Installation, Configuration and Basic Test of IBM MQ 9.0 and 9.2 Advanced Message Security (AMS) in Linux

https://www.ibm.com/support/pages/node/598373

Date last updated: 01-May-2022

Angel Rivera
IBM MQ Support
https://www.ibm.com/products/mq/support
Find all the support you need for IBM MQ

+++ Thanks to Bob Gibson for his suggestions for improving this tutorial!

+ Update on 30-Apr-2020:
- MQ 9.2.5 CD was used under RHEL 8.5 to validate the scenarios.
- Minor corrections and improvements were done (Thank you Bob Gibson!)
- Only the users who are going to put/get messages from an AMS protected queue need to create keystore and certificates.
The queue manager does NOT use the certificates from these users.

+++ Objective

The objective of this technical document is to describe in detail how to install and configure for first usage the MQ Advanced Message Security (AMS) on a queue manager at version 9.0 in Linux.

The queue manager will have 2 queues, one that is not protected by AMS, and the other queue is protected by AMS.

This document also shows how to perform a basic test using the following samples (which use local bindings mode) amqsput and amqsget by 3 users:
- one authorized to put,
- another authorized to get, and
- another that is not authorized.
To keep the scope as simple as possible for this tutorial the 3 users are in the same server as the queue manager. That is, they are local users and they are not using server-connection channels.

MQ provides transport-level security with the feature of TLS over channels. However, by default, MQ does not provide a method to encrypt and secure access to messages while they are at rest on queues. If AMS is used in an MQ environment, it is now possible to implement full end-to-end security.

The chapters in this techdoc are:

+ Update on 08-Jul-2020:

a) New diagram of topology to clarify that the scenarios are using 2 users that connect via local bindings in the same server as the queue manager.

b) Reference to new tutorial in which the 2 users connect from remote servers and use server-connection channels:

https://www.ibm.com/support/pages/node/6244608
Configuration and basic test of remote clients for MQ 9.1 Advanced Message Security (AMS) in Linux

c) New Chapter 12 about troubleshooting

+ Update from 16-Aug-2018
Additional information on the performance improvements.

In Chapter 11 a table shows the performance improvement:

Queue Name Protected by KeyReuse Time to put Time to get
By AMS 10k messages 10k messages
Q1 No not applicable 0.097445 S 0.112199 S
Q.AMS Yes 0 (default) 7.542336 S 12.026407 S
Q.AMS Yes 50 0.189219 S 0.290232 S

Notice that the 1st row is the baseline (no AMS) and the time in column 4 shows that it took around 0.1 second to put 10,000 messages.
The 2nd row is the pre-9.0 function of AMS, and it took around 7.5 seconds to do the same task. Notice that the difference with the baseline is really big!

The 3rd row exploits the new option in 9.0 and it took 0.19 seconds, almost double than the baseline in the 1st row but far less than the one for the 2nd row.

https://www.ibm.com/developerworks/community/blogs/messaging/entry/AMS_Confidentiality_Performance?lang=en
AMS Confidentiality Performance
Sam Massey | July 27 2016
https://www.ibm.com/developerworks/community/blogs/messaging/entry/Bitesize_Blogging_MQ_V9_Fast_encrypted_messages_with_MQ_Introducing_AMS_Confidentiality_Policies?lang=en
Bitesize Blogging: MQ V9 Fast encrypted messages with MQ -Introducing AMS Confidentiality Policies
Jonathan Rumsey | June 1 2016
https://ibm-messaging.github.io/mqperf/
New site for MQ Performance Reports
AMS
MQ V9 delivered a new AMS Quality of Protection called ‘Confidentiality’. A performance whitepaper has been produced that illustrates the performance profile this new mode brings by comparing it to existing AMS and non AMS scenarios.
File: https://ibm-messaging.github.io/mqperf/AMS.pdf

+ Topology

The testing in this tutorial will use transport type of “bindings”, using the local samples amqsput and amqsget by 3 users:
- one authorized to put,
- another authorized to get, and
- another that is not authorized.

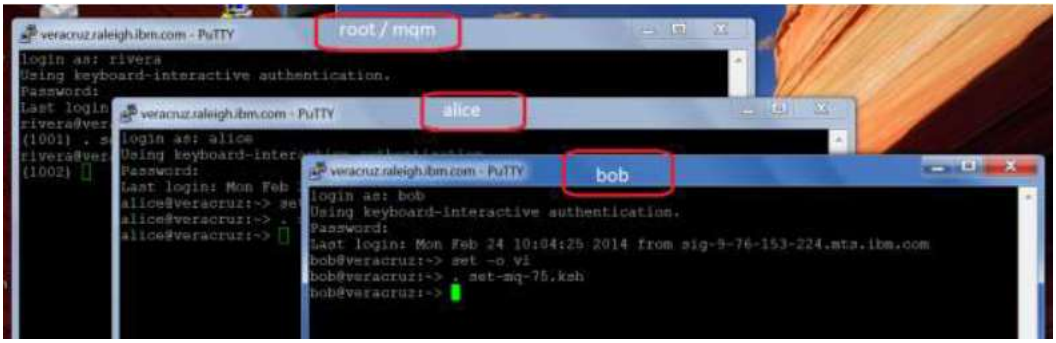
To keep the scope as simple as possible for this tutorial the 3 users are in the same server as the queue manager. That is, they are local users and they are not using server-connection channels.

Topology, 1 single host (2 local users)

Host-1

+ Test recommendation: to have 3 separate command prompt windows
Because this scenario describes the tasks done by multiple users, it is best to create at least three (3) separate command prompt windows, which helps to reduce confusion.

Window 1: for users “root” and “mqm”
Window 2: for user “alice”
Window 3: for user “bob”



+ References

The material in this techdoc is based on the following chapter:

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.sec.doc/q014700_.htm
IBM MQ > IBM MQ 9.0.x > IBM MQ > Security > Advanced Message Security > Advanced Message Security overview > User scenarios for Advanced Message Security > Quick Start Guide for AMS on UNIX platforms

+ Reference of older techdoc

http://www-01.ibm.com/support/docview.wss?uid=swg27041465
Installation, Configuration and Basic Test of WebSphere MQ Advanced Message Security 7.5 in Linux

++
+++ Chapter 1: Installing the AMS code and establishing MQ environment in a session
++

UNIX host: Linux SLES 12 SP 1, x86-64-bit
MQ 9.0.0.0

This chapter describes the installation of the AMS components.
You also need to install the MQ samples, which include amqsput and amqsget.

+ Use Window 1 and log in as root.
Starting with MQ 7.5, the AMS code has been incorporated into the main product and the AMS code is now obtained with the download images from the IBM Passport Advantage site.

In MQ AMS 7.5 and later for Linux, the filesets for AMS are packaged with the MQ server filesets.

You need to log in as user “root” to install the MQ filesets.

The following free redbook has an overview of the installation steps which they apply to MQ 8.0 and 9.0.

http://www.redbooks.ibm.com/redpieces/abstracts/sg248087.html?Open
WebSphere MQ V7.1 and V7.5 Features and Enhancements

… specifically, the section:
Section 16.1 (Page 232) WebSphere MQ Advanced Message Security installation

The following names of the AMS packages on UNIX and Linux are used:
AIX: mqm.ams.rte
HP-UX: MQSERIES.MQM-AMS
Linux: MQSeriesAMS
Solaris: mqams

In the host of the queue manager, there are several versions of MQ running at the same time. MQ 9.0.0.0 is available in Installation3 under /opt/mqm90.

It is necessary to establish the proper set of environment variables for MQ within each Unix command prompt.
To facilitate this task, a shell script was used and the contents is shown below.

Shell script (located in /usr/local/bin)
Name: set-mq-90.ksh

```
# Name: set-mq-90
# Purpose: to setup the environment to run MQ 9.0
. /opt/mqm90/bin/setmqenv -n Installation3
# Additional MQ directories for the PATH
export
PATH=$PATH:$MQ_INSTALLATION_PATH/bin:$MQ_INSTALLATION_PATH/java/bin:$MQ_
INSTALLATION_PATH/samp/bin:$MQ_INSTALLATION_PATH/samp/jms/samples:
# Add local directory for running Java/JMS programs
export CLASSPATH=$CLASSPATH:.
# Display the full fix pack level
dspmqver -f 2
# end
```

+ Example usage
Note that upon initiating a command prompt session, there are no MQ environment variables:

```
$ set | grep MQ
$ echo $PATH
/home/mqm/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/games
```

Issue the script that establishes the environment variables for MQ:

You MUST enter the dot followed by a space, before the script name.

```
$ . set-mq-90
Version: 9.0.0.0
```

Notice that now there are MQ environment variables

```
$ set | grep MQ
MQ_DATA_PATH=/var/mqm
MQ_ENV_MODE=64
MQ_INSTALLATION_NAME=Installation3
MQ_INSTALLATION_PATH=/opt/mqm90
MQ_JAVA_DATA_PATH=/var/mqm
MQ_JAVA_INSTALL_PATH=/opt/mqm90/java
MQ_JAVA_LIB_PATH=/opt/mqm90/java/lib64
MQ_JRE_PATH=/opt/mqm90/java/jre64/jre
MQ_RETVAL=0
```

Notice that the PATH includes now the MQ commands

```
$ echo $PATH
/opt/mqm90/bin:/home/mqm/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:/usr/ga
mes:/opt/mqm90/bin:/opt/mqm90/java/bin:/opt/mqm90/samp/bin:/opt/mqm90/sa
mp/jms/samples:
```

++
+++ Chapter 2: Creating a queue manager and a queue
++

++ Example of the line commands to create a queue manager

+ Use Window 1 and log in as user “mqm”.
You need to log in as user “mqm” or a member of the MQ Administration group (group “mqm”).

-Establish the environment variables for MQ

. set-mq-90

-Create the queue manager.

crtmqm -u DLQ QM_VERIFY_AMS

The -u flag indicates which queue is going to be the dead letter queue (DLQ).
Hint: Many MQ Explorer users hide the SYSTEM* queues and thus, if you use the SYSTEM.DEAD.LETTER.QUEUE as the DLQ, then it will be hidden and you might not notice if there are messages in the dead letter queue

-Start the queue manager

strmqm QM_VERIFY_AMS

-Configure the queue manager

runmqsc QM_VERIFY_AMS

Define a normal queue which will NOT be protected by AMS

define qlocal(Q1)

Define the testing queue which will be protected by AMS

define qlocal(Q.AMS)

Define a listener. It is a good idea to specify the port number in the name in that way a quick look at the list of listeners will tell you the port number right away.
The default port is 1414, however here the port 1456 will be used instead in this test.

define listener(LISTENER.1456) trptype(tcp) control(qmgr) port(1456)

start listener(LISTENER.1456)

```
## Define a channel to be used by a remote MQ Explorer

define channel(SYSTEM.ADMIN.SVRCONN) chltype(SVRCONN)


## Define the DLQ

define qlocal(DLQ) like(SYSTEM.DEAD.LETTER.QUEUE)

## For MQ 7.1 and later and if desiring to allow remote connections by
an MQ Administrator (to avoid return code 2035). This is OK for test queue managers.
This security feature does NOT interfere at all with AMS.

set CHLAUTH(*) TYPE(BLOCKUSER) USERLIST('nobody','*MQADMIN')
set CHLAUTH(SYSTEM.ADMIN.*) TYPE(BLOCKUSER) USERLIST('nobody')

## For MQ 8.0 and later to disable password for remote MQ administrators.
This security feature does NOT interfere at all with AMS.

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

## Display the attribute SPLCAP, which is the attribute that indicates if AMS is enabled
(the fact that the MQ AMS fileset is installed, that is considered to be “enabled”).

display qmgr SPLCAP

AMQ8408: Display Queue Manager details.
QMNAME(QM_VERIFY_AMS) SPLCAP(ENABLED)

## Display the 2 system queues used by AMS

display ql(SYSTEM.PROTECTION*)

AMQ8409: Display Queue details.
QUEUE(SYSTEM.PROTECTION.ERROR.QUEUE) TYPE(QLOCAL)
AMQ8409: Display Queue details.
QUEUE(SYSTEM.PROTECTION.POLICY.QUEUE) TYPE(QLOCAL)

## exit runmqsc

end
```

++
+++ Chapter 3: Creating and authorizing users
++

++ Creating users

+ Window 1: User root
Log in as user “root”.

Use line commands or the YAST GUI or another administrative tool to create:

Group:
mqusers => groupadd -g 1005 mqusers

Users:
Alice => useradd -u 1008 -g mqusers -s /bin/bash -d /home/alice -m alice
bob => useradd -u 1009 -g mqusers -s /bin/bash -d /home/bob -m bob
fulano => useradd -u 1021 -g mqusers -s /bin/bash -d /home/fulano -m fulano

Notice that the user “fulano” will be used in the
chapter that shows what happens
when an unauthorized user tries to browse the AMS protected messages.

For the scenarios described in this document, these users are NOT MQ administrators, therefore they should NOT belong to the group “mqm”.
Remember that in UNIX, any member of the group “mqm” (either as primary or a set of groups), is automatically an MQ administrator.
In this scenario, the users are members of the group “mqusers”.

id alice
uid=1008(alice) gid=1005(mqusers) groups=1005(mqusers)

id bob
uid=1009(bob) gid=1005(mqusers) groups=1005(mqusers)

id fulano
uid=1021(fulano) gid=1005(mqusers) groups=1005(mqusers)

++ Authorizing users

+ Window 1: User mqm
Log in as user “mqm”

The following commands were used to authorize the users to connect to the queue Manager.
Notice that you can have multiple instances of the -p parameter:
setmqaut -m QM_VERIFY_AMS -t qmgr -p alice -p bob +connect +inq +dsp

And to work with the queue Q.AMS: alice can put and bob can get.

setmqaut -m QM_VERIFY_AMS -n Q.AMS -t queue -p alice +put +browse +dsp
setmqaut -m QM_VERIFY_AMS -n Q.AMS -t queue -p bob +get +browse +dsp

The following commands are for the advanced testing done in the last chapter, in which user fulano has normal non-AMS authorities, but is not explicitly authorized by AMS.

setmqaut -m QM_VERIFY_AMS -t qmgr -p fulano +connect +inq +dsp
setmqaut -m QM_VERIFY_AMS -n Q.AMS -t queue -p fulano +put +browse +dsp
setmqaut -m QM_VERIFY_AMS -n Q.AMS -t queue -p fulano +get +browse +dsp

Note:
Technically speaking, the authority in MQ is based on the group membership of the user. Thus, the setmqaut command for user alice actually has the side effect of giving authority to ALL the users who belong to the same primary group as alice, that is 'mqusers'. This means that users bob and fulano will automatically be authorized similar to alice. This is equivalent to use the -g flag (for group) in setmqaut.

Additionally, it is necessary to allow the two users alice and bob (but not user fulano) to browse the AMS system policy queue, and put messages on the AMS error queue.

setmqaut -m QM_VERIFY_AMS -t queue -n SYSTEM.PROTECTION.POLICY.QUEUE -p alice -p bob +browse

setmqaut -m QM_VERIFY_AMS -t queue -n SYSTEM.PROTECTION.ERROR.QUEUE -p alice -p bob +put

++ Verification that users alice and bob can put/get messages using the unprotected queue Q.AMS (at this point, the queue has not been configured to be protected by AMS – this will be done later on).

Before proceeding with the AMS example, let's use the amqsput and amqsget samples to verify that the users can put and get messages:

+ Window 2: User alice
Log in as user “alice”

Select to work with the MQ 9.0 environment:

. set-mq-90
Put a message to the unprotected queue Q.AMS:

amqsput Q.AMS QM_VERIFY_AMS

Sample AMQSPUT0 start
target queue is Q.AMS
test-AMS

Sample AMQSPUT0 end

+ Window 3: User bob
Log in as user “bob”

Select to work with the MQ 9.0 environment:

. set-mq-90

Get a message from the unprotected queue Q.AMS:

amqsget Q.AMS QM_VERIFY_AMS

Sample AMQSGET0 start
message <test-AMS>
no more messages
Sample AMQSGET0 end

++
+++ Chapter 4: Creating key database and certificates for alice and bob
++

To encrypt the message, the AMS interceptors require the public key of the sending users. Thus, the key database of user identities mapped to public and private keys must be created.

In this scenario, we are using self-signed certificate which can be created without using a Certificate Authority. For production systems, it is advisable not to use self-signed certificates however instead rely on certificates signed by a Certificate Authority.

+ Window 2: User alice
This is the window where you have already log in as alice

The umask used in this example is the following:

umask

0022

Note: This umask is used by the operating system to setup the permissions when creating files. The following is an example in which a file is created with 644 (rw-r--r--) file permissions:
alice@mosquito:~> touch file.txt
alice@mosquito:~> ls -l file.txt
-rw-r--r--1 alice mqusers 0 Apr 22 10:52 file.txt

Create a new key database for user alice
The -p flag will create intermediate directories, if they do not yet exist. It is useful when dealing a deep directory tree.

mkdir /home/alice/.mqs -p

runmqakm -keydb -create -db /home/alice/.mqs/alicekey.kdb -pw passw0rd -stash

The following are the directories and files that were created:

ls -dl /home/alice/.mqs

drwxr-xr-x 2 alice mqusers 86 Apr 22 10:54 /home/alice/.mqs

ls -l /home/alice/.mqs

-rw-------1 alice mqusers 88 Apr 22 10:54 alicekey.crl
-rw-------1 alice mqusers 88 Apr 22 10:54 alicekey.kdb
-rw-------1 alice mqusers 88 Apr 22 10:54 alicekey.rdb
-rw-------1 alice mqusers 129 Apr 22 10:54 alicekey.sth

Create a self-signed certificate identifying the user alice for use in encryption

runmqakm -cert -create -db /home/alice/.mqs/alicekey.kdb -pw passw0rd -label Alice_Cert -dn "CN=alice,O=IBM,C=GB" -default_cert yes

Notes:
-The 'label' parameter specifies the name for the certificate, which interceptors will look up to receive necessary information.
-The 'DN' parameter specifies the details of the Distinguished Name (DN), which must be unique for each user.

Notice the increase in size for alicekey.kdb, which indicates that the new certificate is stored in that file.

ls -l /home/alice/.mqs
-rw-------1 alice mqusers 88 Apr 22 10:54 alicekey.crl
-rw-------1 alice mqusers 5088 Apr 22 10:59 alicekey.kdb
-rw-------1 alice mqusers 88 Apr 22 10:54 alicekey.rdb
-rw-------1 alice mqusers 129 Apr 22 10:54 alicekey.sth

+ Window 3: User bob
This is the window where you have already log in as bob

The umask used in this example is:

umask
0022

Create a new key database for the user bob

mkdir /home/bob/.mqs -p

runmqakm -keydb -create -db /home/bob/.mqs/bobkey.kdb -pw passw0rd -stash

The following are the directories and files that were created:

```
ls -dl /home/bob/.mqs
drwxr-xr-x 2 bob mqusers 78 Apr 22 11:00 /home/bob/.mqs

ls -l /home/bob/.mqs
-rw-------1 bob mqusers 88 Apr 22 11:00 bobkey.crl
-rw-------1 bob mqusers 88 Apr 22 11:00 bobkey.kdb
-rw-------1 bob mqusers 88 Apr 22 11:00 bobkey.rdb
-rw-------1 bob mqusers 129 Apr 22 11:00 bobkey.sth
```

Create a certificate identifying the user bob for use in encryption

```
runmqakm -cert -create -db /home/bob/.mqs/bobkey.kdb -pw passw0rd -label
Bob_Cert -dn "CN=bob,O=IBM,C=GB" -default_cert yes

ls -l /home/bob/.mqs
-rw-------1 bob mqusers 88 Apr 22 11:00 bobkey.crl
-rw-------1 bob mqusers 5088 Apr 22 11:01 bobkey.kdb
-rw-------1 bob mqusers 88 Apr 22 11:00 bobkey.rdb
-rw-------1 bob mqusers 129 Apr 22 11:00 bobkey.sth
```

++
+++ Chapter 5: Creating keystore.conf for alice and bob
++

You must point MQ Advanced Message Security interceptors to the directory
where the key databases and certificates are located. This is done via
the keystore.conf file, which hold that information in the plain text form.

Each user must have a separate keystore.conf file.
Therefore, this step should be done for both alice and bob.

The content of keystore.conf must be of the form:

cms.keystore = <dir>/keystore_file
cms.certificate = certificate_label

Notes:
- The path to the keystore file must be provided with no file extension.
- $HOME/.mqs/keystore.conf is the default location where MQ Advanced Message Security searches for the keystore.conf file.

+ Window 2: User alice
Create file:

vi /home/alice/.mqs/keystore.conf

The contents is:

cms.keystore = /home/alice/.mqs/alicekey
cms.certificate = Alice_Cert

ls -l /home/alice/.mqs

-rw-------1 alice mqusers 88 Apr 22 10:54 alicekey.crl
-rw-------1 alice mqusers 5088 Apr 22 10:59 alicekey.kdb
-rw-------1 alice mqusers 88 Apr 22 10:54 alicekey.rdb
-rw-------1 alice mqusers 129 Apr 22 10:54 alicekey.sth

-rw-r--r--1 alice mqusers 70 Apr 22 11:02 keystore.conf

+ Window 3: User bob
Create file:

vi /home/bob/.mqs/keystore.conf

The contents is:

cms.keystore = /home/bob/.mqs/bobkey
cms.certificate = Bob_Cert

ls -l /home/bob/.mqs
-rw-------1 bob mqusers 88 Apr 22 11:00 bobkey.crl
-rw-------1 bob mqusers 5088 Apr 22 11:01 bobkey.kdb
-rw-------1 bob mqusers 88 Apr 22 11:00 bobkey.rdb
-rw-------1 bob mqusers 129 Apr 22 11:00 bobkey.sth
-rw-r--r--1 bob mqusers 64 Apr 22 11:04 keystore.conf

++
+++ Chapter 6: Sharing Certificates
++

It is necessary to share the certificates between the two key databases so that each user can successfully identify each other.

Because these users are located in the same host, the directory /tmp will be used as the neutral directory to exchange the certificates between the users.

But if the users were located in different boxes, then you will need to use ftp and specify the file transfer as “ascii”.

+ Window 2: User alice
Export the certificate identifying alice to a file located in /tmp.
The resulting file will be written as ascii text, which is the default (-format ascii).

runmqakm -cert -extract -db /home/alice/.mqs/alicekey.kdb -pw passw0rd -label Alice_Cert -target /tmp/alice_public.arm

Allow the certificate to be read by others

Notice that in this case, the file permissions will not allow bob to read the file!
ls -l /tmp/*.arm
-rw-------1 alice mqusers 782 Apr 22 11:06 /tmp/alice_public.arm

Thus, it is necessary to allow members of the Unix group and others to read the file.
chmod 644 /tmp/alice_public.arm
ls -l /tmp/*.arm
-rw-r--r--1 alice mqusers 782 Apr 22 11:06 /tmp/alice_public.arm

+ Window 3: User bob
Add the certificate from alice into bob's keystore:
runmqakm -cert -add -db /home/bob/.mqs/bobkey.kdb -pw passw0rd -label Alice_Cert -file /tmp/alice_public.arm

Notice that the size for bobkey.kdb has increased, to reflect the added certificate:
ls -l /home/bob/.mqs
-rw-------1 bob mqusers 88 Apr 22 11:00 bobkey.crl
-rw-------1 bob mqusers 10088 Apr 22 11:09 bobkey.kdb
-rw-------1 bob mqusers 88 Apr 22 11:00 bobkey.rdb
-rw-------1 bob mqusers 129 Apr 22 11:00 bobkey.sth
-rw-r--r--1 bob mqusers 64 Apr 22 11:04 keystore.conf

Print the details of the certificate for alice, to verify that it is indeed in the keystore,

runmqakm -cert -details -db /home/bob/.mqs/bobkey.kdb -pw passw0rd
-label Alice_Cert

+ begin excerpt
Label : Alice_Cert
Key Size : 1024
Version : X509 V3
Serial : 7bdb43f5424cd529
Issuer : CN=alice,O=IBM,C=GB
Subject : CN=alice,O=IBM,C=GB
Not Before : April 21, 2017 10:59:15 AM EDT

Not After : April 22, 2018 10:59:15 AM EDT

Public Key
30 81 9F 30 0D 06 09 2A 86 48 86 F7 0D 01 01 01
05 00 03 81 8D 00 30 81 89 02 81 81 00 B1 F9 C3
62 2A C0 96 62 BB 0E 05 A8 90 AF 2B 84 66 B5 2D
80 6E 9B 46 32 4E D9 F9 31 EA 02 3C E6 D8 9A 1E
C1 43 3A AC 87 F3 D9 78 23 DB 22 45 25 90 C3 6E
4D B3 62 3F 7A 8D F8 07 A7 13 CE 39 04 B1 25 05
86 9C AD 27 36 59 D8 12 9D 67 01 A5 84 15 24 21
BD 49 E7 82 19 20 91 AB E5 D7 A8 6F 71 50 EF 01
5A AB 0C E5 8F 8B 58 FC D1 5E DC 46 8C 6E 9A 52
22 F3 BD 53 07 68 E5 2C 2C B8 9A C6 8F 02 03 01
00 01

Public Key Type : RSA (1.2.840.113549.1.1.1)

Fingerprint : SHA1 :
34 CA DC C0 41 0D C5 23 1B EC CC 63 06 C4 46 B1
69 25 72 5A

Fingerprint : MD5 :
0F C7 0C 1D EA 0C B1 48 02 1D 50 09 44 31 83 A5

Fingerprint : SHA256 :
38 B9 32 3C 45 31 5A D1 4E 0B FD 6C 0E AE 98 A5
72 3E 42 1F 06 61 B4 4B E6 E0 27 B0 6D C0 2D 77

Extensions
SubjectKeyIdentifier

keyIdentifier:
AD 2E 0C 38 46 2E 69 F7 C7 75 1A 28 14 61 C9 C0
DE 02 A5 29
AuthorityKeyIdentifier

keyIdentifier:
AD 2E 0C 38 46 2E 69 F7 C7 75 1A 28 14 61 C9 C0
DE 02 A5 29

authorityIdentifier:

authorityCertSerialNumber:
Signature Algorithm : SHA1WithRSASignature (1.2.840.113549.1.1.5)
Value

51 92 97 C8 46 92 C2 17 77 B9 77 C2 79 D1 A1 AE
FF D4 1C 85 F9 F6 BB 95 C5 68 6F CA C8 02 32 E6
83 4C B9 AC DE 2B C7 DC C4 0F C4 4E 3F 35 66 DC
D3 E1 0F D3 45 F7 BD D7 B0 01 3F 80 78 1F 32 20
2B 15 4E 30 4D 08 D1 86 51 DF 70 73 92 C6 EE 36
2F 21 0F 11 10 9C 06 CD 52 BA B1 F4 00 43 79 81
89 5F 3F 6E A9 76 9E F7 14 FB D4 AB D9 C9 C8 28
78 05 7C 78 0E 33 4E C2 51 0F 84 55 0B 24 3B D6

Trust Status : Enabled

+ end excerpt

Export the certificate identifying bob to a file located in /tmp:

runmqakm -cert -extract -db /home/bob/.mqs/bobkey.kdb -pw passw0rd -label Bob_Cert -target /tmp/bob_public.arm

Allow the certificate to be read by others

chmod 644 /tmp/bob_public.arm

ls -l /tmp/*.arm

-rw-r--r--1 alice mqusers 782 Apr 22 11:06 /tmp/alice_public.arm
-rw-r--r--1 bob mqusers 778 Apr 22 11:13 /tmp/bob_public.arm

+ Window 2: User alice

Add the certificate for bob to alice's keystore:

```
runmqakm -cert -add -db /home/alice/.mqs/alicekey.kdb -pw passw0rd -label
Bob_Cert -file /tmp/bob_public.arm
ls -l /home/alice/.mqs
-rw-------1 alice mqusers 88 Apr 22 10:54 alicekey.crl
-rw-------1 alice mqusers 10088 Apr 22 11:14 alicekey.kdb
-rw-------1 alice mqusers 88 Apr 22 10:54 alicekey.rdb
-rw-------1 alice mqusers 129 Apr 22 10:54 alicekey.sth
-rw-r--r--1 alice mqusers 70 Apr 22 11:02 keystore.conf
```

Print the details

```
runmqakm -cert -details -db /home/alice/.mqs/alicekey.kdb -pw passw0rd -label
Bob_Cert
```

(Similar results as for Alice_Cert)

+ begin excerpt

```
Label : Bob_Cert
Key Size : 1024
Version : X509 V3
Serial : 64dc10c73a9ed1bf
Issuer : CN=bob,O=IBM,C=GB
Subject : CN=bob,O=IBM,C=GB
Not Before : April 21, 2017 11:01:20 AM EDT

Not After : April 22, 2018 11:01:20 AM EDT
```

+ end excerpt

++
+++ Chapter 7: Defining queue policy for AMS
++

Let's define protection policies using the “setmqspl” command.

Each policy name must be the same as the queue name it is to be applied to.

+ Window 1: User mqm
Example:
This is an example of a policy defined for the Q.AMS queue.
The messages are signed by the user alice using the SHA1 algorithm, and encrypted using the AES 256-bit algorithm.
The new MQ 9.0 attribute key reuse count “-c” is specified, but for now it is set to 0 (which is the default value, for backwards compatibility –
keys cannot be reused).
The user alice is the only valid sender and the user bob is the only receiver of the messages on this queue:

setmqspl -m QM_VERIFY_AMS -p Q.AMS -s SHA1 -a "CN=alice,O=IBM,C=GB" -e AES256 -r "CN=bob,O=IBM,C=GB" -c 0

Note: The DNs need to match exactly those specified in the receptive user's certificate from the key database.

Verify the policy:

dspmqspl -m QM_VERIFY_AMS

$ dspmqspl -m QM_VERIFY_AMS
Policy Details:
Policy name: Q.AMS
Quality of protection: PRIVACY
Signature algorithm: SHA1
Encryption algorithm: AES256
Signer DNs:
CN=alice,O=IBM,C=GB

Recipient DNs:
CN=bob,O=IBM,C=GB
Key reuse count: 0
Toleration: 0

You could also use runmqsc:

```
SET POLICY('Q.AMS') SIGNALG(SHA1) ENCALG(AES256) SIGNER('CN=alice,O=IBM,C=GB')
RECIP('CN=bob,O=IBM,C=GB') KEYREUSE(DISABLED) ENFORCE ACTION(REPLACE)
AMQ9084: IBM MQ Advanced Message Security policy set.

DISPLAY POLICY(*)
AMQ9086: Display IBM MQ Advanced Message Security policy details.

POLICY(Q.AMS) SIGNALG(SHA1)
ENCALG(AES256) SIGNER(CN=alice,O=IBM,C=GB)
RECIP(CN=bob,O=IBM,C=GB) KEYREUSE(DISABLED)
ENFORCE
```

++
+++ Chapter 8: Basic testing of the setup
++

Let's test the setup by putting a message as user alice and reading the message as user bob.

+ Window 2: User alice
As user alice, put a message using a sample application. Type the text of the message, then press Enter.

amqsput Q.AMS QM_VERIFY_AMS

Sample AMQSPUT0 start
target queue is Q.AMS
this is a test

Sample AMQSPUT0 end

+ Window 3: User bob
As user bob, get a message using a sample application:

amqsget Q.AMS QM_VERIFY_AMS

Sample AMQSGET0 start
message <this is a test>
no more messages
Sample AMQSGET0 end

Conclusion: User alice was able to put a message, and bob was able to read it.

++
+++ Chapter 9: Confirming the encryption of the messages at rest in the queue
++

To verify that the encryption is occurring as expected, create an alias queue which references the original queue Q.AMS.

This alias queue will have no security policy and so no user will have the information to decrypt the message and therefore the encrypted data will be shown.

+ Window 1: User mqm
Create an alias queue
runmqsc QM_VERIFY_AMS
  DEFINE QALIAS(TEST.ALIAS) TARGET(Q.AMS)
  end

Grant bob access to browse from the alias queue

setmqaut -m QM_VERIFY_AMS -n TEST.ALIAS -t queue -p bob +browse

+ Window 2: User alice
As user alice, put another message:
amqsput Q.AMS QM_VERIFY_AMS

+ Window 3: User bob
As user bob, browse the message via the alias queue:
amqsbcg TEST.ALIAS QM_VERIFY_AMS

The output from amqsbcg application shows the encrypted data that is on the queue proving that the message has been encrypted:

+ begin output
AMQSBCG0 -starts here
****
MQOPEN -'TEST.ALIAS'
MQGET of message number 1, CompCode:0 Reason:0
****Message descriptor****
StrucId : 'MD ' Version : 2
Report : 0 MsgType : 8
Expiry : -1 Feedback : 0
Encoding : 546 CodedCharSetId : 1208
Format : ' '
Priority : 0 Persistence : 0
MsgId : X'414D5120514D5F5645524946595F414DE265FB5897D66C25'
CorrelId : X'000000000000000000000000000000000000000000000000'
BackoutCount : 0

```
ReplyToQ : ' '
ReplyToQMgr : 'QM_VERIFY_AMS '
** Identity Context
UserIdentifier : 'alice '
AccountingToken :

X'0431303038000000000000000000000000000000000000000000000000000006'
ApplIdentityData : ' '
** Origin Context
PutApplType : '6'
PutApplName : 'amqsput '
PutDate : '20170422' PutTime : '15511382'
ApplOriginData : ' '

GroupId : X'000000000000000000000000000000000000000000000000'
MsgSeqNumber : '1'
Offset : '0'
MsgFlags : '0'
OriginalLength : '-1'

**** Message ****

length -1310 of 1310 bytes

00000000: 5044 4D51 0200 0200 7000 0000 7000 0000 'PDMQ....p...p...'
00000010: 0800 0000 B804 0000 0F00 0000 0000 0000 '................'
00000020: 4D51 5354 5220 2020 0000 0000 0000 0000 'MQSTR ........'
00000030: 0000 0000 0000 0000 2020 2020 2020 2020 '........ '
00000040: 2020 2020 2020 2020 2020 2020 2020 2020 ' '
00000050: 2020 2020 2020 2020 2020 2020 2020 2020 ' '
00000060: 2020 2020 2020 2020 0000 0000 0000 0000 ' ........'
00000070: 3082 04AA 0609 2A86 4886 F70D 0107 03A0 '0.....*.H.......'
00000080: 8204 9B30 8204 9702 0100 3181 CF30 81CC '...0......1..0..'
…

00000500: 26E2 A40E A81E BC7A 0315 0B7B 4679 F833 '&......z...{Fy.3'
00000510: 5258 34EA E264 43F6 6BAE 3006 D4E8 'RX4..dC.k.0... '

No more messages
MQCLOSE

+ end output
```

++
+++ Chapter 10: Advanced testing
++

+++ Scenario A: not authorized by AMS to view messages

Let's explore what happens when other users, who are not authorized explicitly to use the queues protected by AMS, try to view the messages.

+ Window 2: User alice
As user alice, put a message using a sample application. Type the text of the message, then press Enter.

amqsput Q.AMS QM_VERIFY_AMS

Sample AMQSPUT0 start
target queue is Q.AMS
this is another test
Sample AMQSPUT0 end

+ Window 1: User fulano
Log in as user fulano and ensure to set up the environment for using MQ 9.0:

. set-mq-90

Try to put, browse or get a message from the queue. These actions will fail.

Even though the setmqaut was given for user fulano to get messages from the queue Q.AMS, the AMS policies do not include user fulano as an authorized user:

amqsput Q.AMS QM_VERIFY_AMS

Sample AMQSPUT0 start
target queue is Q.AMS
MQOPEN ended with reason code 2035
unable to open queue for output
Sample AMQSPUT0 end

amqsbcg Q.AMS QM_VERIFY_AMS
AMQSBCG0 -starts here
****
MQOPEN -'Q.AMS'
MQOPEN ended with reason code 2035

amqsget Q.AMS QM_VERIFY_AMS
Sample AMQSGET0 start
MQOPEN ended with reason code 2035
unable to open queue for input
Sample AMQSGET0 end

Notice that the reason code is 2035. You can use the following MQ command to get the short name for a reason code, in order to get a rough idea of that the problem is:

mqrc 2035
2035 0x000007f3 MQRC_NOT_AUTHORIZED

+ Window 1: User mqm
Log in as user mqm
As user mqm try to browse the message:
amqsbcg Q.AMS QM_VERIFY_AMS

AMQSBCG0 -starts here
****
MQOPEN -'Q.AMS'
MQOPEN failed with CompCode:2, Reason:2035

NOTE:
The user mqm, even though it is an MQ administrator, is NOT authorized to read the messages.

+ Error messages in the queue manager error log
Let's look at the error messages in the queue manager error log:
cd /var/mqm/qmgrs/QM_VERIFY_AMS/errors
tail AMQERR01.LOG

We will see the security errors for both users: fulano and mqm

+ begin excerpt
04/22/2017 11:55:46 AM -Process(25010.1) User(fulano) Program(amqsput)
Host(mosquito) Installation(Installation3)
VRMF(9.0.0.0) QMgr(QM_VERIFY_AMS)

AMQ9062: The IBM MQ security policy interceptor could not read the keystore configuration file: /home/fulano/.mqs/keystore.conf.
EXPLANATION:
The IBM MQ security policy interceptor could not read the keystore configuration file: /home/fulano/.mqs/keystore.conf.
ACTION:
Make sure that the user who executes the IBM MQ application has permissions to read the configuration file. Check if the configuration file is not corrupted

or empty. If the problem persists, contact your local IBM service representative.

04/22/2017 11:57:15 AM -Process(25014.1) User(mqm) Program(amqsbcg) Host(mosquito) Installation(Installation3)
VRMF(9.0.0.0) QMgr(QM_VERIFY_AMS)

AMQ9062: The IBM MQ security policy interceptor could not read the keystore configuration file: /home/mqm/.mqs/keystore.conf.
EXPLANATION:
The IBM MQ security policy interceptor could not read the keystore configuration file: /home/mqm/.mqs/keystore.conf.

+ end excerpt

Conclusions:
-Only users alice and bob, who are fully authorized to put/get messages in the Q.AMS are allowed to put and get messages.
-Not even the user “mqm”, who is MQ administrator is able to browse, put or get messages from the protected queue Q.AMS

+++ Scenario B: User alice is not authorized by AMS to read messages signed by bob

Only one AMS policy has been created for this technical document.
In this policy the user "alice" was explicitly indicated as a "signer" and user "bob" was indicated as a "reader".

Now, let's explore the following scenario, which is NOT covered by the above policy: the user "bob" puts a message as a signer and user "alice" tries to read it.
Because there is no explicit policy for this case, the error message that we get will be 2063:

2063 0x0000080f MQRC_SECURITY_ERROR

Window 3 (bob)

As user bob put a message into Q.AMS. This is successful.
The message is encrypted and placed encrypted in the queue.

```
$ amqsput Q.AMS QM_VERIFY_AMS
Sample AMQSPUT0 start
target queue is Q.AMS
testing
Sample AMQSPUT0 end
```

Window 2 (alice)

As user alice try to browse message from Q.AMS.
This is not successful.

```
$ amqsbcg Q.AMS QM_VERIFY_AMS
AMQSBCG0 -starts here
**********************
MQOPEN -'Q.AMS'
MQGET 1, failed with CompCode:2 Reason:2063
MQCLOSE
```

The reason code 2063 means: MQRC_SECURITY_ERROR

It is necessary to view the queue manager error log to get more details.
The last item in the EXPLANAION section, number 4, is the one that applies to this situation:

(4) receiver is not among the recipients of the message.

04/22/2017 12:11:19 PM -Process(25080.1) User(alice) Program(amqsbcg)
Host(mosquito) Installation(Installation3)
VRMF(9.0.0.0) QMgr(QM_VERIFY_AMS)
AMQ9017: IBM MQ security policy internal error: message could not be unprotected: GSKit error code 851968, reason 43.
EXPLANATION:
The IBM MQ security policy interceptor could not verify or decrypt a message because the indicated GSKit error occurred. This can happen for several reasons, all of which are internal failures:
(1) the message is not a valid PKCS#7 message;
(2) the sender's certificate does not have the required key usage bit to be able to encrypt the message;
(3) the sender's certificate was not recognized as a trusted certificate;
(4) receiver is not among the recipients of the message.
ACTION:
Consult the GSKit information in the Information Center for the explanation of the GSKit reason code and take corrective action. If the problem persists, contact your IBM service representative.

+++ Scenario C: User bob is not authorized by AMS to read messages signed by bob

As mentioned in the previous scenario in this chapter, only one AMS policy has been created for this technical document.
In this policy the user "alice" was explicitly indicated as a "signer" and user "bob" was indicated as a "reader".

Now, let's explore the following scenario, which is NOT covered by the above policy: the user "bob" puts a message as a signer and the same user "bob" tries to read it. Because there is no explicit policy for this case, the error message that we get will be 2063:

2063 0x0000080f MQRC_SECURITY_ERROR

This may seem a bit strange: unless there is a policy in place, user bob CANNOT browse the encrypted messages generated by himself!

Window 3 (bob)

As user bob put a message into Q.AMS. This is successful.
The message is encrypted and placed encrypted in the queue.

```
$ amqsput Q.AMS QM_VERIFY_AMS
Sample AMQSPUT0 start
target queue is Q.AMS
testing
Sample AMQSPUT0 end
```

Now, again as user bob, try to browse the message:

```
$ amqsbcg Q.AMS QM_VERIFY_AMS
AMQSBCG0 -starts here
**********************
MQOPEN -'Q.AMS'
MQGET 1, failed with CompCode:2 Reason:2063
```

Let's take a look at the queue manager error log to get more details:

```
AMQ9035: Message signer is not in the list of authorised signers.
EXPLANATION:
The MQ security policy interceptor detected that the message is
signed by an unauthorised party.
ACTION:
Establish whether the identity associated with the sender of the message is
authorized to send messages to this application. Ensure the sender is named in
the list of allowed signers on the security policy for the queue.
```

++
+++ Chapter 11: Testing performance improvement of new feature in MQ 9.0
++

The objective of this chapter is to provide you with a rough comparison of 2 scenarios, one which is used a baseline and the other which exploits a new option for AMS added in MQ 9.0 to improve performance.

++ Reference

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.pro.doc/q113120_.htm#q113120___amsprot
IBM MQ > IBM MQ 9.0.x > IBM MQ > Product overview > What's new and changed in IBM MQ Version 9.0 > What's new in Version 9.0.0
New family features

Additional quality of protection for AMS

+ begin excerpt

To complement the existing Integrity and Privacy privacy policies, Advanced Message Security (AMS) provides a new, third alternative, Confidentiality (Encryption only with optional key reuse), in IBM MQ Version 9.0.
Significant CPU cost savings can be made with Confidentiality policies through symmetric key reuse. This new mode of operation continues to use the PKCS#7 format to share a symmetric encryption key. However, there is no digital signature, which eliminates some of the per message asymmetric key operations. The symmetric key still needs to be encrypted with asymmetric key operations for each recipient, but the symmetric key can be optionally reused over multiple messages that are destined for the same recipients. If key reuse is permitted by policy, then only the first message requires asymmetric key operations. Subsequent messages only need to use symmetric key operations. For more information, see Qualities of protection available with AMS.

+ end excerpt

https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_9.0.0/com.ibm.mq.sec.doc/q127085_.htm
IBM MQ 9.0.x / IBM MQ / Securing / Advanced Message Security / AMS overview / Qualities of protection available with AMS

++ Scenarios

The MQ sample “amqsblst” (also called “Blast”) will be used to test putting/getting a large quantity of messages (10,000) into the queue.

In Unix the following 3 commands were used. Note that “date” in Unix displays both the date and time.

Blast putting 10000 messages of size 2K queue Q1 on queue manager
date; amqsblst QM_VERIFY_AMS Q1 -W -c 10000 -s 2048; date

The important line from the execution is the one that shows the “elapsed time”.

Blast> elapsed time = 0.142514 S

The objective of this scenario is to take measurements of the time that it takes to perform the tasks mentioned in the table below.

Queue Name Protected by KeyReuse Time to put Time to get
By AMS 10k messages 10k messages
Q1 No not applicable 0.097445 S 0.112199 S
Q.AMS Yes 0 (default) 7.542336 S 12.026407 S
Q.AMS Yes 50 0.189219 S 0.290232 S
Conclusion:

The new feature for AMS provides a faster response when using AMS protected queues.

+ Window 1 (mqm)
As MQ administrator alter the maximum amount of messages that can be held. The default is 5,000 which is a bit short for this type of test.

alter ql(Q1) MAXDEPTH(11000)
alter ql(Q.AMS) MAXDEPTH(11000)

Notice that:
-Q1 is NOT protected by AMS.
-Q.AMS is protected by AMS.

++ Baseline test for PUT/GET, using queue Q1 (not protected by AMS)

+ PUT 10,000 messages
Blast putting 10000 messages of size 2K queue Q1

```
mqm@mosquito: /home/mqm
$ date; amqsblst QM_VERIFY_AMS Q1 -W -c 10000 -s 2048; date
Tue Apr 25 07:30:47 EDT 2017
welcome to blast
Blast> successfully opened queue <Q1>
Blast> 10000 messages sent

Blast> elapsed time = 0.097445 S

Blast> ended
Blast> 10000 messages have been put
Blast> 0 messages have been got
Tue Apr 25 07:30:47 EDT 2017
```

+ GET 10,000 messages
Blast getting 10000 messages of size 2K queue Q1

```
mqm@mosquito: /home/mqm
$ date; amqsblst QM_VERIFY_AMS Q1 -R; date
Tue Apr 25 07:34:12 EDT 2017
welcome to blast
Blast> successfully opened queue <Q1>
Blast> 100 messages received
Blast> 200 messages received

…


Blast> 9900 messages received
Blast> 10000 messages received

Blast> elapsed time = 0.112199 S

Blast> ended
Blast> 0 messages have been put
Blast> 10000 messages have been got
Tue Apr 25 07:34:12 EDT 2017
```

++ Test 1: PUT/GET using queue Q.AMS (protected by AMS), KeyCount=0 (default)

Queue Q.AMS is protected by AMS by the policy that uses:
signature algorithm SIGNALG(SH1) and key reuse KEYREUSE(DISABLED)

Line command:

setmqspl -m QM_VERIFY_AMS -p Q.AMS -s SHA1 -a "CN=alice,O=IBM,C=GB" -e AES256 -r "CN=bob,O=IBM,C=GB" -c 0

Under runmqsc:

DISPLAY POLICY(*)

AMQ9086: Display IBM MQ Advanced Message Security policy details.
POLICY(Q.AMS) SIGNALG(SHA1)
ENCALG(AES256) SIGNER(CN=alice,O=IBM,C=GB)
RECIP(CN=bob,O=IBM,C=GB) KEYREUSE(DISABLED)
ENFORCE

+ Window 2: alice -PUT 10,000 messages
Blast putting 10000 messages of size 2K queue Q1

alice@mosquito:~> date; amqsblst QM_VERIFY_AMS Q.AMS -W -c 10000 -s 2048; date
Tue Apr 25 08:00:15 EDT 2017
welcome to blast
Blast> successfully opened queue <Q.AMS>
Blast> 10000 messages sent

Blast> elapsed time = 7.542336 S

Blast> ended
Blast> 10000 messages have been put
Blast> 0 messages have been got
Tue Apr 25 08:00:23 EDT 2017

+ Window 3: bob -GET 10,000 messages
Blast getting 10000 messages of size 2K queue Q1

bob@mosquito:~> date; amqsblst QM_VERIFY_AMS Q.AMS -R; date
Tue Apr 25 08:01:40 EDT 2017
welcome to blast
Blast> successfully opened queue <Q.AMS>
Blast> 100 messages received
Blast> 200 messages received

…

Blast> 10000 messages received

Blast> elapsed time = 12.026407 S

Blast> ended
Blast> 0 messages have been put
Blast> 10000 messages have been got
Tue Apr 25 08:01:52 EDT 2017

++ Test 2: PUT/GET using queue Q.AMS (protected by AMS), KeyCount=50

Queue Q.AMS is protected by AMS by the policy that uses:
signature algorithm SIGNALG(NONE) and key reuse KEYREUSE(50)
Line command:

setmqspl -m QM_VERIFY_AMS -p Q.AMS -s NONE -e AES256 -r "CN=bob,O=IBM,C=GB" -c 50

Under runmqsc:

display policy(*)

1 : display policy(*)
AMQ9086: Display IBM MQ Advanced Message Security policy details.
POLICY(Q.AMS) SIGNALG(NONE)
ENCALG(AES256) RECIP(CN=bob,O=IBM,C=GB)
KEYREUSE(50) ENFORCE

+ Window 2: alice -PUT 10,000 messages
Blast putting 10000 messages of size 2K queue Q1

alice@mosquito:~> date; amqsblst QM_VERIFY_AMS Q.AMS -W -c 10000 -s 2048; date
Tue Apr 25 07:54:53 EDT 2017
welcome to blast
Blast> successfully opened queue <Q.AMS>
Blast> 10000 messages sent

Blast> elapsed time = 0.189219 S

Blast> ended
Blast> 10000 messages have been put
Blast> 0 messages have been got
Tue Apr 25 07:54:53 EDT 2017

+ Window 3: bob -GET 10,000 messages
Blast getting 10000 messages of size 2K queue Q1

bob@mosquito:~> date; amqsblst QM_VERIFY_AMS Q.AMS -R; date
Tue Apr 25 07:56:25 EDT 2017
welcome to blast
Blast> successfully opened queue <Q.AMS>
Blast> 100 messages received
…
Blast> 9900 messages received

Blast> elapsed time = 0.290232 S

Blast> ended
Blast> 0 messages have been put
Blast> 9999 messages have been got
Tue Apr 25 07:56:25 EDT 2017

++
+++ Chapter 12: Basic troubleshooting information
++

a) If you are using a remote MQ Client for AMS activities and the return code is related to security, such as 2035 (MQRC_NOT_AUTHORIZED), then you should check the general error log for the remote machine:
/var/mqm/errors/AMQERR01.LOG

b) It is a best practice for queue managers to have a Dead Letter Queue (DLQ) enabled, which can be used for routing messages that could not be delivered to the desired destination queue.
By default, the queue manager does not use any DLQ, but the following will enable the use of the queue SYSTEM.DEAD.LETTER.QUEUE (which is created when a queue manager is created) to serve as a DLQ:
ALTER QMGR DEADQ(SYSTEM.DEAD.LETTER.QUEUE)

c) When using MQ AMS, the code uses another queue which is similar to the DLQ in nature: the AMS code will route messages that failed to meet the security requirements for the destination queue.
This queue is named: SYSTEM.PROTECTION.ERROR.QUEUE

d) Thus, an AMS queue manager can use 2 different DLQs:
- SYSTEM.DEAD.LETTER.QUEUE => for messages that could not be delivered to the destination queue (for reasons not related to AMS).
- SYSTEM.PROTECTION.ERROR.QUEUE => for messages that failed to meet security requirements.

e) For more details on how to find out the reason code for which a message is sent to an DLQ see:

Handling undelivered messages in MQ 7: Dead Letter Queue, Poison Messages
http://www-01.ibm.com/support/docview.wss?uid=swg27039569
This WSTE discusses practical information on how to handle undelivered messages: Dead Letter Queue, Poison Messages. It also discusses the configuration for handling poison messages by the MQ JMS provider in WebSphere Application Server.
Level of Difficulty: Intermediate
- The steps to find the reason code for the message to be sent to the DLQ are described in pages 21 thru 30 in the PDF file.
- The discussion on the Dead Letter Queue handler (runmqdlq) is in pages 43 thru 47.

f) For more details see the following page from the online manual for MQ 9.1:
https://www.ibm.com/support/knowledgecenter/SSFKSJ_9.1.0/com.ibm.mq.sec.doc/q014595_.htm
IBM MQ 9.1.x / IBM MQ / Securing / Advanced Message Security / Overview of Advanced Message Security /
Error handling
.
+ begin excerpt
.
IBM® MQ Advanced Message Security defines an error handling queue to manage messages that contain errors or messages that cannot be unprotected.

Defective messages are dealt with as exceptional cases. If a received message does not meet the security requirements for the queue it is on, for example, if the message is signed when it should be encrypted, or decryption or signature verification fails, the message is sent to the error handling queue. A message might be sent to the error handling queue for the following reasons:

- Quality of protection mismatch - a quality of protection (QOP) mismatch exists between the received message and the QOP definition in the security policy.
- Decryption error - the message cannot be decrypted.
- PDMQ header error - the Advanced Message Security (AMS) message header cannot be accessed.
- Size mismatch - length of a message after decryption is different than expected.
- Encryption algorithm strength mismatch - the message encryption algorithm is weaker than required.
- Unknown error - unexpected error occurred.

AMS uses the SYSTEM.PROTECTION.ERROR.QUEUE as its error handling queue.
All messages put by IBM MQ AMS to the SYSTEM.PROTECTION.ERROR.QUEUE are preceded by an MQDLH header.

Your IBM MQ administrator can also define the SYSTEM.PROTECTION.ERROR.QUEUE as an alias queue pointing to another queue.
.
+ end excerpt

g) Example:
.
In Linux host "mosquito", which uses MQ 9.0 with AMS.
In one test scenario that failed, a message was sent to the SYSTEM.PROTECTION.ERROR.QUEUE
.
mqm@mosquito: /home/mqm
$ runmqsc QM_VERIFY_AMS
.

```
display qstatus(SYSTEM.PROTECTION.ERROR.QUEUE)
     2 : display qstatus(SYSTEM.PROTECTION.ERROR.QUEUE)
AMQ8450: Display queue status details.
   QUEUE(SYSTEM.PROTECTION.ERROR.QUEUE)    TYPE(QUEUE)
   CURDEPTH(1)
```

Using amqsbcg to browse that message:

```
$ amqsbcg SYSTEM.PROTECTION.ERROR.QUEUE QM_VERIFY_AMS

AMQSBCG0 - starts here
**********************
 MQOPEN - 'SYSTEM.PROTECTION.ERROR.QUEUE'
 MQGET of message number 1, CompCode:0 Reason:0
****Message descriptor****
  StrucId  : 'MD  '  Version : 2
  Report   : 0  MsgType : 8
  Expiry   : -1  Feedback : 0
  Encoding : 546  CodedCharSetId : 1208
  Format : 'MQDEAD  '

...

****   Message      ****

 length - 2694 of 2694 bytes

00000000:  444C 4820 0100 0000 0F08 0000 512E 414D          'DLH ........Q.AM'
00000010:  5300 0000 0000 0000 0000 0000 0000 0000          'S...............'
00000020:  0000 0000 0000 0000 0000 0000 0000 0000          '................'
00000030:  0000 0000 0000 0000 0000 0000 514D 5F56          '............QM_V'
00000040:  4552 4946 595F 414D 5320 2020 2020 2020          'ERIFY_AMS       '
00000050:  2020 2020 2020 2020 2020 2020 2020 2020          '                '
00000060:  2020 2020 2020 2020 2020 2020 2202 0000          '            "...'
00000070:  B804 0000 2020 2020 2020 2020 0600 0000          '....        ....'
00000080:  616D 7173 626C 7374 2020 2020 2020 2020          'amqsblst        '
00000090:  2020 2020 2020 2020 2020 2020 3230 3137          '            2017'
000000A0:  3034 3232 3136 3433 3234 3436 5044 4D51          '042216432446PDMQ'
000000B0:  0300 0200 7000 0000 7000 0000 6000 0000          '....p...p...`...'
000000C0:  B804 0000 0008 0000 4A01 0000 4D51 5354          '........J...MQST'
...
.
To interpret the reason code for sending the message into this queue, get the values for the
bytes 9 thru 12:
00000000:  444C 4820 0100 0000 0F08 0000 512E 414D          'DLH ........Q.AM'
                          *********
```

The desired bytes are:
0F08 0000
.
Because the host used in this example is based on the Intel architecture, it is necessary to reverse the byte order:
0F08 0000 => 0000 080F
.
The value is in hex and you can use the MQ utility "mqrc" to get an idea of the reason code:
mqrc 0x0000080F
.
```
$ mqrc 0x0000080F
   2063  0x0000080f  MQRC_SECURITY_ERROR
```
.
I noticed too that there was an entry in the error log of the queue manager.
.
```
04/22/2017 12:43:39 PM - Process(25211.1) User(bob) Program(amqsblst)
                Host(mosquito) Installation(Installation3)
                VRMF(9.0.0.0) QMgr(QM_VERIFY_AMS)
AMQ9035: Message signer is not in the list of authorised signers.
EXPLANATION:
The IBM MQ security policy interceptor detected that the message is signed by
an unauthorised party.
ACTION:
Establish whether the identity associated with the sender of the message is
authorized to send messages to this application. Ensure the sender is named in
the list of allowed signers on the security policy for the queue.
----- smqigeta.c : 571 --------------------------------------------------
04/22/2017 12:43:39 PM - Process(25211.1) User(bob) Program(amqsblst)
                Host(mosquito) Installation(Installation3)
                VRMF(9.0.0.0) QMgr(QM_VERIFY_AMS)
AMQ9044: The IBM MQ security policy interceptor has put a defective message on
error handling queue SYSTEM.PROTECTION.ERROR.QUEUE.
EXPLANATION:
This is an informational message that indicates the IBM MQ security policy put
a message it could not interpret onto the specified error handling queue, or
returned the message to the original queue if the MQGET of the message was part
of a Unit of Work.
ACTION:
Make sure only valid messages are put onto queues protected by IBM MQ security
policies.
```

+ Scenario: User alice deletes bob_cert from keystore

That is, what happens if user “alice” does not have the public certificate for the user who is going to read the message?

We have tested the scenario in which “bob_cert” is in the keystore and the put of the message was successful.

Now let’s proceed to list the certificates and remove “bob_cert”

```
alice@florencia1.fyre.ibm.com: /home/alice
runmqckm -cert  -list    -db /home/alice/.mqs/alicekey.kdb -pw passw0rd
```

Notice that the syntax for -details and -delete is the same.
Because we have used the -details before, here it is:

```
runmqakm -cert -details -db /home/alice/.mqs/alicekey.kdb  -pw passw0rd -label bob_cert
```

We just have to replace “-details” and use “-delete”:

```
runmqakm -cert  -delete  -db /home/alice/.mqs/alicekey.kdb -pw passw0rd -label bob_cert
```

Let’s try to put a message.
Notice that it will fail:

```
amqsput Q.AMS QM_VERIFY_AMS
Sample AMQSPUT0 start
target queue is Q.AMS
MQOPEN ended with reason code 2063
unable to open queue for output
Sample AMQSPUT0 end
```

The reason code 2063 is like 2035, in the sense that it is super vague and does not provide more details, because it has many possible causes:

```
mqrc 2063
     2063  0x0000080f  MQRC_SECURITY_ERROR
```

The MQ Administrator needs to look at the error log of the queue manager to find more details:

```
mqm@florencia1.fyre.ibm.com: /var/mqm/qmgrs/QM_VERIFY_AMS/errors
$ tail -20  AMQERR01.LOG
----- smqigeta.c : 2607 --------------------------------------------------------
05/01/2022 12:28:01 PM - Process(8985.1) User(alice) Program(amqsput)
                    Host(florencia1.fyre.ibm.com) Installation(Installation1)
                    VRMF(9.2.5.0) QMgr(QM_VERIFY_AMS)
                    Time(2022-05-01T19:28:01.571Z)
                    ArithInsert1(57)
                    CommentInsert1(CN=bob,O=IBM,C=GB)

AMQ9021E: An error occured during the certificate import for the following DN:
CN=bob,O=IBM,C=GB, result: 57

EXPLANATION:
The distinguished name is not present in the keystore or invalid.
ACTION:
Consult the GSKit appendix in the Information Center for the explanation of the
GSKit reason code and take corrective action. If the problem persists, contact
your IBM service representative.


+++ end +++
```