IBM MQ Security: Overview & recap

Rob Parker

Important Disclaimer

- THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.
- WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.
- IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE.
- IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.
- NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:
  - CREATING ANY WARRANTY OR REPRESENTATION FROM IBM (OR ITS AFFILIATES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS); OR
  - ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF IBM SOFTWARE.
- Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

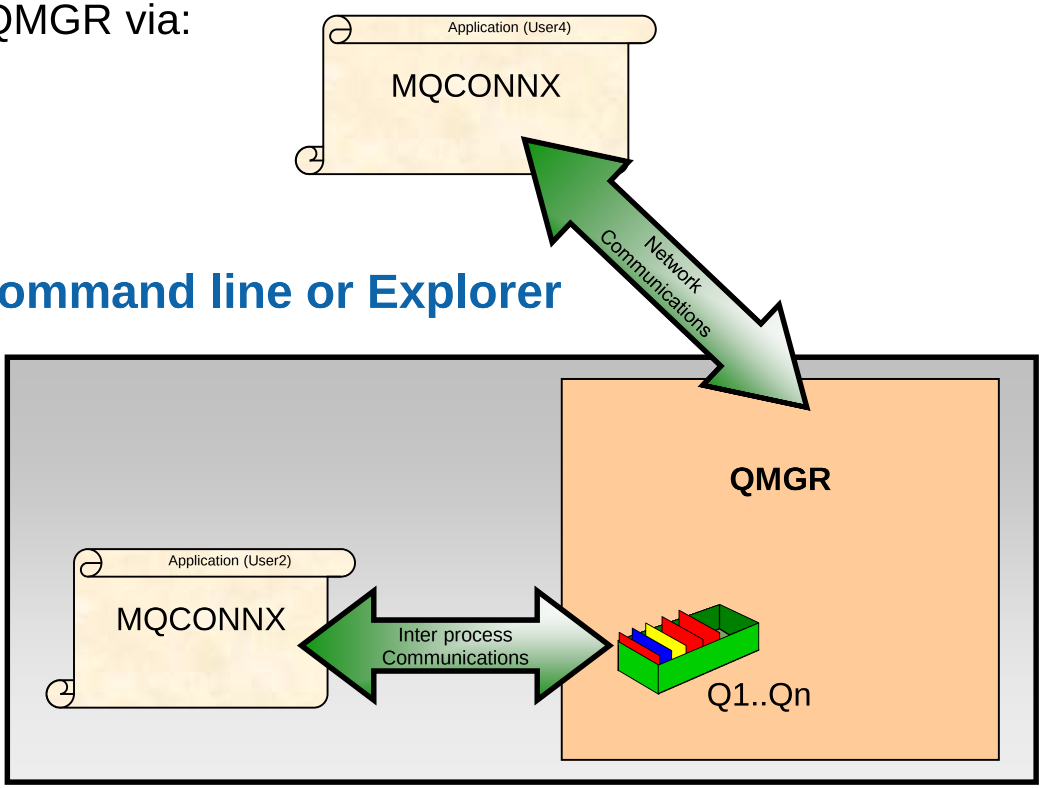
Agenda

- Introduction
- Connection Authentication
- Authorization
- Transport Layer Security
- Channel Authentication
- Security Exits
- AMS

Introduction – Typical MQ

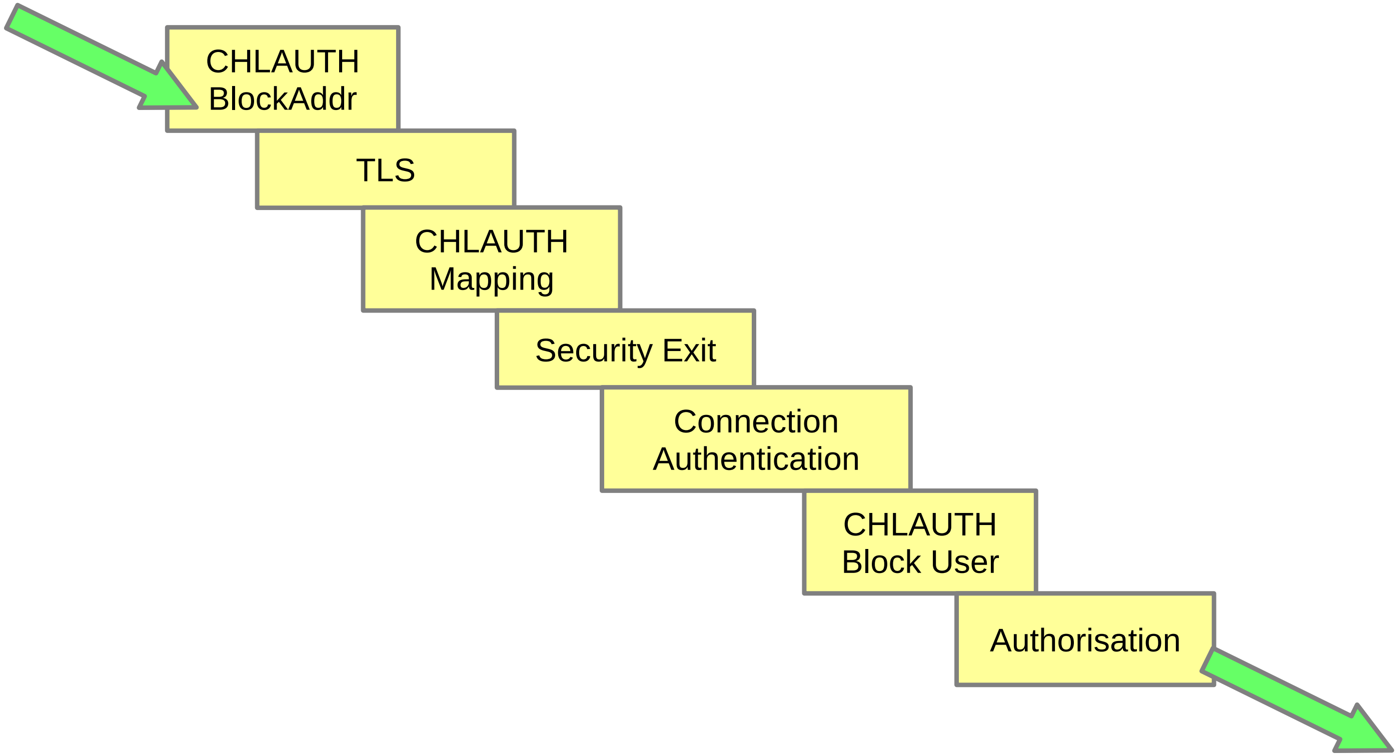
- **In a Typical MQ setup there is:**
  - A Queue Manager (QMGR)
  - A number of Queues
  - Applications that connect to the QMGR via:
    - Local Bindings
    - Client connections

- **Configuration is updated via Command line or Explorer**



Introduction – Security Checks (Client)

- **When a user connects via client:**



CHLAUTH BlockAddr

TLS

CHLAUTH Mapping

Security Exit

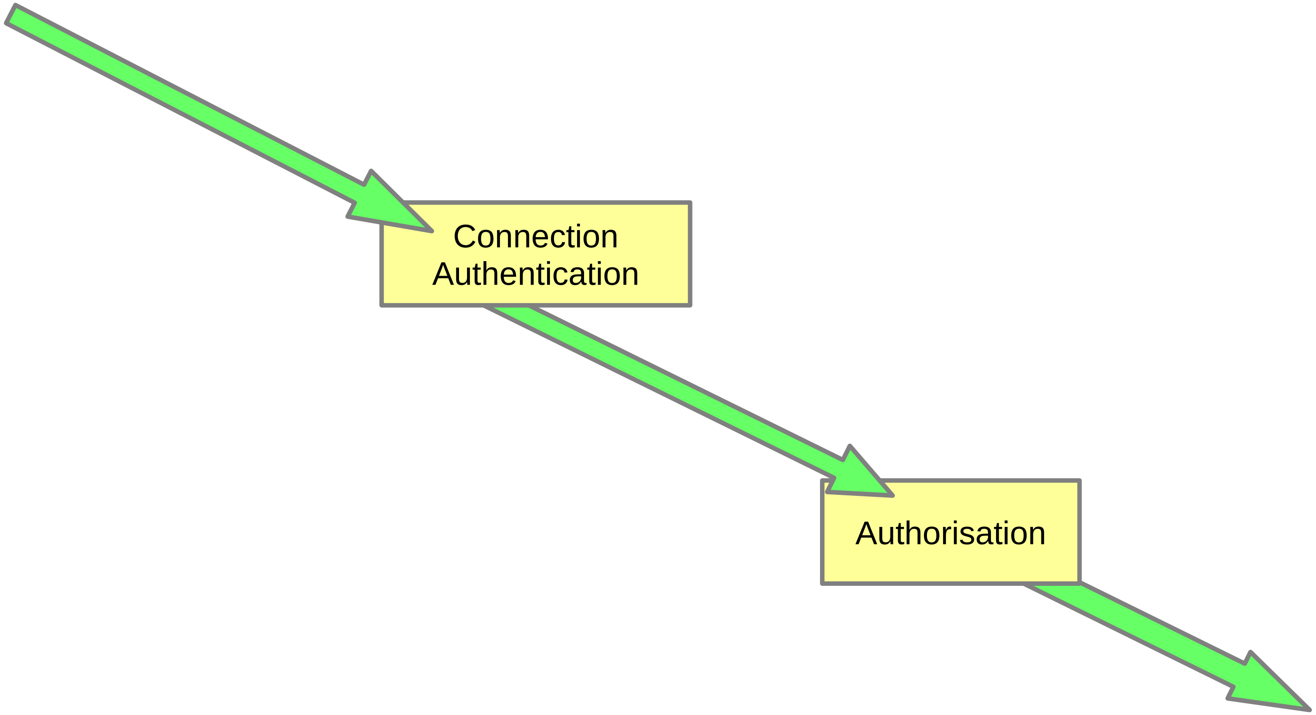
Connection Authentication

CHLAUTH Block User

Authorisation

Introduction - Security Checks (Local)

- **When a user connects via local bindings:**



Connection Authentication

Authorisation

AUTHENTICATION

Details

- **We use Authentication to ask clients connecting to prove they are who they say they are.**
  - Usually used in combination with authorisation to limit user's abilities.

- **Connection authentication feature available in MQ v8 and above.**
  - Allows authentication user credentials supplied by client applications.
  - User credentials can be local OS users or LDAP users.
  - A failure to authenticate results in a MQRC_NOT_AUTHORIZED 2035 error being returned.

- **IBM MQ now can send two different userids in the connection data.**
  - The userid that is running the application.
  - The userid and password that the application wants to authenticate with.

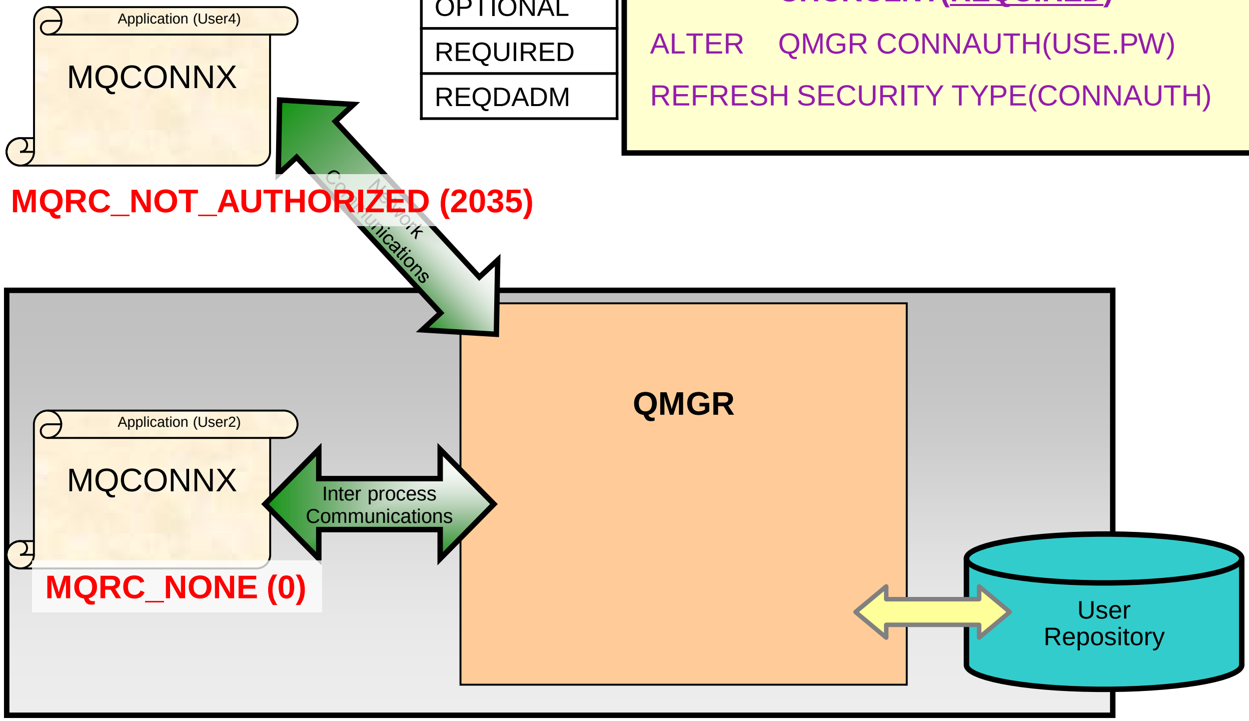
Configuration

CHCK…
NONE
OPTIONAL
REQUIRED
REQDADM

```
DEFINE   AUTHINFO(USE.PW) AUTHTYPE(xxxxxx)
         CHCKLOCL(OPTIONAL)
         CHCKCLNT(REQUIRED)

ALTER    QMGR CONNAUTH(USE.PW)

REFRESH SECURITY TYPE(CONNAUTH)
```



Application (User4)

MQCONNX

Network Communications

MQRC_NOT_AUTHORIZED (2035)

QMGR

Application (User2)

MQCONNX

Inter process Communications

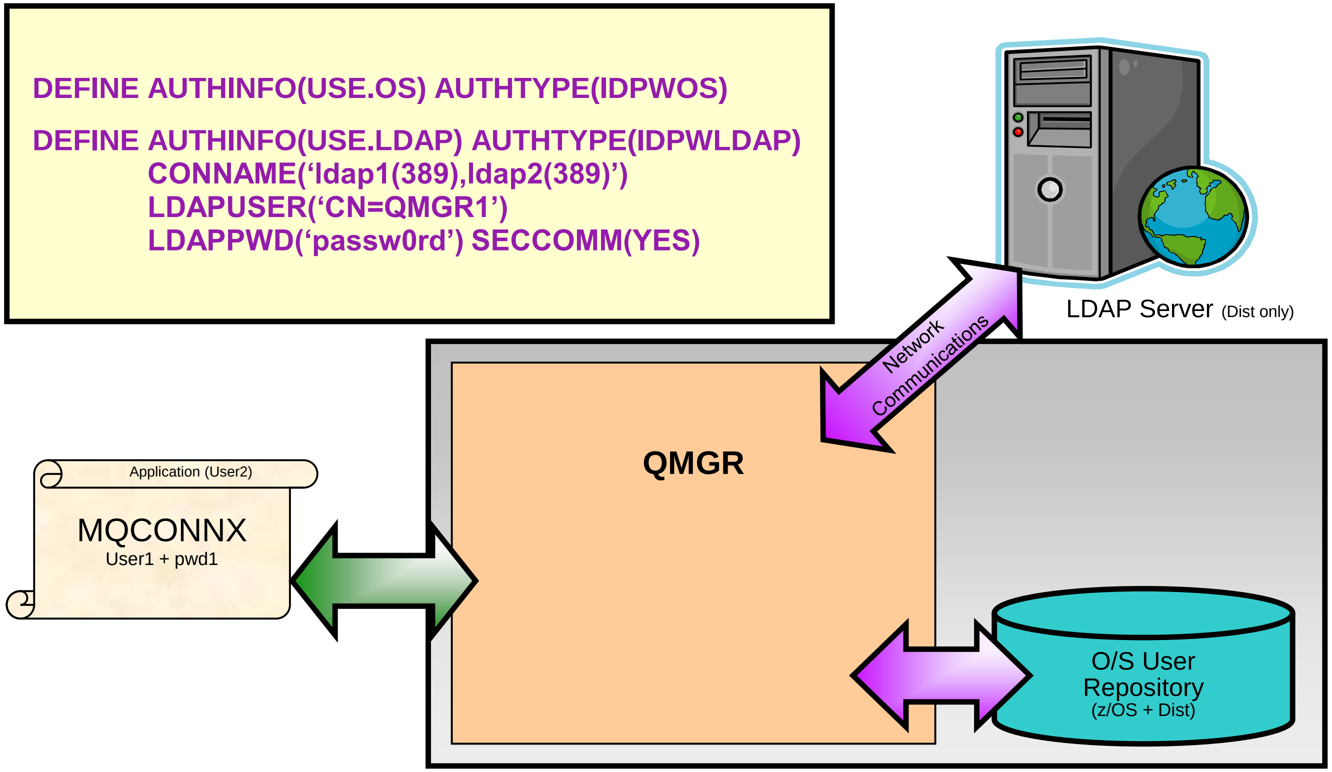
MQRC_NONE (0)

User Repository

Configuration

```
DEFINE AUTHINFO(USE.OS) AUTHTYPE(IDPWOS)

DEFINE AUTHINFO(USE.LDAP) AUTHTYPE(IDPWLDAP)
        CONNAME('ldap1(389),ldap2(389)')
        LDAPUSER('CN=QMGR1')
        LDAPPWD('passw0rd') SECCOMM(YES)
```



AUTHORIZATION

Details

- **We use Authorization to limit what connected users can and cannot do.**

- **This is performed by creating authority records**
  - We create authority records for a specific user or group.
  - User level authority records not available on Linux.

- **Authority is given on MQ objects and dictate what actions they can performed (PUT, GET, OPEN, etc)**

- **If a user or group does not have authority to do what they are trying to do, they get blocked.**
  - MQRC_NOT_AUTHORIZED (2035)
  - Users who are members of the mqm group have full administrator access.

- **A channel or channel authentication rule can change the userid used for authority checks**

Which user will be used for authorization?

Method	Notes
Client machine user ID flowed to server	This will be over-ridden by anything else. Rarely do you want to trust an unauthenticated client side user ID.
MCAUSER set on SVRCONN channel definition	A handy trick to ensure that the client flowed ID is never used is to define the MCAUSER as ‘rubbish’ and then anything that is not set appropriately by one of the next methods cannot connect.
MCAUSER set by ADOPTCTX(YES)	The queue manager wide setting to adopt the password authenticated user ID as the MCAUSER will over-ride either of the above.
MCAUSER set by CHLAUTH rule	To allow more granular control of MCAUSER setting, rather than relying on the above queue manager wide setting, you can of course use CHLAUTH rules
MCAUSER set by Security Exit	Although CHLAUTH gets the final say on whether a connection is blocked (security exit not called in that case), the security exit does get called with the MCAUSER CHLAUTH has decided upon, and can change it.

Configuration

- **Authority records are created or modified using one of three tools:**

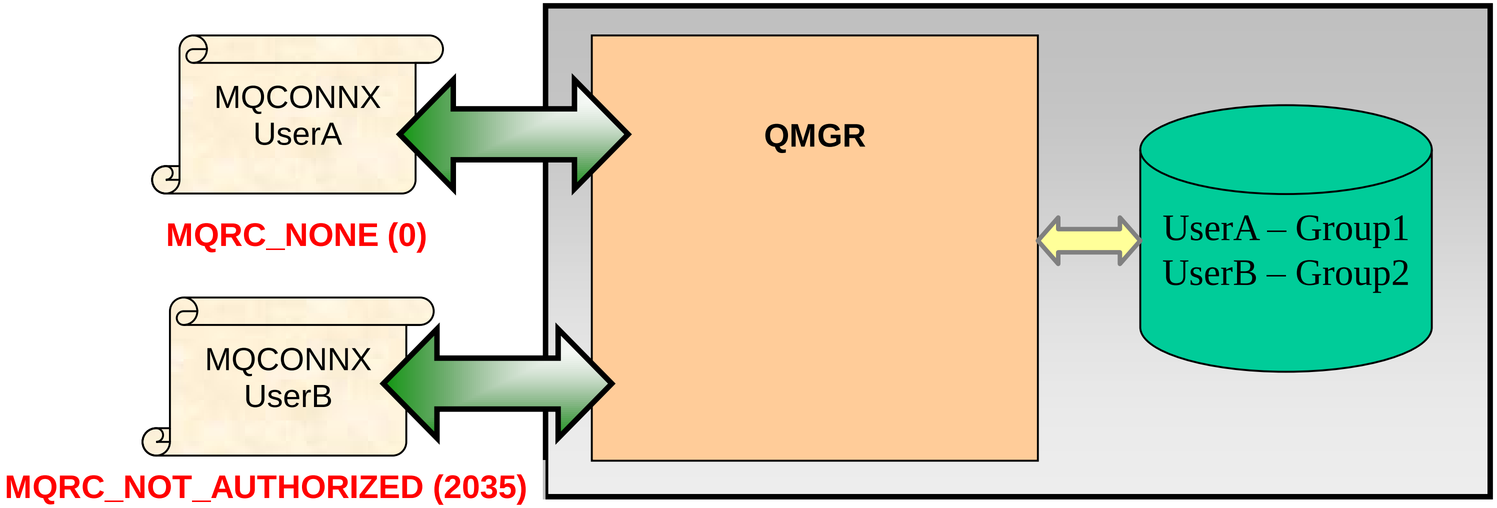
- **runmqsc**
  - `SET AUTHREC(<Object name>) OBJTYPE(<Object type>) GROUP(<group name>) AUTHADD|AUTHRMV(<authority to give|remove>)`

- **setmqaut**
  - `setmqaut –m <QM name> -n <Object name> -t <Object type> -g <Group name> <authorizations to give or remove>`

- **IBM MQ Explorer**
  - By right clicking on the object you want to grant/remove authorities for and selecting “`Object Authorities -> Manage Authority Records`”

Configuration

```
SET AUTHREC  OBJTYPE(QMGR)          +
             GROUP('Group1')        +
             AUTHADD(CONNECT)
```



TRANSPORT LAYER SECURITY (TLS)

Details

- **TLS uses Private-Public asymmetric keys to exchange symmetric keys used to encrypt data.**
  - The symmetric keys exchanged are referred to as “session keys”.
  - The asymmetric keys are associated with a certificate that is used for identity.

- **IBM MQ’s integration of TLS provides the following two features:**
  - Encryption of transmissions between client/queue manager to queue manager.
  - [optional] Authentication with a queue manager.

Details

- **Certificates are created, stored and managed using tools supplied with IBM MQ**
  - runmqakm
  - runmqckm
  - iKeyman (strmqikm)

- **Certificates must be stored in a keystore format recognised by the queue manager (CMS)**
  - The keystore password must also be available to the queue manager in a secure stash file.

- **IBM MQ Channels can only have a single CipherSpec set on them**
  - A CipherSpec is a string which details the hashing and encryption algorithm to use.
  - A list of the cipher strings you can supply are detailed on the knowledge centre.

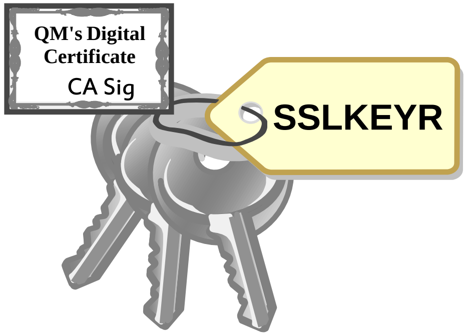
Details

- **IBM MQ allows clients to either connect anonymously or with mutual authentication**
  - If a client connects with a certificate then it must be known and trusted by the queue manager.

- **CipherSpec lists are updated when new vulnerabilities arise**
  - In later versions of IBM MQ you may notice the list size changing.
  - We do not delete CipherSpecs, we disable them by default.

- **MQv8 added in multiple certificates feature**
  - Allows you to specify a different certificate to use at the channel level
  - Allows you to specify a certificate to use on the queue manager
    - Before you would be forced to name your certificate **`ibmwebspheremq<QM name>`**

Configuration

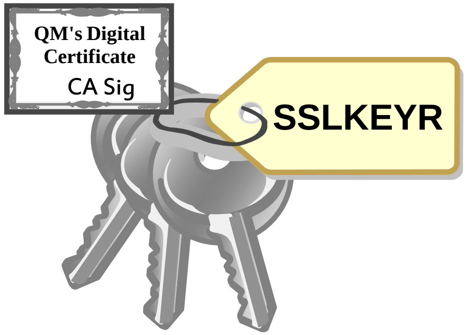


- **Once you have created a Key store for the server to use:**
  - `ALTER QMGR SSLKEYR(<location of keystore>)`
- **Once you have created the certificate for the server to use (MQv8+ only)**
  - `ALTER QMGR CERTLABL(<certificate label>)`

```
ALTER QMGR
SSLKEYR('var/mqm/qmgrs/QM1/ssl/key')
CERTLABL(‘QM1Certificate’)

REFRESH SECURITY TYPE(SSL)
```

Configuration



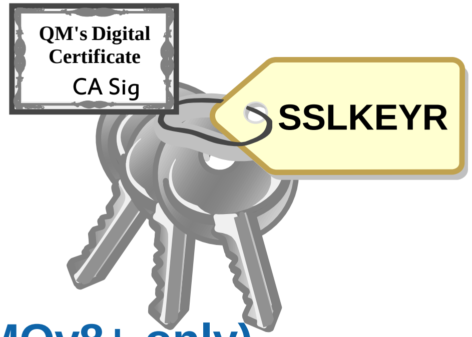
- **Once you have created a Key store for the server to use:**
  - `ALTER QMGR SSLKEYR(<location of keystore>)`

- **Once you have created the certificate for the server to use (MQv8+ only)**
  - `ALTER QMGR CERTLABL(<certificate label>)`

- **To enable TLS on a channel, specify a CipherSpec to use.**
  - `ALTER CHANNEL(<channel name>) CHLTYPE(<channel type>) SSLCIPH(<Cipher string>)`

```
ALTER CHANNEL(X) SSLCAUTH(REQUIRED)
SSLCIPH(TLS_RSA_WITH_AES_128_CBC_SHA256)
```

Configuration



- **Once you have created a Key store for the server to use:**
  - `ALTER QMGR SSLKEYR(<location of keystore>)`

- **Once you have created the certificate for the server to use (MQv8+ only)**
  - `ALTER QMGR CERTLABL(<certificate label>)`

- **To enable TLS on a channel, specify a CipherSpec to use.**
  - `ALTER CHANNEL(<channel name>) CHLTYPE(<channel type>) SSLCIPH(<Cipher string>)`

- **To force clients to connect with a mutual authentication, set the SSLCAUTH to REQUIRED**
  - `ALTER CHANNEL(<channel name>) CHLTYPE(<channel type>) SSLCAUTH(OPTIONAL|REQUIRED)`

- **To set a different certificate to use on a channel (MQv8+ only)**
  - `ALTER CHANNEL(<channel name>) CHLTYPE(<channel type>) CERTLABL(<certificate label>)`

CHANNEL AUTHENTICATION

Details

- **Channel authentication rules are filters that can be applied for incoming connections**
    - Whitelisting – Allow connections based on a filter
    - Blacklisting – Block a connection based on a filter

- **The filters are applied on channels and are applied to all incoming connections for that channel**
    - The filter can be either very specific or generic. (Exact channel name or wildcard)

Details

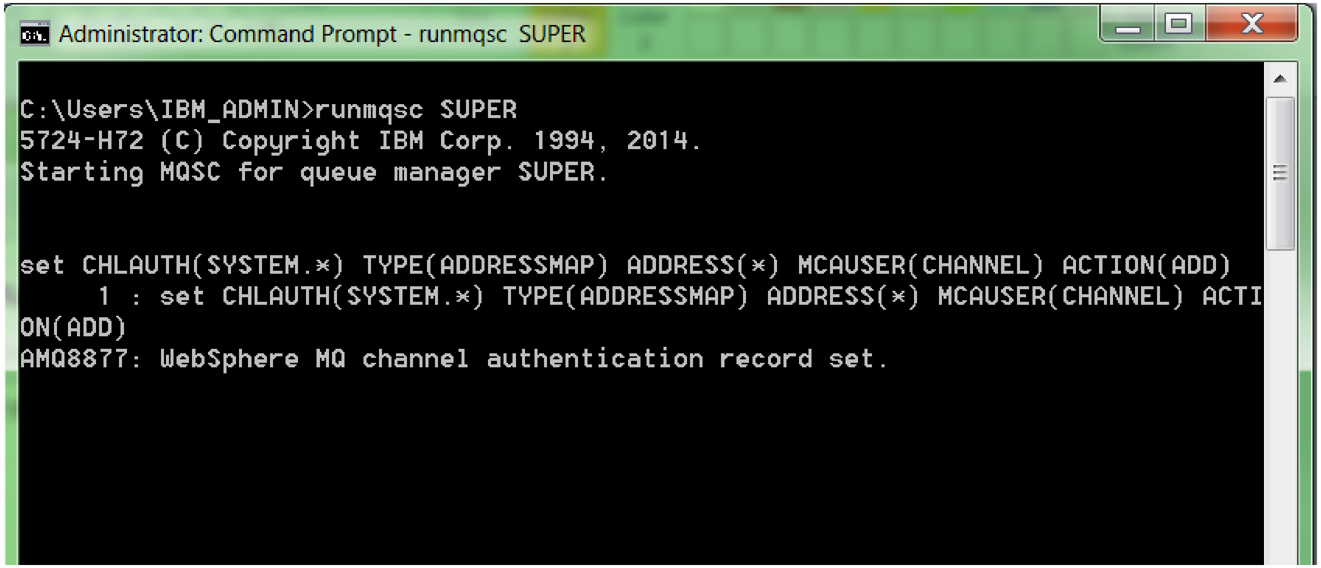
- **There are four types of filters:**
  - TLS Distinguished name (Issuer and Subject)
  - Client User ID name
  - Remote Queue Manager name
  - IP/Hostname

- **For IP/Hostname the connection can be allowed/blocked at the listener or channel**

- **For Client user ID, the userid blocked can be the userid connected with or the final adopted userid**

Configuration

```
SET CHLAUTH(<Channel name>) TYPE(<channel authentication type>)
<extra parameters> ACTION(ADD|REMOVE|REMOVEALL|REPLACE)
```



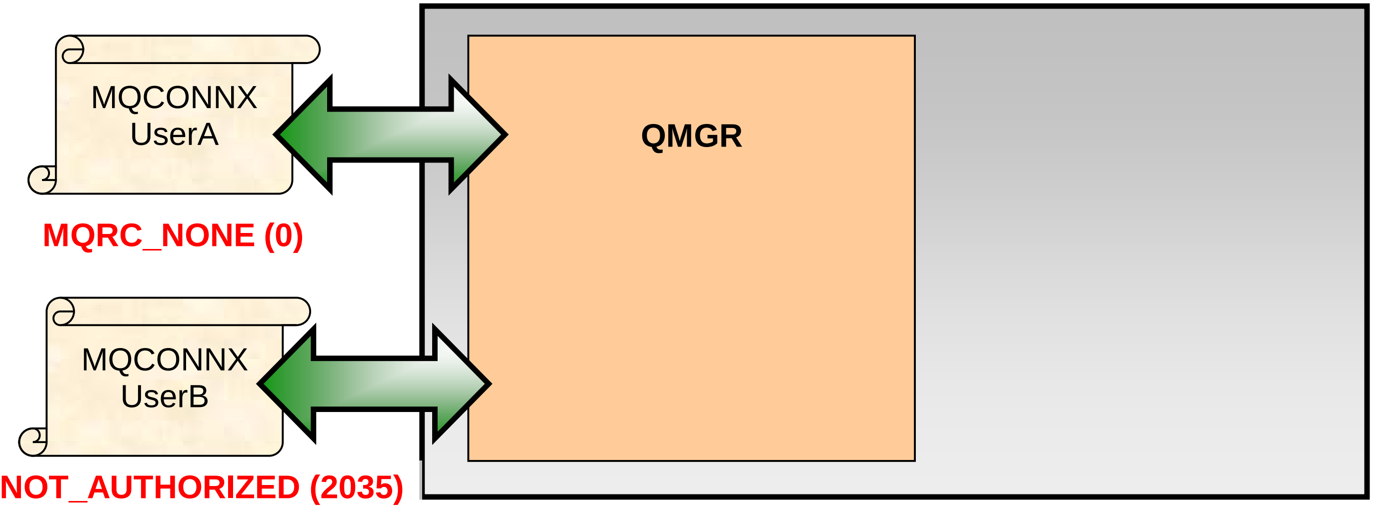
```
C:\Users\IBM_ADMIN>runmqsc SUPER
5724-H72 (C) Copyright IBM Corp. 1994, 2014.
Starting MQSC for queue manager SUPER.


set CHLAUTH(SYSTEM.*) TYPE(ADDRESSMAP) ADDRESS(*) MCAUSER(CHANNEL) ACTION(ADD)
     1 : set CHLAUTH(SYSTEM.*) TYPE(ADDRESSMAP) ADDRESS(*) MCAUSER(CHANNEL) ACTI
ON(ADD)
AMQ8877: WebSphere MQ channel authentication record set.
```

Configuration

```
SET CHLAUTH(*) TYPE(USERMAP)
               CLNTUSER(*)
               USERSRC(NOACCESS)
               ACTION(ADD)

SET CHLAUTH(*) TYPE(USERMAP)
                CLNTUSER(‘UserA’)
                USERSRC(CHANNEL)
                ACTION(ADD)
```



MQRC_NONE (0)

MQRC_NOT_AUTHORIZED (2035)

SECURITY EXITS

Details

- **Security exits are bespoke, customer created exits that are ran during the security checking.**

- **MQ comes with an API that can interact with MQ to provide extra control over a connection.**
  - They allow customers to expand MQ's security to suit their needs.
  - For example a customer could write a security exit to only allow connection to a channel during 08:00 to 17:00.

- **Before MQ v8 they could be used to provide connection authentication functionality.**

- **When executed the security exit will have access to the channel definition, information about the incoming connection and information**
  - It will also have a piece of data passed to it that is set on the channel - SCYDATA

Creation and configuration

- **To create a Security exit, first create a C file with the following skeleton code.**

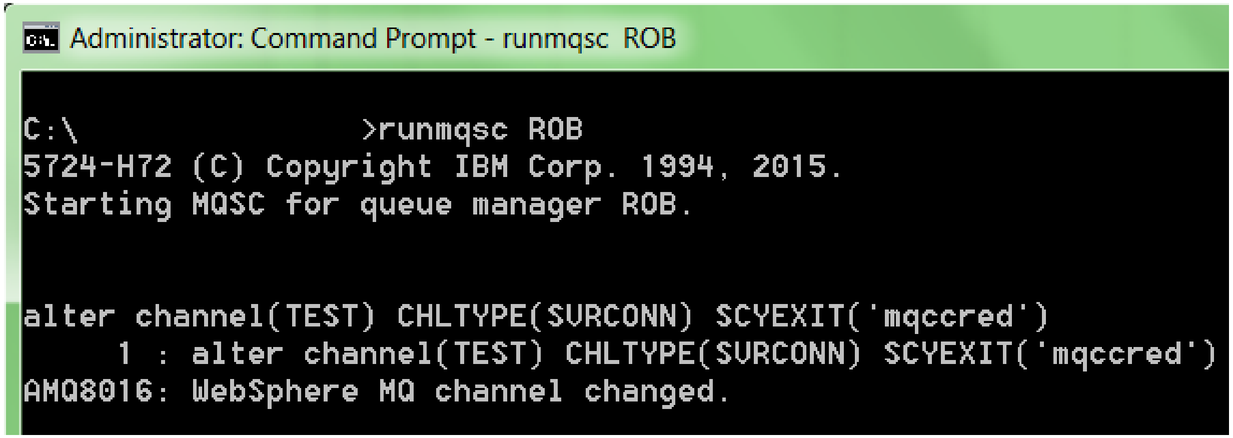
```
void MQENTRY MQStart() {;}
void MQENTRY EntryPoint (PMQVOID pChannelExitParms,
                         PMQVOID pChannelDefinition,
                         PMQLONG pDataLength,
                         PMQLONG pAgentBufferLength,
                         PMQVOID pAgentBuffer,
                         PMQLONG pExitBufferLength,
                         PMQPTR  pExitBufferAddr)
{
  PMQCXP pParms = (PMQCXP)pChannelExitParms;
  PMQCD  pChDef = (PMQCD)pChannelDefinition;
  /* Add Security Exit Code Here */
}
```

Creation and configuration

- **Next compile the C file into a dll and place it into:**
  - `<MQ Data Root>/exits/<Installation Name>`

- **With the exit in place you can now edit the channel configuration you want the exit to be invoked on**

  -
    ```
    ALTER CHANNEL(<channel name>) CHLTYPE(<channel type>)
    SCYEXIT(<exit filename without extension>)
    SCYDATA(<Data to pass to security exit>)
    ```



```
C:\                   >runmqsc ROB
5724-H72 (C) Copyright IBM Corp. 1994, 2015.
Starting MQSC for queue manager ROB.


alter channel(TEST) CHLTYPE(SVRCONN) SCYEXIT('mqccred')
     1 : alter channel(TEST) CHLTYPE(SVRCONN) SCYEXIT('mqccred')
AMQ8016: WebSphere MQ channel changed.
```

AMS

Details

- **AMS stands for Advanced Message Security**
  - It is message level security
  - It is a separate licensable feature - included in MQ Advanced

- **AMS is an end-to-end security model, messages stay signed/encrypted through the whole lifetime of a message**
  - In transit
  - At rest

- **With AMS you can create policies for a queue that describe how messages should be protected when applications put or get messages using that queue name.**
  - Signing
  - Encryption
  - Both

Details

- **AMS does not perform any access control:**
  - Only privacy and integrity protection
  - Should be used with existing access control, authentication, etc

- **Encryption level protection prevents unauthorised users reading message data.**
  - Including MQ administrators.

- **Signing protection prevents messages from being altered.**

- **Signing & Encryption use certificates – Same as TLS.**

- **No application code changes required to use AMS.**

Configuration

- **Differences between AMS & TLS configuration**
  - Both sides must have a certificate
  - Both sides must have exchanged the public certificate
  - The full certificate chain must be present in the key store

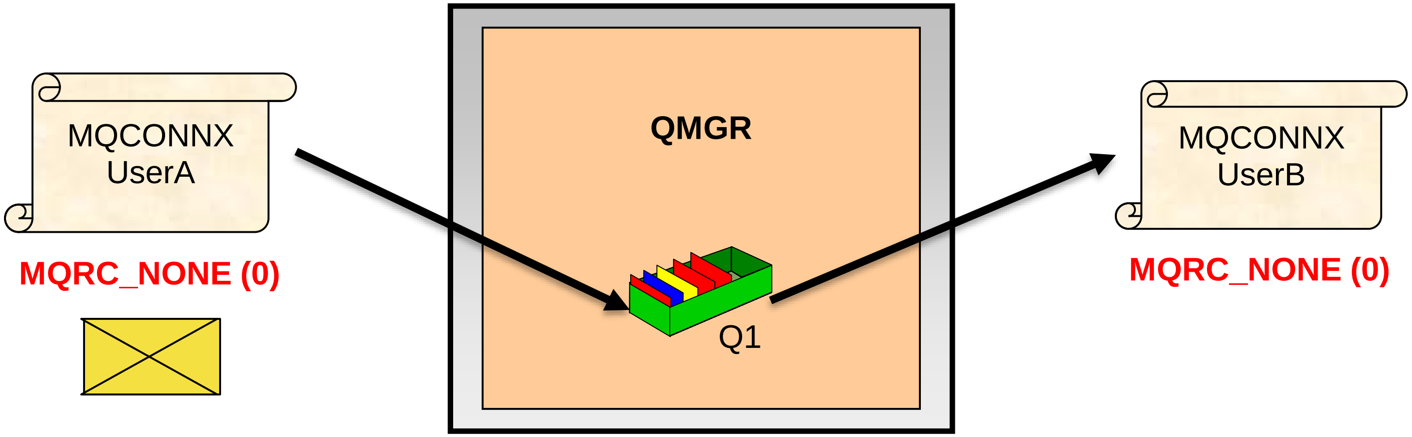
- **Policies can be created in explorer, runmqsc or using setmqspl**

  -
    ```
    setmqspl –m <QM name> -p <Q Name> -s <Signing algorithm>
    -a <Authorised signers> -e <Encryption algorithm> -r <Recipients>
    ```

  -
    ```
    SET POLICY(<Q NAME>) SIGNALG(<Signing algorithm>)
    ENCALG(<Encryption algorithm>) SIGNER(<Authorised signers>)
    RECIP(<Recipients>) ACTION(ADD|REPLACE|REMOVE)
    ```

Configuration

```
SET POLICY(Q1) SIGNALG(SHA512)
               SIGNER('CN=UserA,O=IBM,C=UK')
               ACTION(ADD)
```



Questions & Answers

Other IBM MQ security sessions

- **MQ Security Deep Dive including IBM MQ AMS – Rob Parker**
  - Monday 13:00 – 14:10, Leopardwood room
  - Tuesday 08:30 – 09:40, Leopardwood room

- **Integrating MQ with Directory Services – Mark Taylor**
  - Wednesday 09:50 – 11:00, Indigo Bay

- **Authentication in MQ – Morag Hughson**
  - Tuesday 09:50 – 11:00, Leopardwood room
  - Wednesday 13:00 – 14:10, Salon E

- **Securing your z/OS Queue – Morag Hughson**
  - Monday 14:30 – 15:40, Rosewood room
  - Tuesday 14:30 – 15:30, Aloeswood room

- **Check list of MQ top MQ security outstanding bugs/issues/gotchas – T.Rob**
  - Monday 16:00 – 17:10, Aloeswood room
  - Wednesday 08:30 – 09:40, Salon E

Trying Something New

MQ Labs Experiment

Booth next to Aloeswood

MQ for z/OS Images

MQ for Distributed Image

Monday & Tuesday Morning – drop in