IMQ09 - IBM MQ V9 for z/OS Wildfire Workshop

L12 – Advance Message 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 the functions and capabilities of MQ Advanced Message Security on z/OS. The RACF resources required by Advanced Message Security (AMS) will be defined and AMS security policies will be configured during this exercise. MVS batch jobs and Windows scripts will then be used to test the RACF resource and AMS policies by putting messages on the AMS protected queues and then trying to get the messages using a variety of authorization identities.

General Exercise Information and Guidelines

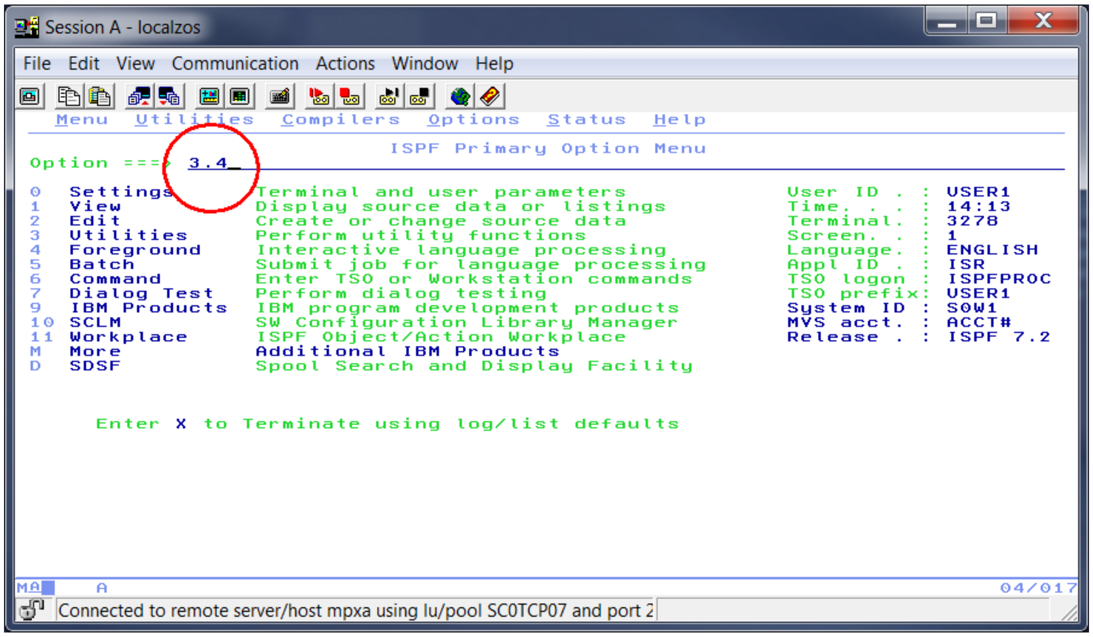
- ✓ This exercise requires using TSO user *USER1* on the local z/OS system.
- ✓ The TSO password for this exercise will be provided by the lab instructor.
- ✓ This exercise uses data set *USER1.AMS.JCL*.
- ✓ Case (upper or lower) is very important when invoking the *RACDCERT* and *keytool* commands in this exercise. When you see the use of upper characters in these commands enter the commands with the upper-case characters exactly as they appear in the text (in upper case). The lower-case characters in RACDCERT commands can be entered in either case.
- ✓ Anytime the results are not expected, try clearing the queues of all messages and restarting the test. Sometimes old messages on the queues will cause unexpected results.
- ✓ 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 RACF resources and AMS policies

In this step you will define RACF digital certificates and key rings required by AMS. Also the AMS policies that are required for message integrity and message privacy will also be configured.

____1. Log on to TSO.

____2. On the *ISPF Primary Option Menu* screen, enter ***3.4*** after the *Option* prompt and press the **Enter** key.



```
  Menu  Utilities  Compilers  Options  Status  Help
                          ISPF Primary Option Menu
Option ===> 3.4_

0  Settings      Terminal and user parameters          User ID . : USER1
1  View          Display source data or listings        Time. . . : 14:13
2  Edit          Create or change source data           Terminal. : 3278
3  Utilities     Perform utility functions              Screen. . : 1
4  Foreground    Interactive language processing        Language. : ENGLISH
5  Batch         Submit job for language processing     Appl ID . : ISR
6  Command       Enter TSO or Workstation commands      TSO logon : ISPFPROC
7  Dialog Test   Perform dialog testing                 TSO prefix: USER1
9  IBM Products  IBM program development products       System ID : SOW1
10 SCLM          SW Configuration Library Manager       MVS acct. : ACCT#
11 Workplace     ISPF Object/Action Workplace           Release . : ISPF 7.2
M  More          Additional IBM Products
D  SDSF          Spool Search and Display Facility

     Enter X to Terminate using log/list defaults

MA      A                                                          04/017
Connected to remote server/host mpxa using lu/pool SC0TCP07 and port 2
```

Tech-Tip: The copyright message can be dismissed by pressing the **Enter** key. In this and subsequent labs the instructions refer to the IBM 3270 Keyboard names or labels. So if the instructions say press the enter key this means that the IBM 3270 Keyboard Enter key should be pressed. For this 3270 emulator, the IBM 3270 Keyboard Enter key is mapped to the right Ctrl key. This was done so the 3270 new line key could be mapped to the key labeled *Enter*.

____3. On the *Data Set List Utility* screen enter ***USER1.AMS*** in the *Dsname level* field. Press the **Enter** key to continue.

____4. Edit data set *USER1.AMS.JCL* PDS by placing an '**E**' in the *Command* field beside the data set name. Press the **Enter** key to continue.

____5. Select member ***AMSQUES***. This is batch job that defines the queues that will be used in this exercise. Queue *AMSDEMO.PRIVACY.QUEUE* will be used for messages whose contents should be private and only readable by the designated recipient (Message Privacy) and queue *AMSDEMO.INTEGRITY.QUEUE* which will be used for messages that required protection from modification (Message Integrity).

Tech-Tip: The warning messages about CAPS OFF or STATS can be dismissed by entering command RESET on the command line.

Tech-Tip: The warning message about the UNDO command can be addressed by entering the *RECOVERY ON* command at the command prompt and pressing **Enter**. This will enable the saving of a recovery copy of a member in case a change needs to be reversed or in other recovery situations.

```
DEFINE QLOCAL(AMSDEMO.INTEGRITY.QUEUE) +
 DESCR('AMS DEMO INTEGRITY QUEUE') +
 QSGDISP(QMGR) +
 PUT(ENABLED) +
 GET(ENABLED) +
 DEFPSIST(YES) +
 MSGDLVSQ(FIFO) +
 NOTRIGGER +
 USAGE(NORMAL) +
 REPLACE +
 SHARE

 DEFINE QLOCAL(AMSDEMO.PRIVACY.QUEUE) +
 DESCR('AMS DEMO PRIVACY QUEUE') +
 QSGDISP(QMGR) +
 PUT(ENABLED) +
 GET(ENABLED) +
 DEFPSIST(YES) +
 MSGDLVSQ(FIFO) +
 NOTRIGGER +
 USAGE(NORMAL) +
 REPLACE +
 SHARE

DEFINE REPLACE +
 QALIAS('AMSDEMO.PRIVACY.ALIAS') +
 TARGTYPE(QUEUE) +
 PUT(ENABLED) +
 GET(ENABLED) +
 TARGET('AMSDEMO.PRIVACY.QUEUE')

DEFINE REPLACE +
 QALIAS('AMSDEMO.INTEGRITY.ALIAS') +
 TARGTYPE(QUEUE) +
 PUT(ENABLED) +
 GET(ENABLED) +
 TARGET('AMSDEMO.INTEGRITY.QUEUE')
```

Tech-Tip: The two alias queues defined by *AMSQUES* will used to display the AMS protected queues in MQ Explorer. They are not required for AMS per se but will prove useful later in this exercise.

____6. Submit *AMSQUES* for execution and verify that it completes with a condition code of zero.

____7. Next select member ***AMSCA***. This job uses the RACF *RACDCERT* command to create the certificate authority (CA) certificate that will be used to validate the personal certificates used by AMS. This certificate will be exported to a MVS data set for sending to Windows later in this exercise.

```
RACDCERT CERTAUTH +
 GENCERT SUBJECTSDN(CN('AMSCA') O('IBM') C('US')) +
 WITHLABEL('AMSCA') KEYUSAGE(CERTSIGN) NOTAFTER(DATE(2020/12/31))

RACDCERT CERTAUTH   EXPORT(LABEL('AMSCA')) DSN(AMSCA.CACERT.ARM)
```

____8. Submit *AMSCA* for execution and verify that it completes with a condition code of zero.

____9. Next select member ***USERCRTS***. This job uses the RACF *RACDCERT* command to

1. Create the personal certificates for each of the test users (USER1 and USER2)
2. Signs these personal certificates using the AMSCA CA certificate created earlier
3. Alters these certificates to make then trusted

```
RACDCERT ID(USER1) -
GENCERT SUBJECTSDN(CN('USER1') O('IBM') C('US')) -
WITHLABEL('USER1') SIGNWITH(CERTAUTH LABEL('AMSCA')) -
KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN) NOTAFTER(DATE(2020/12/31))

RACDCERT ID(USER2) -
GENCERT SUBJECTSDN(CN('USER2') O('IBM') C('US')) -
WITHLABEL('USER2') SIGNWITH(CERTAUTH LABEL('AMSCA')) -
KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN) NOTAFTER(DATE(2020/12/31))

RACDCERT ID(USER1) ALTER (LABEL('USER1')) TRUST
RACDCERT ID(USER2) ALTER (LABEL('USER2')) TRUST
```

____10. Submit *USERCRTS* for execution and verify that it completes with a condition code of zero.

____11. Next select member ***ADDRINGS***. This job uses the RACF *RACDCERT* command to:

a) Define key rings for the AMS started task user (START1) and each of our test users (USER1 and USER2).
b) Connect the test user's personal certificates to the AMS started task keyring.
c) To connect the AMSCA CA certificate and each user's personal certificate to their key ring.
d) To display the details of each keyring.

```
RACDCERT ID(START1) ADDRING(drq.ams.keyring)
RACDCERT ID(USER1) ADDRING(drq.ams.keyring)
RACDCERT ID(USER2) ADDRING(drq.ams.keyring)

RACDCERT ID(START1) CONNECT(RING(drq.ams.keyring) -
   LABEL('AMSCA') CERTAUTH USAGE(CERTAUTH)
RACDCERT ID(START1) CONNECT(RING(drq.ams.keyring) -
   ID(USER1) LABEL('USER1') USAGE(SITE)
RACDCERT ID(START1) CONNECT(RING(drq.ams.keyring) -
   ID(USER2) LABEL('USER2') USAGE(SITE)

RACDCERT ID(USER1) CONNECT(ID(USER1) RING(drq.ams.keyring) -
   LABEL('USER1') DEFAULT USAGE(PERSONAL)

RACDCERT ID(USER2) CONNECT(ID(USER2) RING(drq.ams.keyring) -
   LABEL('USER2') DEFAULT USAGE(PERSONAL)                      )

RACDCERT ID(START1) LISTRING(*)
RACDCERT ID(USER1)  LISTRING(*)
RACDCERT ID(USER2)  LISTRING(*)
```

____12. Submit *ADDRINGS* for execution and verify that it completes with a condition code of zero.

____13. Next select member ***CSQZPARM***. This job reassembles the *CSQZPARM* module and enables the checking of AMS policies in the queue manager (SPLCAP=YES). Change the *CSQ6SYSP* macro program SPLCAP from *NO* to ***YES***. Submit this job and verify that it completes with a condition code of zero.

____14. Use MVS command ***QMZ1 STOP QMGR*** to shut down the active queue manager. Once the queue manager is down, use MVS command ***QMZ1 START QMGR*** to restart the QMZ1 queue manager. Verify that task *QMZ1AMSM* is now active.

```
CSQY031I QMZ1 CSQ0AMST QUEUE MANAGER WAITING FOR AMS INITIALIZATION
CSQY028I QMZ1 CSQ0AMST AMS HAS STARTED
```

____15. Next select member ***ADDPOLCY***. This job uses the AMS ***setmqspl*** command to define the security policies for each of our test queues.

1. Queue *AMSDEMO.INTEGRITY.QUEUE* is configured so that it signed but not encrypted and only messages placed there by *USER1* can be retrieved
2. Queue *AMSDEMO.PRIVACY.QUEUE* is configured so its messages are signed and encrypted. Only **USER2** can place and/or retrieve messages from this queue.

```
setmqspl -m QMZ1
 -p AMSDEMO.INTEGRITY.QUEUE
 -s MD5
 -e NONE
 -a CN=USER1,O=IBM,C=US

 setmqspl -m QMZ1
 -p AMSDEMO.PRIVACY.QUEUE
 -s MD5
 -e AES256
 -a CN=USER2,O=IBM,C=US
 -r CN=USER2,O=IBM,C=US

 dspmqspl -m QMZ1 -p AMSDEMO.INTEGRITY.QUEUE
 dspmqspl -m QMZ1 -p AMSDEMO.PRIVACY.QUEUE
```

Tech-Tip: There are two members provided for reference purposes. Member *DELPOLCY* can be used to delete or remove the AMS policies from the configuration and member *DSPPOLC*Y and be used to display the current AMS policies.

____16. Submit *ADDPOLCY* for execution and verify that it completes with a condition code of zero.

____17. Use MVS command ***F QMZ1AMSM,REFRESH ALL*** to update the active AMS policies and RACF KEYRING configuration.

```
CSQ0635I QMZ1 CSQ0DCNS POLICY refresh complete
CSQ0653I QMZ1 CSQ0DCNS CRL checking disabled
CSQ0637I QMZ1 CSQ0DCNS KEYRING refresh complete
CSQ0641I QMZ1 CSQ0DCNS REFRESH command completed successfully
```

Part 2 – Testing AMS Security Policies on z/OS

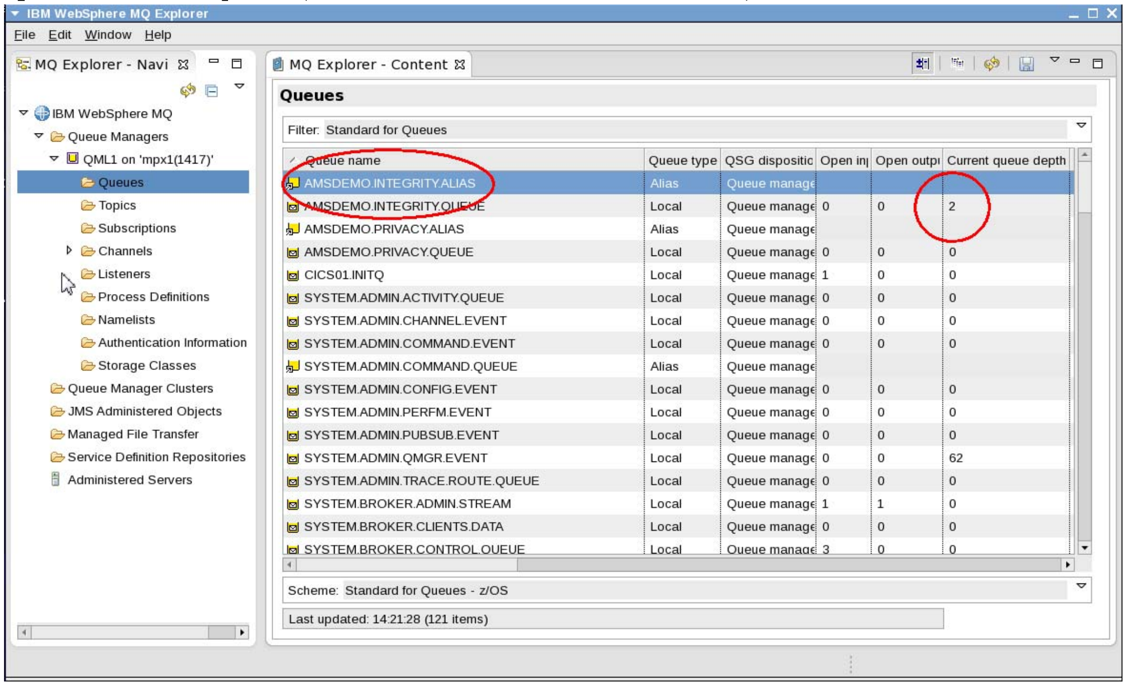
Batch jobs will be submitted in this part of the exercise. These batch jobs will run under different RACF authorities and will put messages or get messages using the protected queues. The results of these jobs will be reviewed to see what implications the various AMS security policies had on their execution.

____1. Select member ***PUTMSGI***. This job runs under security identity *USER1* and will put two messages on the *AMSDEMO.INTEGRITY.QUEUE* queue. Submit this job for execution and once complete there will be two STDOUT DD output streams in the job's output (Hint: use SDSF ***?*** line command to display the job's output files). You should see something like what is below in these output streams. These are the random messages that have been written to the message integrity queue.

```
Success:  Obtained a server reference to the Queue manager: QMZ1
Success:  Obtained a reference to the Queue: AMSDEMO.INTEGRITY.QUEUE
Success:  The MQ Message has been created: Wed Jan 13 19:15:33 UTC 2016-- Never
Success:  The message 'Wed Jan 13 19:15:33 UTC 2016-- Never miss a good chance to
Success:  Closed the queue
Success:  Disconnected from the Queue Manager

Success:  Obtained a server reference to the Queue manager: QMZ1
Success:  Obtained a reference to the Queue: AMSDEMO.INTEGRITY.QUEUE
Success:  The MQ Message has been created: Wed Jan 13 19:15:47 UTC 2016-- Cry 'H
Success:  The message 'Wed Jan 13 19:15:47 UTC 2016-- Cry 'Havoc'! and let slip
Success:  Closed the queue
Success:  Disconnected from the Queue Manager
```

____2. Open the *MQ Explorer* on the desktop and connect to the *QMZ1* queue manger and then expand the queues to display queue *AMSDEMO.INTEGRITY.ALIAS* (see below). Note that the current depth of the base queue (*AMSDEMO.INTEGRITY.QUEUE*) for this alias should be at least 2.



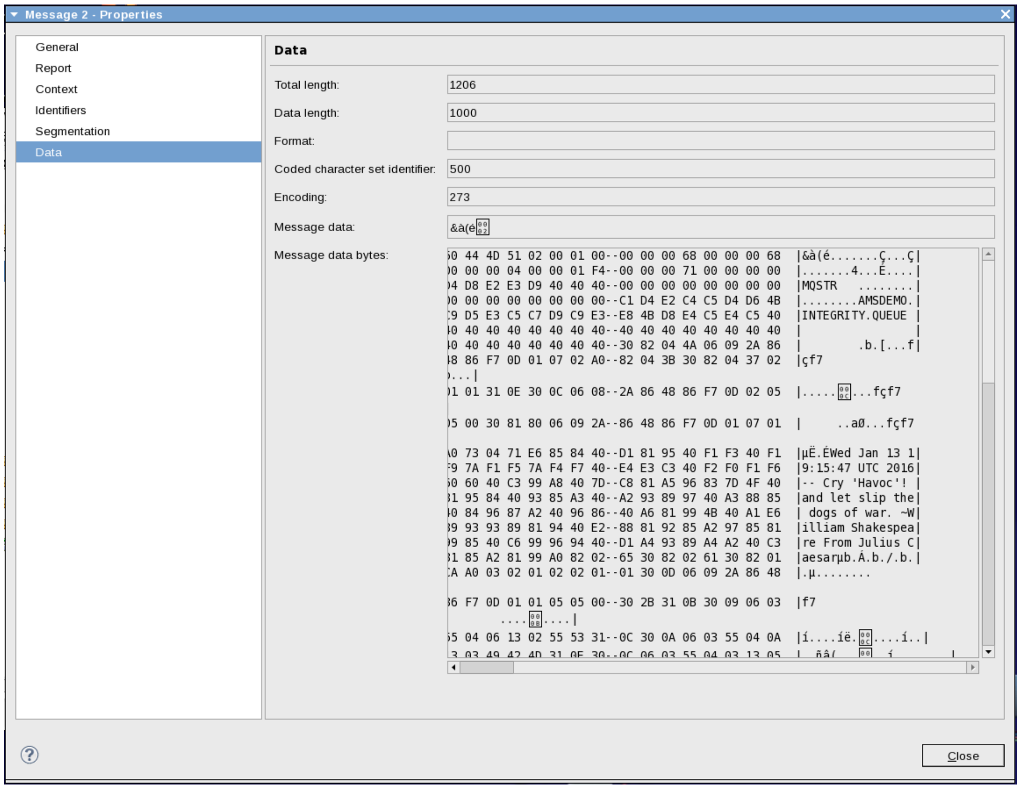
Filter: Standard for Queues

Queue name	Queue type	QSG dispositic	Open in	Open outp	Current queue depth
AMSDEMO.INTEGRITY.ALIAS	Alias	Queue manage			
AMSDEMO.INTEGRITY.QUEUE	Local	Queue manage	0	0	2
AMSDEMO.PRIVACY.ALIAS	Alias	Queue manage			
AMSDEMO.PRIVACY.QUEUE	Local	Queue manage	0	0	0
CICS01.INITQ	Local	Queue manage	1	0	0
SYSTEM.ADMIN.ACTIVITY.QUEUE	Local	Queue manage	0	0	0
SYSTEM.ADMIN.CHANNEL.EVENT	Local	Queue manage	0	0	0
SYSTEM.ADMIN.COMMAND.EVENT	Local	Queue manage	0	0	0
SYSTEM.ADMIN.COMMAND.QUEUE	Alias	Queue manage			
SYSTEM.ADMIN.CONFIG.EVENT	Local	Queue manage	0	0	0
SYSTEM.ADMIN.PERFM.EVENT	Local	Queue manage	0	0	0
SYSTEM.ADMIN.PUBSUB.EVENT	Local	Queue manage	0	0	0
SYSTEM.ADMIN.QMGR.EVENT	Local	Queue manage	0	0	62
SYSTEM.ADMIN.TRACE.ROUTE.QUEUE	Local	Queue manage	0	0	0
SYSTEM.BROKER.ADMIN.STREAM	Local	Queue manage	1	1	0
SYSTEM.BROKER.CLIENTS.DATA	Local	Queue manage	0	0	0
SYSTEM.BROKER.CONTROL.QUEUE	Local	Queue manage	3	0	0

Scheme: Standard for Queues - z/OS

Last updated: 14:21:28 (121 items)

____3. *Browse the Messages* on the alias queue and then select one of the messages to display its properties. Select the *Data* contents on the P*roperties* screen and you should see something like the below:



Data

Total length: 1206
Data length: 1000
Format:
Coded character set identifier: 500
Encoding: 273
Message data: &à(é□
Message data bytes:

```
0 44 4D 51 02 00 01 00--00 00 00 68 00 00 00 68  |&à(é.......Ç...Ç|
0 00 00 04 00 00 01 F4--00 00 00 71 00 00 00 00  |.......4...É....|
4 D8 E2 E3 D9 40 40 40--00 00 00 00 00 00 00 00  |MQSTR   ........|
0 00 00 00 00 00 00 00--C1 D4 E2 C4 C5 D4 D6 4B  |........AMSDEMO.|
9 D5 E3 C5 C7 D9 C9 E3--E8 4B D8 E4 C5 E4 C5 40  |INTEGRITY.QUEUE |
0 40 40 40 40 40 40 40--40 40 40 40 40 40 40 40  |                |
0 40 40 40 40 40 40 40--30 82 04 4A 06 09 2A 86  |        .b.[...f|
8 86 F7 0D 01 07 02 A0--82 04 3B 30 82 04 37 02  |çf7
...|
1 01 31 0E 30 0C 06 08--2A 86 48 86 F7 0D 02 05  |.....□...fçf7

5 00 30 81 80 06 09 2A--86 48 86 F7 0D 01 07 01  |     ..aØ...fçf7

0 73 04 71 E6 85 84 40--D1 81 95 40 F1 F3 40 F1  |µË.ÉWed Jan 13 1|
9 7A F1 F5 7A F4 F7 40--E4 E3 C3 40 F2 F0 F1 F6  |9:15:47 UTC 2016|
0 60 40 C3 99 A8 40 7D--C8 81 A5 96 83 7D 4F 40  |-- Cry 'Havoc'! |
1 95 84 40 93 85 A3 40--A2 93 89 97 40 A3 88 85  |and let slip the|
0 84 96 87 A2 40 96 86--40 A6 81 99 4B 40 A1 E6  | dogs of war. ~W|
9 93 93 89 81 94 40 E2--88 81 92 85 A2 97 85 81  |illiam Shakespea|
9 85 40 C6 99 96 94 40--D1 A4 93 89 A4 A2 40 C3  |re From Julius C|
1 85 A2 81 99 A0 82 02--65 30 82 02 61 30 82 01  |aesarµb.Á.b./.b.|
A A0 03 02 01 02 02 01--01 30 0D 06 09 2A 86 48  |.µ........

6 F7 0D 01 01 05 05 00--30 2B 31 0B 30 09 06 03  |f7
        ....□....|
5 04 06 13 02 55 53 31--0C 30 0A 06 03 55 04 0A  |í....íë.□....í..|
3 03 49 42 4D 31 0E 30--0C 06 03 55 04 03 13 05
```

____4. The message is still readable, but you should see that the message has been signed (the unreadable contents) to protect its integrity. If the contents are changed during the transmission AMS will recognized this by verifying the signature and flag this message as having been corrupted.

Tech-Tip: We are able to view this message since we are using an alias for the base queue. This bypasses the AMS interceptors and allows us to look at the message while it is at rest on the base queue.

____5. Go back to TSO and select member ***GETMSGI*** in *USER1.AMS.JCL*. This member contains two jobs. Each job performs retrieval *(GET)* of a message from queue *AMSDEMO.INTEGRITY.QUEUE* but each job runs under different user identities and therefore using different key rings. Submit these jobs and monitor their output. Each job should successfully retrieve one of the messages placed on the queue by the previous execution of *PUTMSGI.*

____6. Now edit member *PUTMSGI* and change all occurrences of ***USER1*** to ***USER2***. Submit this job for execution and monitor it until it completes. As before, you should still find two *STDOUT* output streams showing the two messages placed on the queue. ***Do not save the changed JCL.***

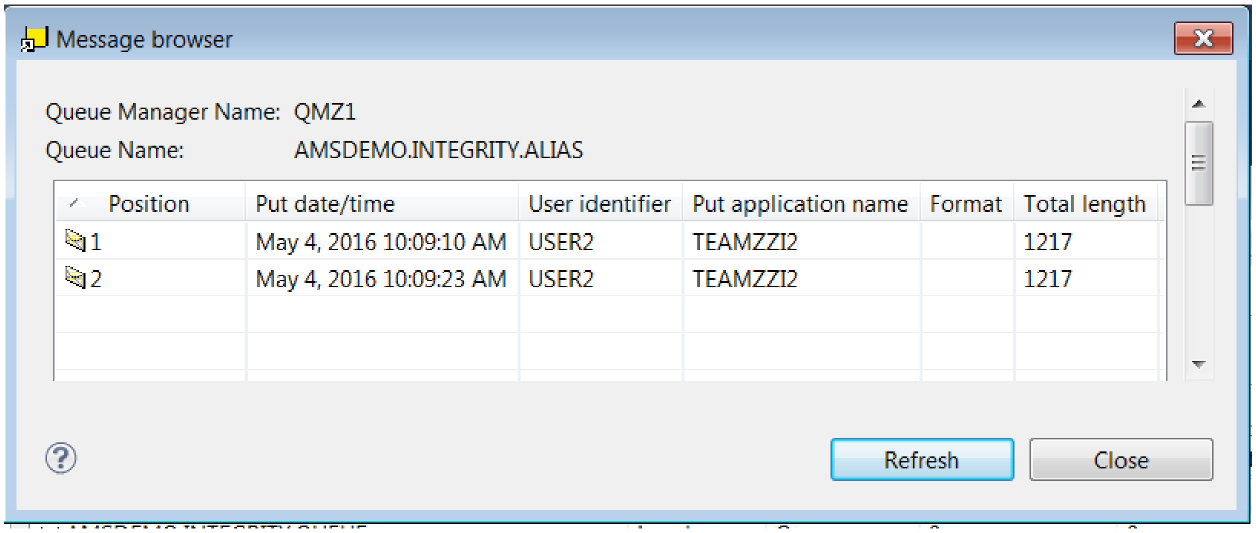
____7. Now submit *GETMSGI* again. As before two jobs will be submitted. Monitor both jobs and review their output and both should fail with message

A WebSphere MQ error occurred: Completion code 2 Reason code 2063

This is because the policy for this queue only allows the retrieval messages placed there by *USER1*.

```
setmqspl -m QMZ1
-p AMSDEMO.INTEGRITY.QUEUE
-s MD5
-e NONE
-a CN=USER1,O=IBM,C=US
```

These messages were placed there by *USER2* and therefore their retrieval is being prevented by AMS policies. Optionally, repeat the last two steps but before submitting *GETMSGI* use MQ Explorer to confirm that the *User identifier* associated with these messages is *USER2*.



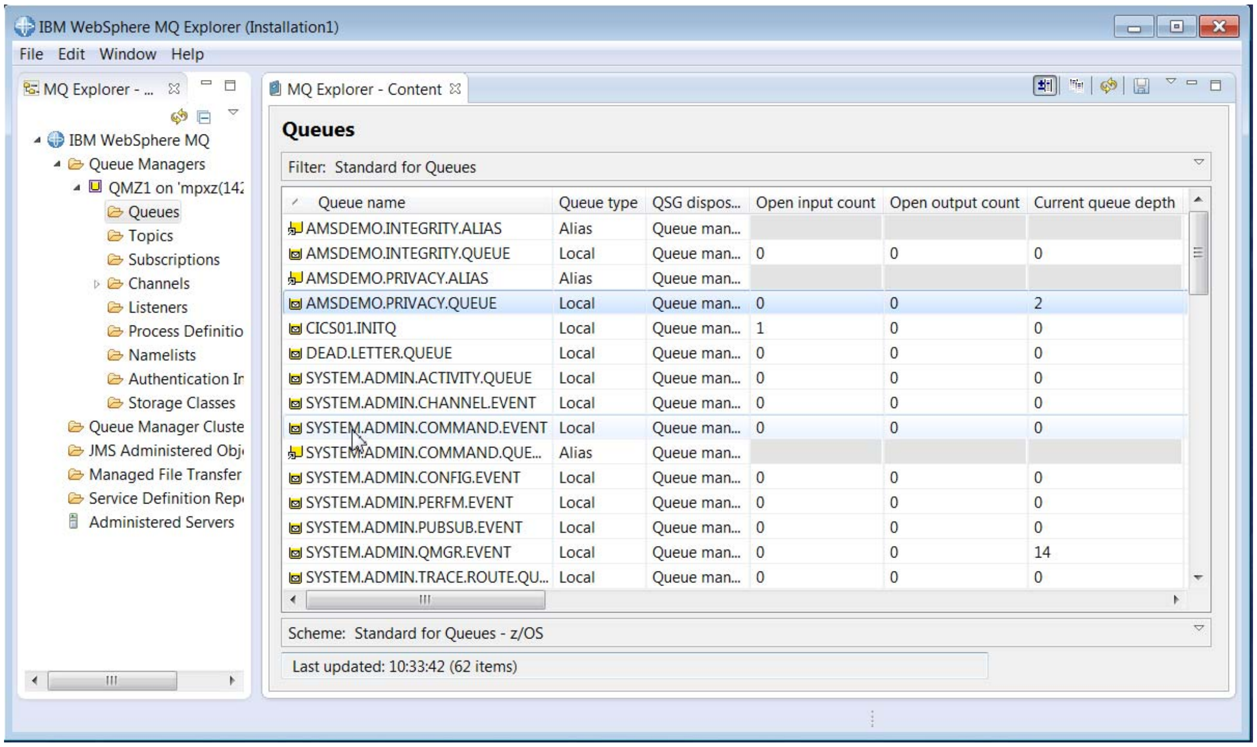
Queue Manager Name: QMZ1
Queue Name: AMSDEMO.INTEGRITY.ALIAS

Position	Put date/time	User identifier	Put application name	Format	Total length
1	May 4, 2016 10:09:10 AM	USER2	TEAMZZI2		1217
2	May 4, 2016 10:09:23 AM	USER2	TEAMZZI2		1217

Now let's move to explore message privacy.

____8. Select member ***PUTMSGP***. This job will put two messages on the *AMSDEMO.PRIVACY.QUEUE* queue. Submit this job for execution and as before there will be two STDOUT DD output streams when the job completes. These show the random messages written to queue *AMSDEMO.PRIVACY.QUEUE.*

____9. Use MQ Explorer to display the *AMSDEMO.PRIVACY.ALIAS* queue (see below). Also note that the current depth of the base queue (*AMSDEMO.PRIVACY.QUEUE*) for this alias should be at least 2.



Queues

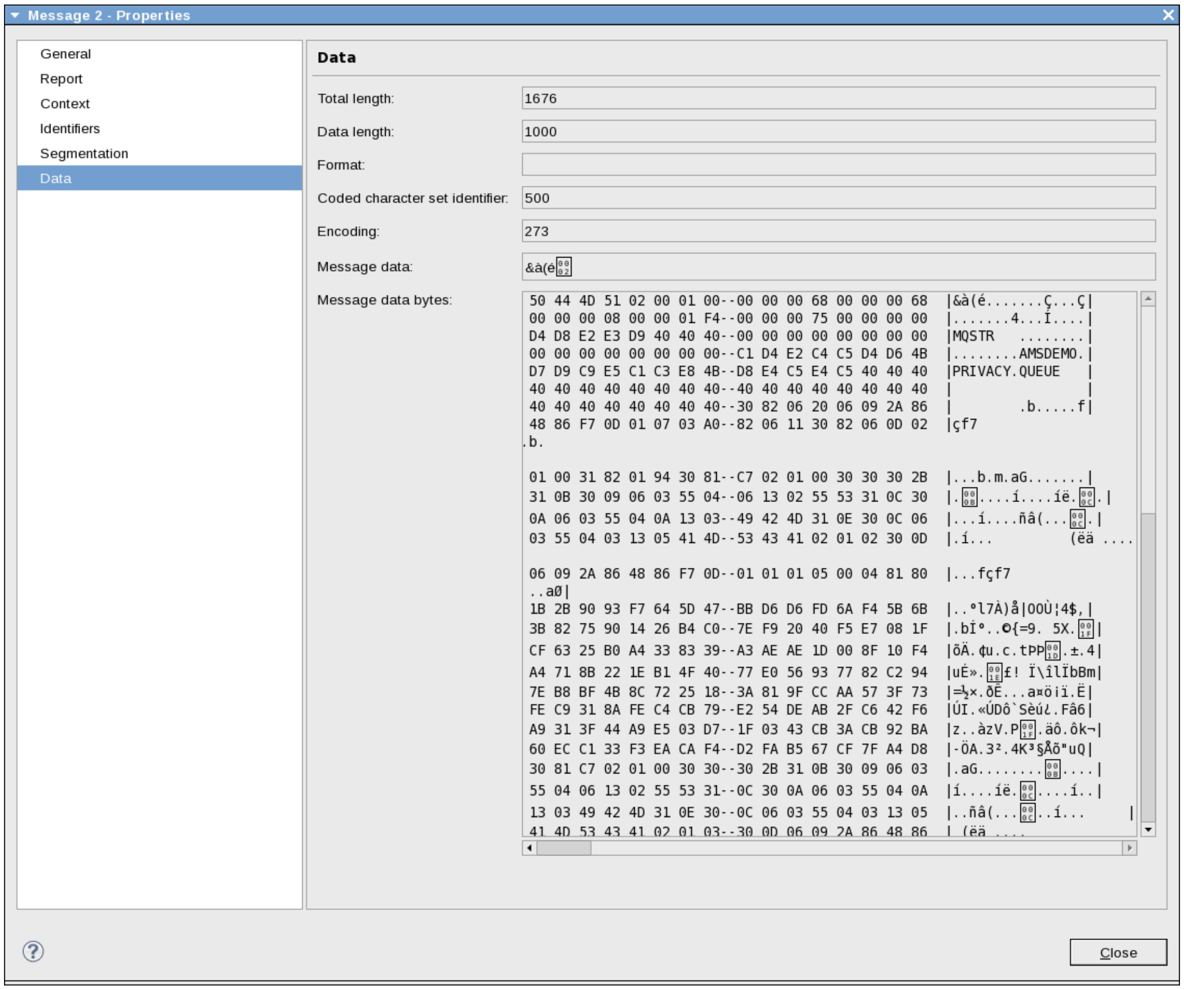
Filter: Standard for Queues

Queue name	Queue type	QSG dispos...	Open input count	Open output count	Current queue depth
AMSDEMO.INTEGRITY.ALIAS	Alias	Queue man...			
AMSDEMO.INTEGRITY.QUEUE	Local	Queue man...	0	0	0
AMSDEMO.PRIVACY.ALIAS	Alias	Queue man...			
AMSDEMO.PRIVACY.QUEUE	Local	Queue man...	0	0	2
CICS01.INITQ	Local	Queue man...	1	0	0
DEAD.LETTER.QUEUE	Local	Queue man...	0	0	0
SYSTEM.ADMIN.ACTIVITY.QUEUE	Local	Queue man...	0	0	0
SYSTEM.ADMIN.CHANNEL.EVENT	Local	Queue man...	0	0	0
SYSTEM.ADMIN.COMMAND.EVENT	Local	Queue man...	0	0	0
SYSTEM.ADMIN.COMMAND.QUE...	Alias	Queue man...			
SYSTEM.ADMIN.CONFIG.EVENT	Local	Queue man...	0	0	0
SYSTEM.ADMIN.PERFM.EVENT	Local	Queue man...	0	0	0
SYSTEM.ADMIN.PUBSUB.EVENT	Local	Queue man...	0	0	0
SYSTEM.ADMIN.QMGR.EVENT	Local	Queue man...	0	0	14
SYSTEM.ADMIN.TRACE.ROUTE.QU...	Local	Queue man...	0	0	0

Scheme: Standard for Queues - z/OS

Last updated: 10:33:42 (62 items)

____10. *Browse the Messages* on the alias queue and then select one of the messages to display its properties. Select the *Data* contents on the P*roperties* screen and you should see something like the below:



Data

Total length: 1676
Data length: 1000
Format:
Coded character set identifier: 500
Encoding: 273
Message data: &à(é□
Message data bytes:

```
 50 44 4D 51 02 00 01 00--00 00 00 68 00 00 00 68  |&à(é.......Ç...Ç|
 00 00 00 08 00 00 01 F4--00 00 00 75 00 00 00 00  |.......4...Í....|
 D4 D8 E2 E3 D9 40 40 40--00 00 00 00 00 00 00 00  |MQSTR   ........|
 00 00 00 00 00 00 00 00--C1 D4 E2 C4 C5 D4 D6 4B  |........AMSDEMO.|
 D7 D9 C9 E5 C1 C3 E8 4B--D8 E4 C5 E4 C5 40 40 40  |PRIVACY.QUEUE   |
 40 40 40 40 40 40 40 40--40 40 40 40 40 40 40 40  |                |
 40 40 40 40 40 40 40 40--30 82 06 20 06 09 2A 86  |        .b.....f|
 48 86 F7 0D 01 07 03 A0--82 06 11 30 82 06 0D 02  |çf7
.b.

 01 00 31 82 01 94 30 81--C7 02 01 00 30 30 30 2B  |...b.m.aG.......|
 31 0B 30 09 06 03 55 04--06 13 02 55 53 31 0C 30  |.□....í....íë.□.|
 0A 06 03 55 04 0A 13 03--49 42 4D 31 0E 30 0C 06  |...í....ñâ(...□.|
 03 55 04 03 13 05 41 4D--53 43 41 02 01 02 30 0D  |.í...          (ëä ....

 06 09 2A 86 48 86 F7 0D--01 01 01 05 00 04 81 80  |...fçf7
 ..aØ|
 1B 2B 90 93 F7 64 5D 47--BB D6 D6 FD 6A F4 5B 6B  |..°l7À)å|OOÙ¦4$,|
 3B 82 75 90 14 26 B4 C0--7E F9 20 40 F5 E7 08 1F  |.bÍ°..©{=9. 5X.□|
 CF 63 25 B0 A4 33 83 39--A3 AE AE 1D 00 8F 10 F4  |õÄ.¢u.c.tÞÞ□.±.4|
 A4 71 8B 22 1E B1 4F 40--77 E0 56 93 77 82 C2 94  |uÉ».□£! Ï\îlÏbBm|
 7E B8 BF 4B 8C 72 25 18--3A 81 9F CC AA 57 3F 73  |=½×.ðË...a¤öiï.Ë|
 FE C9 31 8A FE C4 CB 79--E2 54 DE AB 2F C6 42 F6  |ÚI.«ÚDô`Sèú¿.Fâ6|
 A9 31 3F 44 A9 E5 03 D7--1F 03 43 CB 3A CB 92 BA  |z..àzV.P□.äô.ôk¬|
 60 EC C1 33 F3 EA CA F4--D2 FA B5 67 CF 7F A4 D8  |-ÖA.3².4K³§Åõ"uQ|
 30 81 C7 02 01 00 30 30--30 2B 31 0B 30 09 06 03  |.aG........□....|
 55 04 06 13 02 55 53 31--0C 30 0A 06 03 55 04 0A  |í....íë.□....í..|
 13 03 49 42 4D 31 0E 30--0C 06 03 55 04 03 13 05  |..ñâ(...□..í...     |
 41 4D 53 43 41 02 01 03--30 0D 06 09 2A 86 48 86  | (ëä ....
```

____11. The content of the message is unreadable (not even the message is legible). This confirms that the message has been encrypted to protect its integrity and its contents.

Tech-Tip: Again, we are able to view this message since we are using an alias for the base queue. This bypasses the AMS interceptors and allows us to look at the message while it is on the base queue.

____12. Go back to TSO and select member ***GETMSGP*** in *USER1.AMS.JCL*. This member contains two jobs. Each job performs does retrieves a message from the A*MSDEMO.PRIVACY.QUEUE* queue but each uses a different user identity. Submit these jobs and monitor their output. Each job should attempt to retrieve a message that was placed on the queue by the previous execution of *PUTMSGP*.

____13. Browse the output of the job that ran under USER1's authority and you should see this message.

A WebSphere MQ error occurred: Completion code 2 Reason code 2063

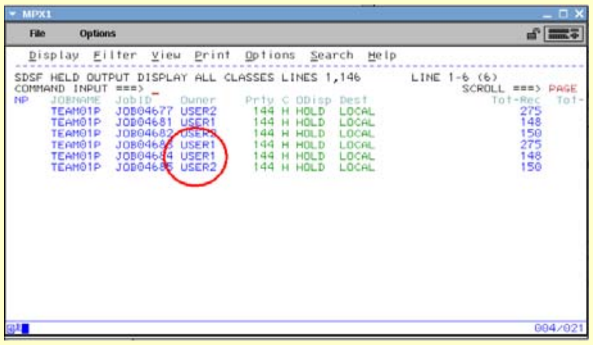
This is because the policy for this queue only allows the retrieval messages placed there by USER2.

```
setmqspl -m QMZ1
 -p AMSDEMO.PRIVACY.QUEUE
 -s MD5
 -e AES256
 -a CN=USER2,O=IBM,C=US
 -r CN=USER2,O=IBM,C=US
```

____14. Browse the output of the job that ran under *USER2*'s authority and you should see that the message was successfully retrieved.

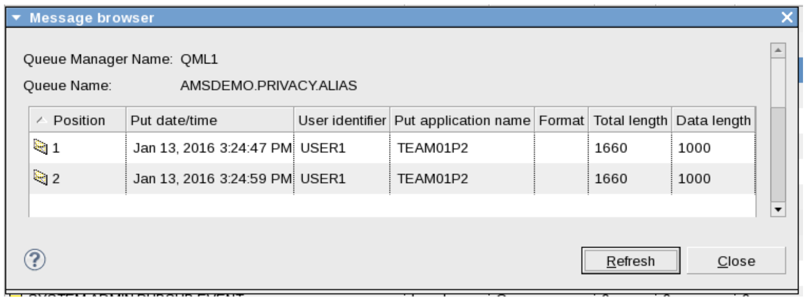
```
Success:  Obtained a reference to the Queue manager: QMZ1
Success:  Obtained a reference to the Queue: AMSDEMO.PRIVACY.QUEUE
Success:  Obtained a reference to the MQ Message
Success:  The message was successfully retrieved from the queue.
=== Wed Jan 13 19:58:20 UTC 2016-- Now is the time for all good men to come to t
Success:  Closed the queue
Success:  Disconnected from the Queue Manager
```

Tech-Tip: Use the *Owner column* in the SDSF output queue to confirm which jobs were submitted by USER2.



```
SDSF HELD OUTPUT DISPLAY ALL CLASSES LINES 1,146        LINE 1-6 (6)
COMMAND INPUT ===>                                        SCROLL ===> PAGE
NP   JOBNAME  JobID    Owner    Prty C ODisp Dest                  Tot-Rec  Tot-
     TEAM01P  JOB04677 USER2     144 H HOLD  LOCAL                     275
     TEAM01P  JOB04681 USER1     144 H HOLD  LOCAL                     148
     TEAM01P  JOB04682 USER2     144 H HOLD  LOCAL                     150
     TEAM01P  JOB04683 USER1     144 H HOLD  LOCAL                     275
     TEAM01P  JOB04684 USER1     144 H HOLD  LOCAL                     148
     TEAM01P  JOB04685 USER2     144 H HOLD  LOCAL                     150
```

____15. Now edit member *PUTMSGP* and change all occurrences of ***USER2*** to ***USER1***. Submit this job for execution and monitor it until it completes. You should still find two STDOUT output streams showing the two messages placed onto the queue. . Use MQ Explorer that there are now two messages on the *AMSDEMO.PRIVACY.ALIAS* queue with *USER1* as the user identifier. ***Do not save the changed JCL***



Queue Manager Name: QML1
Queue Name: AMSDEMO.PRIVACY.ALIAS

Position	Put date/time	User identifier	Put application name	Format	Total length	Data length
1	Jan 13, 2016 3:24:47 PM	USER1	TEAM01P2		1660	1000
2	Jan 13, 2016 3:24:59 PM	USER1	TEAM01P2		1660	1000

____16. Now submit *GETMSGP* again. As before two jobs will be submitted. Monitor their output and both should fail with message

A WebSphere MQ error occurred: Completion code 2 Reason code 2063

This is because the policy for this queue only allows the retrieval messages placed there by *USER2*.

Since these messages were placed there by *USER1* their retrieval is prevented by AMS policies.

Part 3 – Configuring a Client for Message Integrity

In this part of the lab exercise the AMS signer certificates created in RACF will be installed in a Windows key store for accessing messages protected by AMS message integrity. A self-signed personal certificate will be created and sent to the certificate authority (e.g. RACF). The client's personal certificate will be signed and returned to Windows for importing into the key store.

Note: Keystore files and Key Databases both refer to the repository in which digital certificates are maintained in non-RACF managed environments.

___1. Open a command prompt terminal session (icon Command Prompt) and use a cd command to change to directory \z*home*, e.g. ***cd \z\home***. Then use ftp to move the MVS data set containing the AMSCA exported certificates from z/OS to Windows.

```
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, 13:33:10 on 2016-
01-14.
220 Connection will close if idle for more than 5 minutes.
Name (mpx1:zosuser): USER1
331 Send password please.
Password: xxxxxxxx
230 USER1 is logged on.  Working directory is "USER1.".
Remote system type is MVS.
ftp> mget *.cacert.arm
mget AMSCA.CACERT.ARM? y
227 Entering Passive Mode (192,27,216,44,4,12)
125 Sending data set USER1.AMSCA.CACERT.ARM
250 Transfer completed successfully.
878 bytes received in 0.0201 secs (43.74 Kbytes/sec)
ftp> quit
221 Quit command received. Goodbye.
```

Tech-Tip: The client applications used in this exercise assume that the key store file accessed is located in directory \z\home. 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\home\key.jks

Tech-Tip: The trust store/key store file is located by the local AMS interceptors because a *keystore.conf* file has been placed in directory *c:\z\ams\.mqs* and idenitified by environment variable xxxxxx. The *keystore.conf* file provides the name of the keystore file and its directory location, as well as its password.

___2. Import the RACF generated Certificate Authority certificate (ASMCA) using the **keytool import** subcommand to import the signer certificates into local Key Store file **key.jks** and indicate that this is a trusted signer certificates (-trustcacerts).

```
keytool -import  -trustcacerts -alias "AMSCA" -file AMSCA.CACERT.ARM
        -keystore integrity.jks
Enter keystore password:  changeit
Re-enter new password: changeit
Owner: CN=AMSCA, O=IBM, C=US
Issuer: CN=AMSCA, O=IBM, C=US
Serial number: 0
Valid from: 1/12/16 11:00 PM until: 12/31/20 10:59 PM
Certificate fingerprints:
        MD5:  56:47:33:C9:82:F3:0F:5F:60:F2:E9:F6:C7:EF:23:68
        SHA1: DF:FE:5A:03:D1:4B:5A:63:5B:50:30:BD:EC:65:84:95:DC:2E:87:32
        SHA256:
C0:0F:C3:7F:E4:55:BdC:67:DC:11:B8:1E:AC:5F:AE:50:F0:9B:0D:F8:4F:09:EF:69:45:C5:21:F
E:74:00:83:27
        Signature algorithm name: SHA1withRSA
        Version: 3


…..

Trust this certificate? [no]:  yes
Certificate was added to keystore:
```

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

Tech-Tip: Password ***changeit*** is used by convention. It is commonly used a default password for trust/key stores shipped when a new product is installed.

___3. Use the **keytool genkey** subcommand 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.

```
keytool -genkey -alias "USER3" -dname "CN=USER3, O=IBM, C=US"
      -keystore integrity.jks -keyalg RSA
Enter keystore password:  changeit
Enter key password for <USER3>: (press enter)
      (RETURN if same as keystore password):
```

___4. Use the **keytool certreq** subcommand to extract the self-signed certificate to a certificate request file that can be uploaded to z/OS.

```
keytool -certreq -alias "USER3" -file user3.certreq.arm
      -keystore integrity.jks
Enter keystore password: changeit
```

___5. Use **ftp** to move the certificate request file to z/OS.

```
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, 13:45:00 on 2016-01-14.
220 Connection will close if idle for more than 5 minutes.
Name (mpx1:zosuser): USER1
331 Send password please.
Password: xxxxxxxxx
230 USER1 is logged on.  Working directory is "USER1.".
Remote system type is MVS.
ftp> quote site lrecl=256 recfm=vb
200 SITE command was accepted
ftp> mput user3.certreq.arm
mput user3.certreq.arm? y
227 Entering Passive Mode (192,27,216,44,4,13)
125 Storing data set USER1.USER3.CERTREQ.ARM
250 Transfer completed successfully.
1019 bytes sent in 4.2e-05 secs (24261.90 Kbytes/sec)
ftp> quit
221 Quit command received. Goodby
```

___6. Browse **USER1.USER3.CERTREQ.ARM** and you should see something like below:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
 MIICoDCCAYgCAQAwKzELMAkGA1UEBhMCVVMxDDAKBgNVBAoTA0lCTTEOMAwGA1UEAxMFVVNFUjMw
 ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCppsrjNg93WPxI+hsJ4kLyV8kLycmbTKNH
 D10XlXH/pq3k/xZQbzXkv9Dcj/VhwlINQTFihWO1pfwFf+GfNd9taBfIbFdb4aYW9HW9jokHUIbL
 iSoxg3Jh9r2Ud60zs8PAf2qZ/jZx/HJY615v1HIWokS8teOpAEgYzCb7GjR8TOysVOd5nae1rnLC
 7cNUbweE2iCJHg8bYZ7lgwWJWHakGAAuG8pGFBlwt9v8nIp1SAFQx+i+A0wD3jLzQen0LhuvP7fC
O8gphMMQuGBkQ58Y3AeesggO7wgQwb9bJ4hIGsuRtlXQeydJqDFlF8caBMaFGROJ10vHyHV1+ltP
 HXR1AgMBAAGgMDAuBgkqhkiG9w0BCQ4xITAfMB0GA1UdDgQWBBSIp26SUGR/Iu3Us9rvgcU0CHQk
 5DANBgkqhkiG9w0BAQsFAAOCAQEABj2J9J7ZKhpu3tmVTa4RBXaawWYNulojuNUVR+HhoH6x0dC0
 QBP4F4bcrIMpCF6twFJUjs63Mr0hV/qr7PFW0bV/hXVs+XQO+qmdVfW9pkayQjZyRRcNKiFLvay
 15wi1dleQtKnr0UkncBMMUYt4oIu2sMtGmdjxUpqLMCHs0sXP69cpRso1UdIBznM/SOSrU9rKjkf
 unl5xyp6x4mXlyNrDIhzw7in7ERdrnJH3m0PRg6yb6SF2shxvPAr06awbynyt6QTYrMB2CDeasb/
 wqOclRPAWrrPEljUZLbxYuOAEF9eujflaBgpdCLCgzr7jXkpESX2VZOjT1HVdsUC8g==
 -----END NEW CERTIFICATE REQUEST-----
```

___7. After uploading the certificate request to z/OS and use the **RACDCERT GENCERT** command to sign the certificate request with the RACF client signer certificate (AMSCA) and associated it with *USER3*.

```
racdcert id(user3) gencert(user3.certreq.arm) withlabel('USER3')
signwith(certauth label('AMSCA')) notafter(date(2020/12/31))
```

___8. Export the signed client certificate request file to a sequential z/OS dataset using the **RACDCERT EXPORT** command.

```
racdcert id(user3) export(label('USER3')) dsn(user3.cert.arm)
```

___9. Connect USER3's certificate to the RACF keyring of the AMS started tasks using the RACDCERT CONNECT command.

racdcert id(start1) connect(ring(drq.ams.keyring) id(user3) label('USER3') usage(site)

___10. Refresh the AMS task's key ring information with MVS command

F QMZ1AMSM,REFRESH KEYRING

___11. Browse **USER1.USER3.CERT.ARM** and you should see something like below:

```
-----BEGIN CERTIFICATE-----
MIIC1TCCAj6gAwIBAgIBAzANBgkqhkiG9w0BAQUFADArMQswCQYDVQQGEwJVUzEM
MAoGA1UEChMDSUJNMQ4wDAYDVQQDEwVBTVNDQTAeFw0xNjAxMTQwNDAwMDBaFw0y
MTAxMDEwMzU5NTlaMCsxCzAJBgNVBAYTAlVTMQwwCgYDVQQKEwNJQk0xDjAMBgNV
BAMTBVVTRVIzMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAqabK4zYP
d1j8SPobCeJC8lfJC8nJm0yjRw9dF5Vx/6at5P8WUG815L/Q3I/1YcJSDUExYoVj
taX8BX/hnzXfbWgXyGxXW+GmFvR1vY6JB1CGy4kqMYNyYfa9lHetM7PDwH9qmf42
cfxyWOteb9RyFqJEvLXjqQBIGMwm+xo0fEzsrFTneZ2nta5ywu3DVG8HhNogiR4P
G2Ge5YMFiVh2pBgALhvKRhQZcLfb/JyKdUgBUMfovgNMA94y80Hp9C4brz+3wjvI
KYTDELhgZEOfGNwHnrIIDu8IEMG/WyeISBrLkbZV0HsnSagxZRfHGgTGhRkTiddL
x8h1dfpbTx10dQIDAQABo4GEMIGBMD8GCWCGSAGG+EIBDQQyFjBHZW5lcmF0ZWQg
YnkgdGhlIFNlY3VyaXR5IFNlcnZlciBmb3Igei9PUyAoUkFDRikwHQYDVR0OBBYE
FIinbpJQZH8i7dSz2u+BxTQIdCTkMB8GA1UdIwQYMBaAFIRXmy5eUHfdUNfXKVNy
40bI43CnMA0GCSqGSIb3DQEBBQUAA4GBAKjrb44jVYXINhe9H8OF6aX4QmiwjlY3
4AX2Q9hoO3n4VoAFB1uWv8DQVDKYEEYZ8WJQhmXyQJM+caVX9Q1Dz5sFNnAWb1lG
mMRGv2b0frEM17eXds12jWJe/zMp7Wc0ZnoPauvMSnEZ2BG3foEJF4tsybye+Myc
Enktpw8IYSc5
-----END CERTIFICATE-----
```

___12. Use **ftp** to move the signed certificate in dataset **USER1.USER3.CERT.ARM** to Windows (see Step 1 above). Note that the mget command should be ***mget *.cert.arm*** rather than ***mget *.cacert.arm***.

___13. Use the **keytool import** subcommand to import the signed certificate into the Windows JSSE keystore.

```
 keytool -v -import -alias "USER3" -file USER3.CERT.ARM -keystore
integrity.jks
Enter keystore password:  changeit
Certificate reply was installed in keystore
[Storing integrity.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 integrity.jks and deleting the *USER3* certificate with command ***racdcert id(user3) delete(label('USER3')***

___14. Use the **keytool list** subcommand to display the contents of the key store.

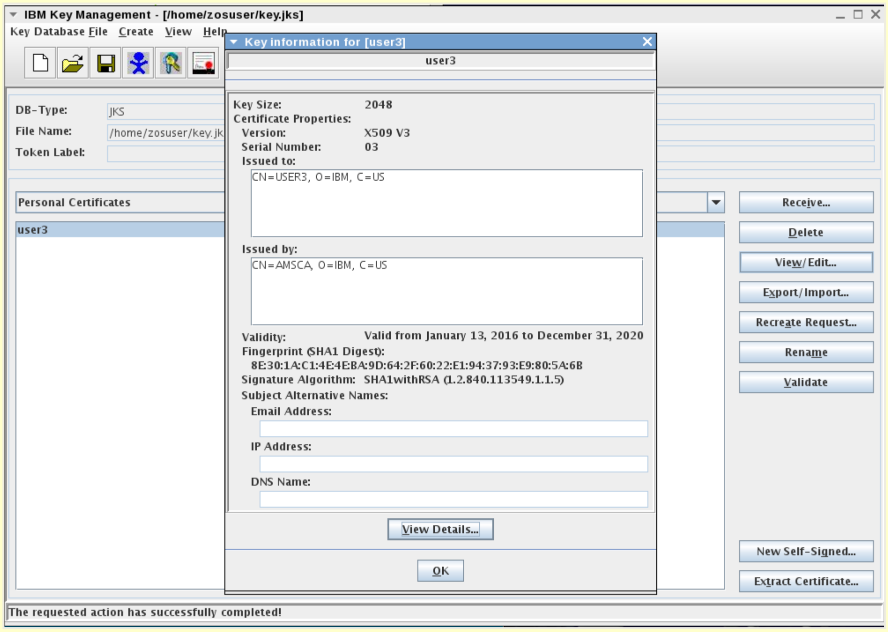
```
keytool -list -keystore integrity.jks
Enter keystore password: changeit

Keystore type: jks
Keystore provider: IBMJCE

Your keystore contains 2 entries

amsca, Jan 14, 2016, trustedCertEntry,
Certificate fingerprint (SHA1):
DF:FE:5A:03:D1:4B:5A:63:5B:50:30:BD:EC:65:84:95:DC:2E:87:32
user3, Jan 14, 2016, keyEntry,
Certificate fingerprint (SHA1):
8E:30:1A:C1:4E:4E:BA:9D:64:2F:60:22:E1:94:37:93:E9:80:5A:6B
```

Tech-Tip: IBM MQ command ***strmqikm*** can be used to start the MQ provided version of the IBM Java iKeyMan GUI tool (see below). This GUI tool can be used as alternative to using the ***keytool*** command.



IBM Key Management - [/home/zosuser/key.jks]

DB-Type: JKS
File Name: /home/zosuser/key.jk
Token Label:

Personal Certificates
user3

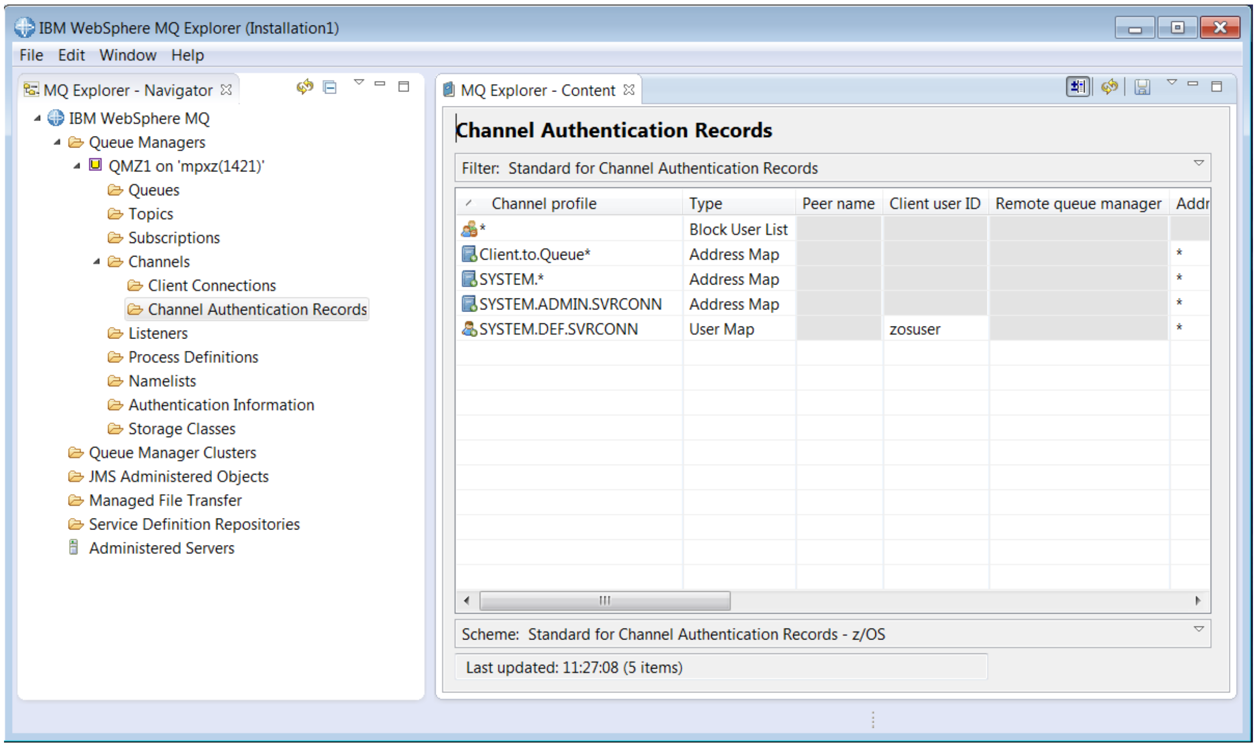
Key information for [user3]

user3

Key Size: 2048
Certificate Properties:
Version: X509 V3
Serial Number: 03
Issued to:
CN=USER3, O=IBM, C=US
Issued by:
CN=AMSCA, O=IBM, C=US
Validity: Valid from January 13, 2016 to December 31, 2020
Fingerprint (SHA1 Digest):
8E:30:1A:C1:4E:4E:BA:9D:64:2F:60:22:E1:94:37:93:E9:80:5A:6B
Signature Algorithm: SHA1withRSA (1.2.840.113549.1.1.5)
Subject Alternative Names:
Email Address:
IP Address:
DNS Name:

The requested action has successfully completed!

___15. Next a Channel Authentication record needs to be added to allow our new user (*USER3*) access to the queue manager. Use MQ Explorer and connect to *QMZ1*. Expand *Channels* and select *Channel Authentication Records*.



Channel Authentication Records

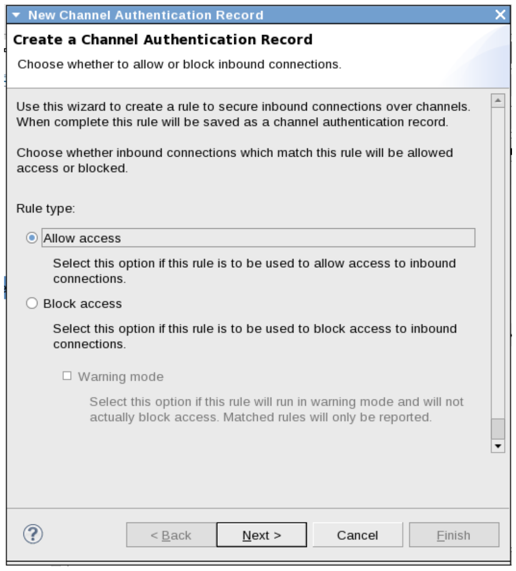
Filter: Standard for Channel Authentication Records

Channel profile	Type	Peer name	Client user ID	Remote queue manager	Addr
*	Block User List				
Client.to.Queue*	Address Map				*
SYSTEM.*	Address Map				*
SYSTEM.ADMIN.SVRCONN	Address Map				*
SYSTEM.DEF.SVRCONN	User Map		zosuser		*

Scheme: Standard for Channel Authentication Records - z/OS

Last updated: 11:27:08 (5 items)

___16. 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.

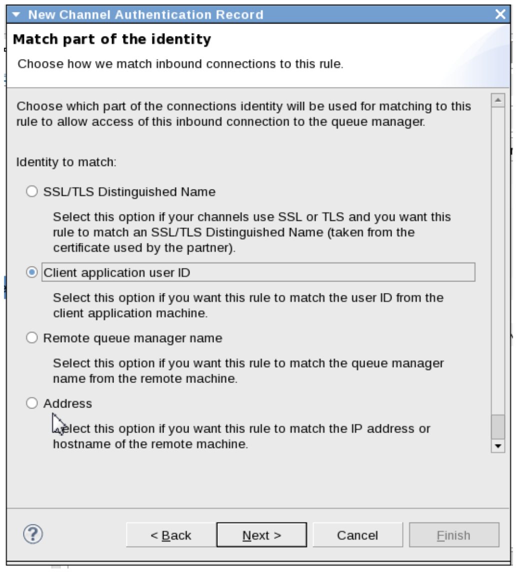
Block access

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.

___17. 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.

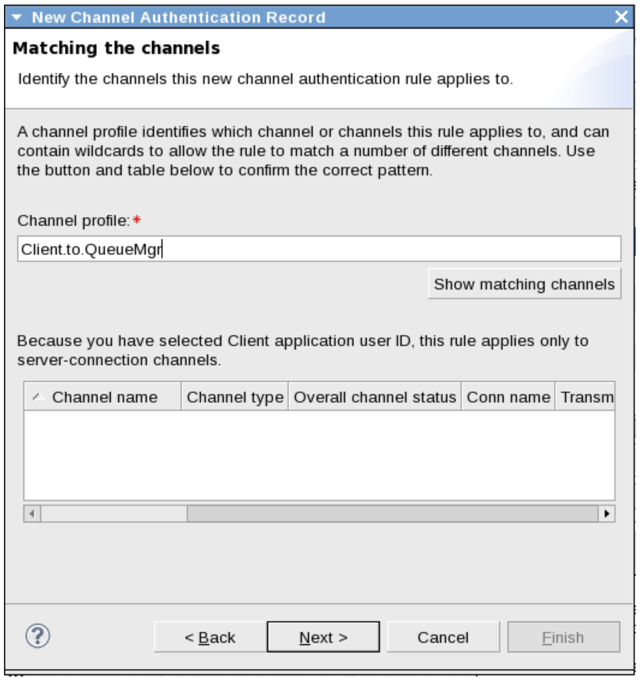
( ) Remote queue manager name

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.

___18. 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.

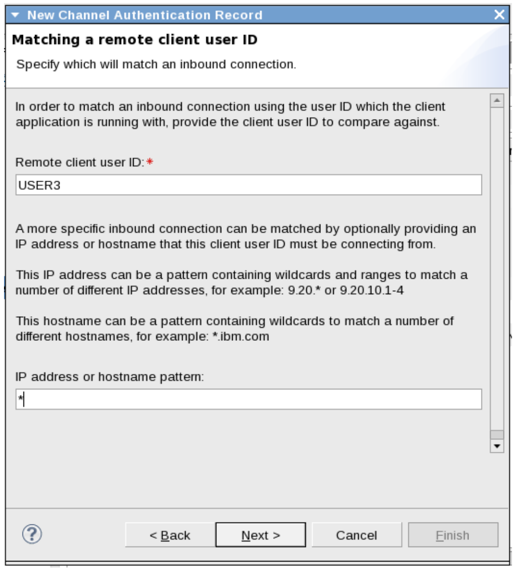
Channel profile: *

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

___19. On the **Matching a remote client user ID** window enter ***USER3*** 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:*

USER3

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.

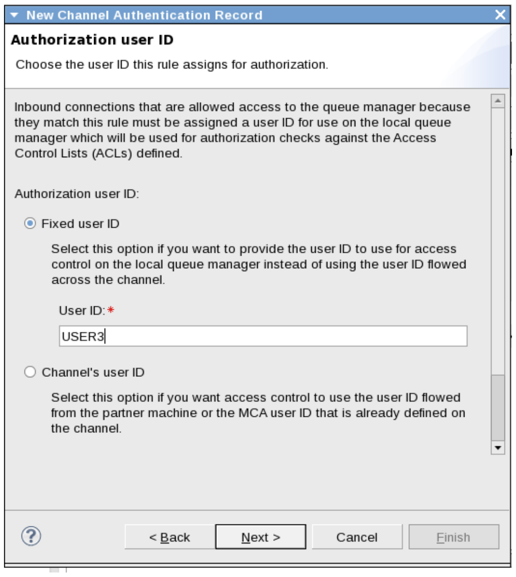
This IP address can be a pattern containing wildcards and ranges to match a number of different IP addresses, for example: 9.20.* or 9.20.10.1-4

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

IP address or hostname pattern:

*

___20. On the *Authorization user ID* window enter ***USER3*** as the *Fixed 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.

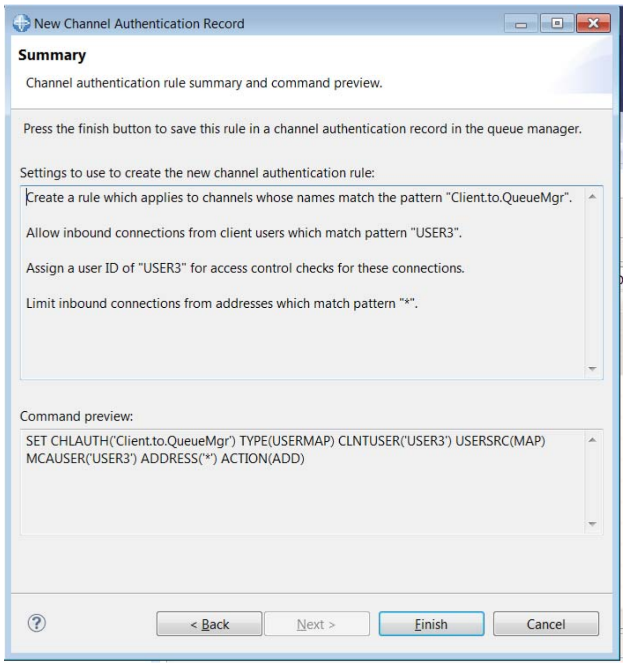
User ID:*

USER3

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.

___21. 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 "USER3".

Assign a user ID of "USER3" 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('USER3') USERSRC(MAP)
MCAUSER('USER3') ADDRESS('*') ACTION(ADD)
```

___22. Ensure there are messages on the AMS protected queues by submitting jobs P*UTMSGI* . Wait until the job completes before continuing.

___23. When job *PUTMSGI* completes, go to directory */z/home* enter command ***getmsg AMSDEMO.INTEGRITY.QUEUE*** in the command prompt window.

```
getmsg AMSDEMO.INTEGRITY.QUEUE
Success:  Obtained a reference to the Queue manager: QMZ1
Success:  Obtained a reference to the Queue: AMSDEMO.INTEGRITY.QUEUE
Success:  Obtained a reference to the MQ Message
Success:  The message was successfully retrieved from the queue.
=== Thu Jan 14 14:54:56 UTC 2016-- When angry, count four; when very angry, swear.
~Samuel Clemens
Success:  Closed the queue
Success:  Disconnected from the Queue Manager
```

Repeat this command until message *A WebSphere MQ error occurred: Completion code 2 Reason code 2033* appears (the queue is empty).

The retrieval of these messages was successful because they were placed on the queue by user *USER1*.

Tech-Tip: *getmsg* and *putmsg* are scripts that set up a Java execution environment and then invoke Java with the command below. The only variable passed to the scripts is the queue name that is the target of either a put or a get request.

java -classpath $CLASSPATH com.ibm.ats.encode.PutMessage client QMZ1 $1 USER3 USER3

___24. Next try putting a message on the *AMSDEMO.INTEGRITY.QUEUE* queue. Enter command ***putmsg AMSDEMO.INTEGRITY.QUEUE*** in the command prompt window.

```
putmsg AMSDEMO.INTEGRITY.QUEUE
Success:  Obtained a client reference to the Queue manager: QMZ1
Success:  Obtained a reference to the Queue: AMSDEMO.INTEGRITY.QUEUE
Success:  The MQ Message has been created: Thu Jan 14 10:02:21 EST 2016-- Clothes
make the man. Naked people have little or no influence in society. ~Samuel Clemens
Success:  The message 'Thu Jan 14 10:02:21 EST 2016-- Clothes make the man. Naked
people have little or no influence in society. ~Samuel Clemens' was successfully
sent to the queueName: 'AMSDEMO.INTEGRITY.QUEUE                         '
Success:  Closed the queue
Success:  Disconnected from the Queue Manager
```

___25. Now try to retrieve the message just placed on the *AMSDEMO.INTEGRITY.QUEUE* queue with command ***getmsg AMSDEMO.INTEGRITY.QUEUE***

```
getmsg AMSDEMO.INTEGRITY.QUEUE
Success:  Obtained a reference to the Queue manager: QMZ1
Success:  Obtained a reference to the Queue: AMSDEMO.INTEGRITY.QUEUE
Success:  Obtained a reference to the MQ Message
A WebSphere MQ error occurred : Completion code 2 Reason code 2063
An MQSeries error occurred : Security Error. Ensure the correct credentials were
used
```

The retrieval of this message failed because messages placed on the queue by *USER3* cannot be retrieved because the AMS policy for this queue does not allow this.

___26. Submit job ***PUTMSGP*** and when it completes, enter command ***getmsg AMSDEMO.PRIVACY.QUEUE*** in the command prompt window.

```
getmsg AMSDEMO.PRIVACY.QUEUE
Success:  Obtained a reference to the Queue manager: QMZ1
A WebSphere MQ error occurred : Completion code 2 Reason code 2063
An MQSeries error occurred : Security Error. Ensure the correct credentials were used
```

The AMS policy for this queue only allows access for *USER2*.

___27. Now edit member *ADDPOLCY* in *USER1.AMS.JCL* and add *USER3* as an authorized signer (-***a***) for queue *AMSDEMO.INTEGRITY.QUEUE* and add *USER3* as an authorized recipient (-***r***) of messages for queue *AMSDEMO.PRIVACY.QUEUE.*

```
setmqspl -m QMZ1
 -p AMSDEMO.INTEGRITY.QUEUE
 -s MD5
 -e NONE
 -a CN=USER1,O=IBM,C=US
 -a CN=USER3,O=IBM,C=US

 setmqspl -m QMZ1
 -p AMSDEMO.PRIVACY.QUEUE
 -s MD5
 -e AES256
 -a CN=USER2,O=IBM,C=US
 -r CN=USER2,O=IBM,C=US
 -r CN=USER3,O=IBM,C=US
```

___28. Submit job *ADDPOLCY* and confirm that it completes with a condition code or zero.

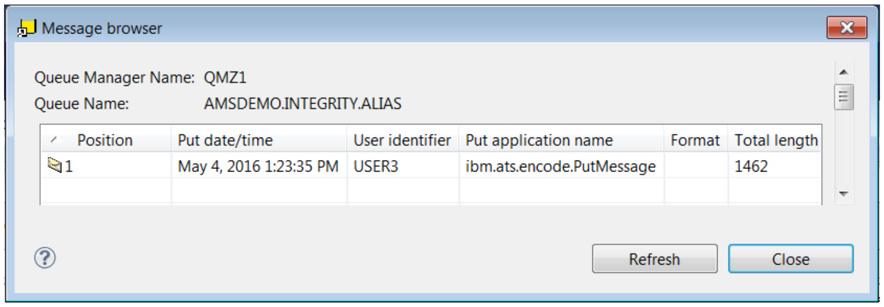
___29. Refresh the AMS task's policy information with MVS command

F QMZ1AMSM,REFRESH POLICY

___30. Repeat the ***putmsg AMSDEMO.INTEGRITY.QUEUE*** command to put a message the *AMSDEMO.INTEGRITY.QUEUE* as *USER3*.

```
putmsg AMSDEMO.INTEGRITY.QUEUE
Success:  Obtained a client reference to the Queue manager: QMZ1
Success:  Obtained a reference to the Queue: AMSDEMO.INTEGRITY.QUEUE
Success:  The MQ Message has been created: Fri Jan 15 12:48:24 EST 2016--
The evil that men do lives after them, The good is oft interred with their
bones, ~William Shakespeare From Julius Caesar
Success:  The message 'Fri Jan 15 12:48:24 EST 2016-- The evil that men do
lives after them, The good is oft interred with their bones, ~William
Shakespeare From Julius Caesar' was successfully sent to the queueName:
'AMSDEMO.INTEGRITY.QUEUE                               '
Success:  Closed the queue
Success:  Disconnected from the Queue Manager
```

___31. Use MQ Explorer to confirm there is a message on queue AMSDEMO.INTEGRITY.QUEUE placed there by *USER3*.



Queue Manager Name: QMZ1
Queue Name: AMSDEMO.INTEGRITY.ALIAS

Position	Put date/time	User identifier	Put application name	Format	Total length
1	May 4, 2016 1:23:35 PM	USER3	ibm.ats.encode.PutMessage		1462

___32. Repeat the ***getmsg AMSDEMO.INTEGRITY.QUEUE*** command to retrieve the message placed on the *AMSDEMO.INTEGRITY.QUEUE* by *USER3* can now be retrieved.

```
getmsg AMSDEMO.INTEGRITY.QUEUE
Success:  Obtained a reference to the Queue manager: QMZ1
Success:  Obtained a reference to the Queue: AMSDEMO.INTEGRITY.QUEUE
Success:  Obtained a reference to the MQ Message
Success:  The message was successfully retrieved from the queue.
=== Fri Jan 15 12:48:24 EST 2016-- The evil that men do lives after them,
The good is oft interred with their bones, ~William Shakespeare From Julius
Caesar
Success:  Closed the queue
Success:  Disconnected from the Queue Manage
```

Part 4 – Configuring a Client for Message Privacy

Message privacy requires that the public and private keys of the sender and recipient of private message be in the respective personal certificates and be available to the AMS task in order to encrypt/decrypt a message. This means that the certificate used in the previous part for message integrity cannot be used since the private key of the message recipient is not available. One method for providing the private key to the AMS task is if the certificate is generated on z/OS. This certificate must be exported to the client for installing in the client's key/trust store file.

In this part of the lab exercise a recipient certificate will be created in RACF so the public and private keys will be available to the AMS task.

First the existing certificate for *USER*3 needs to be deleted and a new certificate generated and then connected to the appropriate key rings by using RACF RACDCERT commands.

___1. Delete the existing *USER3* personal certificate.

```
racdcert id(user3) delete(label('USER3'))
```

___2. Generate a new personal certificate for USER3.

```
racdcert id(user3)gencert subjectsdn(CN('USER3') O('IBM') C('US'))
withlabel('USER3') signwith(certauth label('AMSCA')) keyusage(handshake
dataencrypt docsign) notafter(date(2020/12/31))
```

Tech-Tip: These commands and the commands in Step 3 and 4 are the same commands in the jobs in members USERCRTS and *ADDRING* in data set *USER1.AMS.JCL*. Modify these members and submit them rather than entering these as TSO commands in ISPF.

___3. Create a key ring for *USER*3 and connect *USER*3's personal certificate to this key ring.

```
racdcert id(user3) addring(drq.ams.keyring)

racdcert id(user3) connect(id(user3) ring(drq.ams.keyring) label('USER3')
      default usage(personal)
```

___4. Connect the *USER*3 personal certificate to the AMS task's keyring.

```
racdcert id(start1) connect(ring(drq.ams.keyring) id(user3)
      label('USER3') usage(site)
```

___5. Export the personal certificate of *USER*2 to a MVS data set.

```
racdcert id(user2) export(label('USER2')) dsn(user2.cert.arm)
```

___6. Export the personal certificate (including the private key) of *USER*3 to a MVS data set.

```
racdcert id(user3) export(label('USER3')) dsn(user3.cert.p12)
format(pkcs12der) password('passw0rd')
```

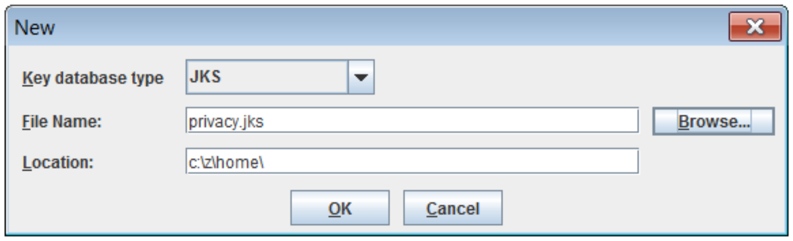
___7. Use MVS command ***F QMZ1AMSM,REFRESH ALL*** to update the active AMS policies and RACF KEYRING configuration.

___8. Go to directory */z/home* and use FTP to move to the local system the exported certificate for USER2 in ASCII mode and the certificate for USER3 in binary mode.

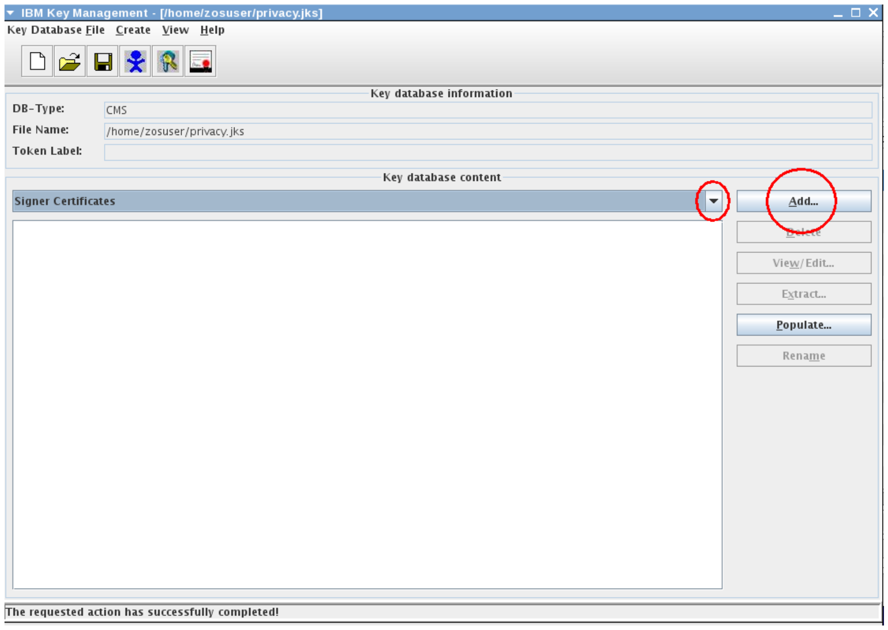
```
ftp wg31.washington.ibm.com
Connected to wg31.washington.ibm.com (192.27.216.44).
220-FTPD1 IBM FTP CS V2R1 at WG31.WASHINGTON.IBM.COM, 19:30:27 on 2016-01-17.
220 Connection will close if idle for more than 5 minutes.
Name (mpx1:zosuser): USER1
331 Send password please.
Password:
230 USER1 is logged on.  Working directory is "USER1.".
Remote system type is MVS.
ftp> mget USER2.CERT.ARM
mget USER2.CERT.ARM? y
227 Entering Passive Mode (192,27,216,44,4,19)
125 Sending data set USER1.USER2.CERT.P12
250 Transfer completed successfully.
ftp> bin
200 Representation type is Image
2306 bytes received in 0.00842 secs (273.97 Kbytes/sec)
ftp> mget USER3.CERT.P12
mget USER3.CERT.P12? y
227 Entering Passive Mode (192,27,216,44,4,19)
125 Sending data set USER1.USER3.CERT.P12
250 Transfer completed successfully.
2306 bytes received in 0.00842 secs (273.97 Kbytes/sec)
ftp> quit
```

___9. Start the *IBM Key Management* tool by entering command ***strmqikm c:\z\home***.

___10. Click on the Key Data File on the tool line and select *New*. On the *New* window use the pull-down arrow to select a *Key database type* of *JKS* and change the *File Name* to ***privacy.jks*** and ensure the Location is set to ***c:\z\home***. Click **OK** to continue and enter a password of ***changeit***. Click **OK** to continue.


New
Key database type: JKS
File Name: privacy.jks
Location: c:\z\home\

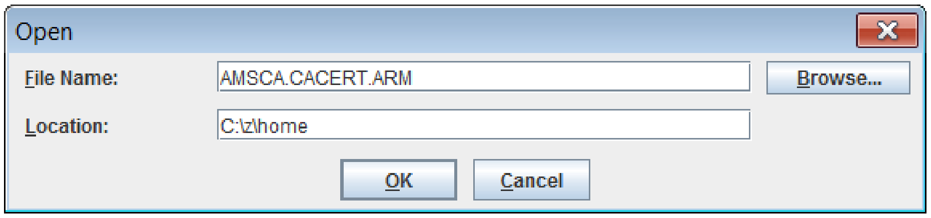
___11. Use the pull down arrow to select *Signer Certificates* and then click on the **Add** button.



DB-Type: CMS
File Name: /home/zosuser/privacy.jks
Token Label:

The requested action has successfully completed!

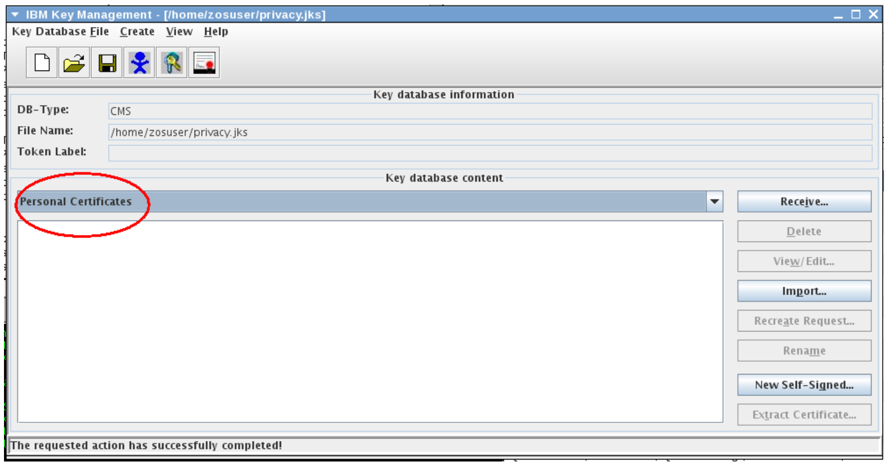
___12. On the *Open* window use the **Browse** button and select file *AMSCA.CACERT.ARM* and click **OK** to continue.



File Name: AMSCA.CACERT.ARM
Location: C:\z\home

___13. Enter ***AMSCA*** as the label of the certificate. (Do not be concerned about the label being folder to lower case.)

___14. Use the pull down arrow and select *Personal Certificates*.

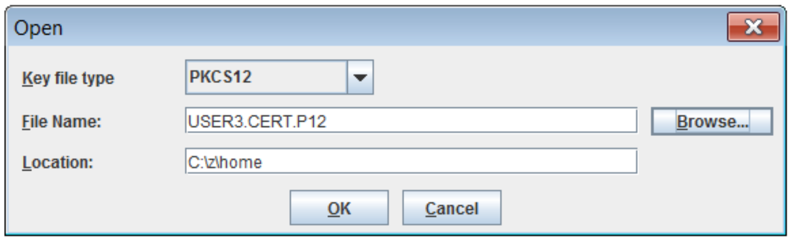


DB-Type: CMS
File Name: /home/zosuser/privacy.jks
Token Label:

The requested action has successfully completed!

___15. Next import the new USER3 certificate that was created by RACF and just downloaded by clicking the **Import** button.

___16. Use the pull down arrow to select a *Key file type* of *PKCS12* and then use the **Browse** button to select files *USER3.CERT.P12*. Click **OK** to continue.



Open

Key file type: PKCS12

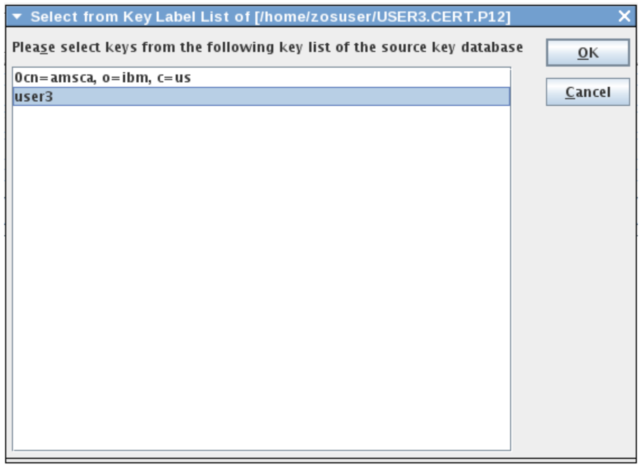
File Name: USER3.CERT.P12

Location: C:\z\home

___17. Enter ***passw0rd*** on the *Password Prompt* window.

Tech-Tip: This is the same password specified in the export command invoked in Step 5 above.

___18. On the *Select from Key Lab List of (c:\z\home\USER3.CERT.P12)* window select *user3* and click **OK** to continue.

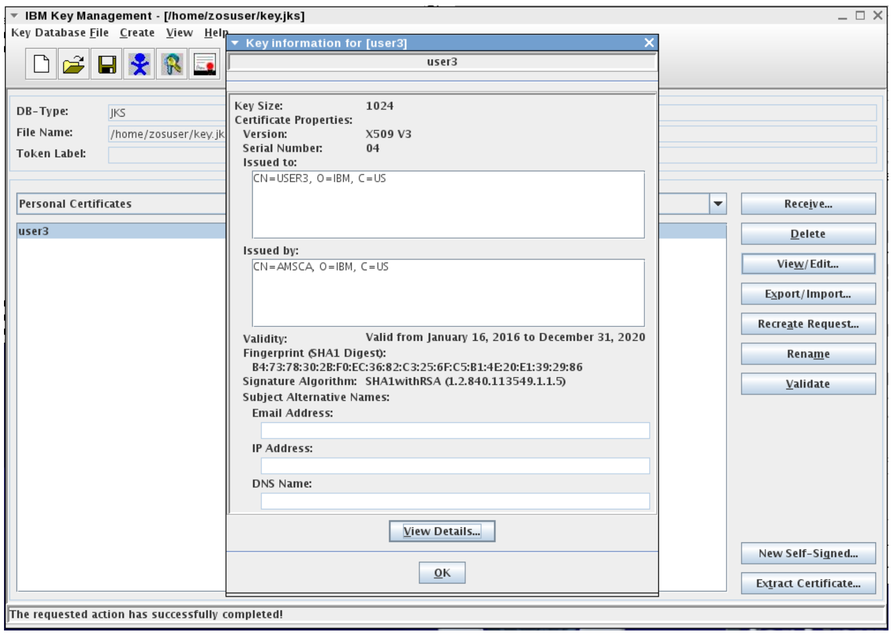


Select from Key Label List of [/home/zosuser/USER3.CERT.P12]

Please select keys from the following key list of the source key database

0cn=amsca, o=ibm, c=us
user3

___19. Click **OK** on the *Change Label* window, use the View/Edit button to display the certificate and then close the *IBM Key Management* tool.



```
Key Size:                 1024
Certificate Properties:
  Version:                X509 V3
  Serial Number:          04
  Issued to:
    CN=USER3, O=IBM, C=US
  Issued by:
    CN=AMSCA, O=IBM, C=US
  Validity:               Valid from January 16, 2016 to December 31, 2020
  Fingerprint (SHA1 Digest):
    B4:73:78:30:2B:F0:EC:36:82:C3:25:6F:C5:B1:4E:20:E1:39:29:86
  Signature Algorithm:  SHA1withRSA (1.2.840.113549.1.1.5)
  Subject Alternative Names:
    Email Address:
    IP Address:
    DNS Name:
```

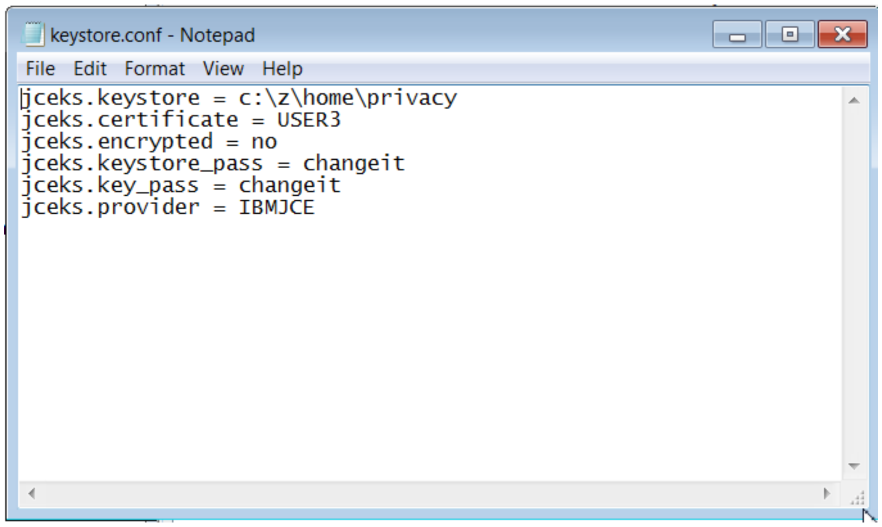
The requested action has successfully completed!

___20. Close the *IBM Key Management* tool.

___21. Use the *keytool* command to import the USER2 certificate into the local keystore.

```
keytool -v -import -alias "USER2" -file USER2.CERT.ARM  -keystore privacy.jks
Enter keystore password:  changeit
Certificate was added to keystore
[Storing privacy.jks]
```

___22. Use the command ***notepad c:\users\workstation\.mqs\keystore.conf*** to open the AMS key store configuration file. Change the *jceks.keystore* directive to this new key store file, c:\**z*****home******privacy***. Save the change by closing the *gedit* window.



```
jceks.keystore = c:\z\home\privacy
jceks.certificate = USER3
jceks.encrypted = no
jceks.keystore_pass = changeit
jceks.key_pass = changeit
jceks.provider = IBMJCE
```

Tech-Tip: The keystore file extension does not need to be specified.

Tech-Tip: If you did not use ***changeit*** for the key store password then directives *jceks.key_pass* and *jceks.keystore_pass* should be updated with the password used.

___23. Use MQ Explorer to clear any existing messages on the *AMSDEMO.PRIVACY.QUEUE* queue.

Tech-Tip: Changes to AMS policies are not retroactive. Any messages on the queue prior to adding access to USER3 would still not be accessible prior and would still generate a 2063 error code if USER3 tried to retrieve those messages.

___24. Back on TSO, submit job *PUTMSGP* in *USER1.AMS.JCL* to place 2 new messages on queue *AMSDEMO.PRIVACY.QUEUE* queue.

___25. Try retrieving the messages from privacy protected queue using the *getmsg* script, e.g. ***getmsg AMSDEMO.PRIVACY.QUEUE***

```
getmsg AMSDEMO.PRIVACY.QUEUE
Success:  Obtained a reference to the Queue manager: QMZ1
Success:  Obtained a reference to the Queue: AMSDEMO.PRIVACY.QUEUE
Success:  Obtained a reference to the MQ Message
Success:  The message was successfully retrieved from the queue.
=== Mon Feb 08 15:40:32 UTC 2016-- When the going gets tough the tough get
going. ~typing exercise
Success:  Closed the queue
Success:  Disconnected from the Queue Manager
getmsg AMSDEMO.PRIVACY.QUEUE
Success:  Obtained a reference to the Queue manager: QMZ1
Success:  Obtained a reference to the Queue: AMSDEMO.PRIVACY.QUEUE
Success:  Obtained a reference to the MQ Message
Success:  The message was successfully retrieved from the queue.
=== Mon Feb 08 15:40:46 UTC 2016-- Cry 'Havoc'! and let slip the dogs of war.
~William Shakespeare From Julius Caesar
Success:  Closed the queue
Success:  Disconnected from the Queue Manager
```

This should be successful since USER3 is a valid recipient of messages protected by this policy.

CONGRATULATIONS!!! You have completed the MQ Advanced Message Security

Summary

In this exercise you defined the security artifacts required for AMS on z/OS. You started with defining the digital certificates (both personal and certificate authority) and the AMS security policies to provide message integrity and privacy.

The artifacts you defined were tested initially using MVS batch jobs and then testing moved to a Windows distributed environment. In both environments you saw the results when AMS either allowed messages to be retrieved or not based the AMS security policies defined in the exercise.