IMQ09 - IBM MQ V9 for z/OS Wildfire Workshop

L13 – Implementing Queue Manager Security

Version V6.0

October 2018

WSC Wildfire Team
IBM z Systems

Table of Contents

Exercise Objectives

The objective of this exercise is to gain experience with protecting access to IBM MQ queues, MQ commands and connection to a queue manager using RACF. In this exercise you will enable the RACF protection of these resources and then provide appropriate access to these resources to different sets of users both local and remote.

General Exercise Information and Guidelines

- ✓ This exercise requires using TSO user *USER1* on the *wg31.washington.ibm.com* z/OS system.
- ✓ The TSO password for this exercise will be provided by the lab instructor.
- ✓ This exercise uses data set *USER1.SECURITY.JCL.*
- ✓ MVS commands are identified by a leading slash (/). This is reminder that these commands need to be entered using SDSF.
- ✓ Case (upper or lower) is very important when invoking the *RACDCERT* and *keytool* commands in this exercise. When you see the use of upper characters in these commands enter the commands exactly as they appear in the text. More than likely if there are any unexpected results in the SSL sections of this exercise they will be caused by case inconsistencies.
- ✓ Text in **bold** and highlighted in **yellow** in this document should be available for copying and pasting in a file named *MQ Exercises Cut Paste* file on the desktop.

Part 1 – Configuring Base RACF resources

In this part of the exercise we begin by reviewing the users and the access their roles require. There are multiple functions that should be protected. The users have been connected to RACF groups and where each groups had access to different functions. The first group (*MQUSERS*) will be the general MQ users that need access to basic MQ functions. They will be able to use MQExplorer and ISPF panels and are limited to what actions they can perform. The second set of users will be MQ administrators (*MQSYSP*). These are users who need the same MQ access as general users but will also need the ability to enter protected MQ commands and full authority in MQExplorer and ISPF panels. Another group will be the identities used for the MQ started tasks (*MQSTC*) and CICS regions (*CICSSTC*). Another set of general uses are CICS users who also need access to CICS resources as well as a set of CICS related MQ resources. A single user could be a member of multiple groups.

____1. Log on to TSO using the *PComm-3270* icon on this desk top

____2. Use the RACF command ***LISTGROUP*** (or ***LG***) to list the users currently connected to the groups described above, e.g. ***LG MQUSERS***. You will see something like the results below.

```
INFORMATION FOR GROUP MQUSERS
     SUPERIOR GROUP=SYS1         OWNER=JOHNSON     CREATED=14.173
     NO INSTALLATION DATA
     NO MODEL DATA SET
     TERMUACC
     NO SUBGROUPS
     USER(S)=      ACCESS=      ACCESS COUNT=      UNIVERSAL ACCESS=
       CICSUSER      USE           000000              NONE
          CONNECT ATTRIBUTES=NONE
          REVOKE DATE=NONE                RESUME DATE=NONE
       USER1        USE           000000              NONE
          CONNECT ATTRIBUTES=NONE
          REVOKE DATE=NONE                RESUME DATE=NONE
```

____3. External security checking is enabled or disabled for a specific queue manager either by the presence or absence of specific *MQADMIN* RACF resources during the queue manager initialization. During startup, the queue manager uses its name (e.g. QML) to look for a specific *MQADMIN* resource. If the resource is defined, then it presence disables the corresponding security checking. For example for queue manager QMZ1, if *MQADMIN* resource ***QMZ1.NO.TOPIC.CHECKS*** is defined then external security checking for topics will be disabled. If *MQADMIN* resource ***QMZ1.NO.QUEUE.CHECKS*** is defined then external security checking for queue access will be disabled.

To determine what *MQADMIN* resources are currently defined use the RACF ***SEARCH*** command to display the currently defined MQADMIN resources for QMZ1.

SEARCH CLASS(MQADMIN) FILTER(QMZ1.**)

The results should look like the list below:

```
QMZ1.NO.ALTERNATE.USER.CHECKS
QMZ1.NO.CMD.RESC.CHECKS
QMZ1.NO.CONTEXT.CHECKS
QMZ1.NO.NLIST.CHECKS
QMZ1.NO.PROCESS.CHECKS
QMZ1.NO.SUBSYS.SECURITY
QMZ1.NO.TOPIC.CHECKS
QMZ1.RESLEVEL
```

Tech-Tip: All RACF resources associated with a specific queue manager are named with a prefix of the queue manager name. This allows the coexistences of multiples RACF resources for the same base resource name, i.e. QMZ1.SYSTEM.DEFAULT.LOCAL.QUEUE will have different security protection than QMZ2.SYSTEM.DEFAULT.LOCAL.QUEUE.

Tech-Tip: The RACF SEARCH command is a useful for managing RACF resources. The FILTER option is flexible for specifying the search criteria and adding the EXEC option provides a means for generating additional commands. For example adding

CLIST('RLIST MQADMIN ', 'AUTHUSER')

will generate a CLIST named EXEC.RACF.CLIST) with a series of RLIST commands for displaying the authorized users for every resource found. Adding

CLIST('RDELETE MQADMIN ')

will generate a CLIST named EXEC.RACF.CLIST with a series of RDELETE commands for deleting the resource found.

____4. The *MQADMIN* resources in this list disable various MQ external security checks. One of these one in particular disables all external checking regardless of any other *MQADMIN* resource and that is *QMZ1.NO.SUBSYS.SECURITY*. The presence of this resource during the queue manager's initialization disables all external security checking as shown by the startup message shown below.

```
CSQH021I QMZ1 CSQHINIT SUBSYSTEM security switch set
OFF, profile 'QMZ1.NO.SUBSYS.SECURITY' found
```

____5. To enable external security checking for the next restart of the queue manager, use the RACF *RDELETE* command to delete this MQADMIN resource.

RDELETE MQADMIN QMZ1.NO.SUBSYS.SECURITY

Tech-Tip: To disable external security checking use the RDEFINE to command to define this resource.

RDEFINE MQADMIN QMZ1.NO.SUBSYS.SECURITY OWNER(SYS1)

____6. Refresh the RACF instorage profiles with the RACF *SETROPTS* command

SETROPTS RACLIST(MQADMIN) REFRESH

Tech-Tip: You probably will receive message *ICH14070I SETROPTS RACLIST REFRESH had no effect on class MQADMIN.* No every RACF configuration requires a refresh of the RACF instorage profiles but it is a good practice to get in the habit of doing a refresh after making changes. This will avoid making a configuration change and not having it made active.

____7. Shutdown the QMZ1 queue manager with MVS command

/QMZ1 STOP QMGR

____8. Before the queue manager is restarted some basic MQ RACF resources should be defined. Select member *MQCMDS* in data set *USER1.SECURITY.JCL.* This JCL defines MQCMDS RACF resources for some of the basic MQ commands and then grants users in group STCMQ and MQADMS access.

```
RDEFINE MQCMDS  QMZ1.DEFINE.** OWNER(SYS1)
PERMIT QMZ1.DEFINE.** CLASS(MQCMDS) RESET
PERMIT QMZ1.DEFINE.** CLASS(MQCMDS) ID(MQSTC,MQSYSP) ACC(ALTER)

RDEFINE MQCMDS  QMZ1.DELETE.** OWNER(SYS1)
PERMIT QMZ1.DELETE.** CLASS(MQCMDS) RESET
PERMIT QMZ1.DELETE.** CLASS(MQCMDS) ID(MQSTC,MQSYSP) ACC(ALTER)

RDEFINE MQCMDS  QMZ1.DISPLAY.** OWNER(SYS1)
PERMIT QMZ1.DISPLAY.** CLASS(MQCMDS) RESET
PERMIT QMZ1.DISPLAY.** CLASS(MQCMDS) ID(MQSTC,MQUSERS) ACC(READ)

RDEFINE MQCMDS  QMZ1.REFRESH.** OWNER(SYS1)
PERMIT QMZ1.REFRESH.** CLASS(MQCMDS) RESET
PERMIT QMZ1.REFRESH.** CLASS(MQCMDS) ID(MQSTC,MQSYSP) ACC(ALTER)

RDEFINE MQCMDS  QMZ1.START.** OWNER(SYS1)
PERMIT QMZ1.START.** CLASS(MQCMDS) RESET
PERMIT QMZ1.START.** CLASS(MQCMDS) ID(MQSTC,MQSYSP) ACC(CONTROL)

RDEFINE MQCMDS  QMZ1.STOP.** OWNER(SYS1)
PERMIT QMZ1.STOP.** CLASS(MQCMDS) RESET
PERMIT QMZ1.STOP.** CLASS(MQCMDS) ID(MQSTC,MQSYSP) ACC(CONTROL)

RDEFINE MQCMDS  QMZ1.SET.** OWNER(SYS1)
PERMIT QMZ1.SET.** CLASS(MQCMDS) RESET
PERMIT QMZ1.SET.** CLASS(MQCMDS) ID(MQSTC,MQSYSP) ACC(CONTROL)

RDEFINE MQCMDS  QMZ1.CLEAR.** OWNER(SYS1)
PERMIT QMZ1.CLEAR.** CLASS(MQCMDS) RESET
PERMIT QMZ1.CLEAR.** CLASS(MQCMDS) ID(MQSTC,MQSYSP) ACC(ALTER)

RDEFINE MQCMDS  QMZ1.** OWNER(SYS1)
PERMIT QMZ1.** CLASS(MQCMDS) RESET
PERMIT QMZ1.** CLASS(MQCMDS) ID(MQSTC,MQSYSP) ACC(READ)

SETROPTS RACLIST(MQCMDS) REFRESH
```

Tech-Tip: The *SEARCH* and *EXEC* commands at the beginning of this job delete all existing MQCMDS profiles for queue manager *QMZ1*.

____9. Submit *MQCMDS* for execution and verify that it completes with a condition code of zero.

____10. Select member *MQCONN* in data set *USER1.SECURITY.JCL.* This JCL defines MQCONN RACF resources required for accessing the queue manager from various sources, e.g., CICS, batch, remote, etc.

```
RDEFINE MQCONN  QMZ1.BATCH OWNER(SYS1)
PERMIT QMZ1.BATCH CLASS(MQCONN) RESET
PERMIT QMZ1.BATCH CLASS(MQCONN) ID(MQSTC,MQUSERS) ACC(READ)

RDEFINE MQCONN  QMZ1.CHIN OWNER(SYS1)
PERMIT QMZ1.CHIN CLASS(MQCONN) RESET
PERMIT QMZ1.CHIN CLASS(MQCONN) ID(MQSTC) ACC(READ)

RDEFINE MQCONN  QMZ1.CICS OWNER(SYS1)
PERMIT QMZ1.CICS CLASS(MQCONN) RESET
PERMIT QMZ1.CICS CLASS(MQCONN) ID(CICSSTC) ACC(READ)

SETROPTS RACLIST(MQCONN) REFRESH
```

____11. Submit *MQCONN* for execution and verify that it completes with a condition code of zero.

____12. Select member *MQQUEUE* in data set *USER1.SECURITY.JCL.* This JCL defines MQQUEUE RACF resources required for accessing the system related queues.

```
RDEFINE MQQUEUE QMZ1.** OWNER(SYS1)
PERMIT QMZ1.** CLASS(MQQUEUE) RESET
PERMIT QMZ1.** CLASS(MQQUEUE) ID(MQSTC) ACC(READ)

RDEFINE MQQUEUE QMZ1.SYSTEM.** OWNER(SYS1)
PERMIT QMZ1.SYSTEM.** CLASS(MQQUEUE) RESET
PERMIT QMZ1.SYSTEM.** CLASS(MQQUEUE) ID(MQSTC) ACC(UPDATE)
PERMIT QMZ1.SYSTEM.** CLASS(MQQUEUE) ID(MQUSERS) ACC(READ)

RDEFINE MQQUEUE QMZ1.SYSTEM.CLUSTER.COMMAND.QUEUE OWNER(SYS1)
PERMIT QMZ1.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) RESET
PERMIT QMZ1.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) +
          ID(MQSTC) ACC(ALTER)
PERMIT QMZ1.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) +
          ID(MQUSERS) ACC(UPDATE)

RDEFINE MQQUEUE QMZ1.SYSTEM.BROKER.** OWNER(SYS1)
PERMIT QMZ1.SYSTEM.BROKER.** CLASS(MQQUEUE) RESET
PERMIT QMZ1.SYSTEM.BROKER.** CLASS(MQQUEUE) +
        ID(MQSTC) ACC(ALTER)

RDEFINE MQQUEUE QMZ1.AMQ.MQEXPLORER.** OWNER(SYS1)
PERMIT QMZ1.AMQ.MQEXPLORER.** CLASS(MQQUEUE) RESET
PERMIT QMZ1.AMQ.MQEXPLORER.** CLASS(MQQUEUE) +
        ID(MQSTC,MQUSERS) ACC(UPDATE)

RDEFINE MQQUEUE QMZ1.SYSTEM.COMMAND.INPUT OWNER(SYS1)
PERMIT QMZ1.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) RESET
PERMIT QMZ1.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) +
        ID(MQSTC,MQUSERS) ACC(UPDATE)

RDEFINE MQQUEUE QMZ1.SYSTEM.CSQUTIL.** OWNER(SYS1)
PERMIT QMZ1.SYSTEM.CSQUTIL.** CLASS(MQQUEUE) RESET
PERMIT QMZ1.SYSTEM.CSQUTIL.** +
           CLASS(MQQUEUE) ID(MQSTC,MQUSERS) ACC(UPDATE)
```

```
RDEFINE MQQUEUE QMZ1.SYSTEM.MQEXPLORER.REPLY.MODEL OWNER(SYS1)
PERMIT QMZ1.SYSTEM.MQEXPLORER.REPLY.MODEL CLASS(MQQUEUE) RESET
PERMIT QMZ1.SYSTEM.MQEXPLORER.REPLY.MODEL +
           CLASS(MQQUEUE) ID(MQSTC,MQUSERS) ACC(UPDATE)

RDEFINE MQQUEUE QMZ1.SYSTEM.PROTECTION.POLICY.QUEUE OWNER(SYS1)
PERMIT QMZ1.SYSTEM.PROTECTION.POLICY.QUEUE CLASS(MQQUEUE) RESET
PERMIT QMZ1.SYSTEM.PROTECTION.POLICY.QUEUE CLASS(MQQUEUE) +
       ID(MQUSERS,MQSTC) ACC(UPDATE)

RDEFINE MQQUEUE QMZ1.DEAD.LETTER.QUEUE OWNER(SYS1)
PERMIT QMZ1.DEAD.LETTER.QUEUE CLASS(MQQUEUE) RESET
PERMIT QMZ1.DEAD.LETTER.QUEUE CLASS(MQQUEUE) +
           ID(MQSTC,CICSUSER) ACC(UPDATE)

RDEFINE MQQUEUE QMZ1.SYSTEM.COMMAND.REPLY.MODEL OWNER(SYS1)
PERMIT QMZ1.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQQUEUE) RESET
PERMIT QMZ1.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQQUEUE) +
         ID(MQSTC,MQUSERS) ACC(UPDATE)

RDEFINE MQQUEUE QMZ1.SYSTEM.CSQOREXX.** OWNER(SYS1)
PERMIT QMZ1.SYSTEM.CSQOREXX.** CLASS(MQQUEUE) RESET

PERMIT QMZ1.SYSTEM.CSQOREXX.** CLASS(MQQUEUE) +
         ID(MQSTC) ACC(ALTER)
PERMIT QMZ1.SYSTEM.CSQOREXX.** CLASS(MQQUEUE) +
         ID(MQUSERS) ACC(UPDATE)

RDEFINE MQQUEUE QMZ1.SYSTEM.PROTECTION.ERROR.QUEUE OWNER(SYS1)
PERMIT QMZ1.SYSTEM.PROTECTION.ERROR.QUEUE CLASS(MQQUEUE) RESET
PERMIT QMZ1.SYSTEM.PROTECTION.ERROR.QUEUE CLASS(MQQUEUE) +
         ID(MQUSERS) ACC(UPDATE)

RDEFINE MQQUEUE  QMZ1.SYSTEM.DEFAULT.LOCAL.QUEUE OWNER(SYS1)
PERMIT  QMZ1.SYSTEM.DEFAULT.LOCAL.QUEUE   CLASS(MQQUEUE) RESET
PERMIT  QMZ1.SYSTEM.DEFAULT.LOCAL.QUEUE   CLASS(MQQUEUE) +
         ID(MQUSERS,MQSTC,MQSYSP) ACC(UPDATE)

RDEFINE MQQUEUE  QMZ1.DEAD.LETTER.QUEUE OWNER(SYS1)
PERMIT  QMZ1.DEAD.LETTER.QUEUE   CLASS(MQQUEUE) RESET
PERMIT  QMZ1.DEAD.LETTER.QUEUE   CLASS(MQQUEUE) +
        ID(MQUSERS,MQSTC,MQSYSP) ACC(UPDATE)

RDEFINE MQQUEUE QMZ1.AMSDEMO.** OWNER(SYS1)
PERMIT QMZ1.AMSDEMO.** CLASS(MQQUEUE) RESET
PERMIT QMZ1.AMSDEMO.** CLASS(MQQUEUE) ID(MQUSERS) ACC(UPDATE)
PERMIT QMZ1.AMSDEMO.** CLASS(MQQUEUE) ID(MQSTC) ACC(UPDATE)

RDEFINE MQQUEUE QMZ1.USER1.** OWNER(SYS1)
PERMIT QMZ1.USER1.** CLASS(MQQUEUE) RESET
PERMIT QMZ1.USER1.** CLASS(MQQUEUE) ID(USER1) ACC(UPDATE)
PERMIT QMZ1.USER1.** CLASS(MQQUEUE) ID(MQSTC) ACC(UPDATE)

SETROPTS RACLIST(MQQUEUE) REFRESH
```

____13. Submit *MQQUEUE* for execution and verify that it completes with a condition code of zero.

This completes the configuration of the RACF resources required to start a basic queue manger.

Part 2 – Testing Local Security to the Queue Manager

Queue manager security has been enabled by the deletion of *MQADMIN* resource *QMZ1.NO.SUBSYS.SECURITY* and some of the initial RACF resources for MQCONN, MQQUEUE and MQCMDS have been defined and access granted. Now restart the queue manager and see if the resources that have been defined are sufficient or if adjustments should be made.

____1. Start the QML queue manager with MVS command ***/QMZ1 START QMGR***

____2. Monitor the startup messages of MQ task *QMZ1MSTR* and note the additional messages that now appear because *QMZ1.NO.SUBSYS.SECURITY* was not found.

```
CSQH024I QMZ1 CSQHINIT SUBSYSTEM security switch set
ON, profile 'QMZ1.NO.SUBSYS.SECURITY' not found
CSQH024I QMZ1 CSQHINIT CONNECTION security switch set
ON, profile 'QMZ1.NO.CONNECT.CHECKS' not found
CSQH024I QMZ1 CSQHINIT COMMAND security switch set ON,
profile 'QMZ1.NO.CMD.CHECKS' not found
CSQH021I QMZ1 CSQHINIT CONTEXT security switch set
OFF, profile 'QMZ1.NO.CONTEXT.CHECKS' found
CSQH021I QMZ1 CSQHINIT ALTERNATE USER security switch
set OFF, profile 'QMZ1.NO.ALTERNATE.USER.CHECKS' found
CSQH021I QMZ1 CSQHINIT COMMAND RESOURCES security
switch set OFF, profile 'QMZ1.NO.CMD.RESC.CHECKS' found
CSQH021I QMZ1 CSQHINIT PROCESS security switch set
OFF, profile 'QMZ1.NO.PROCESS.CHECKS' found
CSQH021I QMZ1 CSQHINIT NAMELIST security switch set
OFF, profile 'QMZ1.NO.NLIST.CHECKS' found
CSQH024I QMZ1 CSQHINIT QUEUE security switch set ON,
profile 'QMZ1.NO.QUEUE.CHECKS' not found
CSQH021I QMZ1 CSQHINIT TOPIC security switch set OFF,
profile 'QMZ1.NO.TOPIC.CHECKS' found
```

- *MQADMIN* resources *QMZ1.NO.CONNECT.CHECKS* was not found so connection security is now enabled.
- *MQADMIN* resource *QMZ1.NO.CMD.CHECKS* was not found so command security is now enabled.
- Finally *MQADMIN* resource *QMZ1.NO.QUEUE.CHECKS* was not found so queue security is enabled.
- Security for the other MQ resources is stilled disabled because of the presence of the *MQADMIN* resources displayed in Part 1 Step 3.

____3. Continue monitoring the startup messages of QMZ1 and eventually you will see these messages.

```
ICH408I USER(START3  ) GROUP(SYS1    ) NAME(#################
  QMZ1.ZCONN2.TRIGGER.INITQ CL(MQQUEUE )
  INSUFFICIENT ACCESS AUTHORITY
  FROM QMZ1.** (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

These message occurred because name of queue *QMZ1.CICS01.INITQ* only matched the generic *MQQUEUE* profile *QMZ1.*** which only provides READ access to a queue. In this case user *START3* needs UPDATE access to this queue. *START3* is a member of group *MQSTC* so this group should be granted this access.

____4. Use RACF command *RDEFINE* to define a profile for this queue giving *START3* update access.

RDEFINE MQQUEUE QMZ1.ZCONN2.TRIGGER.INITQ   OWNER(SYS1)

____5. Use the RACF *PERMT* command to give UPDATE access to this queue to group *MQSTC* and *CICSSTC*.

PERMIT QMZ1. QMZ1.ZCONN2.TRIGGER.INITQ   CLASS(MQQUEUE) ID(MQSTC,CICSSTC) ACC(UPDATE)

Tech-Tip: In this workshop we do not use mixed case MQ resources so the case of the queues, etc. name is not important in this workshop. If you are using mixed case for your MQ resources, there are corresponding RACF resources to the one used in this workshop. For example, MXQUEUE is the mixed case equivalent of MQQUEUE and MXADMIN is the mixed case equivalent of

____6. Refresh the RACF instorage profiles will command RACF command *SETROPTS.*

SETROPTS RACLIST(MQQUEUE) REFRESH

____7. Refresh the queue manager's RACF information with MVS command

/QMZ1 REFRESH SECURITY(MQQUEUE)

You should see these messages in the console.

```
QMZ1 REFRESH SECURITY(MQQUEUE)
CSQH001I QMZ1 CSQHCHK4 Security using uppercase classes
CSQ9022I QMZ1 CSQHSREF ' REFRESH SECURITY' NORMAL COMPLETION
```

____8. Display the new *MQQUEUE* profile just defined with a RACF *RLIST* command

RLIST MQQUEUE QMZ1.ZCONN2.TRIGGER.INITQ AUTHUSER

Notice that user USER1 has explicit alter authority to this resource. This may not be what you want and that is why the PERMIT RESET command is useful in the jobs run earlier.

```
USER      ACCESS   ACCESS COUNT
----      ------   ------ -----
USER1     ALTER       000000
MQSTC     UPDATE      000000
CICSSTC   UPDATE      000000
```

Next enter command MVS command ***/QMZ1 ALTER QMGR MAXCHL(350)***. This request should fail because *USER1* does not have sufficient access to the *MQCMDS* resource *QMZ1.***.

```
QMZ1 ALTER QMGR MAXCHL(350)
ICH408I USER(USER1   ) GROUP(SYS1    ) NAME(                    )
  QMZ1.ALTER.QMGR CL(MQCMDS  )
  INSUFFICIENT ACCESS AUTHORITY
  FROM QMZ1.** (G)
  ACCESS INTENT(ALTER  )  ACCESS ALLOWED(READ    )
CSQ9016E QMZ1 ' ALTER' command request not authorized
```

____9. *MQCMDS* resource *QMZ1.*** is a fail-safe or default profile for MQ commands. If it did not exist and a queue manager did a security check for a command that did not have an explicit *MQCMDS* resource, then this message would appear.

```
ICH408I USER(USER1   ) GROUP(SYS1    ) NAME(
   QMZ1.ALTER.QMGR CL(MQCMDS  )
   PROFILE NOT FOUND - REQUIRED FOR AUTHORITY CHECKING
   ACCESS INTENT(ALTER  )  ACCESS ALLOWED(NONE    )
 CSQ9016E QMZ1 ' ALTER' command request not authorized
 CSQ9023E QMZ1 CSQ9SCND 'ALTER QMGR' ABNORMAL COMPLETION
```

READ access to some MQ commands is sufficient for authority purposes, e.g. *DISPLAY* but commands like *ALTER* should have a higher level of protection. This is why there is a default MQ command profile that only gives READ access.

____10. To give access to the MQ *ALTER* command an explicit *MQCMDS* profile needs to be created with command

RDEFINE MQCMDS QMZ1.ALTER.** OWNER(SYS1)

____11. And groups *MQSTC* and *MQSYSP* given *ALTER* access with command

PERMIT QMZ1.ALTER.** CLASS(MQCMDS) ID(MQSTC,MQSYSP) ACC(ALTER)

____12. Refresh the RACF instorage profiles with RACF command SETROPTS.

SETROPTS RACLIST(MQCMDS) REFRESH

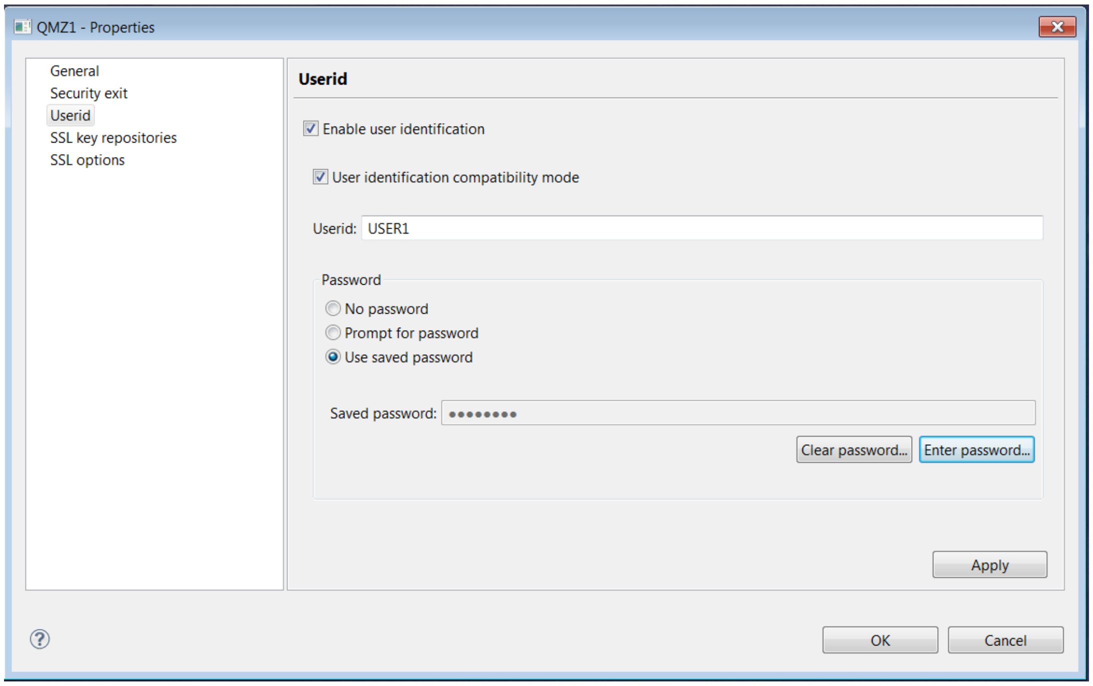
MQCMDS resources cannot be refreshed with a MQ *REFRESH* command. The only way they are update is during a queue manager restart.

____13. Shut the queue manager with MVS command

/QMZ1 STOP QMGR

____14. Restart the queue manager and try to repeat the *ALTER* command. The change to the queue manger should now be successful.

____15. Start MQ Explorer, if it is not already active. Select the QMZ1 queue manger and right mouse button click and select *Connection Details -> Properties*. On the *QMZ1 - Properties* windows select *Userid* on the left-hand side and change the user id from ***USER1*** to ***USER2*** and if necessary use the **Clear password** and **Enter password** buttons to change the password to ***USER2***.



Userid

[✓] Enable user identification

[✓] User identification compatibility mode

Userid: USER1

Password
No password
Prompt for password
[✓ Use saved password]

Saved password: ••••••••

____16. Click **OK** to continue and reconnect to queue manager *QMZ1* as *USER2*.

____17. Try to create a new queue using name ***USER2.TEST.QUEUE***. The request should fail because USER2 does not have *ALTER* authority to *MQCMDS* resource *QMZ1.DEFINE.***.
The messages below should appear in the console.

```
CH408I USER(USER2   ) GROUP(SYS1    ) NAME(####################)
 QMZ1.DEFINE.QUEUE CL(MQCMDS  )
 INSUFFICIENT ACCESS AUTHORITY
 FROM QMZ1.DEFINE.** (G)
 ACCESS INTENT(ALTER  )  ACCESS ALLOWED(NONE   )
```

____18. Try to delete queue USER1.TEST.QUEUE. Again the request should fail because USER2 does not have ALTER authority *MQCMDS* resource to *QMZ1.DELETE.***.

```
ICH408I USER(USER2   ) GROUP(SYS1    ) NAME(####################)
  QMZ1.DELETE.QUEUE CL(MQCMDS  )
  INSUFFICIENT ACCESS AUTHORITY
  FROM QMZ1.DELETE.** (G)
  ACCESS INTENT(ALTER  )  ACCESS ALLOWED(NONE   )
```

____19. Use MQ Explorer to try putting a message to queue *USER1.TEST.QUEUE*. This should fail because of insufficient authority to *MQQUEUE* resource *QMZ1.***.

```
ICH408I USER(USER2  ) GROUP(SYS1    ) NAME(####################)
  QMZ1.USER1.TEST.QUEUE CL(MQQUEUE )
  INSUFFICIENT ACCESS AUTHORITY
  FROM QMZ1.USER1.** (G)
  ACCESS INTENT(UPDATE )  ACCESS ALLOWED(NONE   )
```

____20. Next use MQ Explorer to try putting a message to queue *SYSTEM.DEFAULT.LOCAL.QUEUE*. This should work because there is a *MQQUEUE* RACF resource for queue *QMZ1.SYSTEM.DEFAULT.LOCAL.QUEUE* that allows groups *MQUSERS*, *MQSTC* and *MQSYSP UPDATE* access. Confirm with a *RLIST* RACF command

RLIST MQQUEUE QMZ1.SYSTEM.DEFAULT.LOCAL.QUEUE AUTHUSER

____21. Change the MQ Explorer connection to queue manager QMZ1 back to *USER1* and try creating a queue and then deleting to confirm that USER1 has the authority to perform these functions.

____22. Now select member *PUTMSG* in data set *USER1.SECURITY.JCL*. This JCL places messages on the *SYSTEM.DEFAULT.LOCAL.QUEUE*. This job runs using *USER1*'s authority. Submit the job and confirm that the messages are successfully written to the queue. Again this is successfully because *USER1* is a member of group *MQUSERS* which has access to this queue.

____23. Change all occurrences of *USER1* to *USER3* and resubmit this job for execution. Putting the messages to the queue should fail this time with message.

```
An MQSeries error occurred : Authentication Error. Ensure the correct credentials were used.
A WebSphere MQ error occurred : Completion code 2 Reason code 2035
```

____24. A review of the console will find this message:

```
ICH408I USER(USER3   ) GROUP(SYS1    ) NAME(                    )
  QMZ1.BATCH CL(MQCONN  )
  INSUFFICIENT ACCESS AUTHORITY
  ACCESS INTENT(READ   )  ACCESS ALLOWED(NONE   )
```

____25. *USER3* is not a member of group *MQUSERS* and therefore does not have basic MQ authority. This security issue can be address (assuming USER3 should have access) by connecting USER3 to group MQUSERS with RACF command:

CONNECT USER3 GROUP(MQUSERS)

____26. Resubmit PUTMST under USER3's authority and the messages should now be written to the queue successfully.

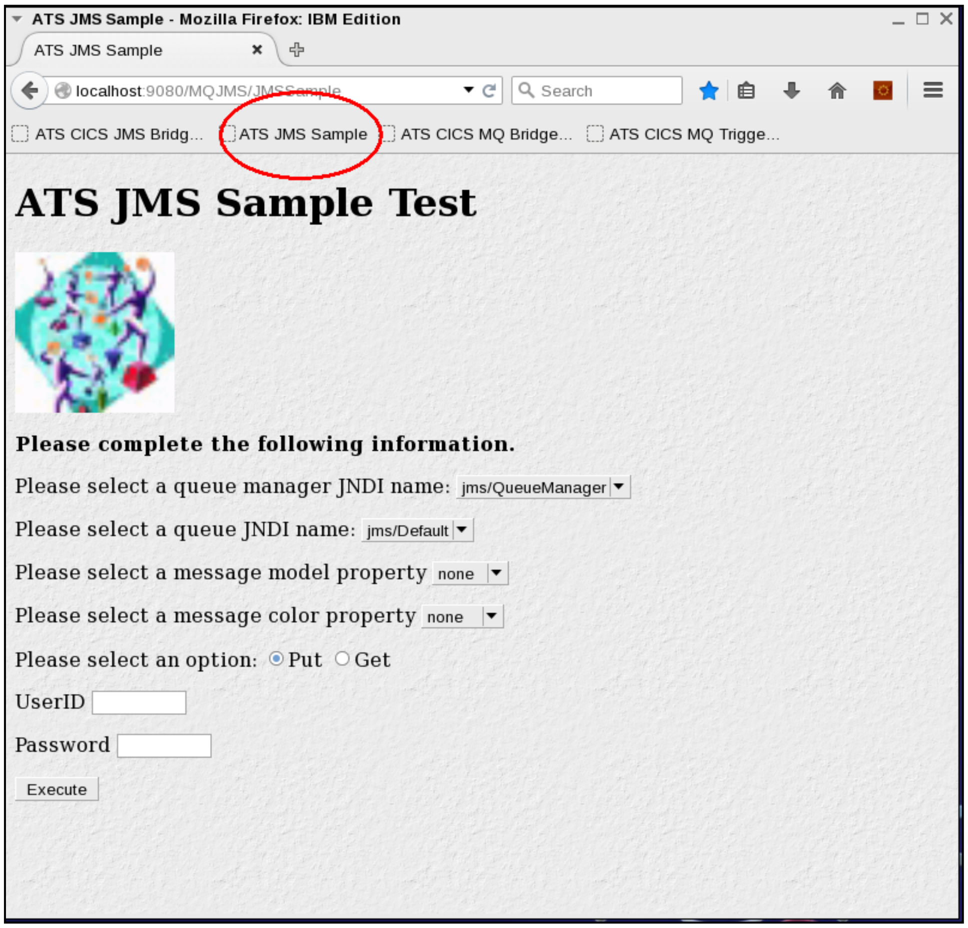
These steps have tested the MQ command authority, queue access and connection resources using the external security manager.

Part 3 – Testing Remote Security to the Queue Manager

In this part of the exercise security when accessing a queue manger from a remote client will be explored. A JMS application running in a Liberty server in the Linux portion of the image will be used to send and retrieve messages from the z/OS queue manager.

____1. Next open the *Mozilla Firefox* on the desktop and selecting the icon and using the right mouse button to select the *Open* option.

____2. On the *Mozilla Firefox* session you should see an option on the *Bookmarks Toolbar* for *ATS JMS Sample* (see below). Click this option and you should see the web page below.



ATS JMS Sample Test

Please complete the following information.

Please select a queue manager JNDI name: jms/QueueManager

Please select a queue JNDI name: jms/Default

Please select a message model property none

Please select a message color property none

Please select an option: [✓ Put] Get

UserID

Password

This application does JNDI lookups for names *jms/QueueManager* and *jms/Default* which are defined in the Liberty *server.xml* file as:

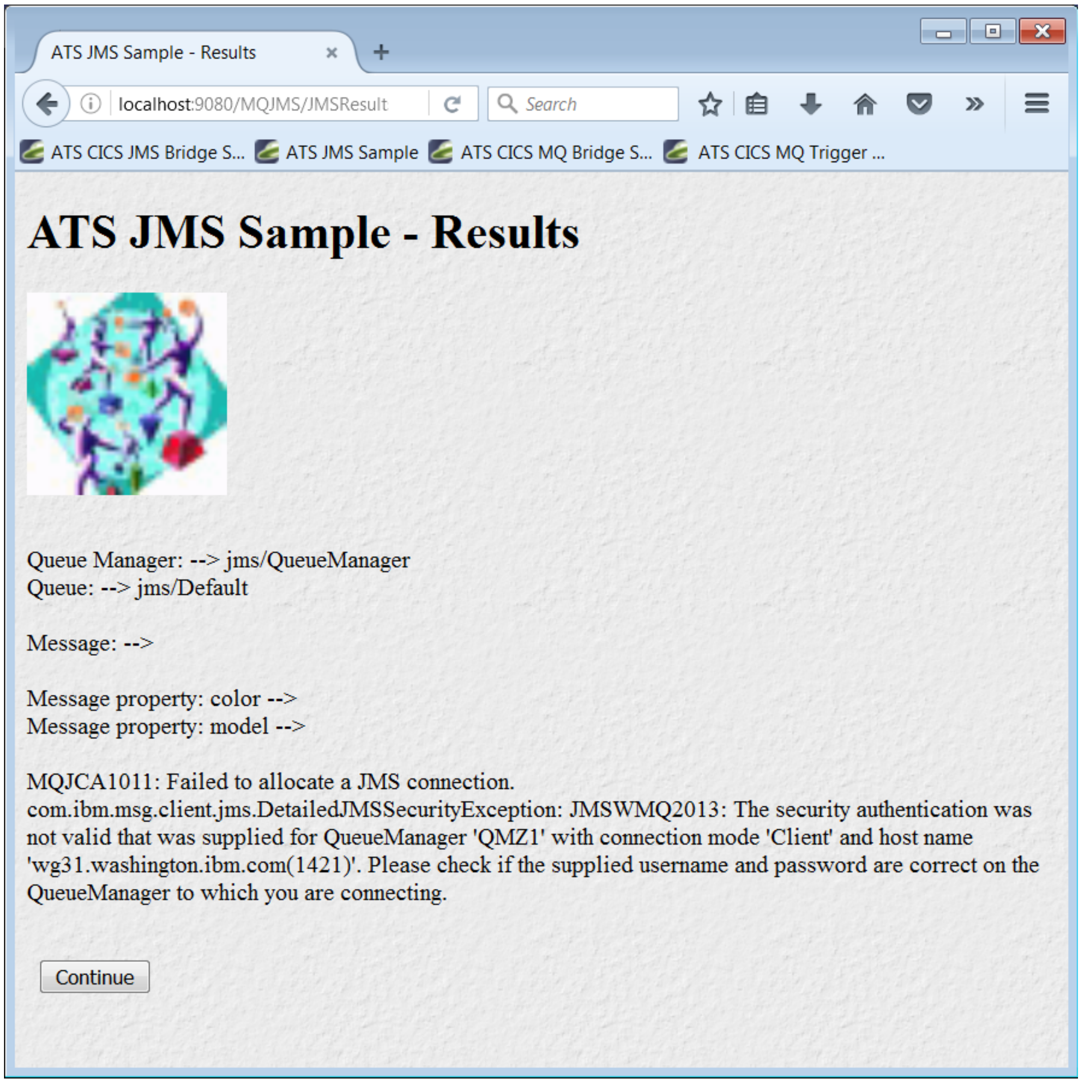
```
<jmsConnectionFactory jndiName="jms/QueueManager">
   <properties.wmqJms channel="Client.to.QueueMgr" hostName="wg.31.washington.ibm.com" port="1421"
            queueManager="QMZ1"/>
 </jmsConnectionFactory>

<jmsQueue id="Default" jndiName="jms/Default">
            <properties.wmqJms baseQueueName="SYSTEM.DEFAULT.LOCAL.QUEUE"/>
</jmsQueue>
```

Channel *Client.to.QueueMgr* is used to connect to the queue manager.

The application does either *put* a message or *get* a message by selecting the appropriate radio button. For either option message properties can be specified by using the pull down arrows to select car model type (e.g. sedan, suv, truck or none) and color (e.g. red, orange, yellow, green, blue, or none). When these properties are selected on a put request the message is written with these properties set. When they are selected on a get request only messages with these properties and values will be retrieved.

____3. Without changing any property click the **Execute** button to see what happens.

____4. The request should fail with the security message shown below:



ATS JMS Sample - Results

Queue Manager: --> jms/QueueManager
Queue: --> jms/Default

Message: -->

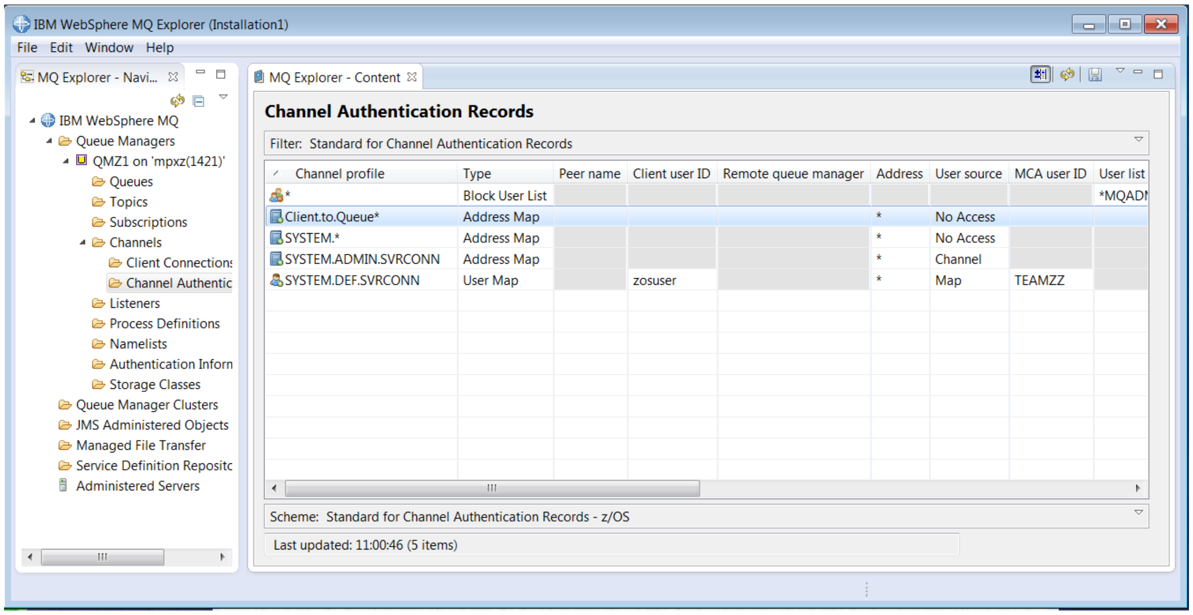
Message property: color -->
Message property: model -->

MQJCA1011: Failed to allocate a JMS connection. com.ibm.msg.client.jms.DetailedJMSSecurityException: JMSWMQ2013: The security authentication was not valid that was supplied for QueueManager 'QMZ1' with connection mode 'Client' and host name 'wg31.washington.ibm.com(1421)'. Please check if the supplied username and password are correct on the QueueManager to which you are connecting.

____5. Review the MVS console and you should fine the message below.

```
+CSQX777E QMZ1 CSQXRESP Channel Client.to.QueueMgr  from 192.168.41.131
 (192.168.41.131) has been blocked due to USERSRC(NOACCESS), Detail:  CLNTUSER(workstation)
```

This occurred because there is a Channel Authentication rule which blocks all connections to generic channel Client.to.Queue*.



Channel Authentication Records

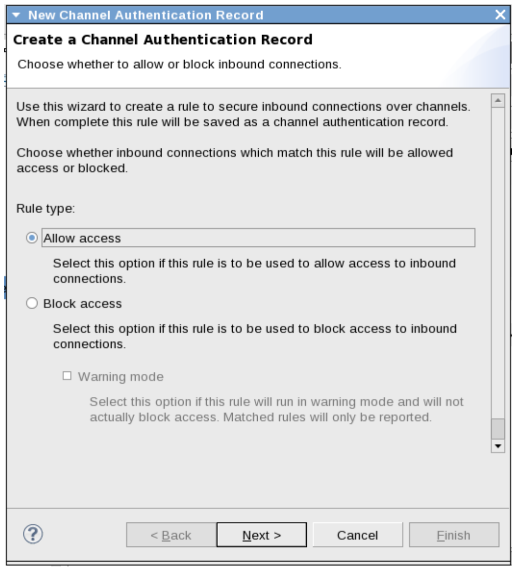
Filter: Standard for Channel Authentication Records

Channel profile	Type	Peer name	Client user ID	Remote queue manager	Address	User source	MCA user ID	User list
*	Block User List							*MQADI
Client.to.Queue*	Address Map				*	No Access		
SYSTEM.*	Address Map				*	No Access		
SYSTEM.ADMIN.SVRCONN	Address Map				*	Channel		
SYSTEM.DEF.SVRCONN	User Map		zosuser		*	Map	TEAMZZ	

Scheme: Standard for Channel Authentication Records - z/OS

Last updated: 11:00:46 (5 items)

____6. Add a new Channel Authentication Record that allows access to channel *Client.to.QueueMgr* for user *USER1*. Expand *Channels* and select *Channel Authentication Records*. Right mouse button click and select *New* -> *Channel Authentication Record* to display the New *Channel Authentication Record – Create a Channel Authentication Record* window.



New Channel Authentication Record

Create a Channel Authentication Record

Choose whether to allow or block inbound connections.

Use this wizard to create a rule to secure inbound connections over channels. When complete this rule will be saved as a channel authentication record.

Choose whether inbound connections which match this rule will be allowed access or blocked.

Rule type:

[✓ Allow access]

Select this option if this rule is to be used to allow access to inbound connections.

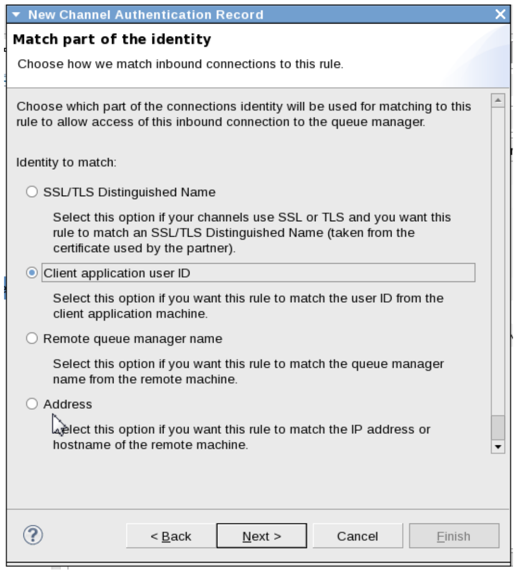
Block access

Select this option if this rule is to be used to block access to inbound connections.

☐ Warning mode

Select this option if this rule will run in warning mode and will not actually block access. Matched rules will only be reported.

____7. Take the default for the *Rule type* (*Allow access*) and press **Next** to continue. On the *Match part of the identity* window select the radio button by *Client application user ID*. Click **Next** to continue.



New Channel Authentication Record

Match part of the identity

Choose how we match inbound connections to this rule.

Choose which part of the connections identity will be used for matching to this rule to allow access of this inbound connection to the queue manager.

Identity to match:

- ○ SSL/TLS Distinguished Name

  Select this option if your channels use SSL or TLS and you want this rule to match an SSL/TLS Distinguished Name (taken from the certificate used by the partner).

- ◉ Client application user ID

  Select this option if you want this rule to match the user ID from the client application machine.

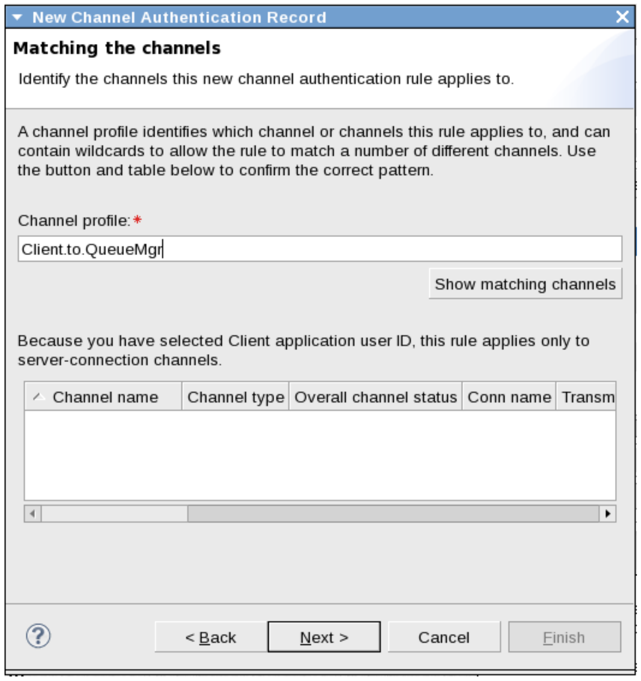
- ○ Remote queue manager name

  Select this option if you want this rule to match the queue manager name from the remote machine.

- ○ Address

  Select this option if you want this rule to match the IP address or hostname of the remote machine.

____8. On the *Matching the channels* window enter ***Client.to.QueueMgr*** in the area under *Channel Profile*. Press Next to continue.



New Channel Authentication Record

Matching the channels

Identify the channels this new channel authentication rule applies to.

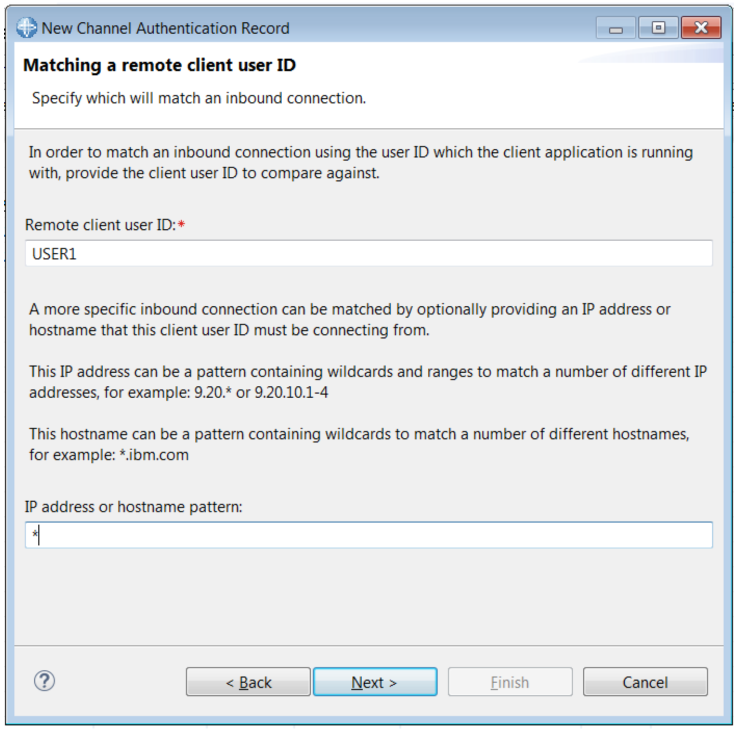
A channel profile identifies which channel or channels this rule applies to, and can contain wildcards to allow the rule to match a number of different channels. Use the button and table below to confirm the correct pattern.

Channel profile:* Client.to.QueueMgr

Because you have selected Client application user ID, this rule applies only to server-connection channels.

Channel name	Channel type	Overall channel status	Conn name	Transm

____9. On the **Matching a remote client user ID** window enter ***USER1*** as the *Remote client user ID* and an asterisk as the *IP address or hostname pattern*.



Matching a remote client user ID

Specify which will match an inbound connection.

In order to match an inbound connection using the user ID which the client application is running with, provide the client user ID to compare against.

Remote client user ID:* USER1

A more specific inbound connection can be matched by optionally providing an IP address or hostname that this client user ID must be connecting from.

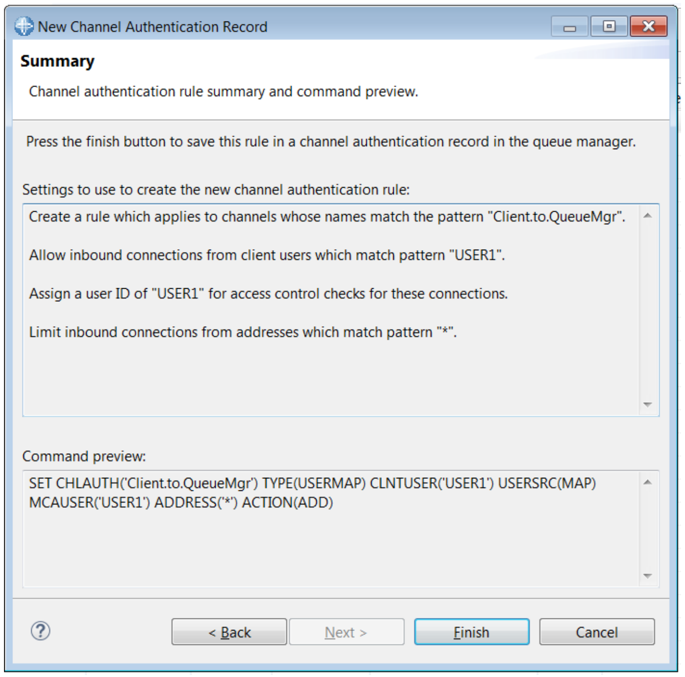
This IP address can be a pattern containing wildcards and ranges to match a number of different IP addresses, for example: 9.20.* or 9.20.10.1-4

This hostname can be a pattern containing wildcards to match a number of different hostnames, for example: *.ibm.com

IP address or hostname pattern: *

____10. On the *Authorization user ID* window enter ***USER1*** as the *Fixed user ID*. Press **Next** 3 times to continue.

____11. Click **Finish** on the *Summary* window to complete the definition of this authentication rule.



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 "Client.to.QueueMgr".

Allow inbound connections from client users which match pattern "USER1".

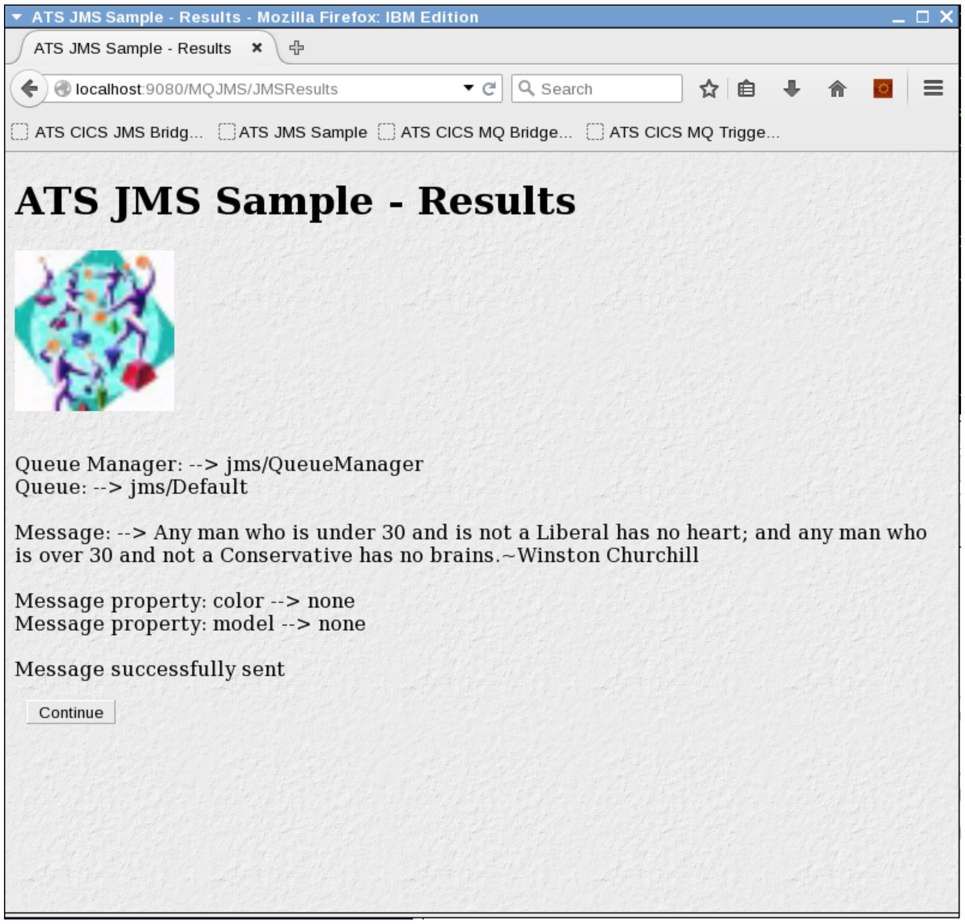
Assign a user ID of "USER1" for access control checks for these connections.

Limit inbound connections from addresses which match pattern "*".

Command preview:

```
SET CHLAUTH('Client.to.QueueMgr') TYPE(USERMAP) CLNTUSER('USER1') USERSRC(MAP)
MCAUSER('USER1') ADDRESS('*') ACTION(ADD)
```

____12. In the Mozilla browser press the **Continue** button to redisplay the initial screen. This time enter your user ID and password and press **Execute** gain. You should see the message *Message successfully sent*.



ATS JMS Sample - Results

Queue Manager: --> jms/QueueManager
Queue: --> jms/Default

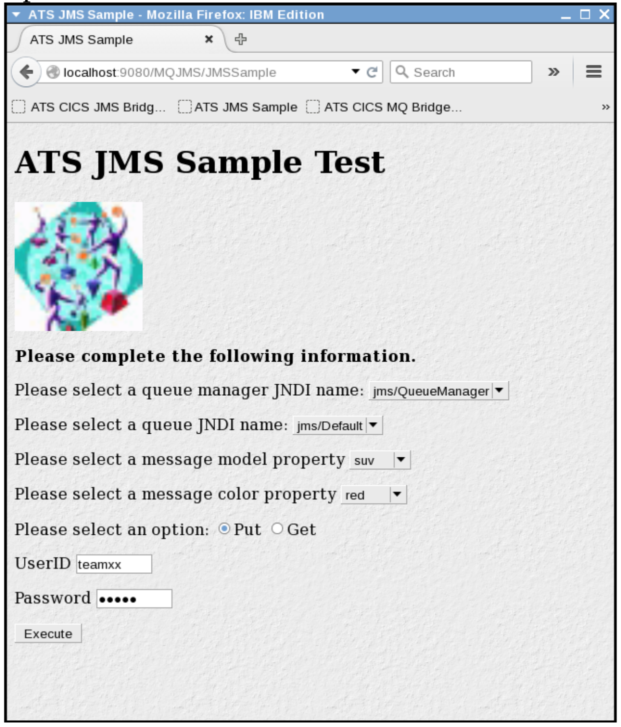
Message: --> Any man who is under 30 and is not a Liberal has no heart; and any man who is over 30 and not a Conservative has no brains.~Winston Churchill

Message property: color --> none
Message property: model --> none

Message successfully sent

____13. If not, review the console messages to see if an error message has occurred, for example message *IRR013I VERIFICATION FAILED. INVALID PASSWORD GIVEN* means that an invalid password was provided. Or if message *CSQX777E QMZ1 CSQXRESP Channel Client.to.QueueMgr from 192.27.141.40) has been blocked due to USERSRC(NOACCESS), Detail: CLNTUSER(USER1* occurs, then the user ID does not have a Channel Authentication Record that allows access.

____14. Put several more messages on the same queue but this time select various combinations of message properties. For example set the model to *suv* and the color to *red* as shown below.



ATS JMS Sample Test

Please complete the following information.

Please select a queue manager JNDI name: jms/QueueManager

Please select a queue JNDI name: jms/Default

Please select a message model property suv

Please select a message color property red

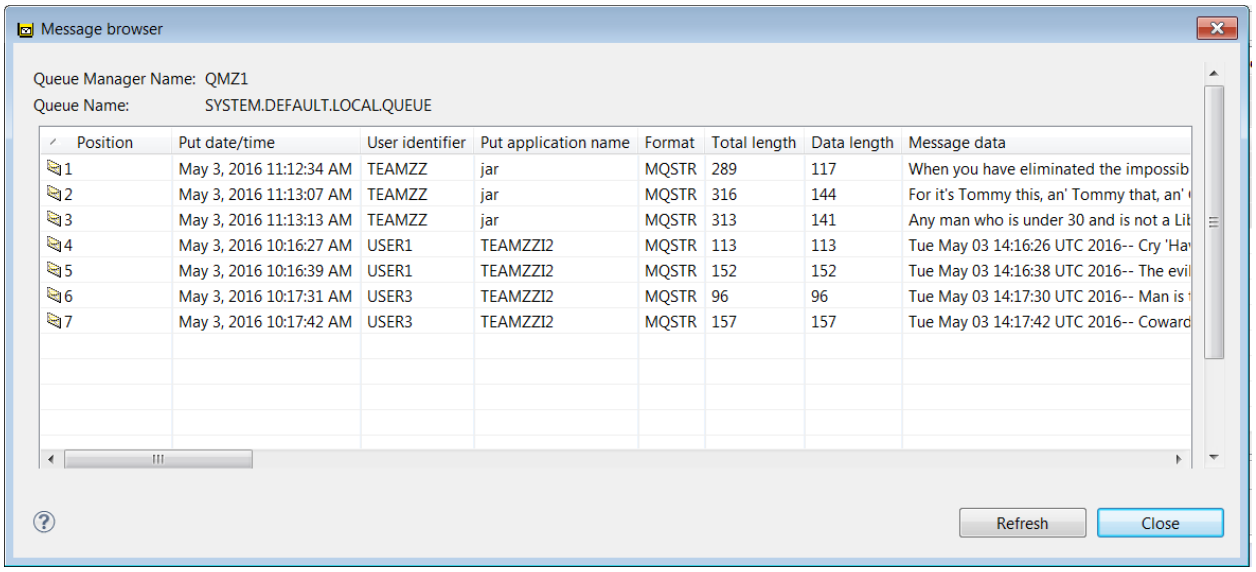
Please select an option: [✓ Put] Get

UserID teamxx

Password •••••

Now use the MQ Explorer to browse the contents on the queue and verify that the message properties were set as specified.

____15. If MQ Explorer is not already active, start it by clicking on the desktop icon. Expand your queue manger and then expand its queues until your queue is displayed. Select your queue and right mouse button and select the *Browse Messages* option. This should display a list of the messages in your queue.

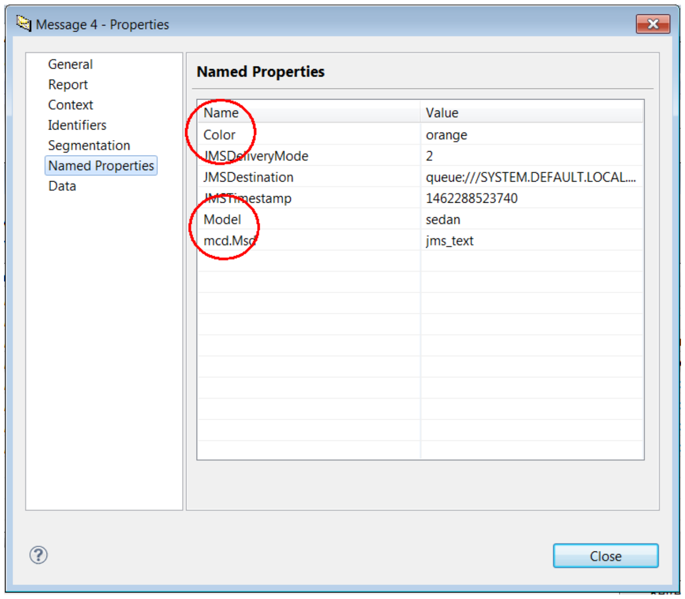


Queue Manager Name: QMZ1

Queue Name: SYSTEM.DEFAULT.LOCAL.QUEUE

Position	Put date/time	User identifier	Put application name	Format	Total length	Data length	Message data
1	May 3, 2016 11:12:34 AM	TEAMZZ	jar	MQSTR	289	117	When you have eliminated the impossib
2	May 3, 2016 11:13:07 AM	TEAMZZ	jar	MQSTR	316	144	For it's Tommy this, an' Tommy that, an'
3	May 3, 2016 11:13:13 AM	TEAMZZ	jar	MQSTR	313	141	Any man who is under 30 and is not a Lil
4	May 3, 2016 10:16:27 AM	USER1	TEAMZZI2	MQSTR	113	113	Tue May 03 14:16:26 UTC 2016-- Cry 'Ha
5	May 3, 2016 10:16:39 AM	USER1	TEAMZZI2	MQSTR	152	152	Tue May 03 14:16:38 UTC 2016-- The evil
6	May 3, 2016 10:17:31 AM	USER3	TEAMZZI2	MQSTR	96	96	Tue May 03 14:17:30 UTC 2016-- Man is
7	May 3, 2016 10:17:42 AM	USER3	TEAMZZI2	MQSTR	157	157	Tue May 03 14:17:42 UTC 2016-- Coward

____16. Next select a recent message placed on the queue by USER1 and right mouse button click. Select the *Named Properties* tab and confirm that the message properties for that message appear. What properties you see will depend entirely on which properties you selected when you put messages on the queue.



Name	Value
Color	orange
JMSDeliveryMode	2
JMSDestination	queue:///SYSTEM.DEFAULT.LOCAL....
JMSTimestamp	1462288523740
Model	sedan
mcd.Msd	jms_text

____17. Once you have a good set of messages with various combinations of message properties start selecting the **Get Message** button while selecting various message properties and verify the message retrieved has properties which match the criteria specified for message properties.

Tip: consider using the *MQ Explorer* to cross check the message properties displayed there versus the properties of the message retrieved. This might require keeping detail notes of which messages are in the queue and what properties they may or not have and considering multiple messages have the same combination of properties.

This completes the configuration of testing the access to the queue manager from a remote client.

Part 4 – Configuring the SSL MQ Connections

Exchanging digital certificates between MQ and MQExplorer requires the use of a trusted certificate authority (CA). The certificate authority's role is to sign (or create) a personal certificate that can be used by an application to identify itself to another application and verify the personal certificate presented at connection time can be trusted. The CA also provides a public CA digital certificate that can be used by application to validate personal certificate issued by that CA.

RACF commands will be used to create a CA signer certificate (known as a CERTAUTH certificate in RACF) that will be used later to sign client certificates used by clients and to create a personal certificate for the MQ channel initiator (CHIN) address space.

The first step to enable digital certificate exchanges between MQExplorer and the queue manager is to create a Certificate Authority (CA) certificate (also known as signer certificates), which will be used to create or sign personal certificates. When a personal certificate is presented to RACF, this CA certificate will be used to ensure the validity of the personal certificate.

____1. Use the TSO command **RACDCERT CERTAUTH GENCERT** command to create a signer certificate with label **'MQ CA'**.

```
racdcert certauth gencert subjectsdn(CN('MQ CA') OU('ATS')
O('IBM') C('US')) withlabel('MQ CA') keyusage(certsign)
notafter(date(2020/12/31))
```

____2. Use the **RACDCERT CERTAUTH GENCERT** command to create a personal certificate for the MQ channel initiator (CHIN) with a label of **'MQCHIN'**.

```
racdcert id(start1) gencert subjectsdn(CN('MQ CHIN') OU('ATS')
O('IBM') C('US')) withlabel('MQ CHIN') signwith(certauth
label('MQ CA')) notafter(date(2020/12/31)
```

Tech-Tip: START1 is the MQ channel initiator region's RACF identity under which the CHIN started task is executing.

Tech-Tip: The common name (CN), organization unite(OU), organization(O) country (C) , labels and aliases are case sensitive. Subsequent RACF or keytool commands referencing a label, an alias or common name, etc. in any command must use the exact same case and spacing when referring to the values of these fields in the certificate (i.e. be consistent).

For example, command ***racdcert certauth list(label('MQ CA'))*** would display this certificate while command ***racdcert certauth list(label('mq CA'))*** would not. Command ***racdcert certauth delete(label('MQ CA'))*** could be used to delete this certificate while command ***racdcert certauth delete(label('mq CA'))*** could not.

____3. Use the **RACDCERT ADDRING** command to create a RACF Key Ring for the MQ CHIN region's RACF identity. The signer certificates and WMQ's personal certificates will be connected to this key ring.

```
racdcert id(start1) addring(MQCHIN.KeyRing)
```

Tech-Tip: RACF key rings are case sensitive so be sure to enter the key ring name, e.g. MQ.KeyRing in exactly the same case in subsequent commands.

Tech-Tip: START1 needs READ access to FACILITY resource IRR.DIGTCERT.LISTRING in order to access this key ring.

____4. Use the **RACDCERT CONNECT** command to connect the MQ signer certificate that was used to sign the   'MQ CA' client's personal certificate to the MQ CHIN's key ring.

```
racdcert id(start1) connect(ring(MQCHIN.KeyRing) label('MQ
CA') certauth usage(certauth))
```

Tech-Tip: A RACF key ring is unique to the owning user. A key ring labeled "MQ.KeyRing" owned by user START1 is not the same key ring owned by user START2. So connecting a signer certificate labeled "MQ CA" with command **RACDCERT ID(START1) CONNECT(RING(MQCHIN.KeyRing)** has no effect on a key ring labeled MQCHIN.KeyRing owned by user START2. The significance of this is more apparent during key ring maintenance when a key ring is updated with new information but the application is actually using another key ring with the same name but owned by a different user.

____5. Use the **RACDCERT CONNECT** command to connect the MQ CHIN region's personal certificate to the MQCHIN.KeyRing.

```
racdcert id(start1) connect(ring(MQCHIN.KeyRing) label('MQ CHIN')
default)
```

Tech-Tip: The ***default*** parameter indicates that this is the personal certificate that will be provided to the remote application during a connection request.

____6. Use the RACF **RACDCERT LISTRING** command to display the MQCHIN.KeyRing contents

```
racdcert id(start1) listring(MQCHIN.KeyRing)

Digital ring information for user START2:

  Ring:
       >MQCHIN.KeyRing<
  Certificate Label Name                Cert Owner    USAGE       DEFAULT
  --------------------------------     ------------  --------    -------
  MQ CA                               CERTAUTH      CERTAUTH    NO
  MQCHIN                              ID(START1)    PERSONAL    YES
```

____7. The MQ signer certificate needs to be exported and installed in the client's trust stores. Use the **RACDCERT CERTAUTH EXPORT** command to export the MQ signer certificate from RACF into a sequential dataset.

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

Tech-Tip: This certificate will be used to validate the personal certificate provided by the CHIN address space during the connection process.

This completes the initial RACF setup of SSL support.

Part 5 – Configuring the MQ Explorer SSL Support

In this part the signer certificate created in RACF in the prior section will be downloaded to the workstation and installed in a trust store file. A self-signed personal certificate will be created in a key store and extracted and then sent to the certificate authority (e.g. RACF). The client's personal certificate will be signed and returned and imported into the key store.

Tech-Tip: Key store and trust store refer to the repository in which digital certificates are maintained in non-RACF managed environments. A key store contains the local personal certificate and its corresponding signer certificate and a trust store contains the signer certificate for personal certificates from remote application. A key store and trust store are functionally equivalent to RACF key rings. This document uses the same file for both the key store and trust store.

Note: The samples below show the commands being entered in a command prompt window. This document was written so show only commands that work on both Windows and Linux. There are GUI tools like **strmqikm** that can be used to maintain key stores and trust stores but, in this exercise, the ***keytool*** command will be used.

____18. Open a command prompt and go to directory /z/home using the cd command, use ***ftp*** to move the exported certificates from z/OS to Windows.

```
C:/z/home>ftp wg31.washington.ibm.com
Connected to wg31.washington.ibm.com (192.27.216.44).
220-FTPD1 IBM FTP CS V2R1 at WG31.WASHINGTON.IBM.COM, 17:28:58 on 2016-
01-19.
220 Connection will close if idle for more than 5 minutes.
Name (wg31.washington.ibm.com:workstation): user1
331 Send password please. xxxxxxxx
Password:
230 USER1 is logged on.  Working directory is "USER1.".
Remote system type is MVS.
ftp> mget mq.cacert.cer
mget MQ.CACERT.CER? y
227 Entering Passive Mode (192,27,216,44,4,38)
125 Sending data set USER1.MQ.CACERT.CER
250 Transfer completed successfully.
878 bytes received in 0.0139 secs (63.20 Kbytes/sec)
ftp> quit
221 Quit command received. Goodbye.
```

Tech-Tip: This document assumes that the key/trust store file accessed is located in directory c:/z/jndi. This means that the keytool command needs to be invoked while in this directory or that the –keystore parameter included the full path name as in -keystore c:/z/jndi/user1.jks

____19. Import the MQ Certificate Authority certificate using the **keytool import** command to import the signer certificate into local trust store file (e.g. **USER1.jks)** and indicate that this is a trusted signer certificates (-trustcacerts).

```
keytool –import -v –trustcacerts –alias "MQ CA" -file MQ.CACERT.CER –keystore
USER1.jks
Enter keystore password:  changeit
Re-enter new password:
Owner: CN=MQ CA, O=IBM, C=US
Issuer: CN=MQ CA, O=IBM, C=US
Serial number: 0
Valid from: 1/18/16 11:00 PM until: 12/31/20 10:59 PM
Certificate fingerprints:
       MD5:  B5:47:87:7E:4C:13:1F:27:9B:A7:4A:FE:8D:55:FE:5B
       SHA1: 89:1B:2F:BC:2D:3E:EC:7F:7B:3B:64:08:37:E9:88:8F:F8:38:05:9F
       SHA256:
A9:62:DC:44:42:CF:7C:35:0F:CB:AC:64:65:5C:D7:27:69:29:27:6F:D3:E3:98:34:29:58:
09:F7:4B:E0:9A:F7
       Signature algorithm name: SHA1withRSA

…..

#4: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 77 d0 79 19 1d e7 f6 44  a1 6a d6 20 0f d1 8a 22  w.y....D.j......
0010: e3 39 31 a8                                        .91.
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
[Storing USER1.jks]
```

Tech-Tip: The key store file USER1.jks is automatically created if it does not already exist.

____20. Use the **keytool genkey** command to generate a self-signed certificate with the desired distinguished name specification. This is required in order to generate a certificate request that will be sent to RACF for signing by the MQ CA certificate.

```
C:\z\jndi> keytool  -genkey -alias "USER1" -dname "CN=USER1, OU=ATS, O=IBM,
C=US" -keystore USER1jks -keyalg RSA
Enter keystore password:  CHANGEIT
Enter key password for <USER1>:
      (RETURN if same as keystore password):
```

____21. Use the **keytool certreq** command to extract the self-signed certificate to a certificate request file that can be uploaded to z/OS.

```
C:\z\jndi> keytool -certreq -alias "USER1" -file certreq.cer        -
keystore USER1.jks
Enter keystore password:  changeit
```

Tech-Tip: IBM Key Management tool can be started by entering command ***strmqikm***. Start IKeyMan and open USER1.jks and explore the store file contents after these commands have been entered.

____22. Use **ftp** to move the certificate request to z/OS.

```
C:\z\jndi> ftp wg31.washington.ibm.com
Connected to mpx1 (192.27.216.44).
220-FTPD1 IBM FTP CS V2R1 at WG31.WASHINGTON.IBM.COM, 17:42:43 on 2016-01-19.
220 Connection will close if idle for more than 5 minutes.
Name (mpx1:zosuser): USER1
331 Send password please. ******
Password:
230 USER1 is logged on.  Working directory is "USER1.".
Remote system type is MVS.
ftp> quote site recfm=vb lrecl=256 blksize=0
200 SITE command was accepted
ftp> mput certreq.cer
mput  certreq.cer? y
227 Entering Passive Mode (192,27,216,44,4,39)
125 Storing data set USER1.CERTREQ.CER
250 Transfer completed successfully.
1019 bytes sent in 4e-05 secs (25475.00 Kbytes/sec)
ftp> quit
221 Quit command received. Goodbye.
```

____23. Browse **USER1.CERTREQ.CER** and you should see something like below:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBbjCB2AIBADAvMQwwCgYDVQQKEwNJQk0xDzANBgNVBAsTBldNQkFETTEOMAwGA1UEAxMFdXNl
cjEwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKD3hxwjl+FxkojgMT58amx3l95yQnStBV8F
wTTeFmA27OUEvtpY0n3YPGYG7ShE0T6idjp8Qfq3MMoiadxlULZ844UMJoB12CRcJdlbIwCuUNTH
6TOjHggl5cNja8H9N4uSaaOK8sPLKByMaE04yXHL44Wkfjgn9HJTvinbZZf5AgMBAAGgADANBgkq
hkiG9w0BAQQFAAOBgQBK2LRYSxMJieQQPgpU42+IukNV49BMIyZFxGhwrB42E8pUZWGLgqOLucjw
LyVuqOHSkIvp4fVp1k+9sMQFhCe+voG/apRDUG2xNBDFKj6DItBldsK4jr3bEh8Oc8Dy47FWiG2B
cKrFi5MmdR5wukFfiHoXjY6RIjNsXfMR1t7C5g==
-----END NEW CERTIFICATE REQUEST-----
```

____24. After uploading the certificate request to z/OS and use the **RACDCERT GENCERT** command to sign the certificate request with the RACF MQ signer certificate and associate this certificate with a valid RACF identity (e.g. *USER1*).

```
racdcert id(USER1) gencert(certreq.cer) withlabel('USER1')
signwith(certauth label('MQ CA')) notafter(date(2020/12/31))
```

____**25.** Export the signed client certificate request file to a sequential z/OS dataset using the **RACDCERT EXPORT** command.

```
racdcert id(USER1) export(label('USER1')) dsn(cert.cer)
```

____26. Browse *USER1.CERT.CER* and you should see something like below:

```
-----BEGIN CERTIFICATE-----
MIICcDCCAdmgAwIBAgIBAjANBgkqhkiG9w0BAQUFADBGMQwwCgYDVQQKEwNJQk0x
EDAOBgNVBAsTB1dNQlVTRVIxJDAiBgNVBAMTG1dNQiBDZXJ0aWZpY2F0aW9uIEF1
dGhvcml0eTAeFw0xMzAxMjIwNDAwMDBaFw0xNjAxMDEwMzU5NTlaMC8xDDAKBgNV
BAoTA0lCTTEPMA0GA1UECxMGV01CQURNMQ4wDAYDVQQDEwV1c2VyMTCBnzANBgkq
hkiG9w0BAQEFAAOBjQAwgYkCgYEAoPeHHCOX4XGSiOAxPnxqbHeX3nJCdK0FXwXB
NN4WYDbs5QS+2ljSfdg8ZgbtKETRPqJ2OnxB+rcwyiJp3GVQtnzjhQwmgHXYJFwl
2VsjAK5Q1MfpM6MeCCXlw2Nrwf03i5Jpo4ryw8soHIxoTTjJccvjhaR+OCf0clO+
Kdtll/kCAwEAAaOBhDCBgTA/BglghkgBhvhCAQ0EMhYwR2VuZXJhdGVkIGJ5IHRo
ZSBTZWN1cml0eSBTZXJ2ZXIgZm9yIHovT1MgKFJBQ0YpMB0GA1UdDgQWBBR5SBn+
45hD2A0BcJ9ChlHEh0zDbTAfBgNVHSMEGDAWgBRZ//MnfdVS1BDsMoppTrw/HWuQ
EDANBgkqhkiG9w0BAQUFAAOBgQB9Xi4ejbGecvOSxuJ8TaFeCTSk4q9+M/zFbRev
/bnJNnX9+UZ5j6WhBZ4HXTxhIJPhge3u+wFiuoSyvx7EKG6PtyoyJSO/ydl6VwmI
Z81XUPsnWYqZmTQ1Lr9iJkV2MBtsXWn73pww2x1wuH3BBd27aNsmd0otonw5W0gS
qh/m4w==
-----END CERTIFICATE-----
```

____27. Use ftp to move the signed certificate in dataset USER1.CERT.CER to Linux (see Step 1 above).

____28. Use the *keytool –import* command to import the signed certificate into the JSSE keystore.

```
C:\z\jndi> keytool -v -import -alias "USER1" -file CERT.CER
       -keystore USER1.jks
Enter keystore password: changeit
Certificate reply was installed in keystore
[Storing USER1.jks]
```

Tech-Tip: If the message "Certificate reply was installed in keystore" is not displayed then there was problem somewhere in this process. Steps 2 to 12 could repeated after removing file user1.jks and deleting the USER1 certificate with command ***racdcert id(USER1) delete(label('USER1')***

____29. Use the *keytool list* command to display the contents of the key store.

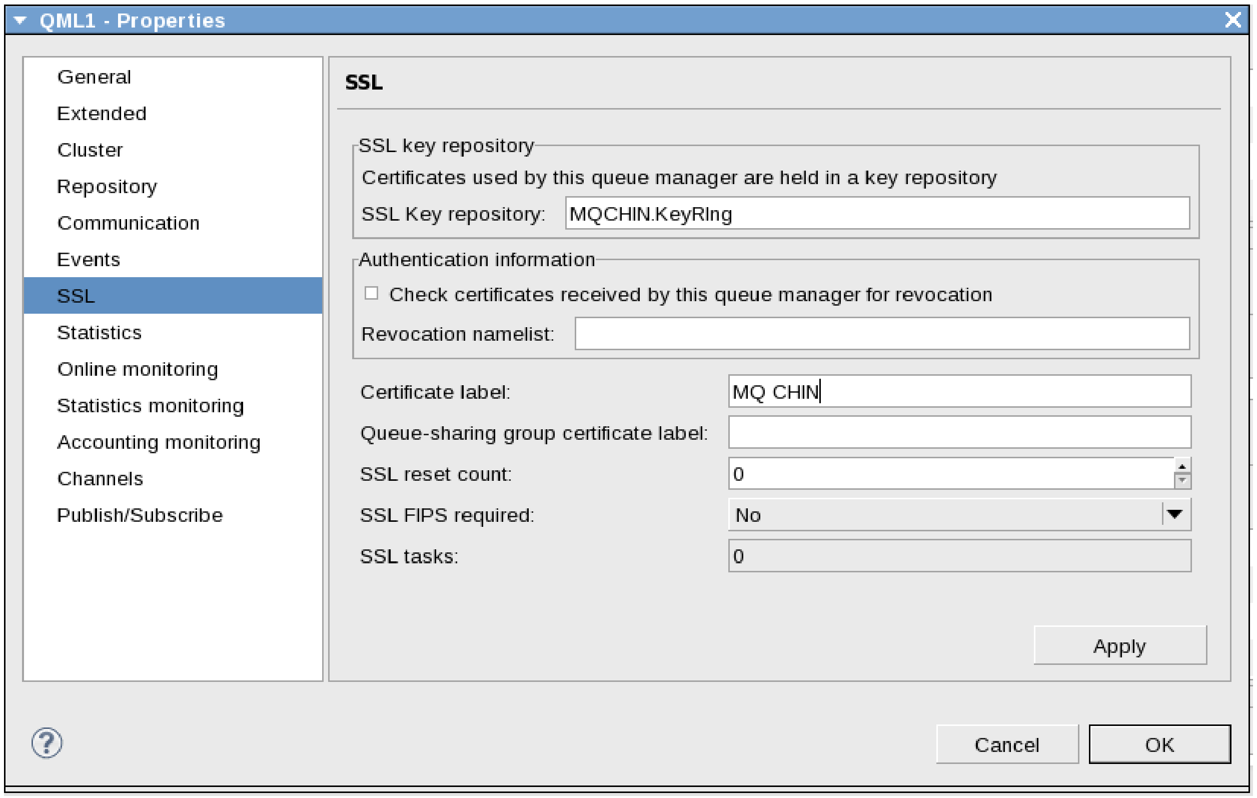
```
C:\z\jndi> keytool -list -keystore USER1.jks
Enter keystore password: changeit

Keystore type: jks
Keystore provider: IBMJCE

Your keystore contains 2 entries

mq ca, Jan 19, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
89:1B:2F:BC:2D:3E:EC:7F:7B:3B:64:08:37:E9:88:8F:F8:38:05:9F
USER1, Jan 19, 2016, keyEntry,
Certificate fingerprint (SHA1):
41:38:59:DE:24:0F:0C:C7:9F:E0:C4:F2:1C:32:0C:45:21:35:DF:93
```

Part 6 – Adding SSL Support to a MQ queue manager

To enable SSL support in a queue manager at least 2 queue manager properties need to be modified. The first is the name of the RACF key ring (*SSLKEYR*) to be accessed for personal and certificate authority digital certificates. In this example the key ring created in Part 3 will be used. Remember the key ring must belong to the user associated with the CHIN address space. The other property is the number of SSL sub tasks (*SSLTASKS*) for processing SSL calls.

Note: Sometimes the term SSL (Secure Socket Layer) is used when discussing digital certificates. SSL does involve the exchange of digital certificates but it also deals with the encryption of the data being sent.

___1. The SSL key ring can be specified using the MQExplorer. Connect to the queue manger in the *Queue Manager* folder and after the connection is established right mouse button click. Select **Properties → SSL** to display the **Properites - SSL** widow (see below). Enter the key ring name ***MQCHIN.KeyRing*** (remember case matters) created in the Part 3, Step 3 in the area beside *SSL Key repository* (see below). Enter ***MQ CHIN*** as the *Certificate label* and click **OK** to continue.



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

___2. The key ring name is case sensitive and probably could not be entered using an MVS modify command. But number of SSL task can be enter using an MVS modify command (be sure to set the number of tasks to a value greater than 2) as in

/QMZ1 ALTER QMGR SSLTASKS(5)

Tech-Tip: Member CSQ4INYG contains an ALTER QMGR command. If this member is present in the CSQINP2 data definition JCL statement in the queue mangers master JCL procedure verify that it does not reset SSLTASKS or the SSLKEYR properties to other values.

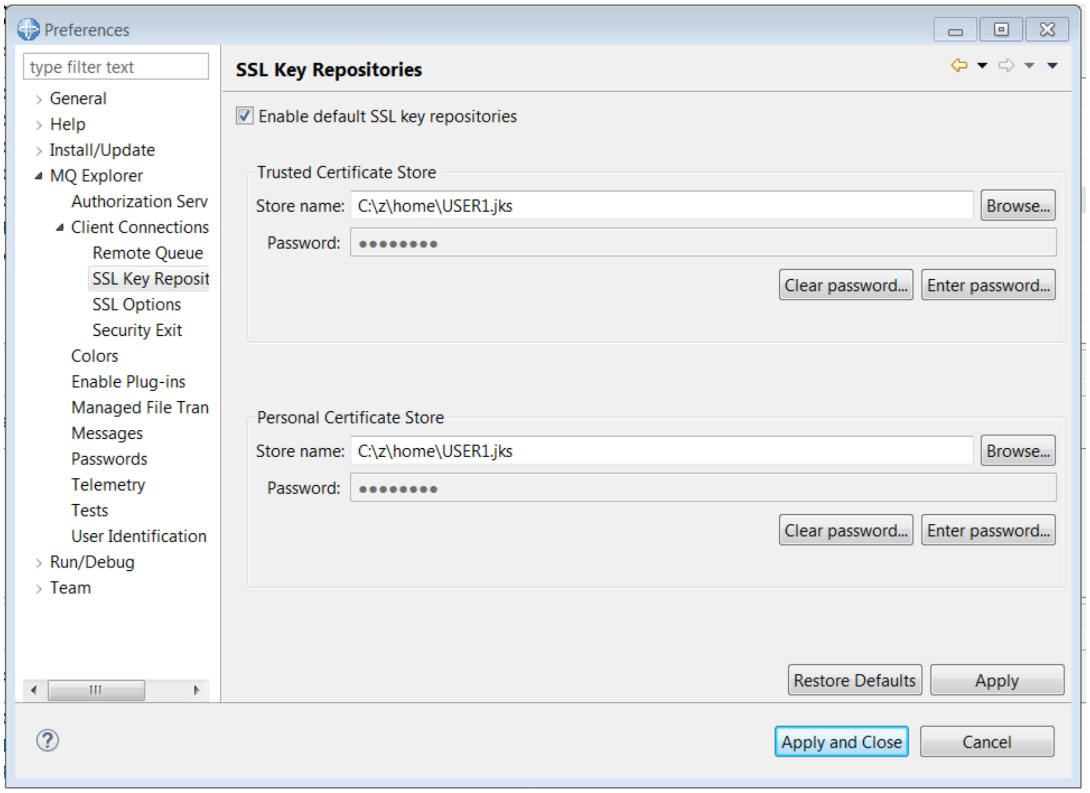
___3. Stop the channel initiator task with MVS command

/QMZ1 STOP CHINIT

___4. When the channel initiator address space (QMZ1CHIN) terminates, restart it with MVS command

/QMZ1 START CHINIT

___5. Start MQ Explorer on the desktop if it is not already active. Click on *Window* on the tool bar and then click on *Preferences*. Expand the MQ Explorer option and then expand *Client Connections*. Select *SSL Key Repositories* preferences. Check the box beside Enable default SSL key repositories and then use the **Browse** buttons to select the key store configured earlier in this exercise (e.g. c:\z\home\USER1.jks) for the *Store names* for *Trusted Certificates* and *Personal Certificates* (see below). Use the **Enter password** button to enter the password, e.g. ***changeit***, for both store files. Click **Apply and Close** to continue.



SSL Key Repositories

☑ Enable default SSL key repositories

Trusted Certificate Store

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

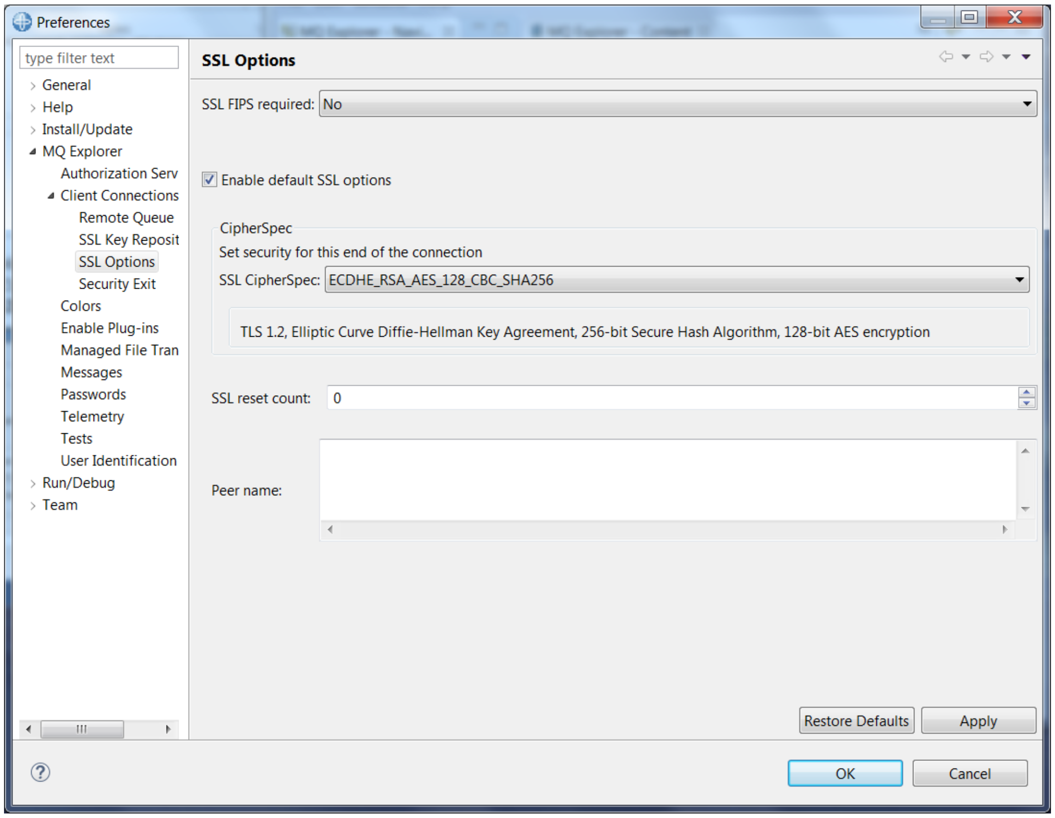
Password: ●●●●●●●●

Personal Certificate Store

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

Password: ●●●●●●●●

___6. Switch to the *SSL Options* preference window. Check the box beside *Enable default SSL options* and then use the pull down to select a *SSL CipherSpec* of **ECDHE_RSA_AES_128_CBC_SHA256** (see below). Click **Apply and Close** to continue.



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

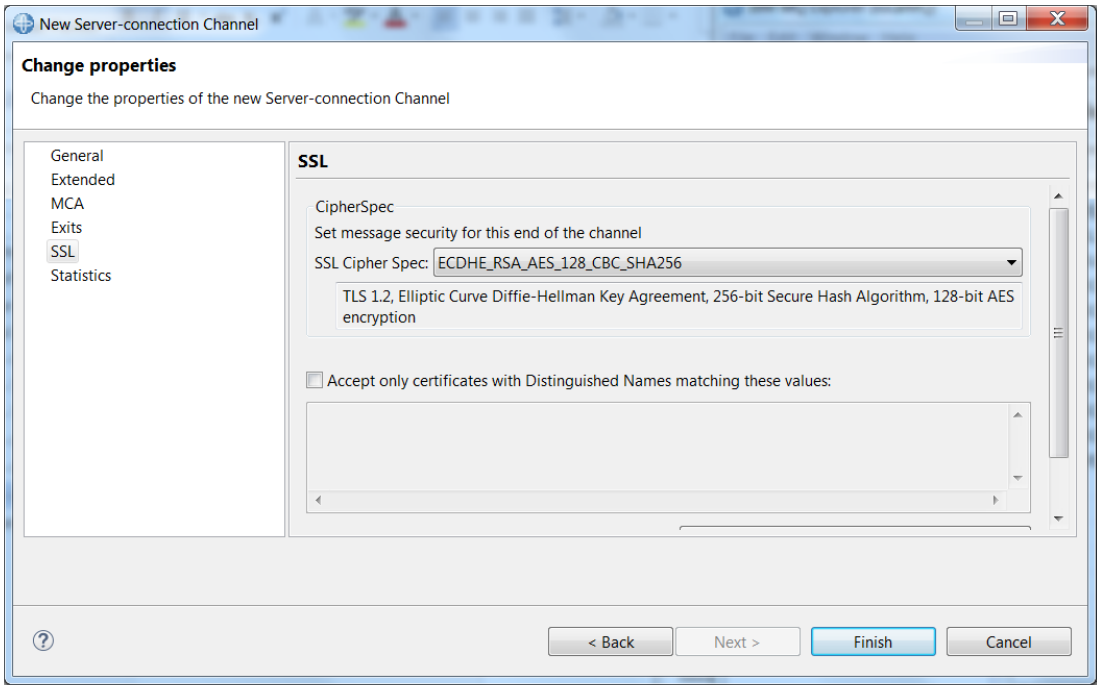
SSL reset count: 0

Peer name:

Tech-Tip: Which cipher suite selected is not important as long as the same cipher is used consistently.

___7. Using an existing non-SSL connection to QMZ1, create a new server connection channel by selecting *Channels* and right mouse button clicking. Select *New -> Server-connection Channel*. Enter a name of ***USER1.SSL.SVRCONN*** and click **Next** to continue.

___8. On the *New Server-connection Channel* window select *SSL* and use the pull down arrow to select *ECDHE_RSA_AES_128_CBC_SHA256* as the *SSL Cipher Spec* (see below). Select **General** to continue.


New Server-connection Channel

Change properties

Change the properties of the new Server-connection Channel

SSL

CipherSpec

Set message security for this end of the channel

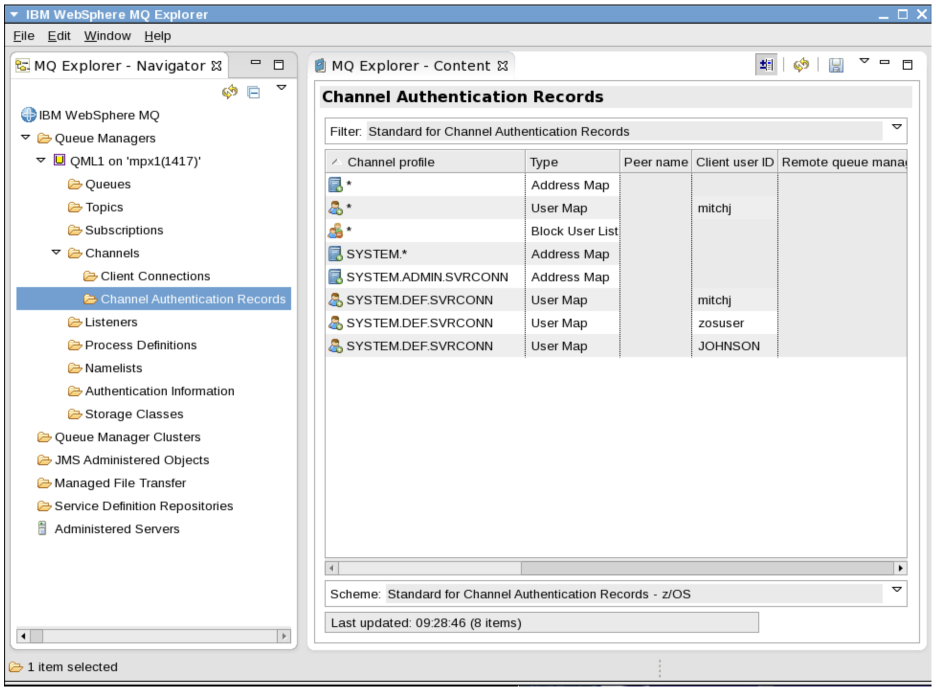
SSL Cipher Spec: ECDHE_RSA_AES_128_CBC_SHA256

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

☐ Accept only certificates with Distinguished Names matching these values:

___9. On the General window use the pull-down arrow to select TCP. Click **Finish** to continue.

___10. Next a Channel Authentication record needs to be added to allow access to the *USER1.SSL.SVRCONN* channel. Using a non-SSL connection to *QMZ1* expand *Channels* and select *Channel Authentication Records*.



Channel Authentication Records

Filter: Standard for Channel Authentication Records

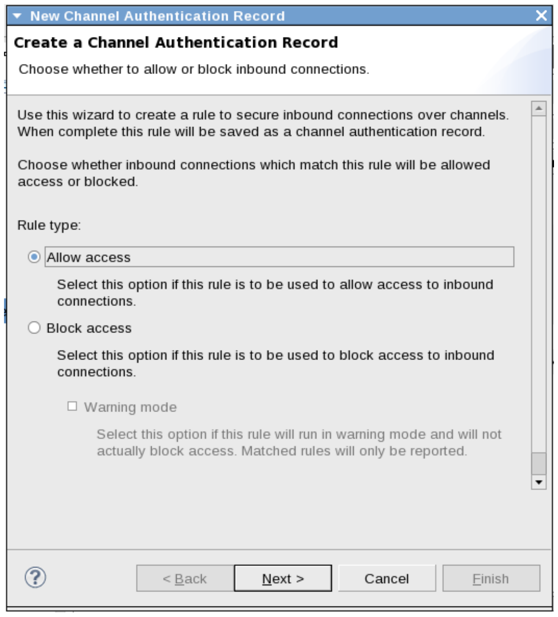
Channel profile	Type	Peer name	Client user ID	Remote queue mana
*	Address Map			
*	User Map		mitchj	
*	Block User List			
SYSTEM.*	Address Map			
SYSTEM.ADMIN.SVRCONN	Address Map			
SYSTEM.DEF.SVRCONN	User Map		mitchj	
SYSTEM.DEF.SVRCONN	User Map		zosuser	
SYSTEM.DEF.SVRCONN	User Map		JOHNSON	

Scheme: Standard for Channel Authentication Records - z/OS

Last updated: 09:28:46 (8 items)

1 item selected

___11. Right mouse button click and select *New -> Channel Authentication Record* to display the New *Channel Authentication Record – Create a Channel Authentication Record* window. Take the default for the *Rule type* (*Allow access*) and press **Next** to continue.



New Channel Authentication Record

Create a Channel Authentication Record

Choose whether to allow or block inbound connections.

Use this wizard to create a rule to secure inbound connections over channels. When complete this rule will be saved as a channel authentication record.

Choose whether inbound connections which match this rule will be allowed access or blocked.

Rule type:

(●) Allow access

Select this option if this rule is to be used to allow access to inbound connections.

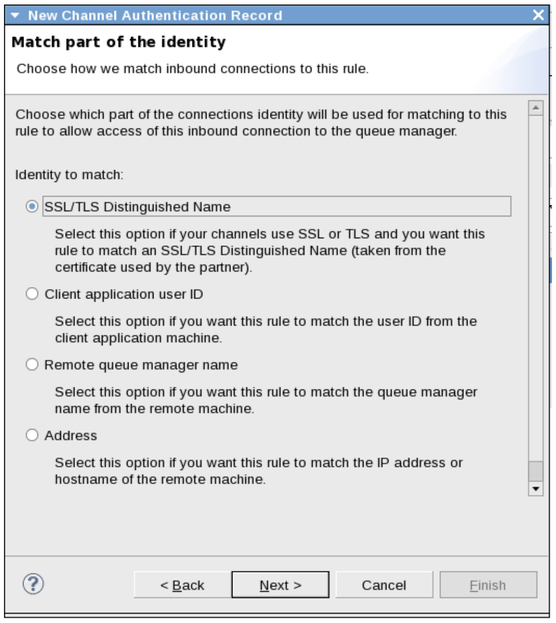
( ) Block access

Select this option if this rule is to be used to block access to inbound connections.

☐ Warning mode

Select this option if this rule will run in warning mode and will not actually block access. Matched rules will only be reported.

___12. On the *Match part of the identity* window select the radio button by *SSL/TLS Distinguished Name*. Click **Next** to continue.



New Channel Authentication Record

Match part of the identity

Choose how we match inbound connections to this rule.

Choose which part of the connections identity will be used for matching to this rule to allow access of this inbound connection to the queue manager.

Identity to match:

(●) SSL/TLS Distinguished Name

Select this option if your channels use SSL or TLS and you want this rule to match an SSL/TLS Distinguished Name (taken from the certificate used by the partner).

( ) Client application user ID

Select this option if you want this rule to match the user ID from the client application machine.

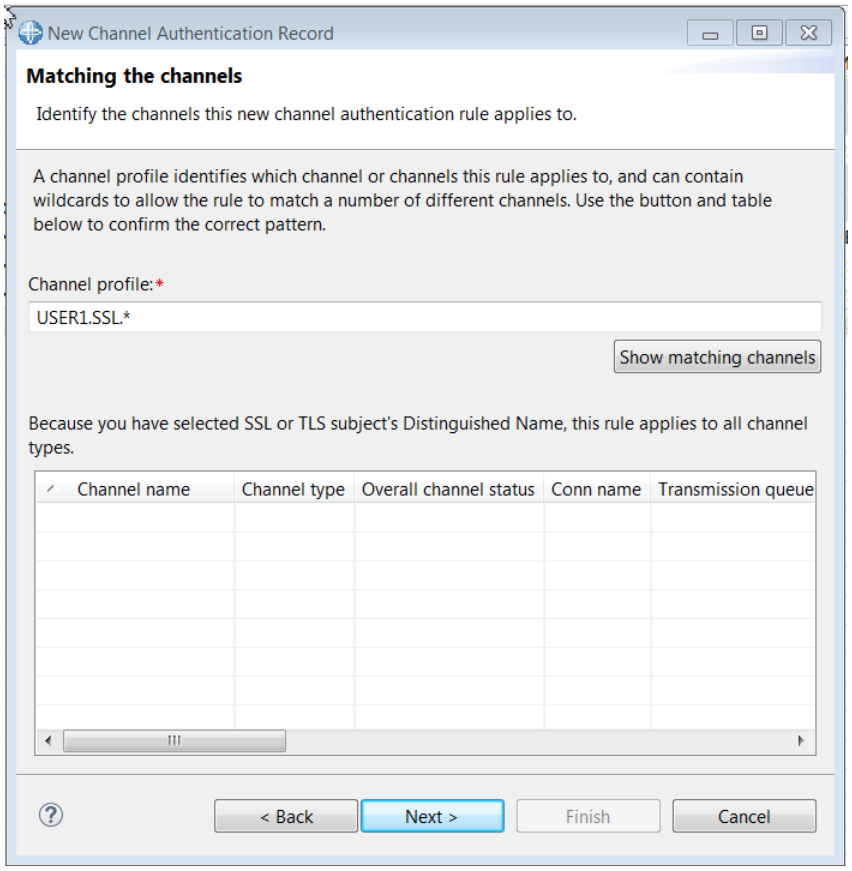
( ) Remote queue manager name

Select this option if you want this rule to match the queue manager name from the remote machine.

( ) Address

Select this option if you want this rule to match the IP address or hostname of the remote machine.

___13. On the *Matching the channels* window enter ***USER1.SSL.**** in the area under *Channel Profile*. Press **Next** to continue.



New Channel Authentication Record

Matching the channels

Identify the channels this new channel authentication rule applies to.

A channel profile identifies which channel or channels this rule applies to, and can contain wildcards to allow the rule to match a number of different channels. Use the button and table below to confirm the correct pattern.

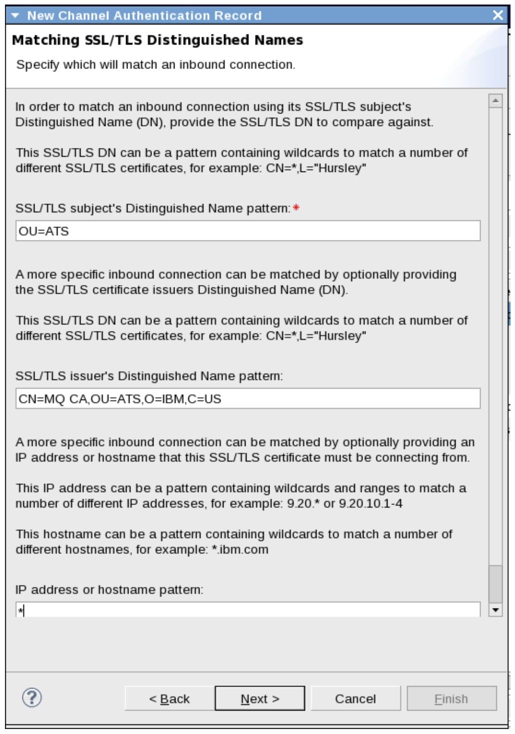
Channel profile:*

USER1.SSL.*

Because you have selected SSL or TLS subject's Distinguished Name, this rule applies to all channel types.

Channel name	Channel type	Overall channel status	Conn name	Transmission queue

___14. On the **Matching SSL/TLS Distinguished Names** window enter ***OU=ATS*** as the *SSL/TLS subject's Distinguished Name pattern* and ***CN=MQ CA,OU=ATS,O=IBM,C=US*** as *the SSL/TLS issuer's Distinguished Name pattern*. Enter an asterisk (*) for the *IP address or hostname pattern*. Click **Next** to continue.



New Channel Authentication Record

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 certificates, for example: CN=*,L="Hursley"

SSL/TLS subject's Distinguished Name pattern:*

OU=ATS

A more specific inbound connection can be matched by optionally providing the SSL/TLS certificate issuers Distinguished Name (DN).

This SSL/TLS DN can be a pattern containing wildcards to match a number of different SSL/TLS certificates, for example: CN=*,L="Hursley"

SSL/TLS issuer's Distinguished Name pattern:

CN=MQ CA,OU=ATS,O=IBM,C=US

A more specific inbound connection can be matched by optionally providing an IP address or hostname that this SSL/TLS certificate must be connecting from.

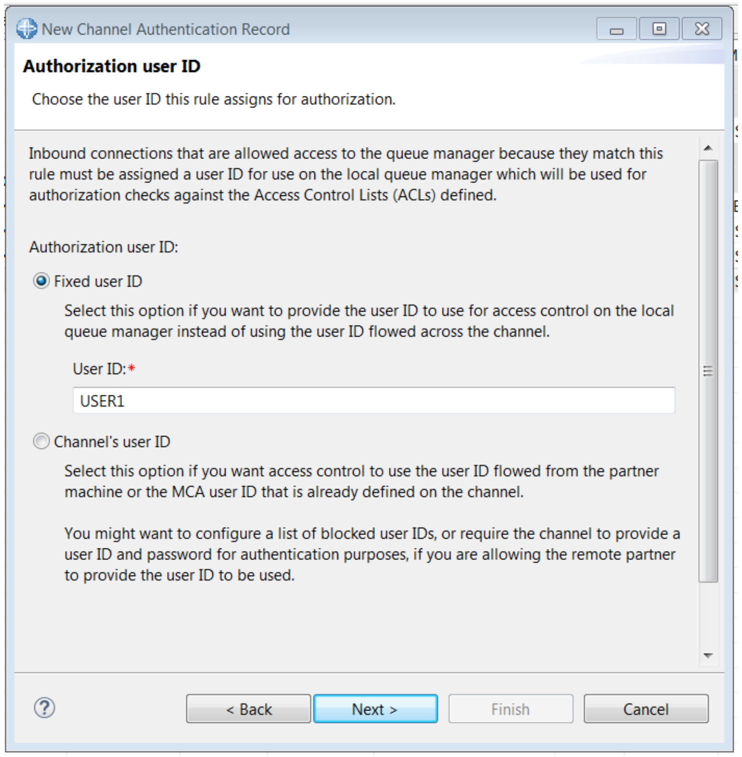
This IP address can be a pattern containing wildcards and ranges to match a number of different IP addresses, for example: 9.20.* or 9.20.10.1-4

This hostname can be a pattern containing wildcards to match a number of different hostnames, for example: *.ibm.com

IP address or hostname pattern:

*

___15. On the *Authorization user ID* window click the radio button beside *Fixed user ID* and enter ***USER1*** as the *User ID*. Press **Next** 3 times to continue.



Authorization user ID

Choose the user ID this rule assigns for authorization.

Inbound connections that are allowed access to the queue manager because they match this rule must be assigned a user ID for use on the local queue manager which will be used for authorization checks against the Access Control Lists (ACLs) defined.

Authorization user ID:

[✓ Fixed user ID]

Select this option if you want to provide the user ID to use for access control on the local queue manager instead of using the user ID flowed across the channel.

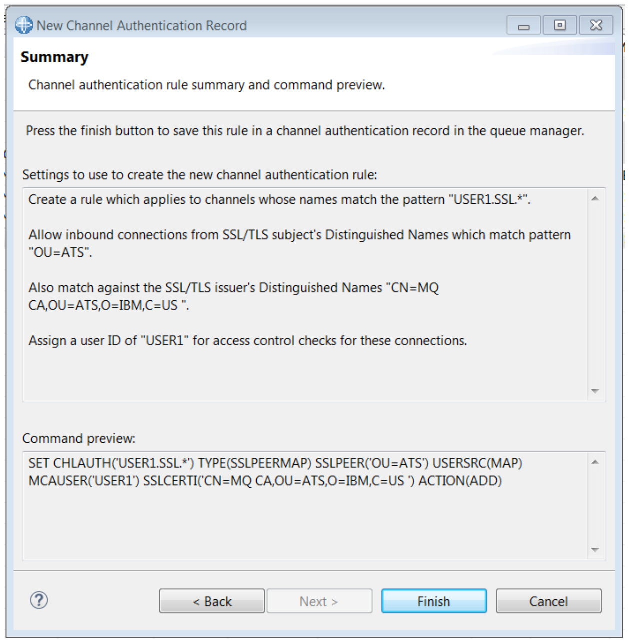
User ID:* USER1

Channel's user ID

Select this option if you want access control to use the user ID flowed from the partner machine or the MCA user ID that is already defined on the channel.

You might want to configure a list of blocked user IDs, or require the channel to provide a user ID and password for authentication purposes, if you are allowing the remote partner to provide the user ID to be used.

___16. Click **Finish** on the Summary window to complete the definition of this authentication rule.



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".

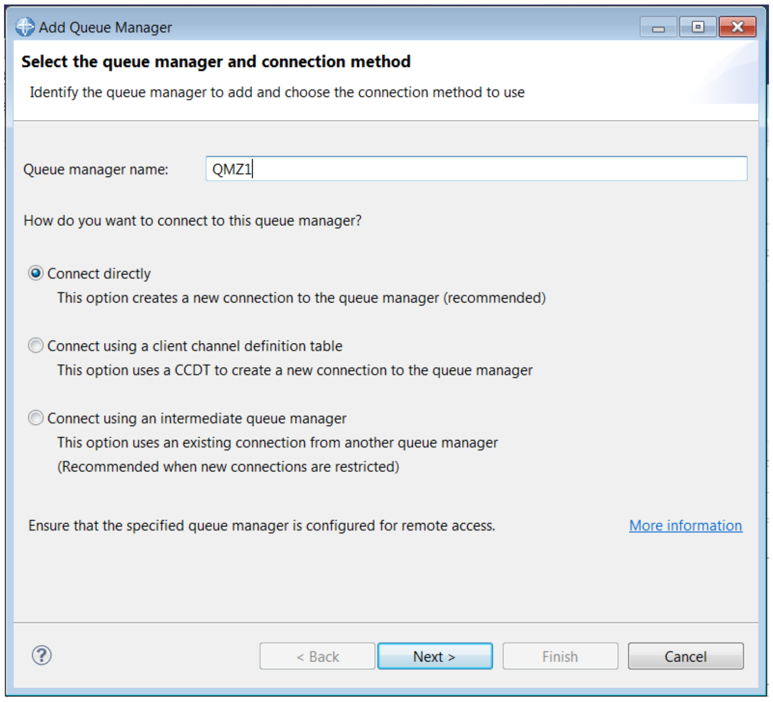
Also match against the SSL/TLS issuer's Distinguished Names "CN=MQ CA,OU=ATS,O=IBM,C=US ".

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)
```

___17. Next create a new remote connection to queue manager QMZ1.



Select the queue manager and connection method

Identify the queue manager to add and choose the connection method to use

Queue manager name: QMZ1

How do you want to connect to this queue manager?

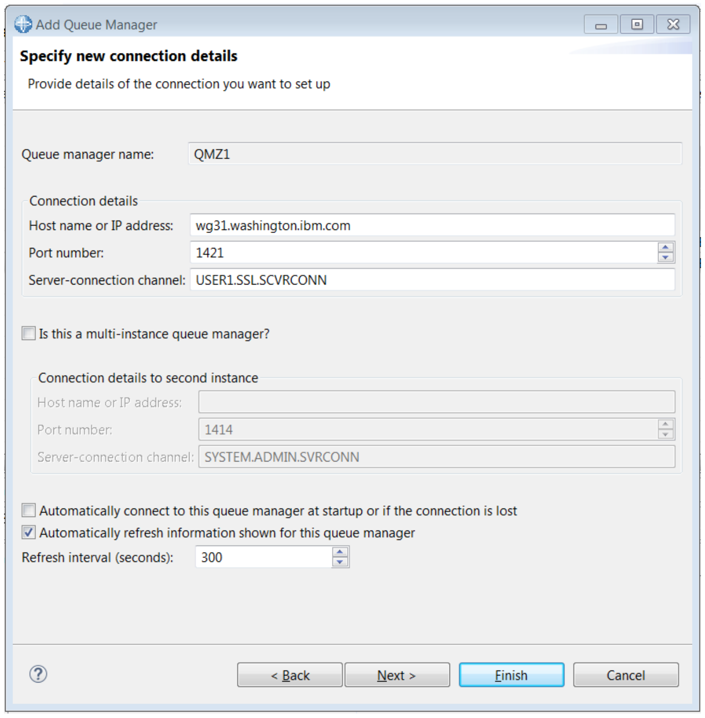
[✓ Connect directly]
This option creates a new connection to the queue manager (recommended)

Connect using a client channel definition table
This option uses a CCDT to create a new connection to the queue manager

Connect using an intermediate queue manager
This option uses an existing connection from another queue manager
(Recommended when new connections are restricted)

Ensure that the specified queue manager is configured for remote access. More information

___18. On the *Specify new connection details* window enter ***wg31.washington.ibm.com*** as the *Host name*, ***1421*** as the *Port Number* and ***USER1.SSL.SVRCONN*** as the *Server-connection channel* name. Click **Next** 3 times to continue.



Specify new connection details

Provide details of the connection you want to set up

Queue manager name: QMZ1

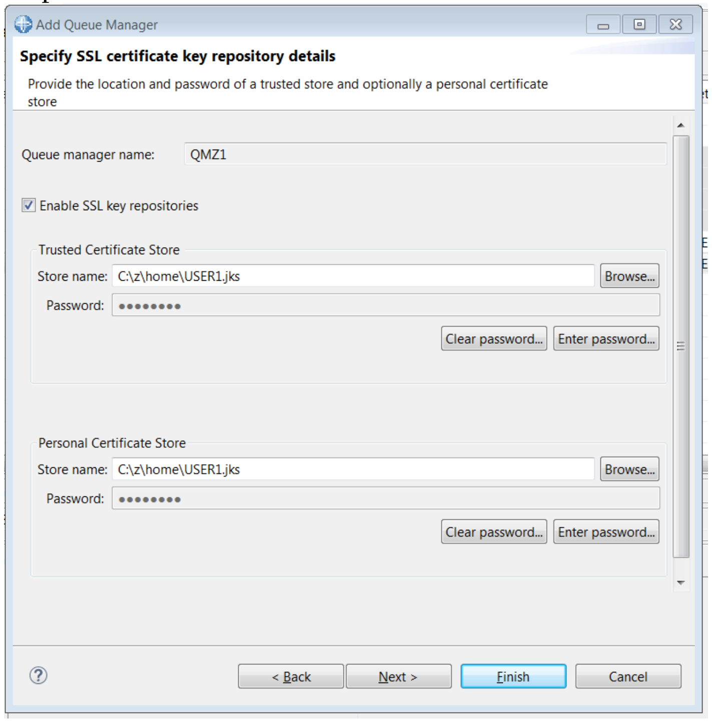
Connection details
Host name or IP address: wg31.washington.ibm.com
Port number: 1421
Server-connection channel: USER1.SSL.SCVRCONN

☐ 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

___19. On the *Specify SSL certificate key repository details* window ensure the box beside *Enable SSL key repositories* is checked. If the *Trusted Certificate Store* is not configured use the **Browse** and **Enter password** buttons to provide this information. Click **Finish** to continue.


Add Queue Manager
Specify SSL certificate key repository details
Provide the location and password of a trusted store and optionally a personal certificate store

Queue manager name: QMZ1

☑ Enable 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: ••••••••

Browse...
Clear password...
Enter password...
< Back
Next >
Finish
Cancel

___20. You should now be connected to the queue manager using SSL. To confirm enter this MVS command:

QMZ1 display chstatus(USER1.*) mcauser sslcertu

The RACF identity associated with the client's certificate is displayed in the SSLCERTU property.

```
QMZ1 DISPLAY CHSTATUS(USER1.*) MCAUSER SSLCERTU
CSQM293I QMZ1 CSQMDRTC 1 CHSTATUS FOUND MATCHING REQUEST
CRITERIA
CSQM201I QMZ1 CSQMDRTC  DISPLAY CHSTATUS DETAILS 954
CHSTATUS(USER1.SSL.SVRCONN)
CHLDISP(PRIVATE)
CONNAME(192.27.216.40)
CURRENT
CHLTYPE(SVRCONN)
STATUS(RUNNING)
SUBSTATE()
STOPREQ(NO)
RAPPLTAG(MQ Explorer 9.1.0)
SSLCERTU(USER1)
MCAUSER(USER1)
 END CHSTATUS DETAILS
CSQ9022I QMZ1 CSQMDRTC ' DISPLAY CHSTATUS' NORMAL
COMPLETION
```

As a test, add and delete queues and other functions that only USER1 should have authority to perform.

CONGRATULATIONS!!! You have completed the Implementing z/OS Queue Manager Security exercise

Summary

In this exercise you enabled external security checking for MQ objects queues, commands and connections. You performed various test to ensure security checking was working as expected and created and changed security resources as required.

Finally, you implemented SSL security between MQExplorer and the queue manager.