IBM Z TechBytes:
MQ for z/OS Security 101

Dorothy Quincy
Lyn Elkins

Washington Systems Center

Agenda

MQ RACF basics

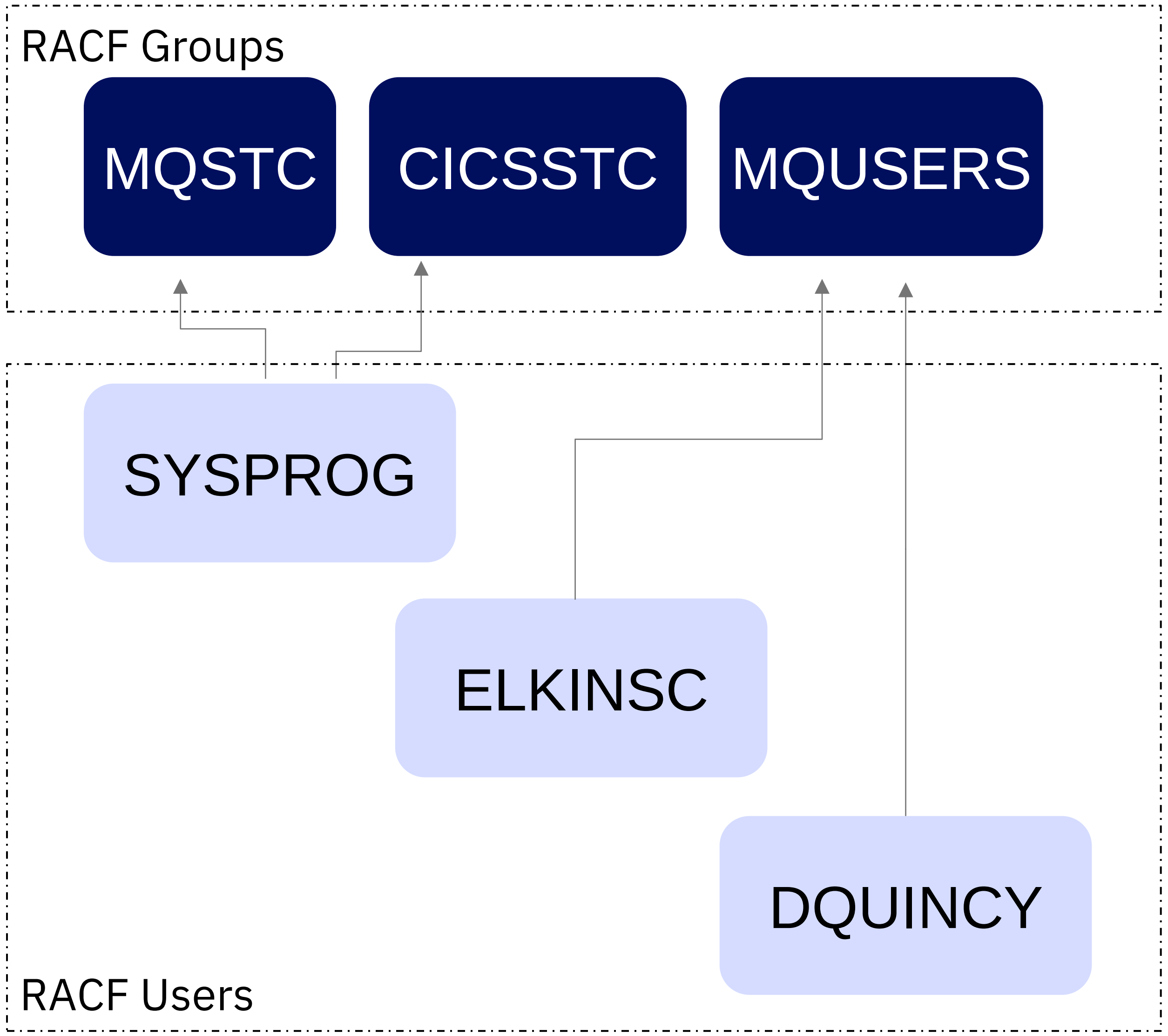
Channel encryption – securing communication

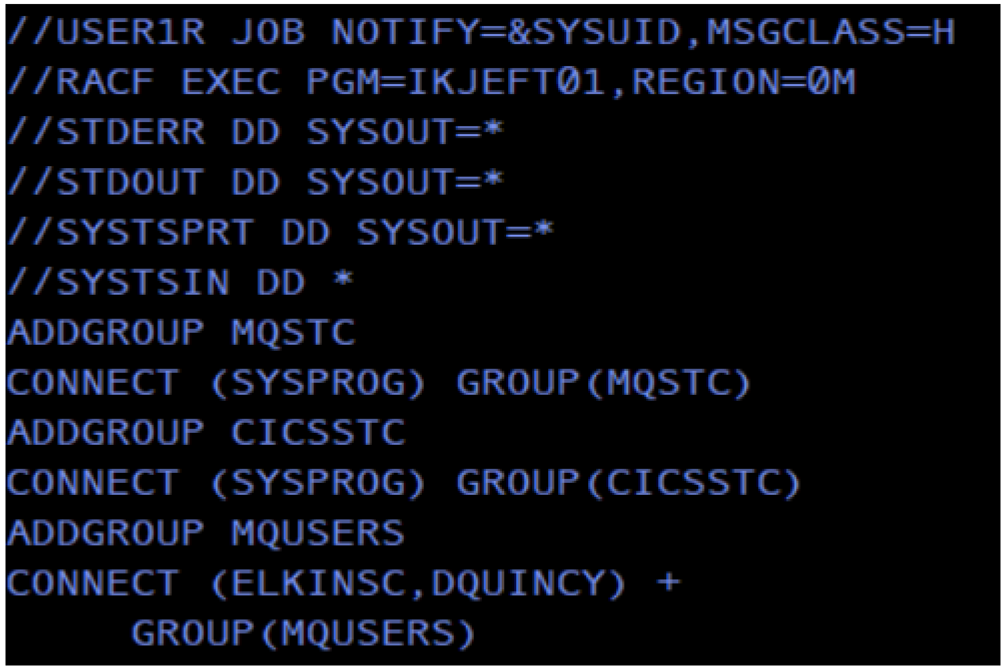
Advanced message security – security end-to-end

Data set encryption – securing data sets

Performance considerations and what’s new

RACF Groups and Users





```
//USER1R JOB NOTIFY=&SYSUID,MSGCLASS=H
//RACF EXEC PGM=IKJEFT01,REGION=0M
//STDERR DD SYSOUT=*
//STDOUT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
ADDGROUP MQSTC
CONNECT (SYSPROG) GROUP(MQSTC)
ADDGROUP CICSSTC
CONNECT (SYSPROG) GROUP(CICSSTC)
ADDGROUP MQUSERS
CONNECT (ELKINSC,DQUINCY) +
     GROUP(MQUSERS)
```

Concept check: enabling security

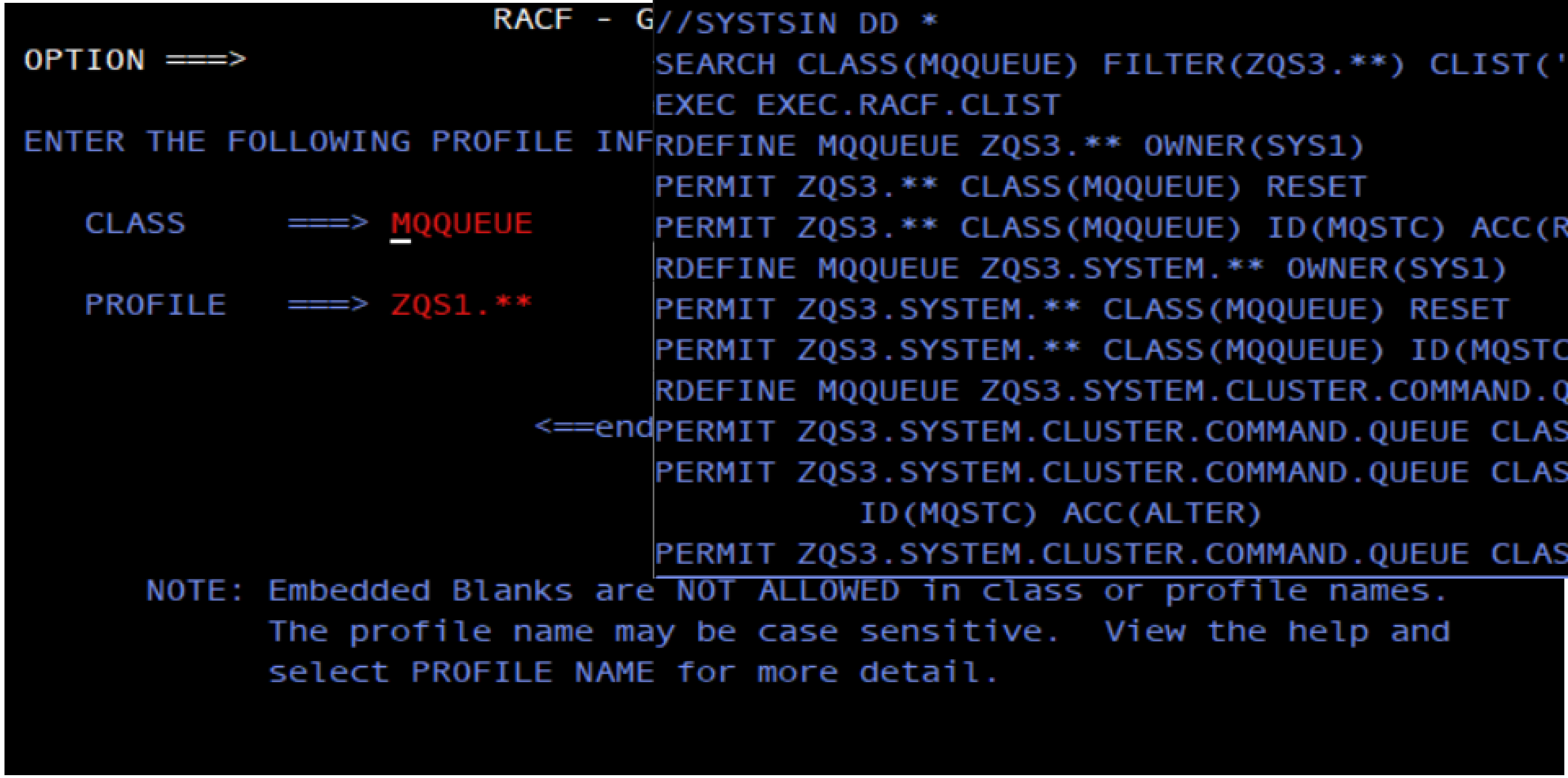
```
ZQS1 DISPLAY SECURITY
CSQH015I ZQS1 Security timeout = 54 minutes
CSQH016I ZQS1 Security interval = 12 minutes
CSQH037I ZQS1 Security using uppercase classes
CSQH030I ZQS1 Security switches ...
CSQH031I ZQS1 SUBSYSTEM: OFF,
'ZQS1.NO.SUBSYS.SECURITY' found
CSQH040I ZQS1 Connection authentication ...
CSQH041I ZQS1 Client checks: OPTIONAL
CSQH042I ZQS1 Local bindings checks: OPTIONAL
CSQ9022I ZQS1 CSQHPDTC ' DISPLAY SECURITY' NORMAL
COMPLETION
```

Enable security

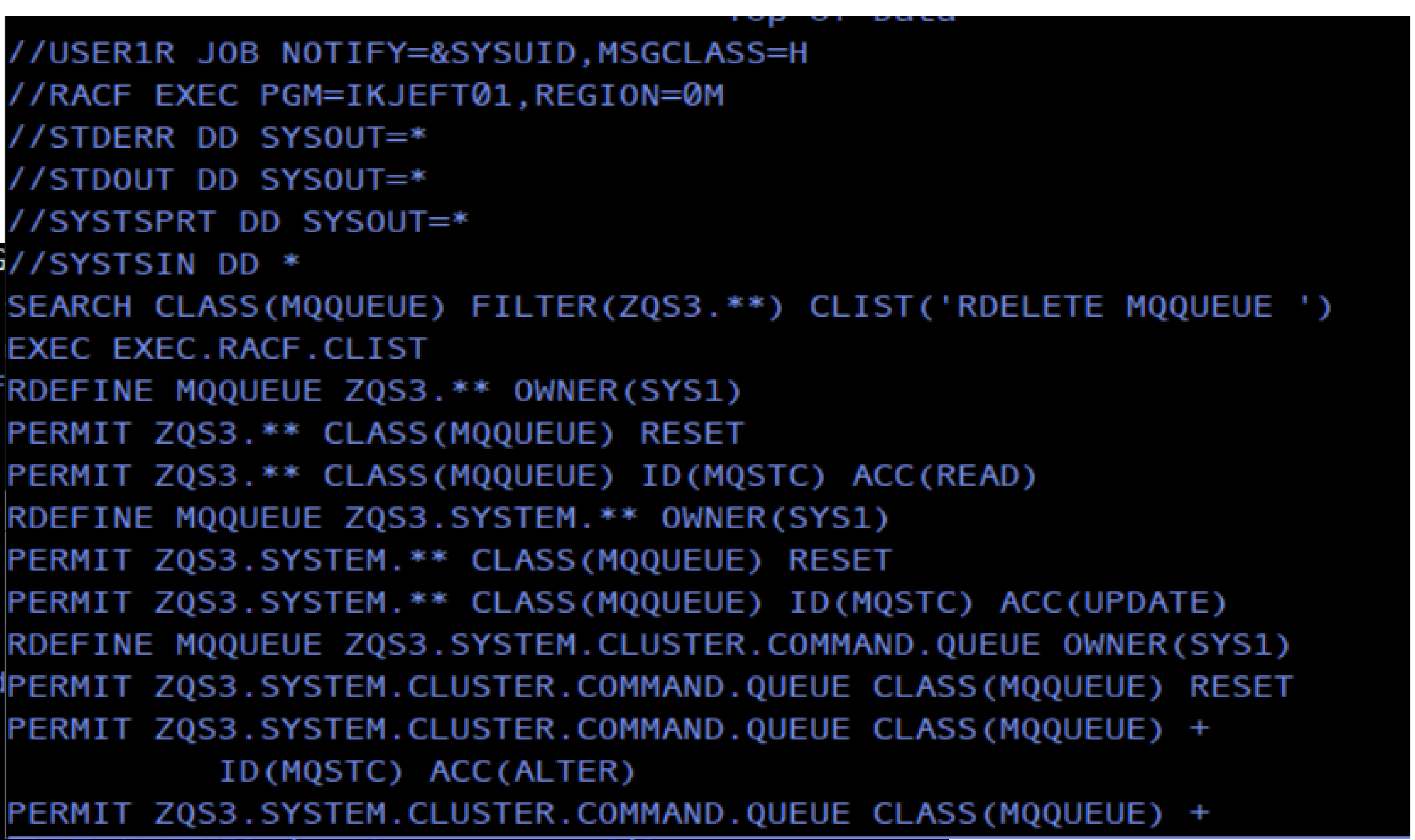
```
CSQH024I ZQS1 CSQHINIT SUBSYSTEM security switch
set ON, profile 'ZQS1.NO.SUBSYS.SECURITY' not found
CSQH024I ZQS1 CSQHINIT CONNECTION security switch
set ON, profile 'ZQS1.NO.CONNECT.CHECKS' not found
CSQH024I ZQS1 CSQHINIT COMMAND security switch set
ON, profile 'ZQS1.NO.CMD.CHECKS' not found CSQH021I
ZQS1 CSQHINIT CONTEXT security switch set OFF,
profile 'ZQS1.NO.CONTEXT.CHECKS' found CSQH021I
ZQS1 CSQHINIT ALTERNATE USER security switch set
OFF, profile 'ZQS1.NO.ALTERNATE.USER.CHECKS' found
CSQH021I ZQS1 CSQHINIT COMMAND RESOURCES security
switch set OFF, profile 'ZQS1.NO.CMD.RESC.CHECKS'
found CSQH021I ZQS1 CSQHINIT PROCESS security
switch set OFF, profile 'ZQS1.NO.PROCESS.CHECKS'
found CSQH021I ZQS1 CSQHINIT NAMELIST security
switch set OFF, profile 'ZQS1.NO.NLIST.CHECKS'
found CSQH024I ZQS1 CSQHINIT QUEUE security switch
set ON, profile 'ZQS1.NO.QUEUE.CHECKS' not found
CSQH021I ZQS1 CSQHINIT TOPIC security switch set
OFF, profile 'ZQS1.NO.TOPIC.CHECKS' found
```

MQ Security Profiles

Security Profile	Corresponds to...	Example of security ON	Example of security OFF
MQADMIN or MXADMIN	All	-	ZQS3.NO.SUBSYS.SECURITY
MQCONN	Connection security	PERMIT ZQS3.BATCH CLASS(MQCONN) ID(MQSTC,MQUSERS) ACC(READ)	ZQS3.NO.CONNECT.CHECKS
MQPROC or MXPROC	Process security	-	ZQS3.NO.PROCESS.CHECKS
MQCMDS	Command security	PERMIT ZQS3.DEFINE.** CLASS(MQCMDS) ID(MQSTC,MQSYSP) ACC(ALTER)	ZQS3.NO.CMD.CHECKS
MQQUEUE or MXQUEUE	Queue security	PERMIT ZQS3.SYSTEM.** CLASS(MQQUEUE) RESET PERMIT ZQS3.SYSTEM.** CLASS(MQQUEUE) ID(MQSTC) ACC(UPDATE)	-
MQNLIST or MXNLIST	Namelist security	-	ZQS3.NO.NLIST.CHECKS
MXTOPIC	Topic security	-	ZQS3.NO.TOPIC.CHECKS



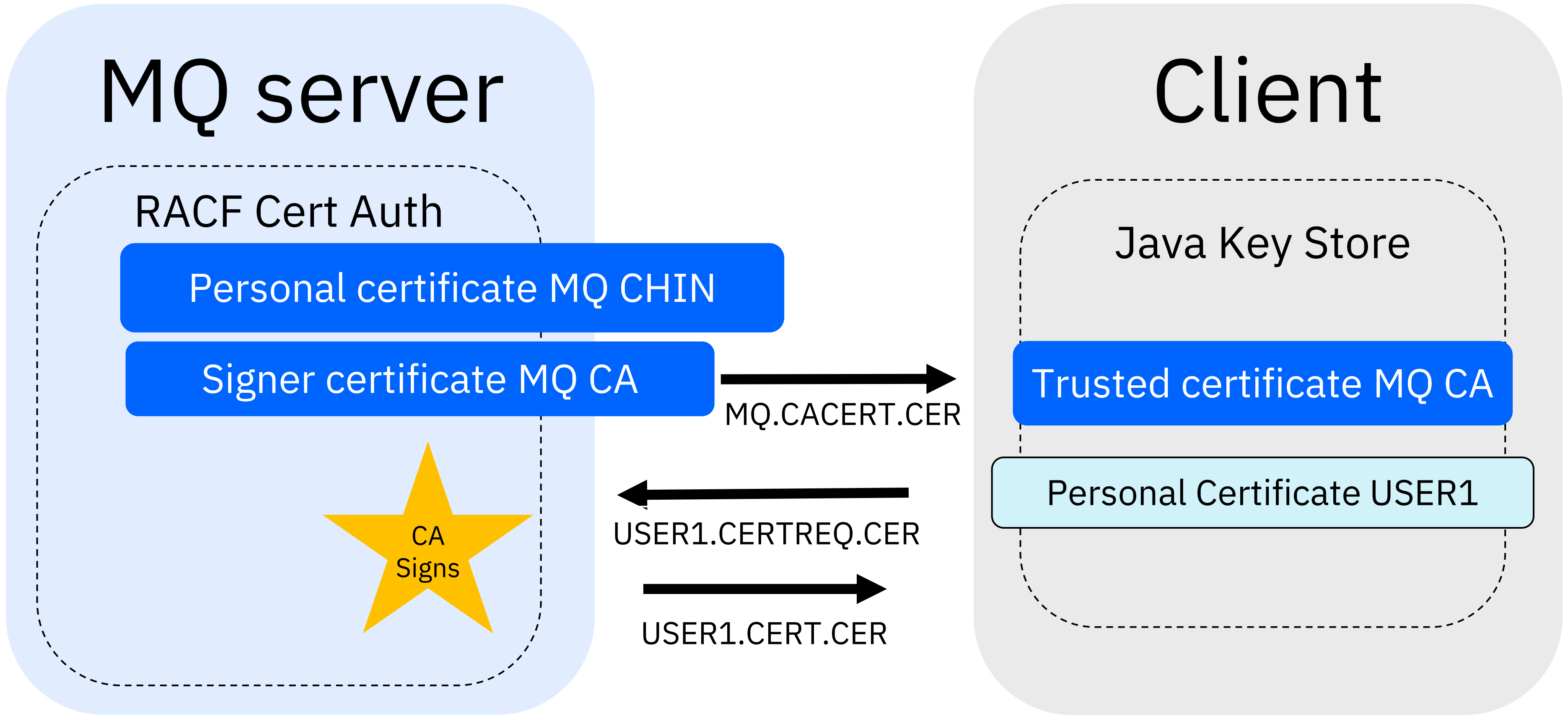
```
                        RACF - G
OPTION ===>

ENTER THE FOLLOWING PROFILE INF

   CLASS      ===> MQQUEUE

   PROFILE    ===> ZQS1.**


                            <==end



         NOTE: Embedded Blanks are NOT ALLOWED in class or profile names.
               The profile name may be case sensitive.  View the help and
               select PROFILE NAME for more detail.
```



```
//USER1R JOB NOTIFY=&SYSUID,MSGCLASS=H
//RACF EXEC PGM=IKJEFT01,REGION=0M
//STDERR DD SYSOUT=*
//STDOUT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
SEARCH CLASS(MQQUEUE) FILTER(ZQS3.**) CLIST('RDELETE MQQUEUE ')
EXEC EXEC.RACF.CLIST
RDEFINE MQQUEUE ZQS3.** OWNER(SYS1)
PERMIT ZQS3.** CLASS(MQQUEUE) RESET
PERMIT ZQS3.** CLASS(MQQUEUE) ID(MQSTC) ACC(READ)
RDEFINE MQQUEUE ZQS3.SYSTEM.** OWNER(SYS1)
PERMIT ZQS3.SYSTEM.** CLASS(MQQUEUE) RESET
PERMIT ZQS3.SYSTEM.** CLASS(MQQUEUE) ID(MQSTC) ACC(UPDATE)
RDEFINE MQQUEUE ZQS3.SYSTEM.CLUSTER.COMMAND.QUEUE OWNER(SYS1)
PERMIT ZQS3.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) RESET
PERMIT ZQS3.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) +
          ID(MQSTC) ACC(ALTER)
PERMIT ZQS3.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) +
```

Setting up the infrastructure for TLS

At a high-level



On z/OS, generate the signer certificate

```
racdcert certauth gencert subjectsdn(CN('MQ CA')
OU('ATS') O('IBM') C('US')) withlabel('MQ CA')
keyusage(certsign) notafter(date(2029/12/31))
```

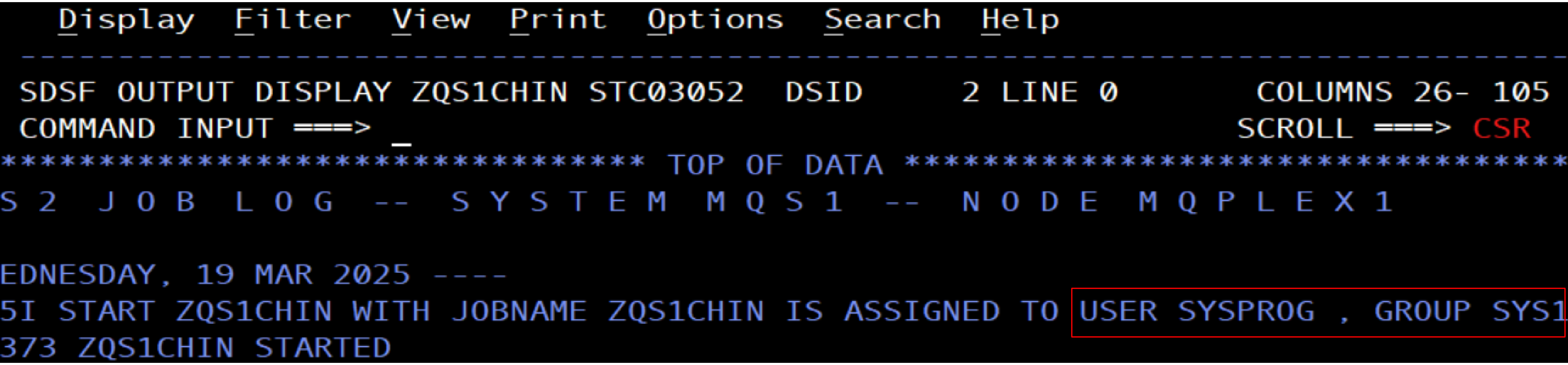
Signer certificate
MQ CA

On z/OS, generate the personal certificate

```
racdcert id(SYSPROG) gencert subjectsdn(CN('MQ CHIN')
OU('ATS') O('IBM') C('US')) withlabel('MQ CHIN')
signwith(certauth label('MQ CA'))
notafter(date(2029/12/31)
```

Personal certificate
MQ CHIN

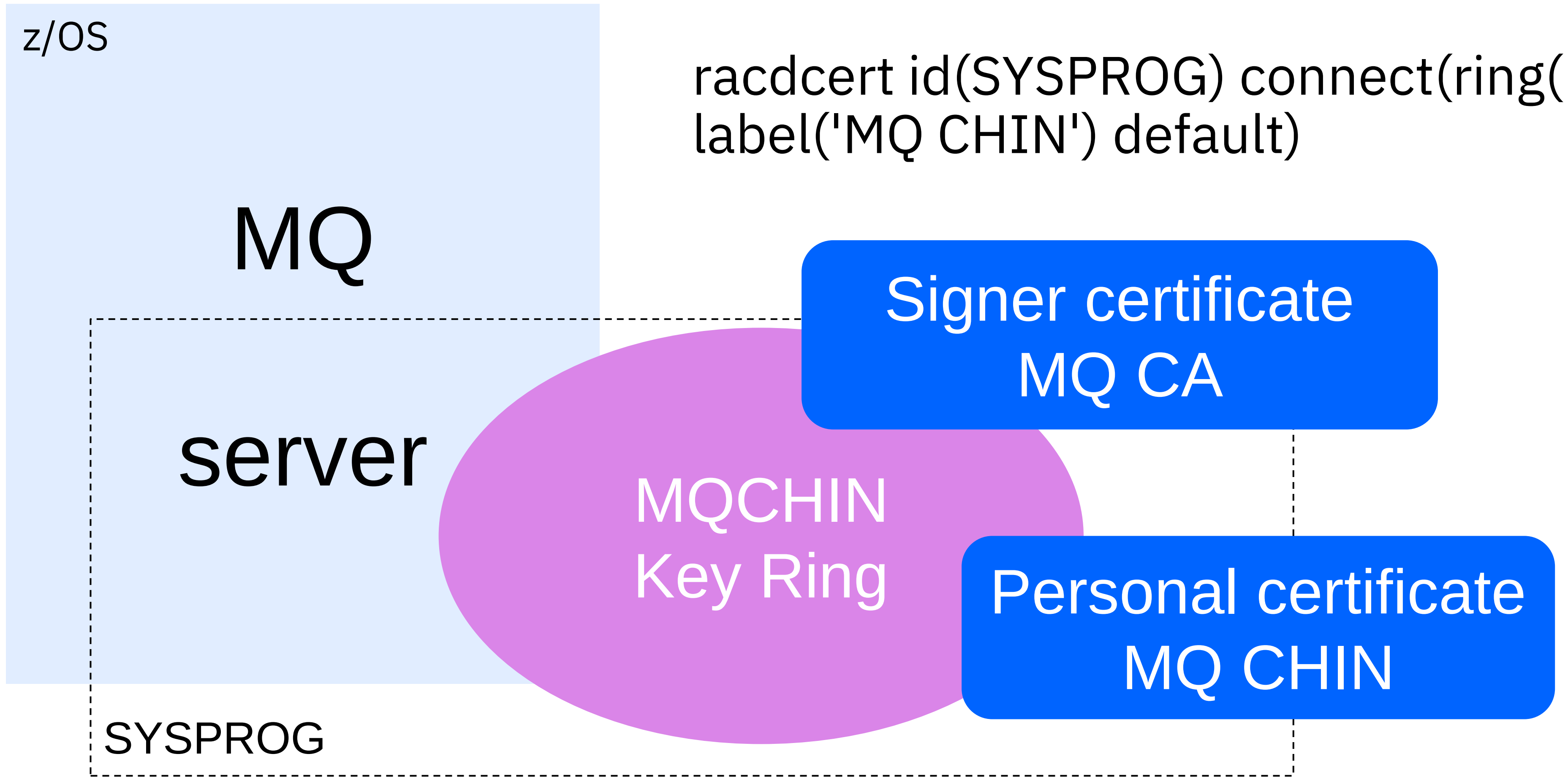
Why is the ID SYSPROG?



```
   Display  Filter  View  Print  Options  Search  Help
 ---------------------------------------------------------------------------------
 SDSF OUTPUT DISPLAY ZQS1CHIN STC03052  DSID     2 LINE 0       COLUMNS 26- 105
 COMMAND INPUT ===> _                                          SCROLL ===> CSR
********************************* TOP OF DATA **********************************
S 2  J O B  L O G  --  S Y S T E M  M Q S 1  --  N O D E  M Q P L E X 1

EDNESDAY, 19 MAR 2025 ----
5I START ZQS1CHIN WITH JOBNAME ZQS1CHIN IS ASSIGNED TO USER SYSPROG , GROUP SYS1
373 ZQS1CHIN STARTED
```

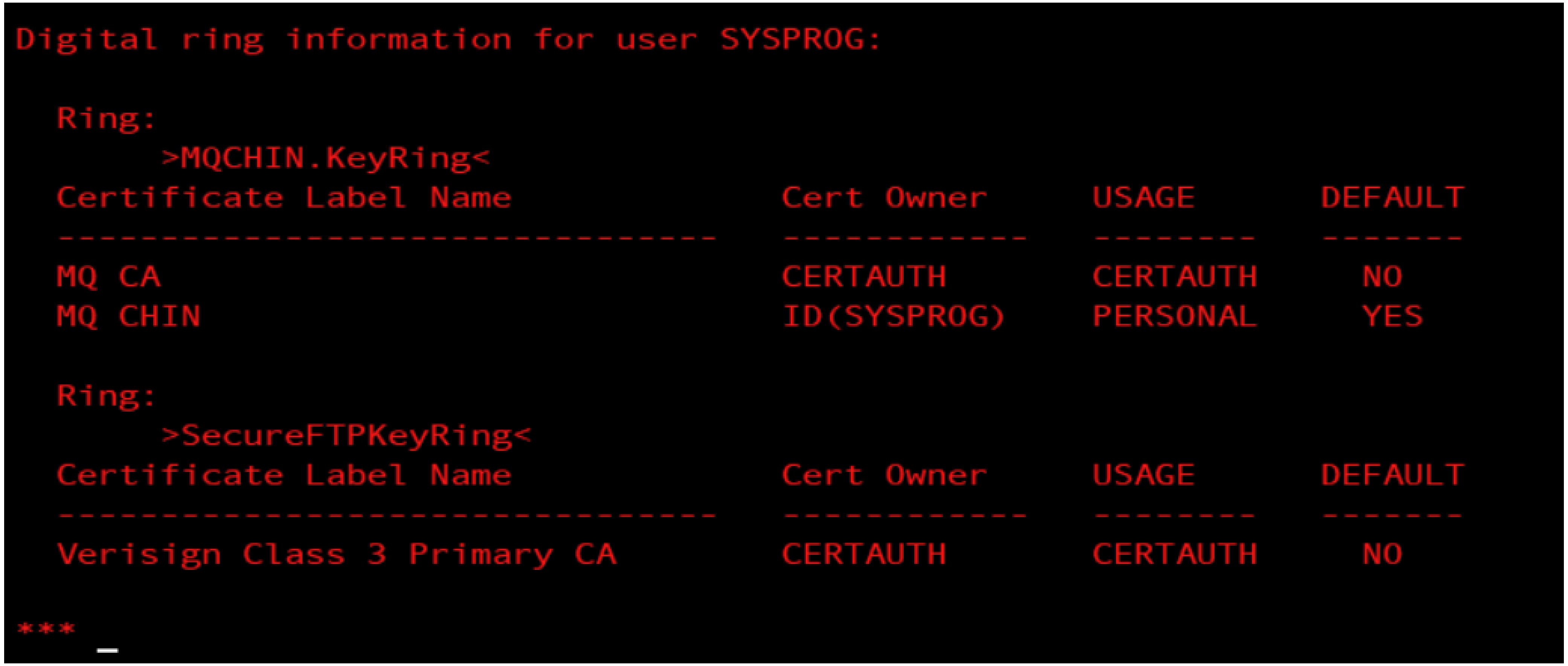
On z/OS, create keyring and add certificates to keyring

```
racdcert id(SYSPROG) addring(MQCHIN.KeyRing)

racdcert id(SYSPROG) connect(ring(MQCHIN.KeyRing)
label('MQ CA') certauth usage(certauth))

racdcert id(SYSPROG) connect(ring(MQCHIN.KeyRing)
label('MQ CHIN') default)
```



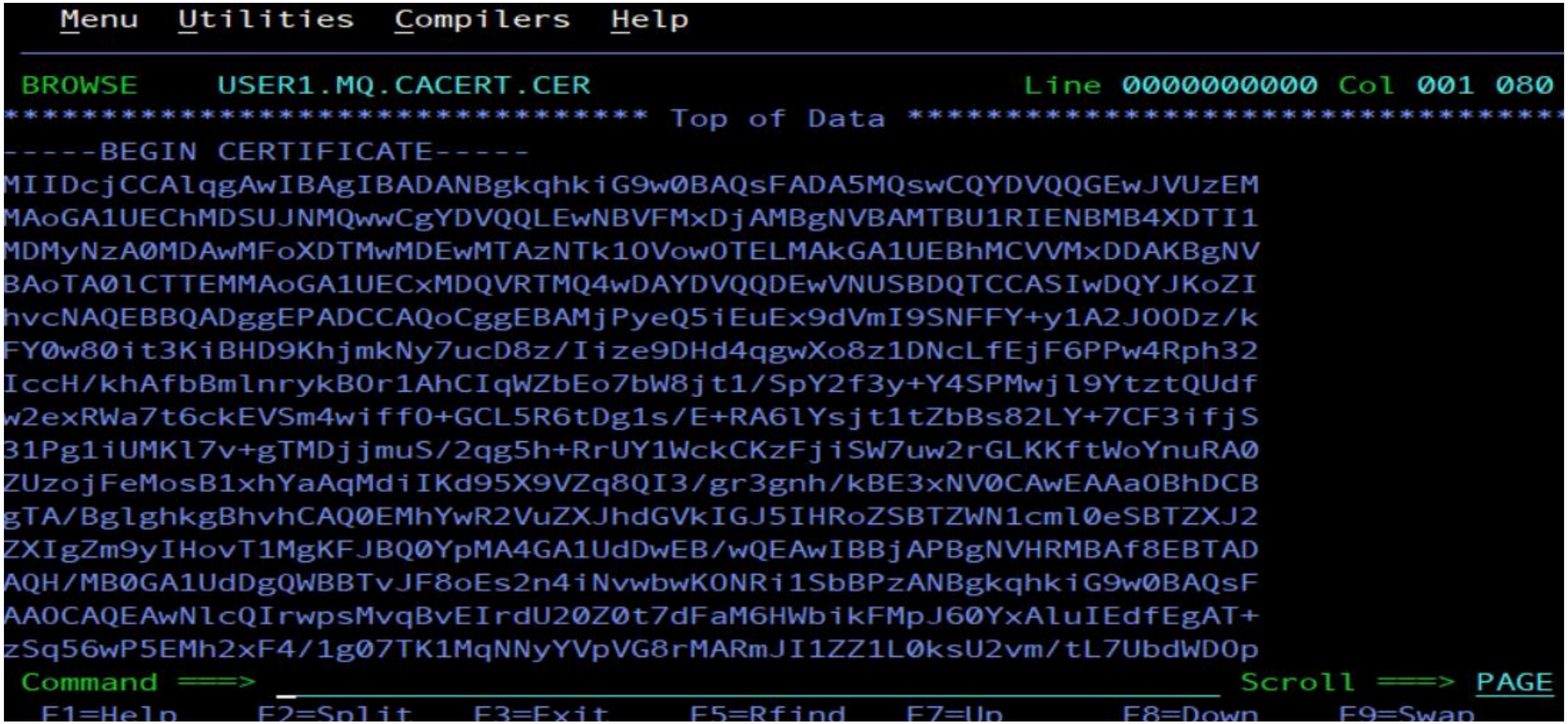
List certificates for SYSPROG’s key ring:

racdcert id(SYSPROG) listring(MQCHIN.KeyRing)



```
Digital ring information for user SYSPROG:

  Ring:
       >MQCHIN.KeyRing<
  Certificate Label Name                 Cert Owner     USAGE      DEFAULT
  --------------------------------       ------------   ---------  -------
  MQ CA                                  CERTAUTH       CERTAUTH     NO
  MQ CHIN                                ID(SYSPROG)    PERSONAL     YES

  Ring:
       >SecureFTPKeyRing<
  Certificate Label Name                 Cert Owner     USAGE      DEFAULT
  --------------------------------       ------------   ---------  -------
  Verisign Class 3 Primary CA            CERTAUTH       CERTAUTH     NO

***
```

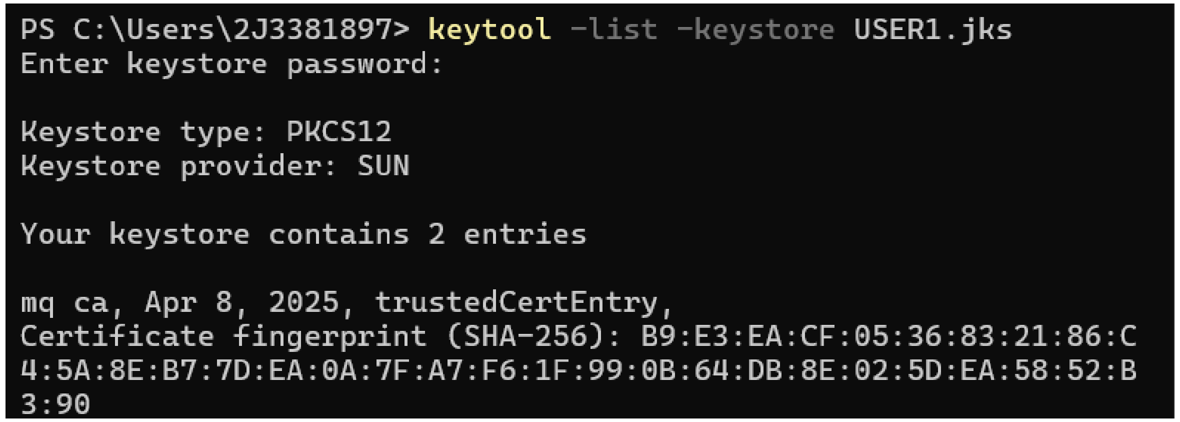
racdcert certauth export(label('MQ CA')) dsn(mq.cacert.cer)



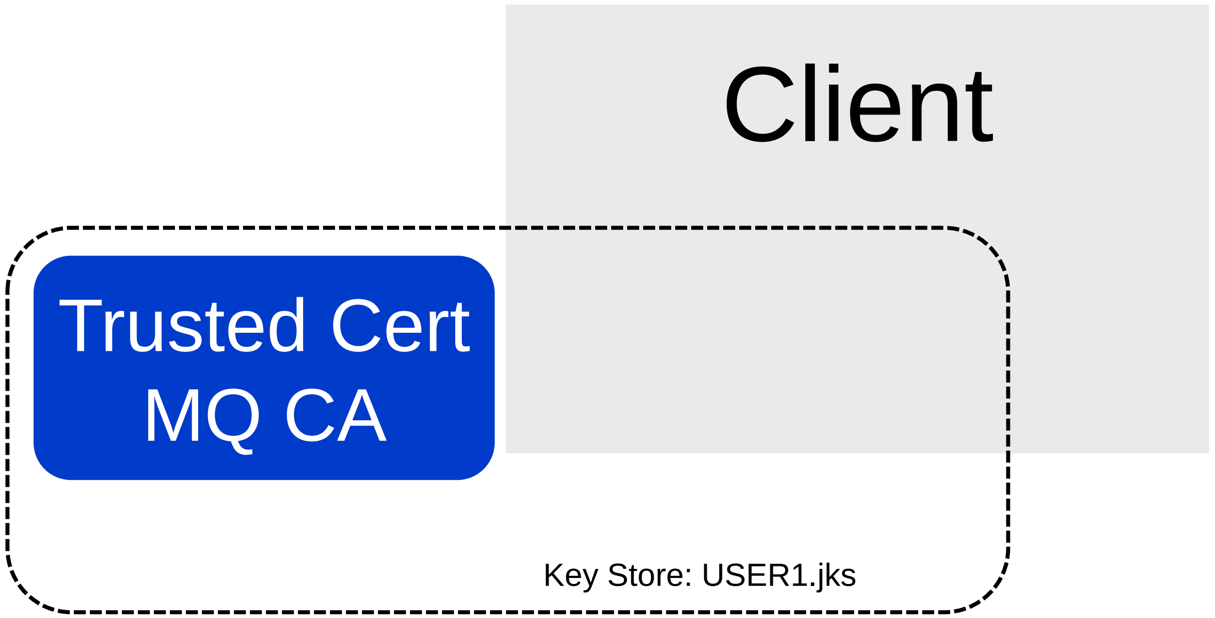
```
   Menu  Utilities  Compilers  Help

 BROWSE    USER1.MQ.CACERT.CER                  Line 0000000000 Col 001 080
********************************* Top of Data **********************************
-----BEGIN CERTIFICATE-----
MIIDcjCCAlqgAwIBAgIBADANBgkqhkiG9w0BAQsFADA5MQswCQYDVQQGEwJVUzEM
MAoGA1UEChMDSUJNMQwwCgYDVQQLEwNBVFMxDjAMBgNVBAMTBU1RIENBMB4XDTI1
MDMyNzA0MDAwMFoXDTMwMDEwMTAzNTk1OVowOTELMAkGA1UEBhMCVVMxDDAKBgNV
BAoTA0lCTTEMMAoGA1UECxMDQVRTMQ4wDAYDVQQDEwVNUSBDQTCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBAMjPyeQ5iEuEx9dVmI9SNFFY+y1A2JO0Dz/k
FY0w80it3KiBHD9KhjmkNy7ucD8z/Iize9DHd4qgwXo8z1DNcLfEjF6PPw4Rph32
IccH/khAfbBmlnrykB0r1AhCIqWZbEo7bW8jt1/SpY2f3y+Y4SPMwjl9YtztQUdf
w2exRWa7t6ckEVSm4wiff0+GCL5R6tDg1s/E+RA6lYsjt1tZbBs82LY+7CF3ifjS
31Pg1iUMKl7v+gTMDjjmuS/2qg5h+RrUY1WckCKzFjiSW7uw2rGLKKftWoYnuRA0
ZUzojFeMosB1xhYaAqMdiIKd95X9VZq8QI3/gr3gnh/kBE3xNV0CAwEAAaOBhDCB
gTA/BglghkgBhvhCAQ0EMhYwR2VuZXJhdGVkIGJ5IHRoZSBTZWN1cml0eSBTZXJ2
ZXIgZm9yIHovT1MgKFJBQ0YpMA4GA1UdDwEB/wQEAwIBBjAPBgNVHRMBAf8EBTAD
AQH/MB0GA1UdDgQWBBTvJF8oEs2n4iNvwbwK0NRi1SbBPzANBgkqhkiG9w0BAQsF
AAOCAQEAwNlcQIrwpsMvqBvEIrdU20Z0t7dFaM6HWbikFMpJ60YxAluIEdfEgAT+
zSq56wP5EMh2xF4/1g07TK1MqNNyYVpVG8rMARmJI1ZZ1L0ksU2vm/tL7UbdWD0p
 Command ===>                                                 Scroll ===> PAGE
  F1=Help     F2=Split    F3=Exit     F5=Rfind    F7=Up       F8=Down     F9=Swap
```

Export the CA certificate to workstation

1. sftp user1@zos
   - cd //’USER1’
   - ls /+mode=text
   - mget mq.cacert.cer
2. keytool –import –v –trustcacerts –alias “MQ CA” –file MQ.CACERT.CER –keystore USER1.jks

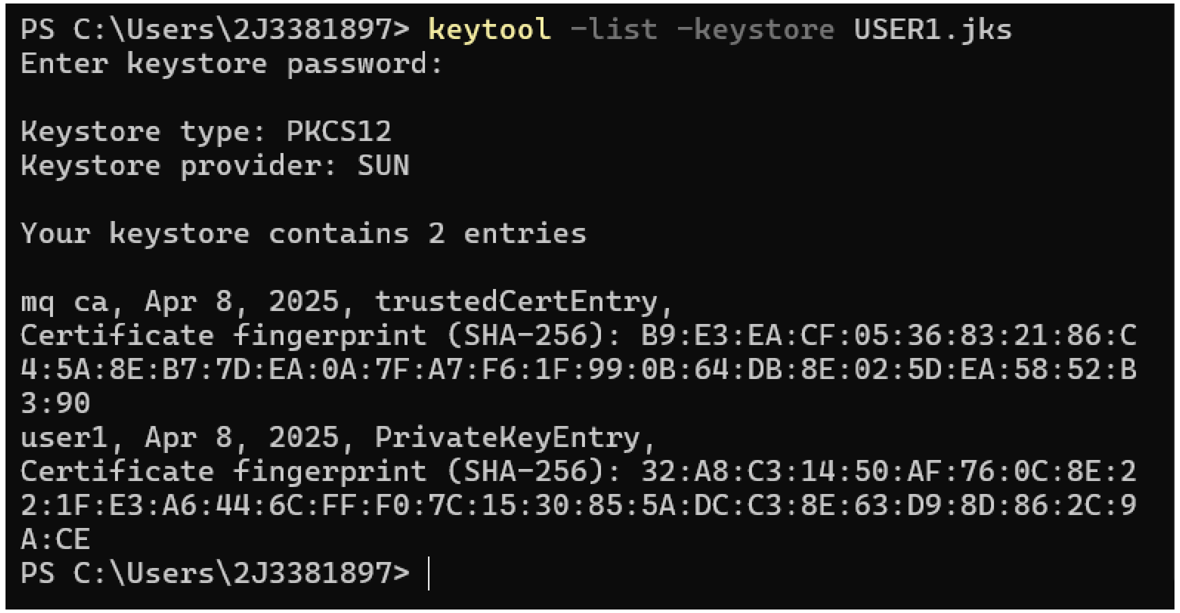


```
PS C:\Users\2J3381897> keytool -list -keystore USER1.jks
Enter keystore password:

Keystore type: PKCS12
Keystore provider: SUN

Your keystore contains 2 entries

mq ca, Apr 8, 2025, trustedCertEntry,
Certificate fingerprint (SHA-256): B9:E3:EA:CF:05:36:83:21:86:C
4:5A:8E:B7:7D:EA:0A:7F:A7:F6:1F:99:0B:64:DB:8E:02:5D:EA:58:52:B
3:90
```

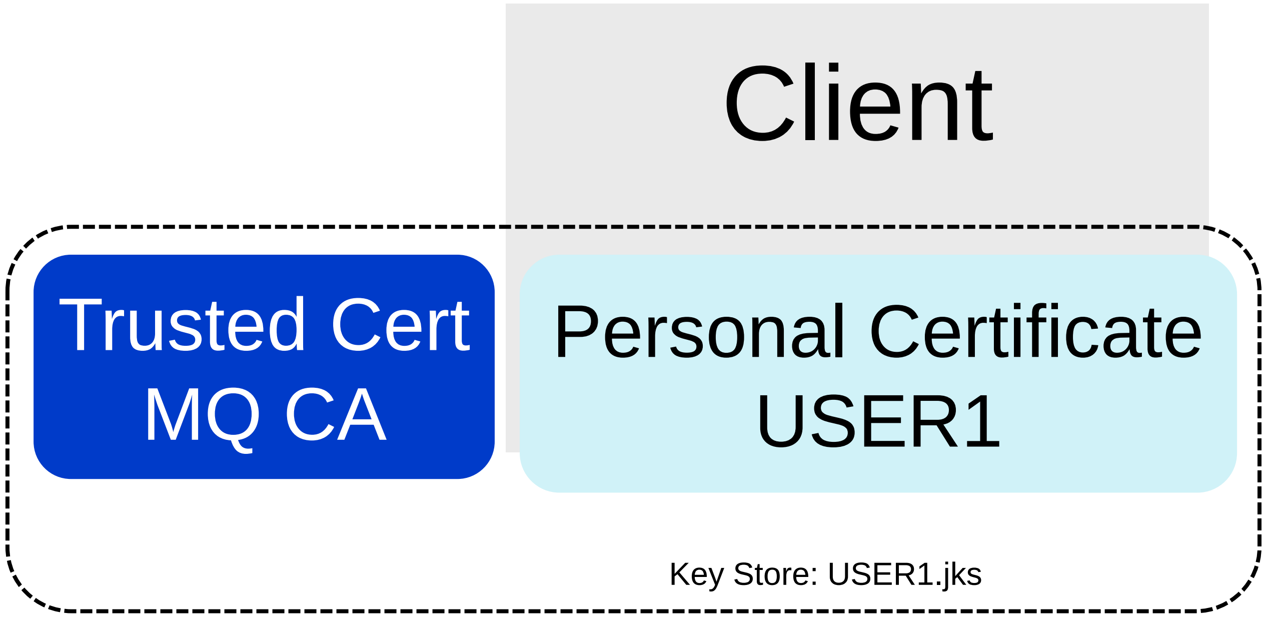


On the client-side, generate self-signed certificate and export certificate to certificate request file to a certificate authority

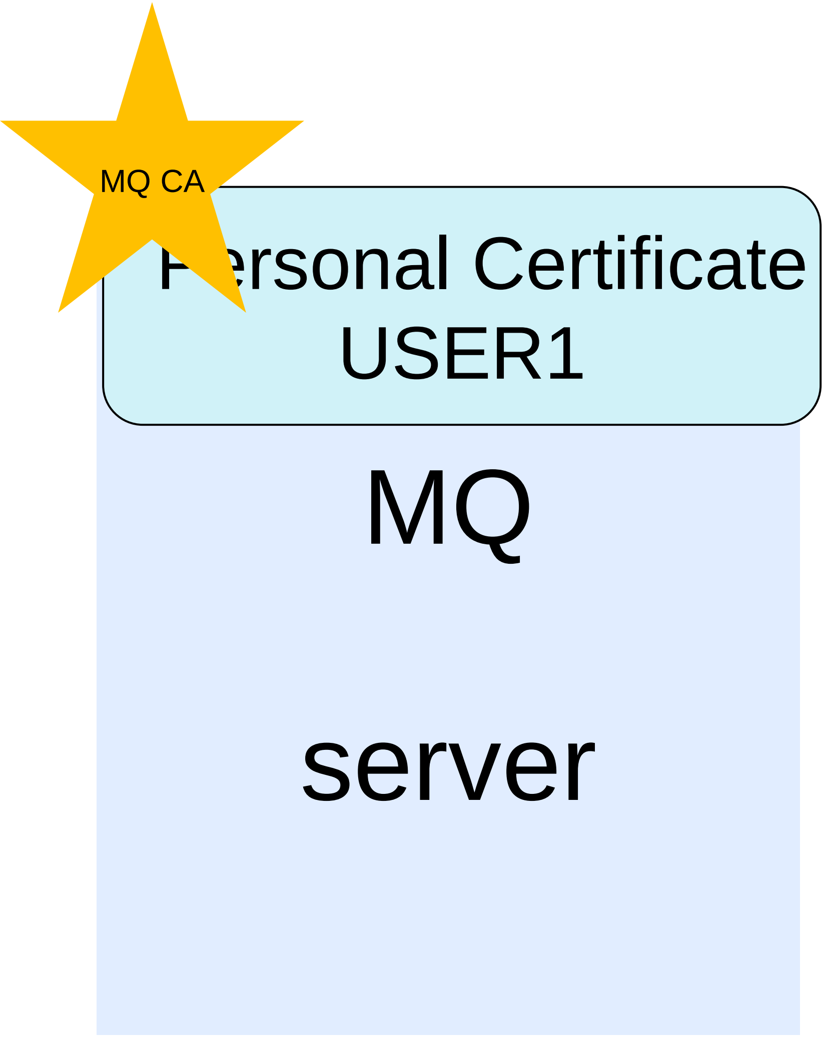
1. keytool –genkeypair –alias “USER1” –dname “CN=USER1, OU=ATS, O=IBM, C=US” –keystore USER1.jks –keyalg RSA
2. keytool –certreq –alias “USER1” –file certreq.cer keystore USER1.jks



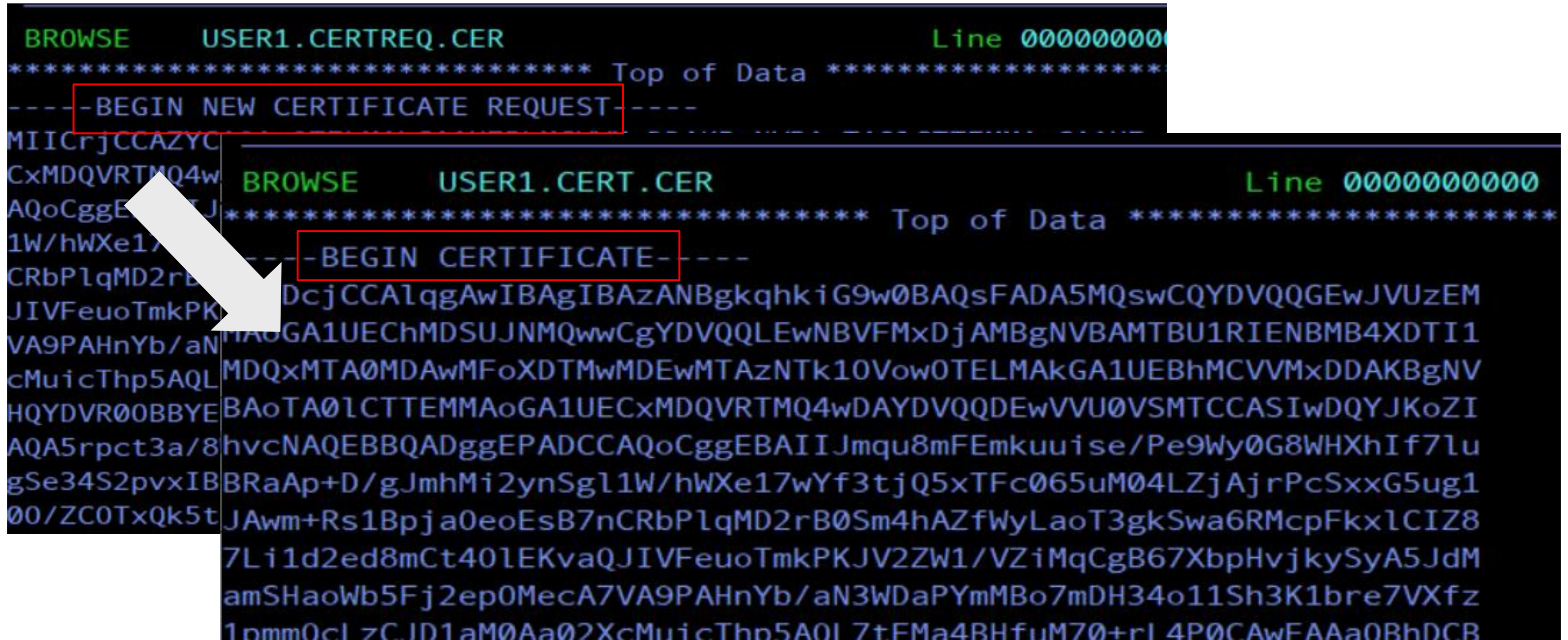
```
PS C:\Users\2J3381897> keytool -list -keystore USER1.jks
Enter keystore password:

Keystore type: PKCS12
Keystore provider: SUN

Your keystore contains 2 entries

mq ca, Apr 8, 2025, trustedCertEntry,
Certificate fingerprint (SHA-256): B9:E3:EA:CF:05:36:83:21:86:C
4:5A:8E:B7:7D:EA:0A:7F:A7:F6:1F:99:0B:64:DB:8E:02:5D:EA:58:52:B
3:90
user1, Apr 8, 2025, PrivateKeyEntry,
Certificate fingerprint (SHA-256): 32:A8:C3:14:50:AF:76:0C:8E:2
2:1F:E3:A6:44:6C:FF:F0:7C:15:30:85:5A:DC:C3:8E:63:D9:8D:86:2C:9
A:CE
PS C:\Users\2J3381897>
```



Move CERTREQ.CER back to z/OS and sign it



1. sftp user1@zos
   - cd //’USER1’
   - ls /+mode=text,lrecl=256,recfm=vb,blksize=0
   - mput certreq.cer
2. racdcert id(USER1) gencert(certreq.cer) withlabel(‘USER1’) signwith(certauth label(‘MQ CA’)) notafter(date(2029/12/31))
3. racdcert id(USER1) export(label(‘USER1’)) dsn(cert.cer)

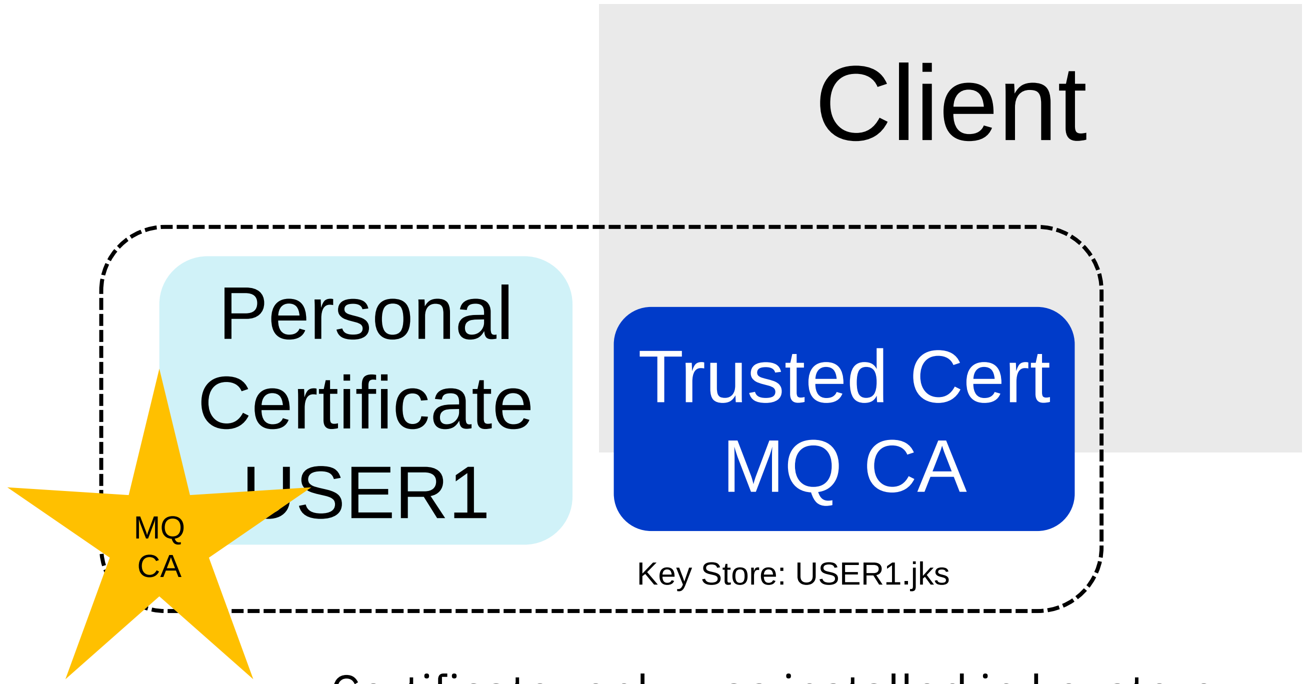


```
BROWSE    USER1.CERTREQ.CER                      Line 00000000
******************************** Top of Data *******************
-----BEGIN NEW CERTIFICATE REQUEST-----
MIICrjCCAZYC
CxMDQVRTMQ4w
AQoCggE
1W/hWXe1
CRbPlqMD2r
JIVFeuoTmkPK
VA9PAHnYb/aN
cMuicThp5AQL
HQYDVR00BBYE
AQA5rpct3a/8
gSe34S2pvxIB
00/ZCOTxQk5t
```

```
BROWSE    USER1.CERT.CER                              Line 0000000000
******************************** Top of Data **********************
-----BEGIN CERTIFICATE-----
   DcjCCAlqgAwIBAgIBAzANBgkqhkiG9w0BAQsFADA5MQswCQYDVQQGEwJVUzEM
MAoGA1UEChMDSUJNMQwwCgYDVQQLEwNBVFMxDjAMBgNVBAMTBU1RIENBMB4XDTI1
MDQxMTA0MDAwMFoXDTMwMDEwMTAzNTk1OVow0TELMAkGA1UEBhMCVVMxDDAKBgNV
BAoTA0lCTTEMMAoGA1UECxMDQVRTMQ4wDAYDVQQDEwVVU0VSMTCCASIwDQYJKoZI
hvcNAQEBBQADggEPADCCAQoCggEBAIIJmqu8mFEmkuuise/Pe9Wy0G8WHXhIf7lu
BRaAp+D/gJmhMi2ynSgl1W/hWXe17wYf3tjQ5xTFc065uM04LZjAjrPcSxxG5ug1
JAwm+Rs1Bpja0eoEsB7nCRbPlqMD2rB0Sm4hAZfWyLaoT3gkSwa6RMcpFkxlCIZ8
7Li1d2ed8mCt40lEKvaQJIVFeuoTmkPKJV2ZW1/VZiMqCgB67XbpHvjkySyA5JdM
amSHaoWb5Fj2ep0MecA7VA9PAHnYb/aN3WDaPYmMBo7mDH34o11Sh3K1bre7VXfz
1pmmQcLzCJD1aM0Aa02XcMuicThp5AQL7tEMa4BHfuMZ0+rL4P0CAwEAAaOBhDCB
```

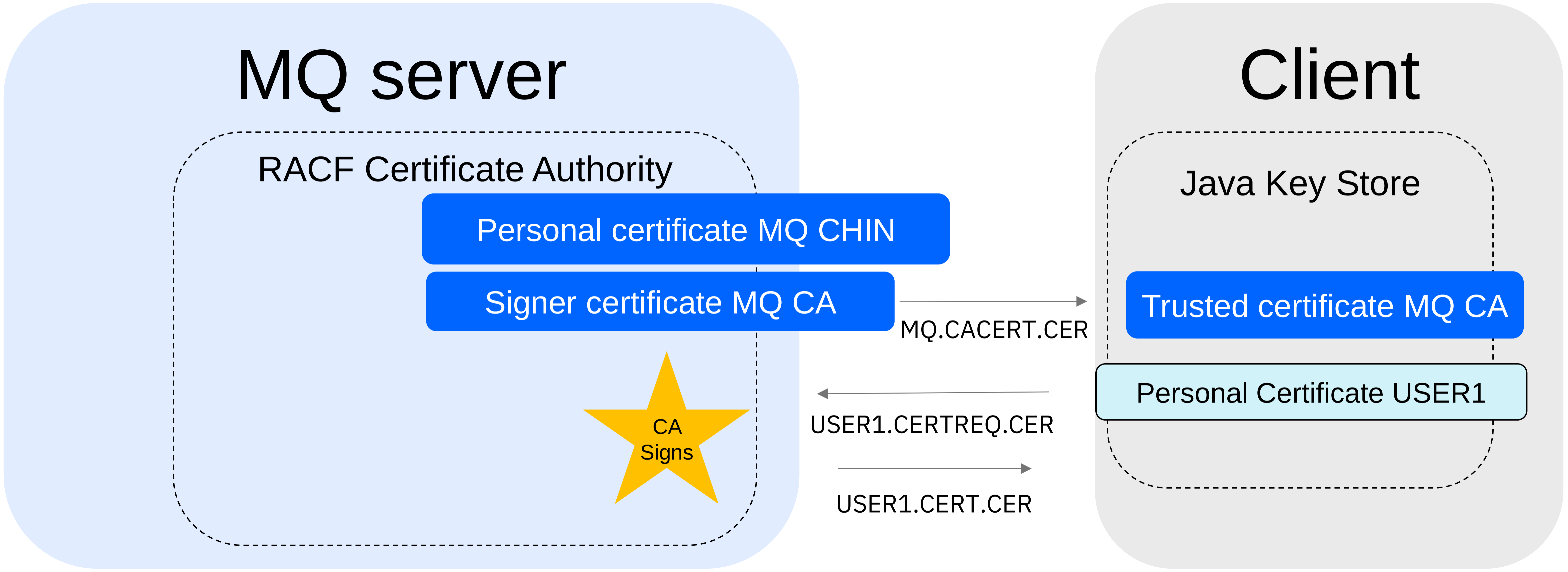
Export signed certificate back to workstation, into USER1.jks

1. keytool –v –import –alias "USER1" –file CERT.CER –keystore USER1.jks
2. keytool –list –keystore USER1.jks



Certificate reply was installed in keystore
[Storing USER1.jks]

At a high-level


MQ server
RACF Certificate Authority
Personal certificate MQ CHIN
Signer certificate MQ CA
CA Signs
MQ.CACERT.CER
USER1.CERTREQ.CER
USER1.CERT.CER
Client
Java Key Store
Trusted certificate MQ CA
Personal Certificate USER1

In labs, we can use self-signed certificates

Use this procedure to create a self-signed personal certificate.

1. Generate a certificate and a public and private key pair using the following command:

```
RACDCERT ID(userid2) GENCERT SUBJECTSDN(CN('common-name') T('title') OU('organizational-unit')
O('organization') L('locality') SP('state-or-province') C('country')) WITHLABEL('label-name’)
```

2. Connect the certificate to your key ring using the following command:

```
RACDCERT ID(userid1) CONNECT(ID(userid2) LABEL('label-name') RING(ring-name) USAGE(PERSONAL))
```

where:

- *userid1* is the user ID of the channel initiator address space or owner of the shared key ring.
- *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space.
  userid1 and *userid2* can be the same ID.
- *ring-name* is the name you gave the key ring in `RACDCERT ID( userid1 ) ADDRING( ring-name )`
- *label-name* must be either the value of the IBM® MQ **CERTLABL** attribute, if it is set, or the default ibmWebSphereMQ with the name of the queue manager appended.

In production, its required to use personal certificates

To apply for a personal certificate, use RACF as follows:

1.Create a self-signed personal certificate. This certificate provides the request with the attribute values for the Distinguished Name.

2.Create a PKCS #10 Base64-encoded certificate request written to a data set, using the following command:

```
RACDCERT ID(userid2) GENREQ(LABEL(' label_name ')) DSN(' output_data_set_name ')
```

where

- *userid2* is the user ID associated with the certificate and must be the user ID of the channel initiator address space
- *label_name* is the label used when creating the self-signed certificate

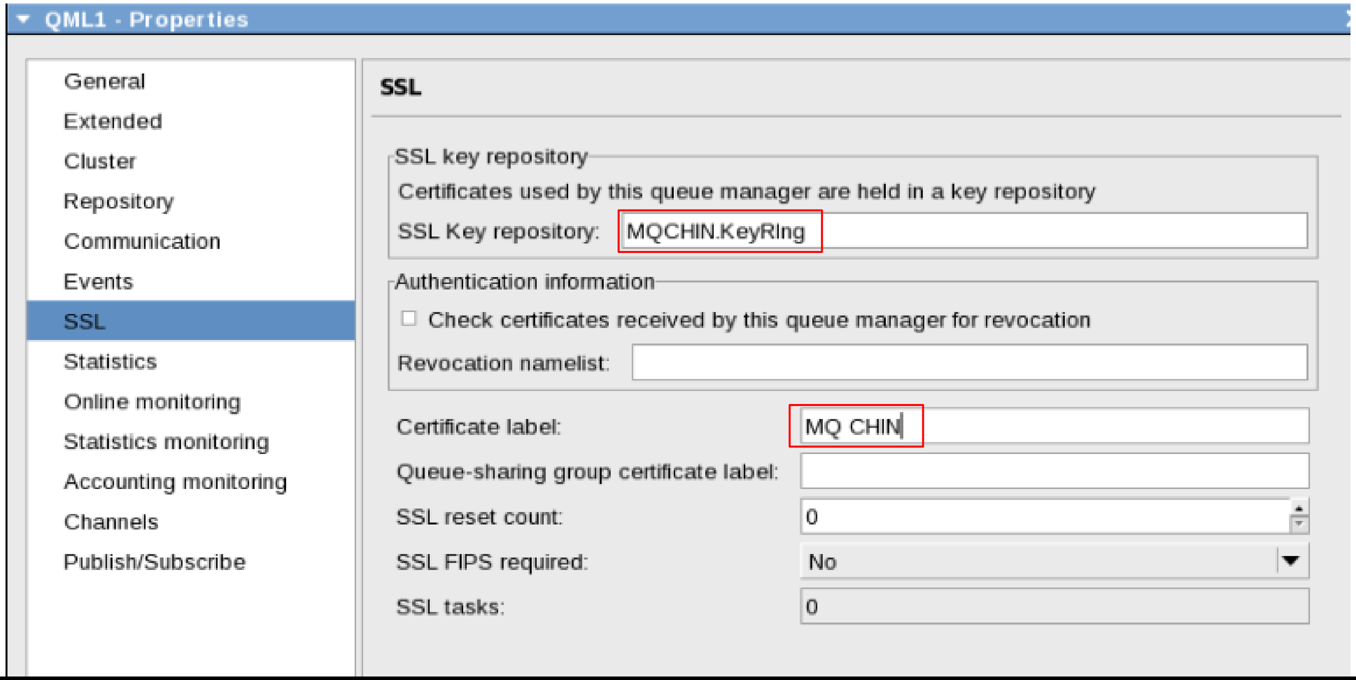
3. Send the data set to a Certificate Authority (CA) to request a new personal certificate.

4.When the signed certificate is returned to you by the Certificate Authority, add the certificate back into the RACF database, using the original label, as described in Adding personal certificates to a key repository on z/OS.

Channel encryption

At this point, we have our certificate set up, but now we need to configure MQ to use the certificate for channels

1. Modify RACF key ring (SSLKEYR) to be accessed for personal and certificate authority digital certificates
2. Modify the number of SSL sub tasks (SSLTASKS) for processing SSL calls
3. Restart CHINIT address space



SSL

SSL key repository

Certificates used by this queue manager are held in a key repository

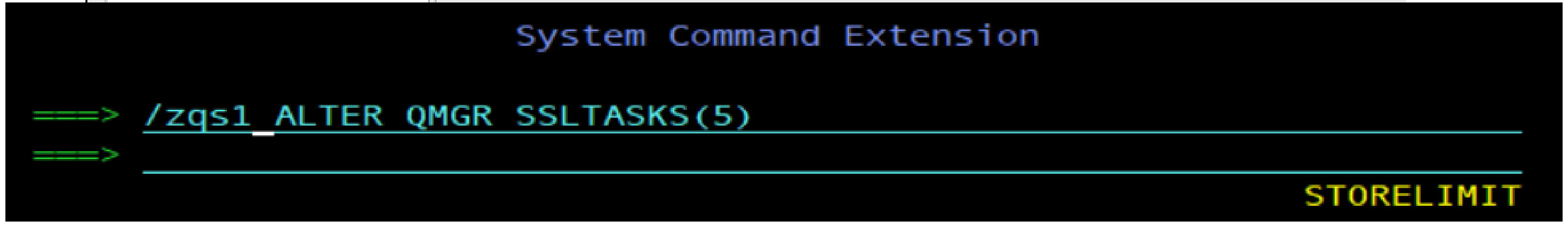
SSL Key repository: MQCHIN.KeyRing

Authentication information

☐ Check certificates received by this queue manager for revocation

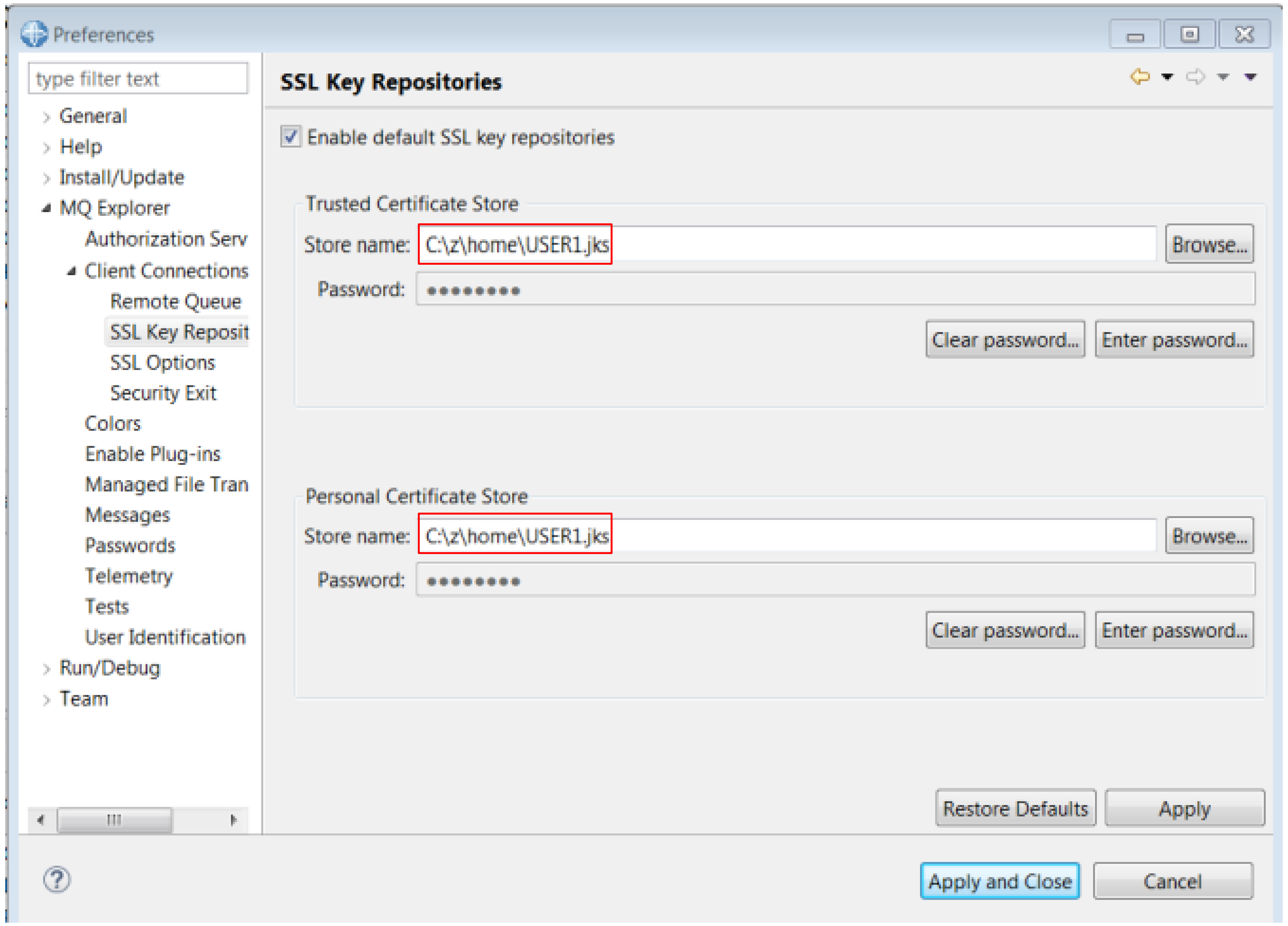
Revocation namelist:

Certificate label:	MQ CHIN
Queue-sharing group certificate label:	
SSL reset count:	0
SSL FIPS required:	No
SSL tasks:	0



```
                    System Command Extension

===> /zqs1 ALTER QMGR SSLTASKS(5)
===>
                                                        STORELIMIT
```

Specify client keystore location on MQ Explorer


Preferences

type filter text

- General
- Help
- Install/Update
- MQ Explorer
  - Authorization Serv
  - Client Connections
    - Remote Queue
    - SSL Key Reposit
    - SSL Options
    - Security Exit
  - Colors
  - Enable Plug-ins
  - Managed File Tran
  - Messages
  - Passwords
  - Telemetry
  - Tests
  - User Identification
- Run/Debug
- Team

SSL Key Repositories

☑ Enable default SSL key repositories

Trusted Certificate Store

Store name: C:\z\home\USER1.jks

Password: ●●●●●●●●●

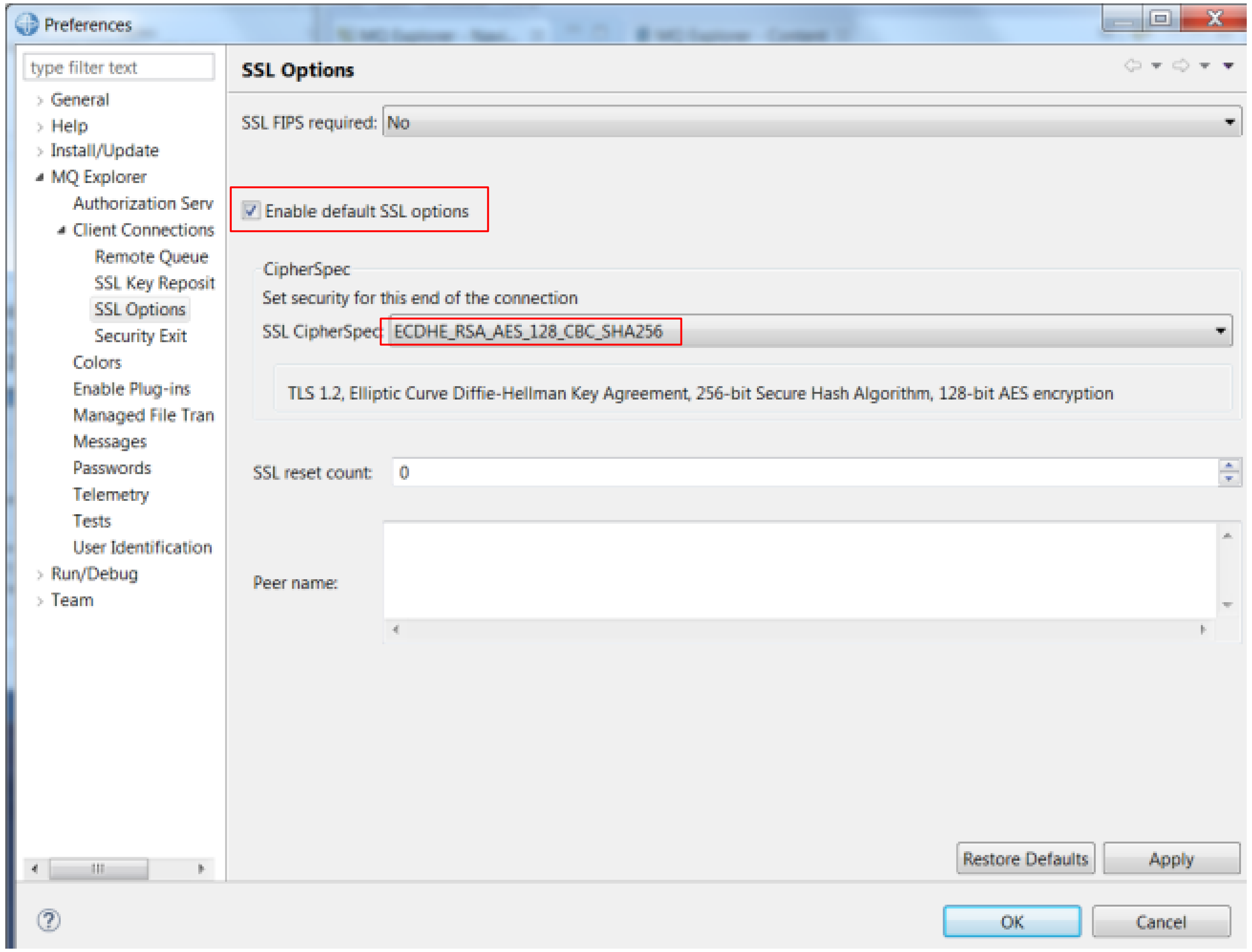
Personal Certificate Store

Store name: C:\z\home\USER1.jks

Password: ●●●●●●●●●

Specify SSL CipherSpec

CipherSpec must be consistent on both ends of a TLS connection



SSL Options

SSL FIPS required: No

☑ Enable default SSL options

CipherSpec

Set security for this end of the connection

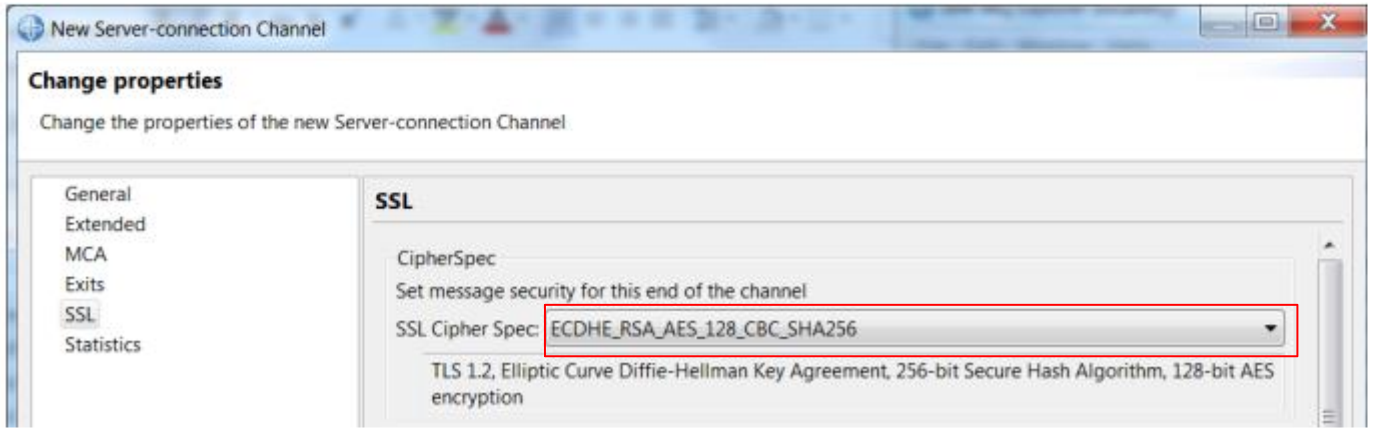
SSL CipherSpec: ECDHE_RSA_AES_128_CBC_SHA256

TLS 1.2, Elliptic Curve Diffie-Hellman Key Agreement, 256-bit Secure Hash Algorithm, 128-bit AES encryption

SSL reset count: 0

Peer name:

Create channel USER1.SSL.SVRCONN under queue manager ZQS1


New Server-connection Channel

Change properties

Change the properties of the new Server-connection Channel

General
Extended
MCA
Exits
SSL
Statistics

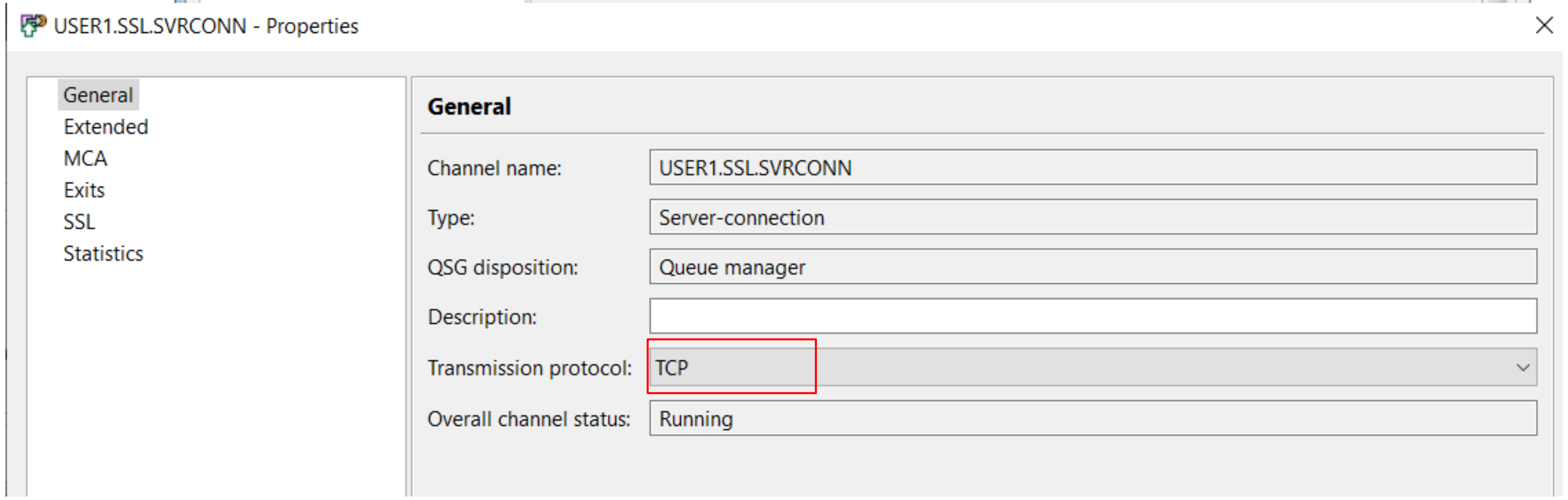
SSL

CipherSpec

Set message security for this end of the channel

SSL Cipher Spec: ECDHE_RSA_AES_128_CBC_SHA256

TLS 1.2, Elliptic Curve Diffie-Hellman Key Agreement, 256-bit Secure Hash Algorithm, 128-bit AES encryption


USER1.SSL.SVRCONN - Properties

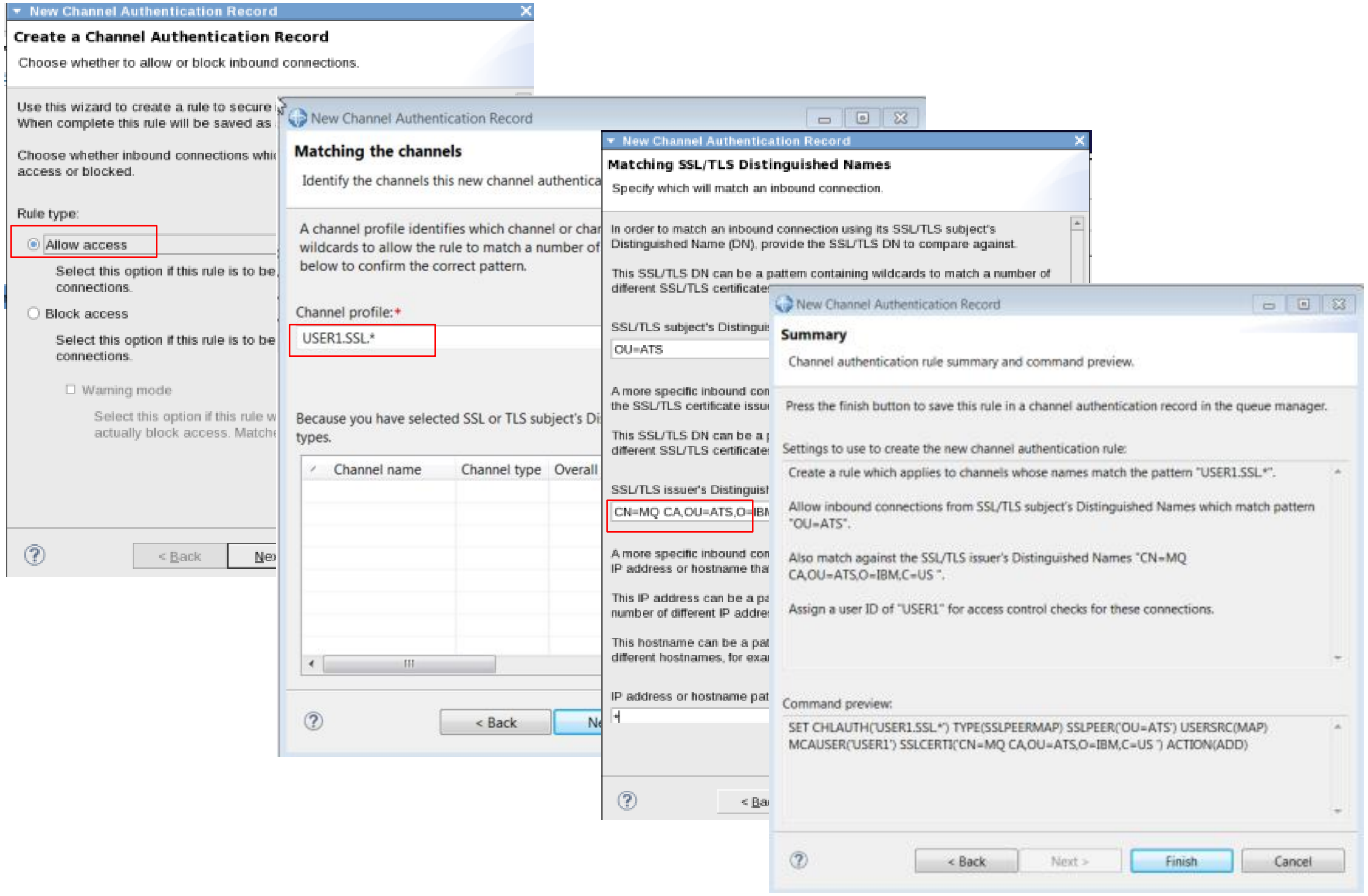
General
Extended
MCA
Exits
SSL
Statistics

General

Channel name:	USER1.SSL.SVRCONN
Type:	Server-connection
QSG disposition:	Queue manager
Description:	
Transmission protocol:	TCP
Overall channel status:	Running

Create channel authentication record under queue manager ZQS1

Channel authentication records allow you to specify how inbound connections to the queue manager should be allowed or blocked, based on identities



Create a Channel Authentication Record

Choose whether to allow or block inbound connections.

Use this wizard to create a rule to secure
When complete this rule will be saved as

Choose whether inbound connections whic
access or blocked.

Rule type:

- (●) Allow access

  Select this option if this rule is to be connections.

- ( ) Block access

  Select this option if this rule is to be connections.

  - ☐ Warning mode

    Select this option if this rule w actually block access. Matche

Matching the channels

Identify the channels this new channel authentica

A channel profile identifies which channel or char wildcards to allow the rule to match a number of below to confirm the correct pattern.

Channel profile:* USER1.SSL.*

Because you have selected SSL or TLS subject's Di types.

Channel name	Channel type	Overall

Matching SSL/TLS Distinguished Names

Specify which will match an inbound connection.

In order to match an inbound connection using its SSL/TLS subject's Distinguished Name (DN), provide the SSL/TLS DN to compare against.

This SSL/TLS DN can be a pattern containing wildcards to match a number of different SSL/TLS certificate

SSL/TLS subject's Distingui: OU=ATS

A more specific inbound con the SSL/TLS certificate issu

This SSL/TLS DN can be a p different SSL/TLS certificate

SSL/TLS issuer's Distinguist: CN=MQ CA,OU=ATS,O=IBM

A more specific inbound con IP address or hostname tha

This IP address can be a pa number of different IP addre

This hostname can be a pat different hostnames, for exa

IP address or hostname pat: *

Summary

Channel authentication rule summary and command preview.

Press the finish button to save this rule in a channel authentication record in the queue manager.

Settings to use to create the new channel authentication rule:

Create a rule which applies to channels whose names match the pattern "USER1.SSL.*".

Allow inbound connections from SSL/TLS subject's Distinguished Names which match pattern "OU=ATS".

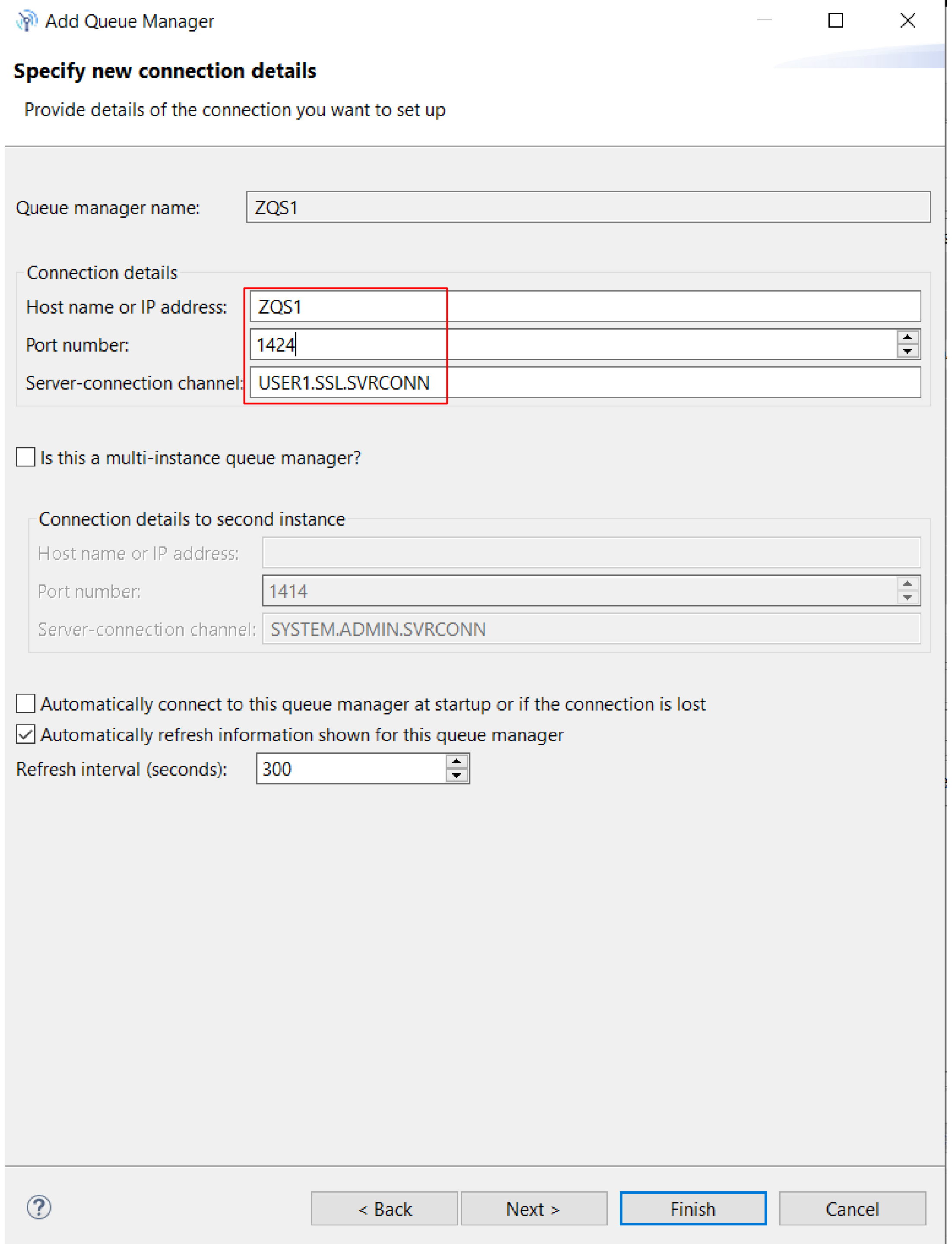
Also match against the SSL/TLS issuer's Distinguished Names "CN=MQ CA,OU=ATS,O=IBM,C=US ".

Assign a user ID of "USER1" for access control checks for these connections.

Command preview:

```
SET CHLAUTH('USER1.SSL.*') TYPE(SSLPEERMAP) SSLPEER('OU=ATS') USERSRC(MAP)
MCAUSER('USER1') SSLCERTI('CN=MQ CA,OU=ATS,O=IBM,C=US ') ACTION(ADD)
```

Create a new remote connection to queue manager ZQS1


Add Queue Manager

Specify new connection details

Provide details of the connection you want to set up

Queue manager name: ZQS1

Connection details

Host name or IP address: ZQS1

Port number: 1424

Server-connection channel: USER1.SSL.SVRCONN

☐ Is this a multi-instance queue manager?

Connection details to second instance

Host name or IP address:

Port number: 1414

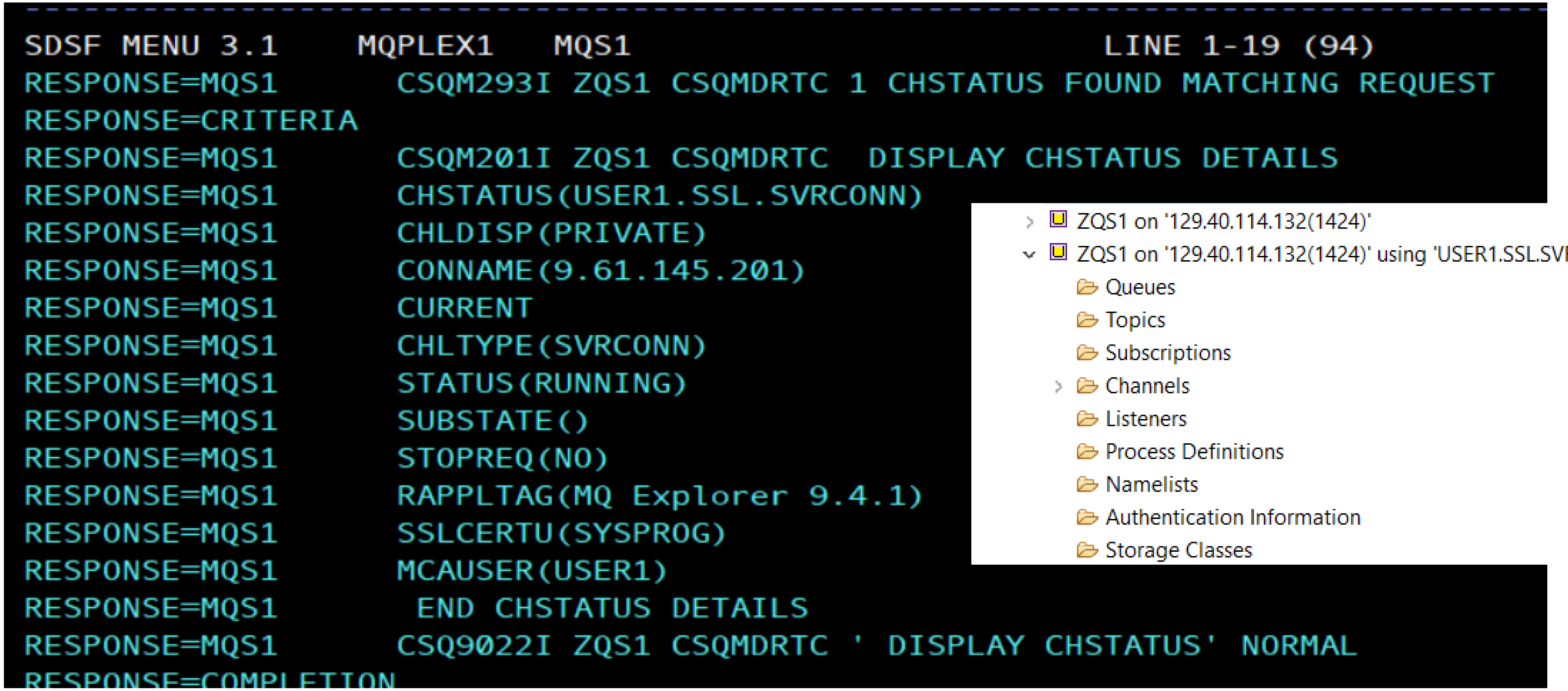
Server-connection channel: SYSTEM.ADMIN.SVRCONN

☐ Automatically connect to this queue manager at startup or if the connection is lost

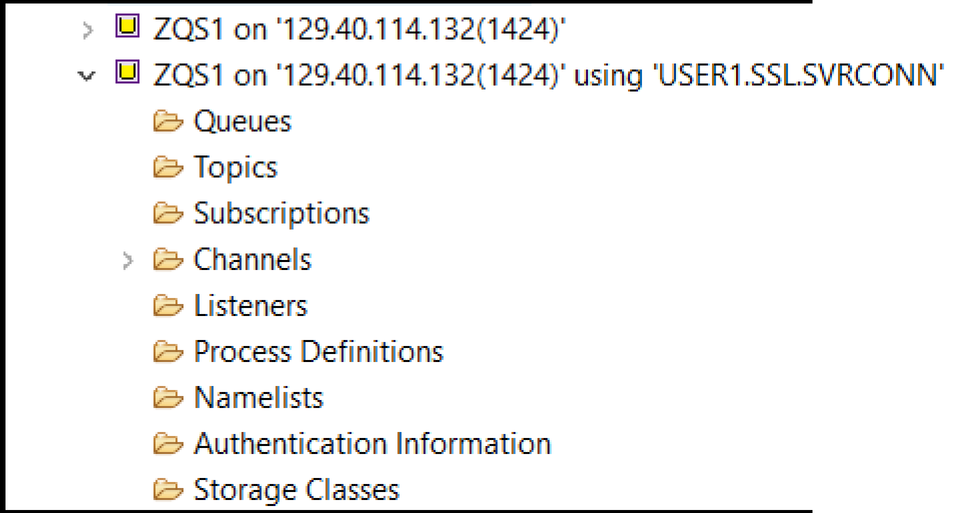
☑ Automatically refresh information shown for this queue manager

Refresh interval (seconds): 300

Woo-hoo!!!

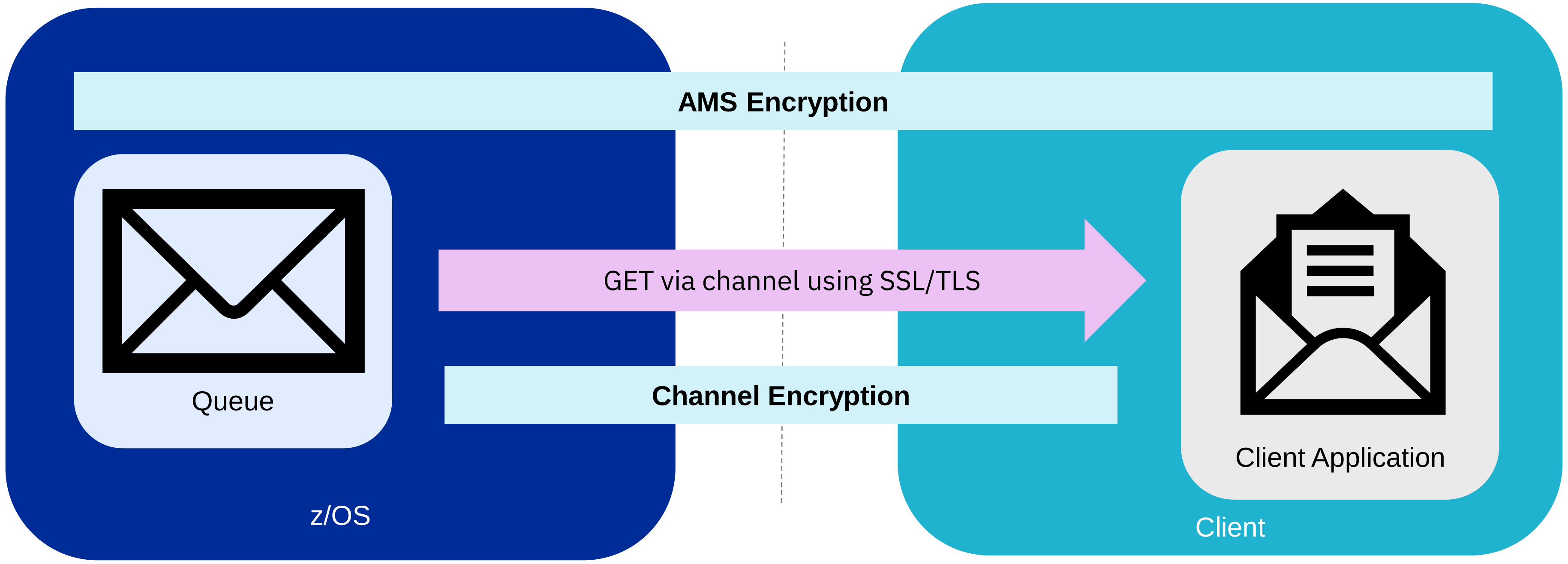


```
SDSF MENU 3.1     MQPLEX1    MQS1                          LINE 1-19 (94)
RESPONSE=MQS1        CSQM293I ZQS1 CSQMDRTC 1 CHSTATUS FOUND MATCHING REQUEST
RESPONSE=CRITERIA
RESPONSE=MQS1        CSQM201I ZQS1 CSQMDRTC  DISPLAY CHSTATUS DETAILS
RESPONSE=MQS1        CHSTATUS(USER1.SSL.SVRCONN)
RESPONSE=MQS1        CHLDISP(PRIVATE)
RESPONSE=MQS1        CONNAME(9.61.145.201)
RESPONSE=MQS1        CURRENT
RESPONSE=MQS1        CHLTYPE(SVRCONN)
RESPONSE=MQS1        STATUS(RUNNING)
RESPONSE=MQS1        SUBSTATE()
RESPONSE=MQS1        STOPREQ(NO)
RESPONSE=MQS1        RAPPLTAG(MQ Explorer 9.4.1)
RESPONSE=MQS1        SSLCERTU(SYSPROG)
RESPONSE=MQS1        MCAUSER(USER1)
RESPONSE=MQS1         END CHSTATUS DETAILS
RESPONSE=MQS1        CSQ9022I ZQS1 CSQMDRTC ' DISPLAY CHSTATUS' NORMAL
RESPONSE=COMPLETION
```



```
> ZQS1 on '129.40.114.132(1424)'
v ZQS1 on '129.40.114.132(1424)' using 'USER1.SSL.SVRCONN'
    Queues
    Topics
    Subscriptions
  > Channels
    Listeners
    Process Definitions
    Namelists
    Authentication Information
    Storage Classes
```

Comparing Channel Encryption to AMS



What does this mean? You have to take into account what you're using for security when the data is at-rest

Advanced message security

Integrity protection is provided by digital signing, which provides assurance on **who created the message**, and that the **message has not been altered**.

Privacy protection is provided by a combination of digital signing and encryption. Encryption ensures that **message data is viewable by only the intended recipient**, or recipients.

Confidentiality protection is provided by **encryption** only

AMS Security Policies

Policies enable us to control on a per-queue level, message integrity, privacy, encryption and get access

Example policies:

```
setmqspl -m ZQS1
-p AMSDEMO.INTEGRITY.QUEUE
-s MD5
-e NONE
-a CN=USER1,O=IBM,C=US
```

```
setmqspl -m ZQS1
-p AMSDEMO.PRIVACY.QUEUE
-s MD5
-e AES256
-a CN=USER2,O=IBM,C=US
-r CN=USER2,O=IBM,C=US
```

```
dspmqspl -m ZQS1 -p AMSDEMO.INTEGRITY.QUEUE
dspmqspl -m ZQS1 -p AMSDEMO.PRIVACY.QUEUE
```

Data set encryption

Encrypts at-rest data in bulk, performing efficiently at speed and for low-cost.

Data set type	**Considerations**
Active and archive logs	4 encrypts and 1 decrypt per message for dual logging
Page set I/O	Depends on types of I/O done to the page set, namely GETs, Immediate WRITEs, and WRITEs
SMDS	Encryption costs mainly charged to application. Decryption costs mainly incurred to QM MSTR; Sufficient buffers can help with encryption costs

Performance considerations

When do you pay for encryption?

1. Starting and stopping of a channel
2. Re-negotiation of secret key
3. Cost of encryption and decryption of data

How can you manage costs?

1. Change re-negotiation frequency to change keys less often
2. Encryption level via CipherSpec
3. Channel start/stop versus long running channels
4. Cost versus data security
5. Consider the use of channel compression
6. Running on the latest possible hardware
7. Offloading work onto Crypto Express cards

The Transport Layer Security (TLS) 1.3 protocol is a major rewrite of prior TLS protocol standards.

TLS 1.3

In z/OS 2.4, System SSL added support for the TLS 1.3 protocol in order for z/OS applications to take advantage of the security updates.

1. All handshake messages after the initial client and server handshake messages are now encrypted.
2. Encrypted handshake messages are presented as payload messages and must be decrypted in order to determine whether the message is a handshake, payload or alert message.
3. The RSA key exchange is no longer supported. It was replaced with Elliptic Curve DiffieHellman Ephemeral (ECDHE), which provides forward secrecy.
4. Prior to TLS 1.3, the negotiated key exchange was part of the cipher suite. In TLS 1.3, the negotiated key exchange is no longer part of the cipher suite and is negotiated separately.

What about quantum?

What's at risk?

Asymmetric encryption. IBM MQ uses asymmetric encryption in:

- TLS communication
- Password protection
- Advanced message security

What can you do?

1. Upgrade to TLS 1.3
2. Consider your management of certificates so you don't become overwhelmed
3. Use AES-128 (or higher) within AMS policies and the TLS Cipher Spec.
4. Use SHA-256 (or higher) within your AMS policies and the TLS Cipher Spec.

Agenda

MQ RACF basics

Channel encryption – securing communication

Advanced message security – security end-to-end

Data set encryption – securing data sets

Performance considerations and what’s new

More resources

Getting SSL to work with MQ for z/OS | Colin Paice

IBM Documentation

IBM Docs | Planning for AMS

MP16: Capacity Planning & Tuning Guide

IBM WSC GitHub

IBM MQ Performance Report | AMS performance