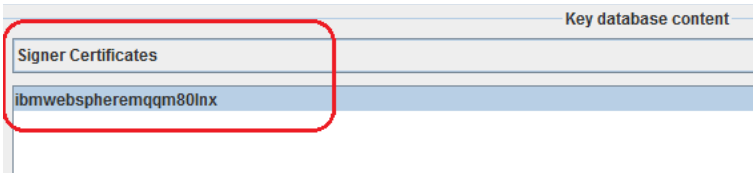
BUT, there is an "Add" under the "Signer Certificates" view!

Click "Add" and then specify the file name:

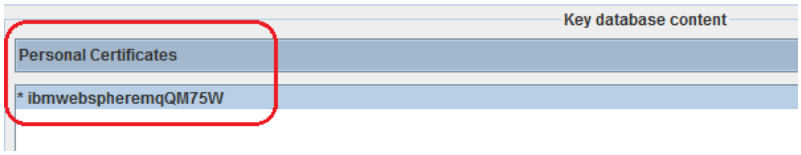


And specify the label:

You will see the added certificate (under the view for "Signer Certificates").



To see the personal self-signed certificate, you have to change the view to "Personal Certificates"



.
+++ end +++