SHARE in Orlando 2015

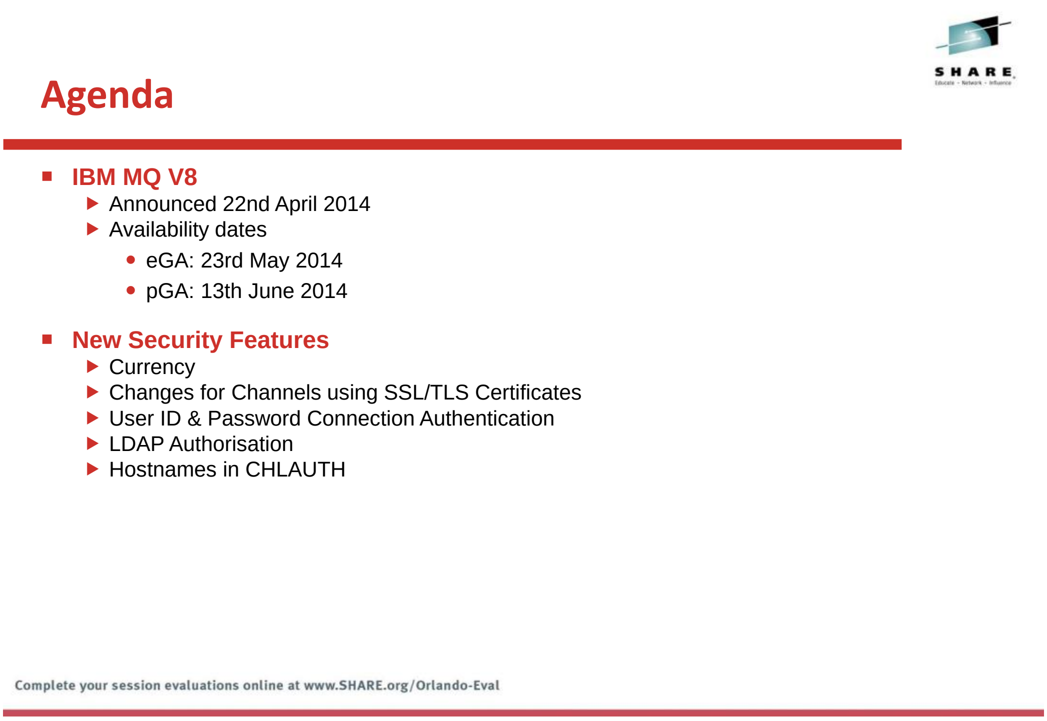
17894: MQ Security - V8 Features Deep Dive

Mark Taylor
marke_taylor@uk.ibm.com
IBM Hursley

#SHAREorg

SHARE is an independent volunteer-run information technology association that provides **education**, professional **networking** and industry **influence**.

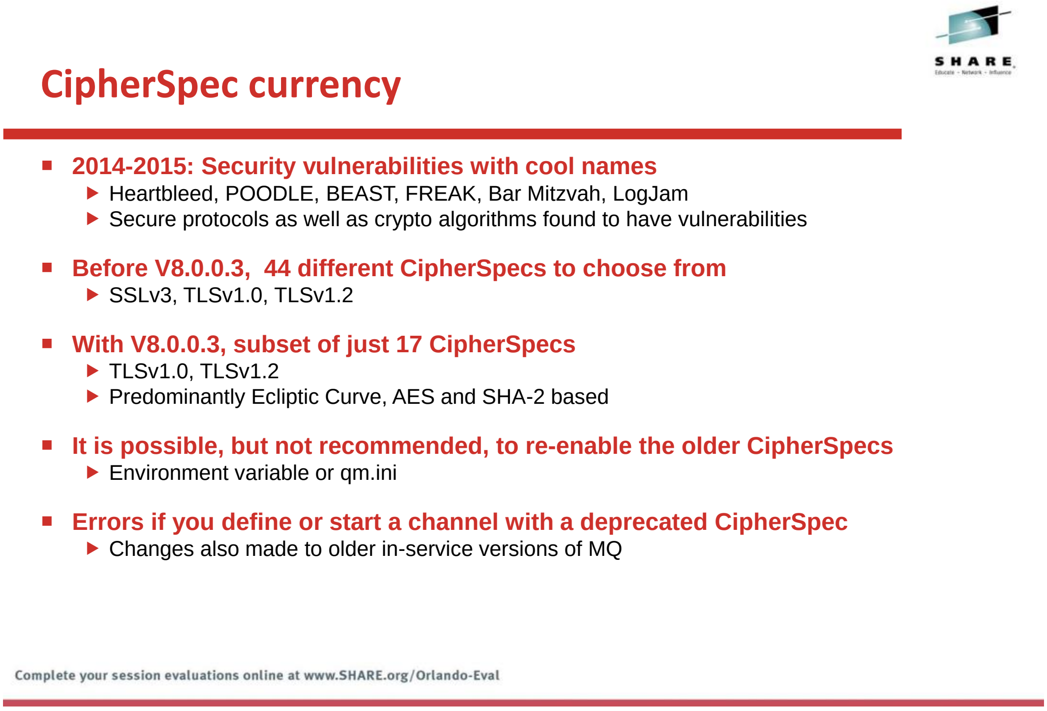
CELEBRATING 60 YEARS OF SHARE
Influencing IT Since 1955



Agenda

- **IBM MQ V8**
  - Announced 22nd April 2014
  - Availability dates
    - eGA: 23rd May 2014
    - pGA: 13th June 2014
- **New Security Features**
  - Currency
  - Changes for Channels using SSL/TLS Certificates
  - User ID & Password Connection Authentication
  - LDAP Authorisation
  - Hostnames in CHLAUTH

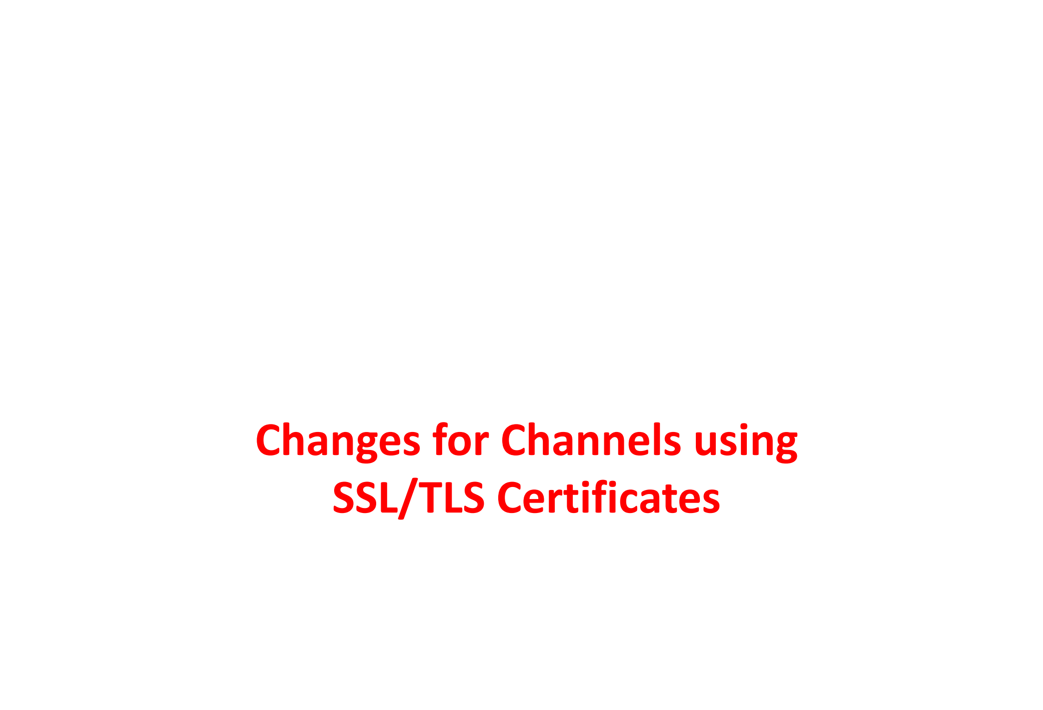
Complete your session evaluations online at www.SHARE.org/Orlando-Eval



CipherSpec currency

- **2014-2015: Security vulnerabilities with cool names**
  - Heartbleed, POODLE, BEAST, FREAK, Bar Mitzvah, LogJam
  - Secure protocols as well as crypto algorithms found to have vulnerabilities
- **Before V8.0.0.3, 44 different CipherSpecs to choose from**
  - SSLv3, TLSv1.0, TLSv1.2
- **With V8.0.0.3, subset of just 17 CipherSpecs**
  - TLSv1.0, TLSv1.2
  - Predominantly Ecliptic Curve, AES and SHA-2 based
- **It is possible, but not recommended, to re-enable the older CipherSpecs**
  - Environment variable or qm.ini
- **Errors if you define or start a channel with a deprecated CipherSpec**
  - Changes also made to older in-service versions of MQ

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

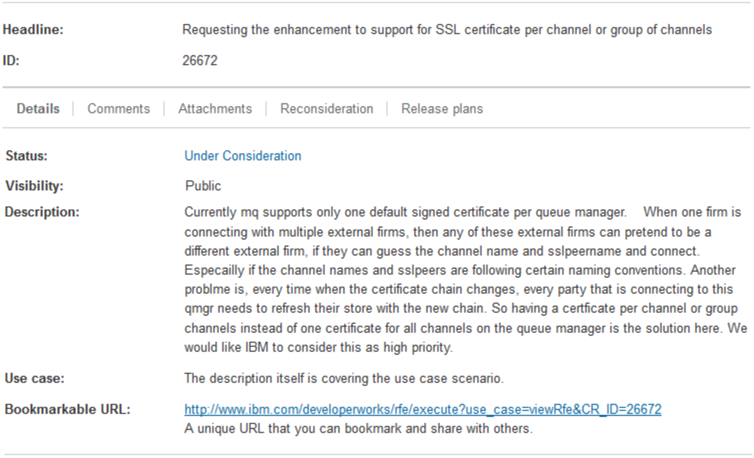


Changes for Channels using SSL/TLS Certificates

Agenda

- **Requests for Enhancement**
- **Changes for Channels using SSL/TLS Certificates**
  - Recap
  - Single Queue Manager Certificate
  - Per Channel Certificate
  - Certificate Matching

Request for Enhancement (26672)



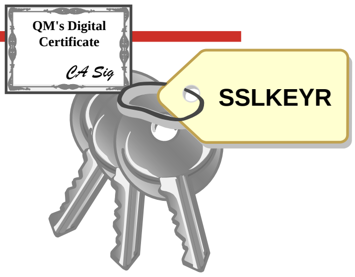
Headline:	Requesting the enhancement to support for SSL certificate per channel or group of channels
ID:	26672

Details | Comments | Attachments | Reconsideration | Release plans

Status:	Under Consideration
Visibility:	Public
Description:	Currently mq supports only one default signed certificate per queue manager. When one firm is connecting with multiple external firms, then any of these external firms can pretend to be a different external firm, if they can guess the channel name and sslpeername and connect. Especailly if the channel names and sslpeers are following certain naming conventions. Another problme is, every time when the certificate chain changes, every party that is connecting to this qmgr needs to refresh their store with the new chain. So having a certficate per channel or group channels instead of one certificate for all channels on the queue manager is the solution here. We would like IBM to consider this as high priority.
Use case:	The description itself is covering the use case scenario.
Bookmarkable URL:	http://www.ibm.com/developerworks/rfe/execute?use_case=viewRfe&CR_ID=26672 A unique URL that you can bookmark and share with others.

Key Repository

- **Contains Entity's own Digital Certificate**
  - z/OS Queue Manager
    - ibmWebSphereMQ<QMgr Name> (mixed case) label
  - Distributed Queue Manager
    - ibmwebspheremq<qmgr name> (lower case) label
  - Client
    - ibmwebspheremq<logon userid> (lower case) label
  - Digital Certificates from various Certification Authorities



- **On z/OS Queue Managers**
  - Keyring name
- **On Unix®, Windows®, iSeries® QMgrs**
  - Key database path
- **Clients: mqclient.ini file**
  - SSL Stanza – SSLKeyRepository
- **MQCONNX (MQSCO structure)**
  - SSLKeyRepository
- **Environment variable**
  - export MQSSLKEYR=/var/mqm/ssl/key

```
ALTER QMGR SSLKEYR(CSQ1RING)

ALTER QMGR
SSLKEYR('/var/mqm/qmgrs/QM1/ssl/key')
```

```
mqclient.ini
SSL:
   SSLKeyRepository=C:\key
```

Key Repository – Notes

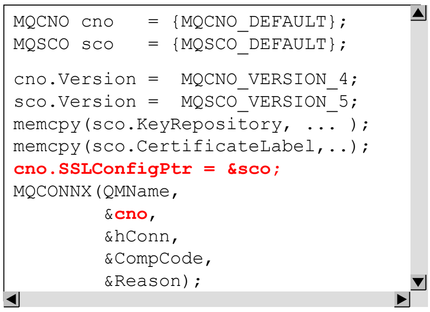
NOTES

- Queue Manager
  - A digital certificate contains the identity of the owner of that certificate. Each MQ queue manager has its own certificate. On all platforms this certificate is stored in a key repository using your digital certificate management tool, e.g. in RACF® (z/OS) or iKeyMan (UNIX and Windows).
  - On z/OS, the required certificate in the key repository is specified with the mixed-case label ibmWebSphereMQ<QMgr Name>. On UNIX, Windows and iSeries, the required certificate in the key repository is specified with the lower-case label ibmwebspheremq<qmgr name> . Note that the certificate label is also sometimes referred to as its "friendly name".
  - The key repository is specified on the MQ QMGR object using the ALTER QMGR command. On z/OS this is the name of the keyring object in the External Security Manager (ESM), and on the distributed platforms this is the path and the stem of the filename for the key database file.
- Client
  - Generally each user of the MQ client has a separate key repository file, with access restricted to that user.
  - This key repository file is accessed using the environment variable MQSSLKEYR, or the MQCONNX SSLKeyRepository parameter.
  - A particular personal certificate within that file is selected for use on the client's SSL channels. Clients use the certificate labeled with ibmwebspheremq followed by the logon userid, wrapped to lower case.

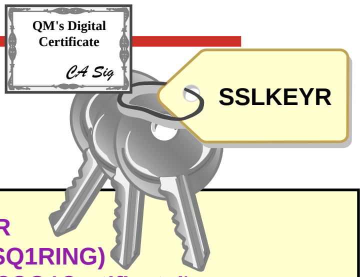
- The key repository generally also contains a number of signed digital certificates from various Certification Authorities which allows it to be used to verify certificates it receives from its partner at the remote end of the connection.

Single Queue Manager Certificate

- **Name Queue Manager Certificate**
  - Using CERTLABL attribute
- **Name Client Certificate**
  - mqclient.ini file SSL Stanza
    - CertificateLabel
  - MQCONNX (MQSCO structure)
    - CertificateLabel
- **Environment variable**
  - export MQCERTLABL=MyCert



```
MQCNO  cno    = {MQCNO_DEFAULT};
MQSCO  sco    = {MQSCO_DEFAULT};

cno.Version =  MQCNO_VERSION_4;
sco.Version =  MQSCO_VERSION_5;
memcpy(sco.KeyRepository, ... );
memcpy(sco.CertificateLabel,..);
cno.SSLConfigPtr = &sco;
MQCONNX(QMName,
        &cno,
        &hConn,
        &CompCode,
        &Reason);
```



ALTER QMGR
SSLKEYR(CSQ1RING)
CERTLABL('CSQ1Certificate')
CERTQSGL('SharedCert')

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

mqclient.ini
SSL:
  SSLKeyRepository=C:\key
  CertificateLabel=MyCert

Single Queue Manager Certificate – Notes

N O T E S

- Before MQ V8, the label name for a digital certificate to be used by the queue manager (or an MQ Client) was fixed by MQ. You had to label your certificate exactly as MQ required it, in order for the certificate to be found. This doesn't always meet customer standards of certificate labelling.
- In MQ V8 you can provide your own label name for the queue manager (or an MQ Client) to use.
- For the queue manager you have a new attribute on ALTER QMGR called CERTLABL (and additionally CERTQSGL on z/OS for a QSG level certificate – previously located with the label ibmMQ<QSG-name>).
- For clients, you can provide the Certificate label in the MQSCO structure (along with the SSLKeyRepository location); or in the SSL stanza in the mqclient.ini file (along with the SSLKeyRepository location), or using the environment variable MQCERTLABL.

Use Cases

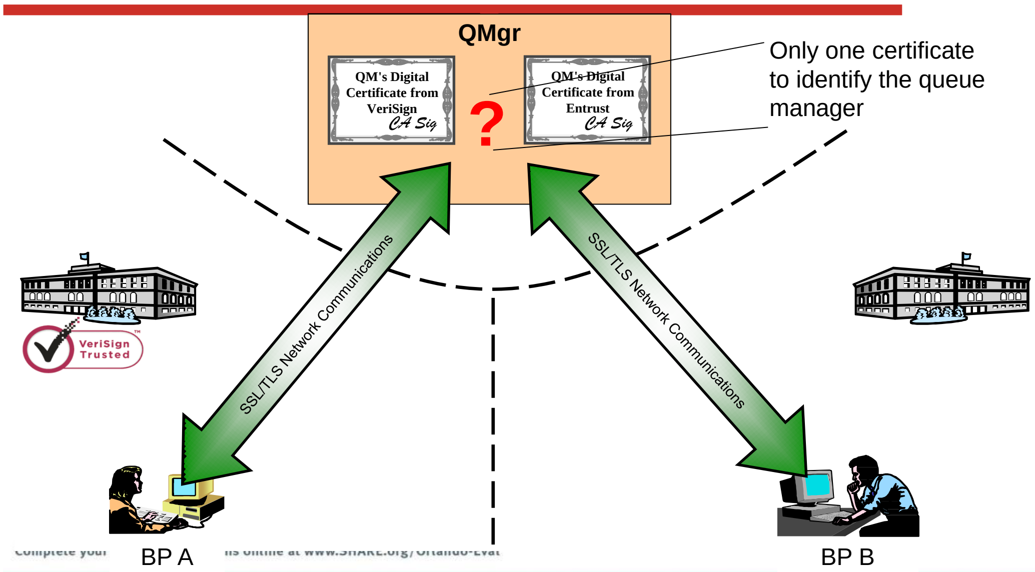
- **Following company policy on certificate labelling**
- **Using the same certificate for more than one queue manager**
  - Not that we would condone this!
- **Migrating over to a new certificate when main certificate is ready to expire**
  - Used to have to issue GSKit/RACF commands to rename certificate
    - ibmwebspheremqqm1 -> ibmwebspheremqqm1old
    - ibmmwebsphereqqm1new -> ibmwebspheremqqm1
    - REFRESH SECURITY TYPE(SSL)
  - Now just MQ commands when the time comes
    - Current label is ‘QM1 Cert 2013’
    - ALTER QMGR CERTLABL(‘QM1 Cert 2014’)
    - REFRESH SECURITY TYPE(SSL)

Use Cases – Notes

N O T E S

- Here we list some of the uses we can imagine for being able to label your own certificate instead of following the pattern mandated in the past by MQ.
- It is worth highlighting here that the change over from using one certificate to another is now a task that can be accomplished by the MQ administrator alone, when he is ready. The job of installing the new certificate can be done at any prior point and labelled however you wish. That label does not now have to change in order to get the queue manager to use it, so it is just a task for the MQ administrator to tell the queue manager which label to use now, and then refresh.

Business Partners with different CA requirements

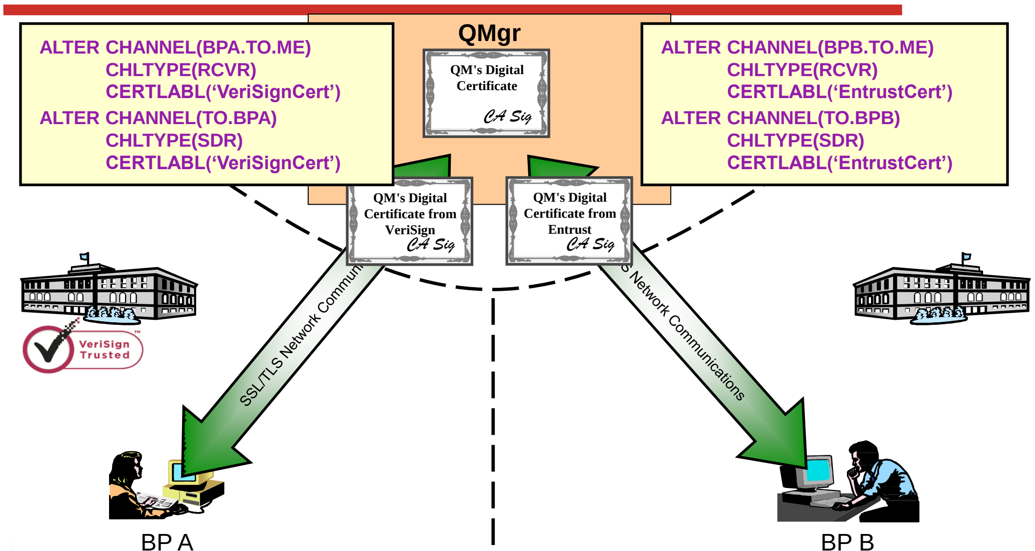

QMgr
QM's Digital Certificate from VeriSign CA Sig
?
QM's Digital Certificate from Entrust CA Sig
Only one certificate to identify the queue manager
VeriSign Trusted
SSL/TLS Network Communications
SSL/TLS Network Communications
BP A
BP B

Business Partners with different CA requirements – Notes

N O T E S

- Imagine the situation where your company has need to communicate securely with two difference business partners. These business partners each have a different requirement about the Certificate Authority (CA) who signs the certificates that they are happy to accept. In our example, Business Partner A will only accept certificates signed by VeriSign, whereas Business Partner B will only accept certificates signed by Entrust.
- In order for your company to be able to communicate with both of these Business Partners, you need a certificate that is signed by VeriSign (to communicate with Business Partner A) and a certificate that is signed by Entrust (to communicate with Business Partner B). However, since a queue manager can only have one certificate, with releases prior to V8 of MQ, you were forced into having two queue managers, one using each certificate. This is less than ideal.
- N.B. Some people also solve this issue by using an MQIPT in front of the queue manager.

Certificate per Channel



```
ALTER CHANNEL(BPA.TO.ME)
          CHLTYPE(RCVR)
          CERTLABL('VeriSignCert')
ALTER CHANNEL(TO.BPA)
          CHLTYPE(SDR)
          CERTLABL('VeriSignCert')
```

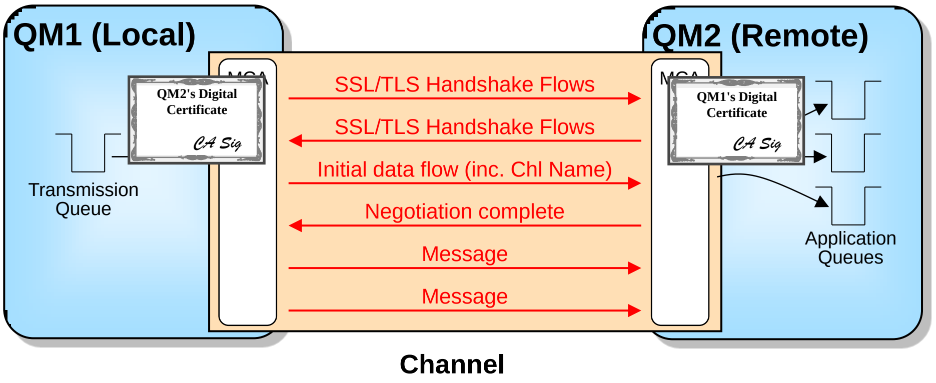
```
ALTER CHANNEL(BPB.TO.ME)
          CHLTYPE(RCVR)
          CERTLABL('EntrustCert')
ALTER CHANNEL(TO.BPB)
          CHLTYPE(SDR)
          CERTLABL('EntrustCert')
```

Certificate per Channel – Notes

N O T E S

- What is required is the ability to indicate that this particular channel should use a different certificate than other channels.
- This is achieved in MQ V8 with an attribute on a channel, CERTLABL, which can either be blank – which means use whatever the queue manager overall is configured to use, or if provided, means that this channel should use the specifically named certificate.
- For reasons explained a little later on, we only allow you to specify a non blank CERTLABL at definition time if you are using a TLS cipherspec.

Why haven’t we always done this?



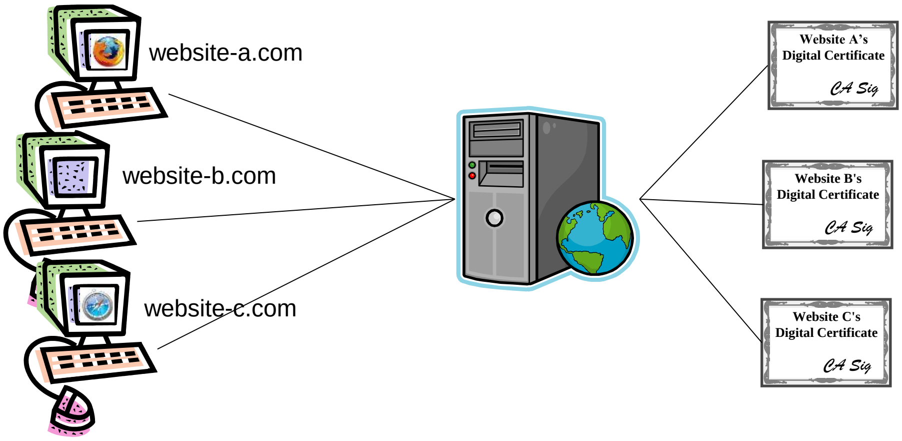
Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Why haven’t we always done this? – Notes

N O T E S

- The SSL/TLS handshake is done as the first thing on a channel, before any of the internal channel FAP flows. If you have ever pointed a web-browser with a https:// address at your MQ listener port, you’ll know this. This means that the certificate is authenticated long before the channel name at the receiver end is known. This made it impossible to choose a certificate to be used for a receiver based on the channel name. The best that could have been done would have been to provide a different certificate per port number and have several different listeners running, each presenting a different certificate.
- Over time however, as SSL/TLS is used by more and more consolidated servers, think HTTP server farms and large application servers, it has become necessary to be able to separate the traffic that is going to a single server into differently authenticated groups.
- Enhancements to the TLS protocol allow the provision of information as part of the TLS handshake which can then be used to determine which certificate should be used for this particular connection.
- This enhancement is known as Server Name Indication (SNI).

Server Name Indication


website-a.com
website-b.com
website-c.com
Website A's Digital Certificate
CA Sig
Website B's Digital Certificate
CA Sig
Website C's Digital Certificate
CA Sig

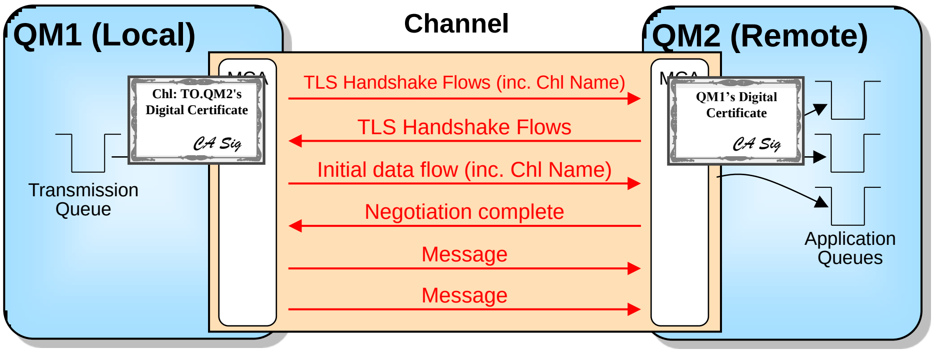
Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Server Name Indication – Notes

N O T E S

- Wikipedia provides a succinct summary of what Server Name Indication (SNI) is.
- The example on this page shows a use case where SNI would be used. We have three websites which each have their own certificate. When they were hosted on individual servers, then this was no problem, each web server has one certificate.
- Now let's think about what happens if we decide to consolidate those web sites onto a single server. How can we maintain the certificate correlation with the website. SNI allows this to be able to happen by providing a place in the TLS handshake for additional data to be flowed. This additional data is the hostname the browser was trying to connect to, thus allowing the certificate to be chosen based off that hostname.

Using Server Name Indication (SNI) with a channel name



- **Both ends of the channel must be at the new release**
- **Only TLS can be used, no SSL**
  - Only certain cipherspecs will be able to supply this behaviour
- **JSSE doesn't yet support SNI**
  - So Java client can't make use of it

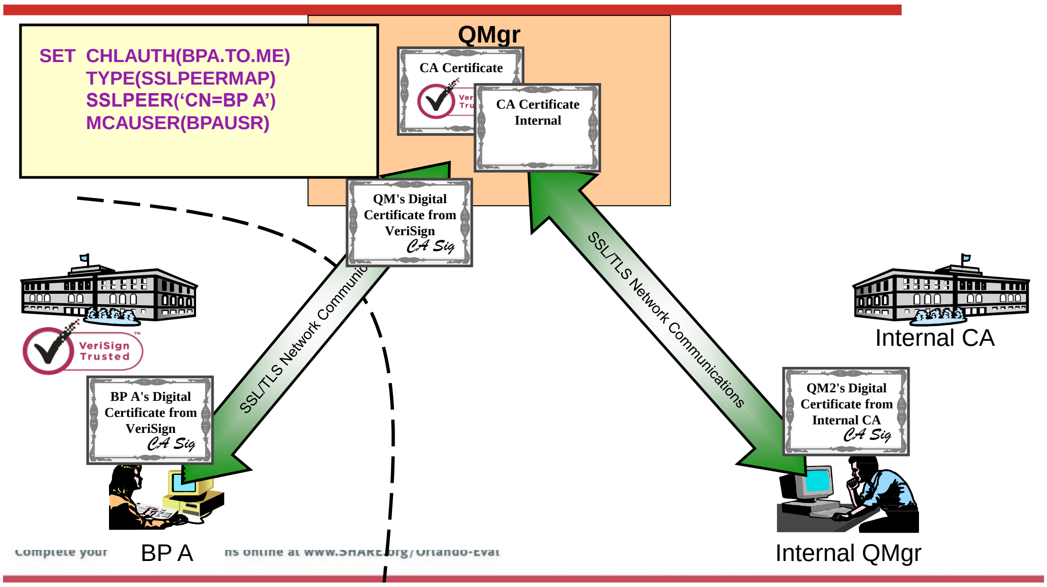
- **If old sender / client / cipherspec used**
  - we only detect that we needed to supply a different certificate after completion of the handshake and so will fail the connection at that point (if it hasn't already failed due to using the wrong certificate!)

Using Server Name Indication (SNI) with a channel name

N O T E S

- MQ V8 uses SNI to provide a channel name instead of a hostname. The sender (or client) end of the channel has been enhanced to put the channel name into the Server Name Indication (SNI) hint for the TLS Handshake.
- The receiver (or server-conn) end of the channel has been enhanced to retrieve the channel name from the SNI hint and select the appropriate certificate based on that information. It is worth nothing that the channel name is now flowing in the clear, although in a tamper-proof manner.
- There are some restrictions to using this feature as listed.
- A back-level queue manager upon receiving a TLS handshake containing SNI, will just ignore what is in the SNI (as it is defined as an optional extension) and use the normal certificate.
- If there are no channels defined on the queue manager with anything in the CERTLABL field, then SNI will not be used by the receiving end. This will leave the behaviour the same as prior releases for certificate selection.

Our Business Partner Scenario again



```
SET  CHLAUTH(BPA.TO.ME)
     TYPE(SSLPEERMAP)
     SSLPEER('CN=BP A')
     MCAUSER(BPAUSR)
```

QMgr
CA Certificate
VeriSign Trusted
CA Certificate Internal
QM's Digital Certificate from VeriSign CA Sig
SSL/TLS Network Communic
SSL/TLS Network Communications
VeriSign Trusted
BP A's Digital Certificate from VeriSign CA Sig
Internal CA
QM2's Digital Certificate from Internal CA CA Sig
BP A
Internal QMgr

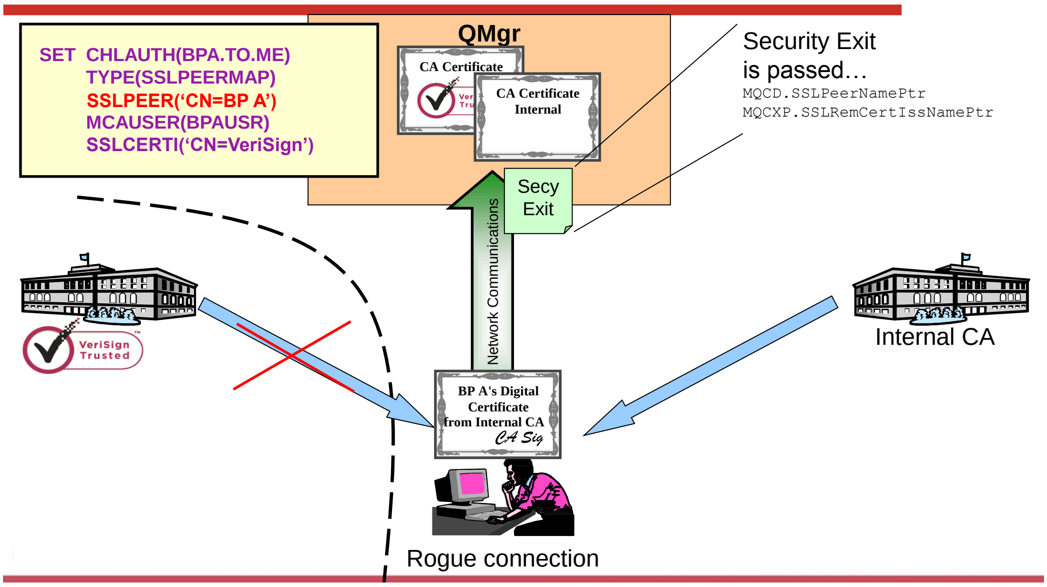
Complete your ... ns online at www.SHARE.org/Orlando-Eval

Our Business Partner Scenario again – Notes

N O T E S

- Let's look again at the business partner scenario again, but this time a little different, with one external CA and one internal CA.
- We've got the system set up so that we're using a Verisign certificate when talking to Business Partner A, and for the rest of our connections we have certificates created by our Internal CA. We've even got CHLAUTH rules in place to ensure that they are only allowed to connect to the queue manager over their appropriate channel.

Ensuring the Correct Certificate



```
SET  CHLAUTH(BPA.TO.ME)
     TYPE(SSLPEERMAP)
     SSLPEER('CN=BP A')
     MCAUSER(BPAUSR)
     SSLCERTI('CN=VeriSign')
```

Security Exit is passed…
`MQCD.SSLPeerNamePtr`
`MQCXP.SSLRemCertIssNamePtr`

Ensuring the Correct Certificate – Notes

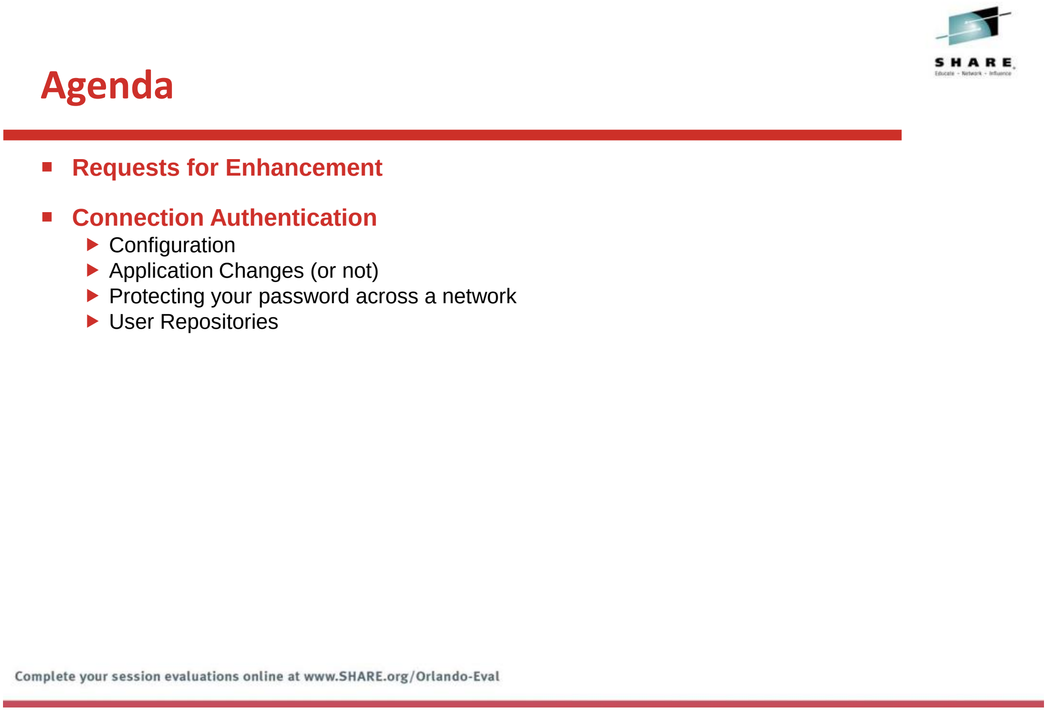
- However, since we now accept certificates which come from two different Certificate Authorities (CAs) we can run foul of another issue.
- One of the benefits of CAs is that they guarantee not to issue the certificates with the same DN as another certificate that they have already issued. So a rogue connection could not obtain a certificate with the same DN as Business Partner A from VeriSign, because VeriSign has already issued one with that DN. Also, one would expect external CA's to do a few more checks than that and not issue certificates with other people's company names in them to people not from that company. However, an internal CA may not be so diligent. Some internal CAs may simply accept what the user requests as their DN, so our rogue could obtain a certificate with Business Partner A's DN from such a CA.
- The only way to solve this issue in the past was to use a security exit, since security exits are presented with both the issuer's and subject's Distinguished Name. However, we are trying to get away from people having to write exits for common security issues, and this very much falls into that category.
- In MQ V8, we can solve this issue by using a new attribute on CHLAUTH rules which matches the issuer's DN – SSLCERTI. Our CHLAUTH rules can now be fully qualfied to use both SSLPEER (the subject's DN) and SSLCERTI (the issuer's DN).

Summary

- **Changes for Channels using SSL/TLS Certificates**
  - Single Queue Manager Certificate
    - `ALTER QMGR CERTLABL('My certificate name')`
  - Per Channel Certificate
    - `ALTER CHANNEL … CERTLABL('This channel certificate')`
  - Certificate Matching

    -
      ```
      SET CHLAUTH('*')
          TYPE(SSLPEERMAP)
          SSLPEER('CN=Mark Taylor')
          SSLCERTI('CN=IBM CA')
          MCAUSER('metaylor')
      ```

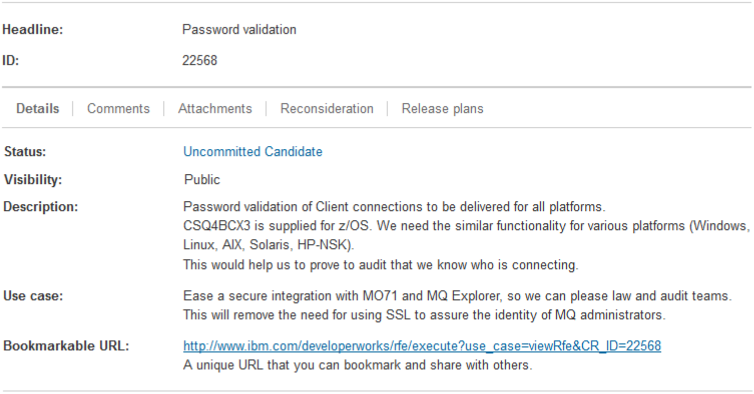
User ID & Password
Connection Authentication



Agenda

- **Requests for Enhancement**
- **Connection Authentication**
  - Configuration
  - Application Changes (or not)
  - Protecting your password across a network
  - User Repositories

Request for Enhancement (22568)

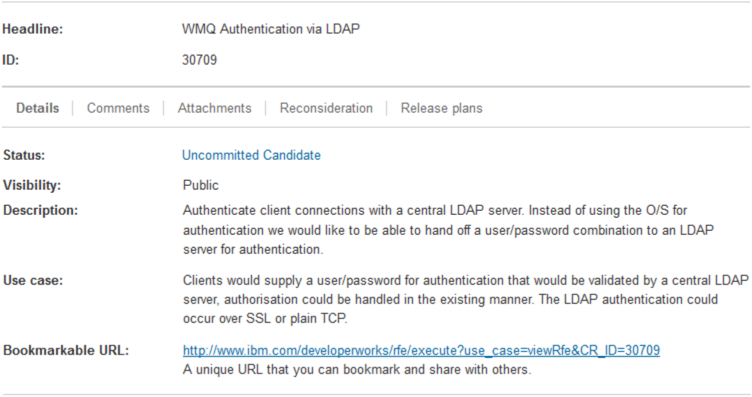


Headline:	Password validation
ID:	22568

Details | Comments | Attachments | Reconsideration | Release plans

Status:	Uncommitted Candidate
Visibility:	Public
Description:	Password validation of Client connections to be delivered for all platforms. CSQ4BCX3 is supplied for z/OS. We need the similar functionality for various platforms (Windows, Linux, AIX, Solaris, HP-NSK). This would help us to prove to audit that we know who is connecting.
Use case:	Ease a secure integration with MO71 and MQ Explorer, so we can please law and audit teams. This will remove the need for using SSL to assure the identity of MQ administrators.
Bookmarkable URL:	http://www.ibm.com/developerworks/rfe/execute?use_case=viewRfe&CR_ID=22568 A unique URL that you can bookmark and share with others.

Request for Enhancement (30709)



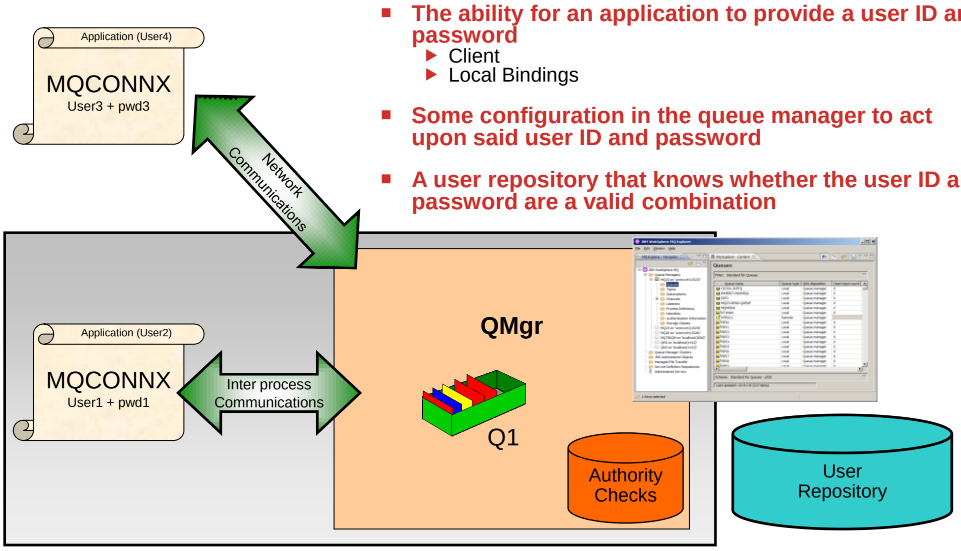
Headline:	WMQ Authentication via LDAP
ID:	30709

Details | Comments | Attachments | Reconsideration | Release plans

Status:	Uncommitted Candidate
Visibility:	Public
Description:	Authenticate client connections with a central LDAP server. Instead of using the O/S for authentication we would like to be able to hand off a user/password combination to an LDAP server for authentication.
Use case:	Clients would supply a user/password for authentication that would be validated by a central LDAP server, authorisation could be handled in the existing manner. The LDAP authentication could occur over SSL or plain TCP.
Bookmarkable URL:	http://www.ibm.com/developerworks/rfe/execute?use_case=viewRfe&CR_ID=30709 A unique URL that you can bookmark and share with others.

Connection Authentication – What is it?

- **The ability for an application to provide a user ID and password**
  - Client
  - Local Bindings
- **Some configuration in the queue manager to act upon said user ID and password**
- **A user repository that knows whether the user ID and password are a valid combination**



Connection Authentication – What is it? – Notes

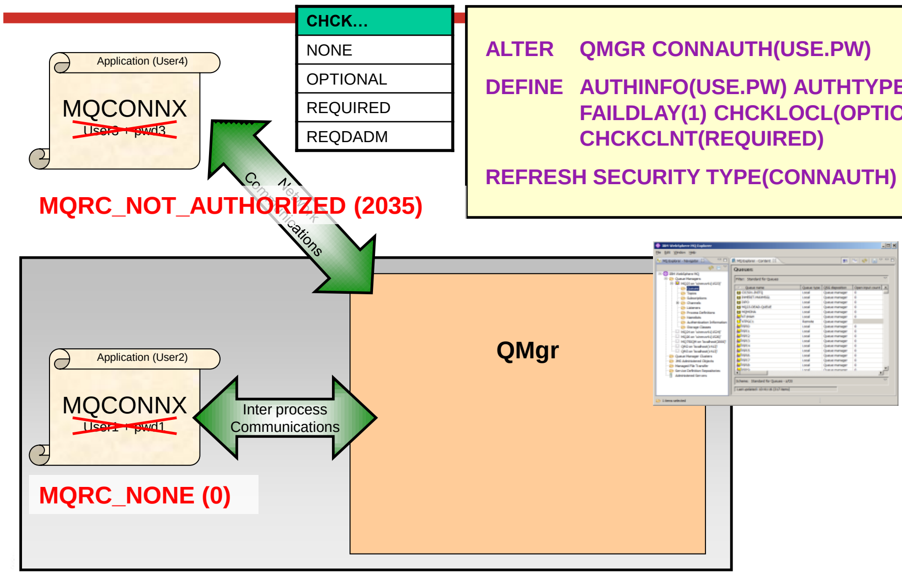
N O T E S

- This picture shows the landscape we're going to use to discuss various patterns and then the changes in MQ V8 in order to support these patterns. Just to ensure everyone is familiar with the parts on the diagram we'll briefly look at them first from left to right.
- On the left of this picture we see applications making connections, one as a client and one using local bindings. These applications could be using a variety of different APIs to connect to the queue manager, but all have the ability to provide a user ID and a password. The user ID that the application is running under (the classic user ID presented to MQ) may be different from the user ID provided by the application along with its password, so we illustrate both on the diagram.
- In the middle we have a queue manager with configuration commands and managing the opening of resources and the checking of authority to those resources. There are lots of different resources in MQ that an application may require authority to, in this diagram we are just going to use the example of opening a queue for output, but the same applies to all others.
- On the right we have a representation of a user repository – i.e. containing user IDs and passwords, more on this later.

Connection Authentication – Configuration

CHCK…
NONE
OPTIONAL
REQUIRED
REQDADM

```
ALTER    QMGR CONNAUTH(USE.PW)

DEFINE   AUTHINFO(USE.PW) AUTHTYPE(xxxxxx)
         FAILDLAY(1) CHCKLOCL(OPTIONAL)
         CHCKCLNT(REQUIRED)

REFRESH SECURITY TYPE(CONNAUTH)
```



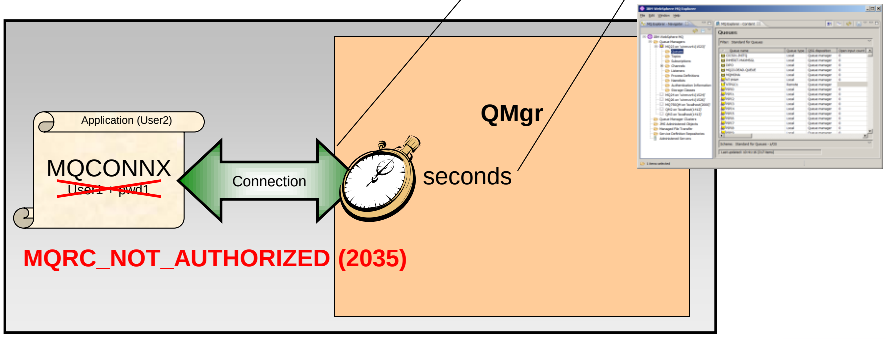
Connection Authentication – Configuration – Notes

NOTES

- We'll start with the basic configuration side of things. How do I turn on this connection authentication feature on the queue manager.
- On the queue manager object there is a new attribute called CONNAUTH (short for connection authentication) which points to an object name. The object name it refers to is an authentication information object – one of two new types. There are two existing types of authentication information objects from earlier releases of MQ, these original two types cannot be used in the CONNAUTH field.
- The two new types are similar in quite a few of the basic attributes so we will look at those first. We'll come back to more of the attributes later. We show here a new authentication information object which has two fields to turn on user ID and password checking, CHCKLOCL (Check Local connections) and CHCKCLNT (Check Client connections). Changes to the configuration of this must be refreshed for the queue manager to pick them up.
- Both of these fields have the same set of attributes, allowing for a strictness of checking. You can switch it off entirely with NONE; set it to OPTIONAL to ensure that if a user ID and password are provided by an application then they must be a valid pair, but that it is not mandatory to provide them – a useful migration setting perhaps; set it to REQUIRED to mandate that all applications provide a user ID and password; and, only on Distributed, REQDADM which says that privileged users must supply a valid user ID and password, but non-privileged users are treated as per the OPTIONAL setting.
- Any application that does not supply a user ID and password when required to, or supplies an incorrect combination even when it is optional will be told 2035 (MQRC_NOT_AUTHORIZED). N.B. When password checking is turned off using NONE – then invalid passwords will not be detected.

Connection Failure Delay

```
ALTER    QMGR CONNAUTH(USE.PW)

DEFINE   AUTHINFO(USE.PW) AUTHTYPE(xxxxxx)
         FAILDLAY(1) CHCKLOCL(OPTIONAL)
         CHCKCLNT(REQUIRED)

REFRESH SECURITY TYPE(CONNAUTH)
```



Application (User2)

MQCONNX

~~User1 + pwd1~~

Connection

QMgr

seconds

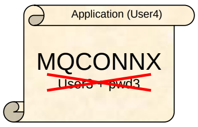
MQRC_NOT_AUTHORIZED (2035)

Connection Failure Delay - Notes

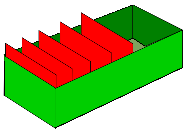
NOTES

- Any failed authentications will be held for the number of seconds in the FAILDLAY attribute before the error is returned to the application – just some protection against a busy loop from an application repeatedly connecting.

Connection Authentication – Error notification



MQRC_NOT_AUTHORIZED (2035)



ALTER QMGR AUTHOREV(ENABLED)

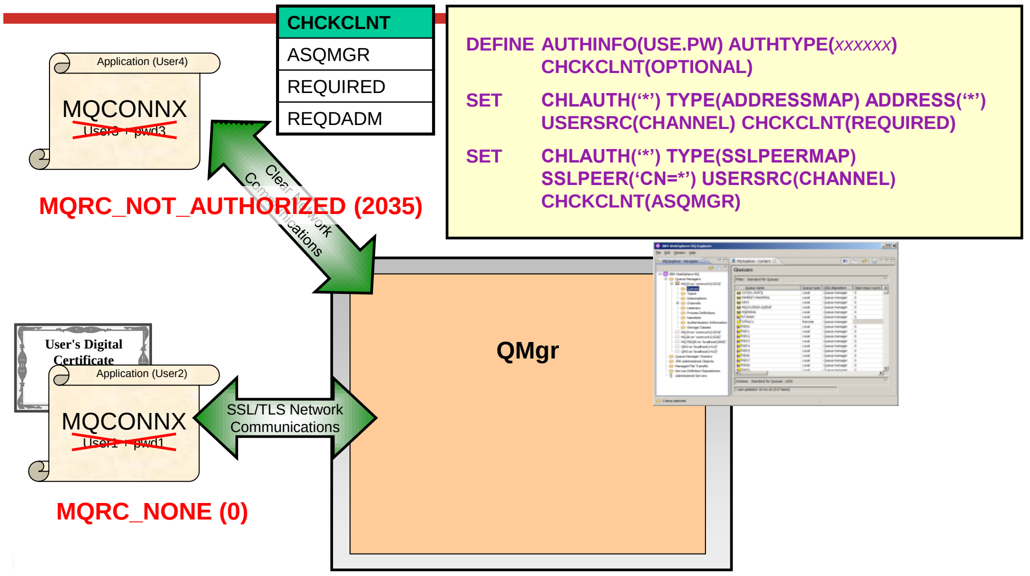
- **Application**
  - MQRC_NOT_AUTHORIZED (2035)
- **Administrator**
  - Error message
- **Monitoring Tool**
  - Not Authorized Event message (Type 1 – Connect)
  - MQRQ_CONN_NOT_AUTHORIZED (existing)
    - Connection not authorized.
  - MQRQ_CSP_NOT_AUTHORIZED (new)
    - User ID and password not authorized.
  - Additional field to existing connect event
    - MQCACF_CSP_USER_IDENTIFIER

Connection Authentication – Error notification – Notes

N O T E S

- When an application provides a user ID and password which fail the password check, the application is returned the standard MQ security error, 2035 – MQRC_NOT_AUTHORIZED.
- The MQ administrator will see this reported in the error log and can therefore see that the application was rejected due to the user ID and password failing the check, rather than, for example, a lack of connection authority (+connect).
- A monitoring tool can also be notified of this failure if authority events are on - ALTER QMGR AUTHOREV(ENABLED) – via an event message to the SYSTEM.ADMIN.QMGR.EVENT queue. This Not Authorized event is a Type 1 – Connect – event and provides all the same fields as the existing Type 1 event, along with one, additional field, the MQCSP user ID provided. The password is not provided in the event message. This means that there are two user IDs in the event message, the one the application is running as and the one the application presented for user ID and password checking.

Connection Authentication – Configuration Granularity


Application (User4)
MQCONNX
~~User3 + pwd3~~
MQRC_NOT_AUTHORIZED (2035)
Clear Network Communications
User's Digital Certificate
Application (User2)
MQCONNX
~~User1 + pwd1~~
SSL/TLS Network Communications
MQRC_NONE (0)
QMgr

CHCKCLNT
ASQMGR
REQUIRED
REQDADM

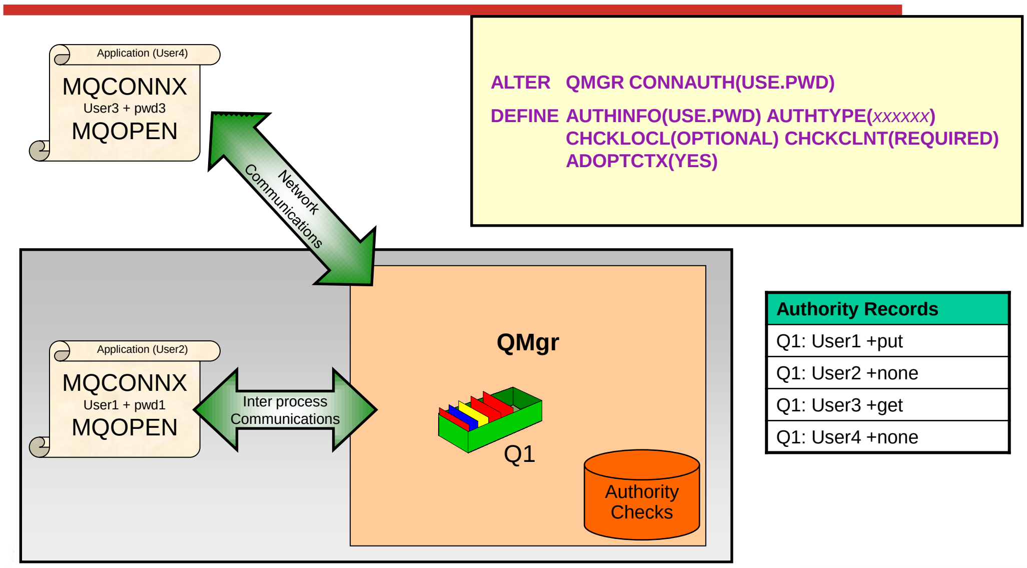
```
DEFINE  AUTHINFO(USE.PW) AUTHTYPE(xxxxxx)
        CHCKCLNT(OPTIONAL)

SET     CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*')
        USERSRC(CHANNEL) CHCKCLNT(REQUIRED)

SET     CHLAUTH('*') TYPE(SSLPEERMAP)
        SSLPEER('CN=*') USERSRC(CHANNEL)
        CHCKCLNT(ASQMGR)
```

Connection Authentication – Configuration Granularity – Notes

N O T E S

- In addition to the two fields that turn this on overall for client and locally bound applications, there are enhancements to the CHLAUTH rules so that more specific configuration can be made using CHCKCLNT. You can set the overall CHCKCLNT value to OPTIONAL, and then upgrade it to be more stringent for certain channels by setting CHCKCLNT to REQUIRED or REQDADM on the CHLAUTH rule. By default, CHLAUTH rules will run with CHCKCLNT(ASQMGR) so this granularity does not have to be used.

Connection Authentication – Relationship to Authorization



```
ALTER   QMGR CONNAUTH(USE.PWD)
DEFINE  AUTHINFO(USE.PWD) AUTHTYPE(xxxxxx)
        CHCKLOCL(OPTIONAL) CHCKCLNT(REQUIRED)
        ADOPTCTX(YES)
```

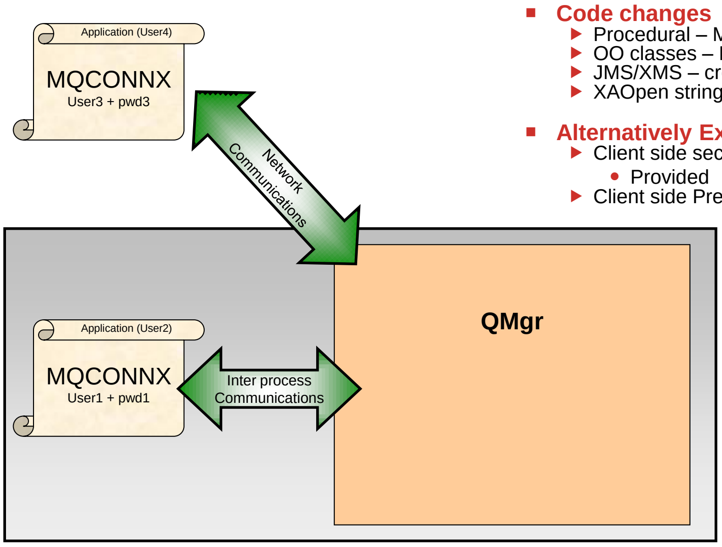
Authority Records
Q1: User1 +put
Q1: User2 +none
Q1: User3 +get
Q1: User4 +none

Connection Authentication – Relationship to Authorization – Notes

N O T E S

- So we have seen that we can configure our queue manager to mandate user IDs and passwords are provided by certain applications. We know that the user ID that the application is running under may not be the same user ID that was presented by the application along with a password. So what is the relationship of these user IDs to the ones used for the authorization checks when the application, for example, opens a queue for output.
- There are two choices, in fact, controlled by an attribute on the authentication information object – ADOPTCTX.
- You can choose to have applications provide a user ID and password for the purposes of authenticating them at connection time, but then have them continue to use the user ID that they are running under for authorization checks. This may be a useful stepping stone when migrating, or even a desirable mode to run in, perhaps with client connections, because authorization checks are being done using an assigned MCAUSER based on IP address or SSL/TLS certificate information.
- Alternatively, you can choose the applications to have all subsequent authorization checks made under the user ID that you authenticated by password by selecting to adopt the context as the applications context for the rest of the life of the connection.
- If the user ID presented for authentication by password is the same user ID that the application is also running under, then of course this setting has no effect.

Connection Authentication – Application changes



- **Code changes**
  - Procedural – MQCSP on MQCONNX
  - OO classes – MQEnvironment
  - JMS/XMS – createConnection
  - XAOpen string
- **Alternatively Exits can provide MQCSP**
  - Client side security exit
    - Provided
  - Client side Pre-conn exit

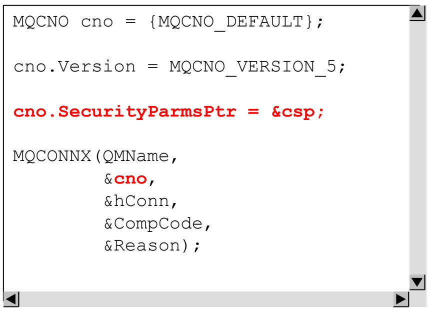
Connection Authentication – Application changes – Notes

N O T E S

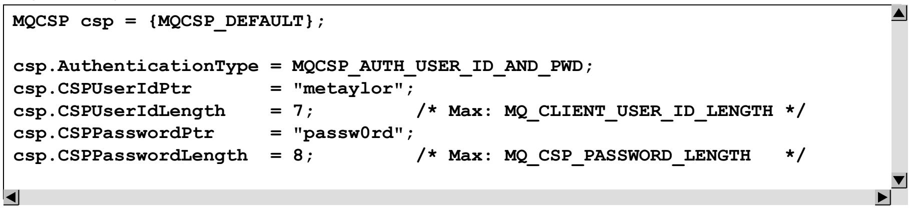
- Since MQ V6.0, an application has been able to provide a user ID and password (in the Connection Security Parameters (MQCSP) structure in the MQCNO) at MQCONNX time. These were passed to a user written plug-point in the OAM on distributed to be checked. If the application was running client bound, this user ID and password were also passed to the client side and server side security exits for processing and can be used for setting the MCAUser attribute of a channel instance. The security exit is called with ExitReason MQXR_SEC_PARMS for this processing.
- This pre-existing feature of the MQI is being used to provide the user ID and password to the queue manager for checking. Previously a custom Authorization Service was required to check this (or a security exit if the applications were connecting as clients), now the Object Authority Manager (OAM) supplied with the queue manager and the z/OS Security component within the queue manager will deal with these user IDs and passwords. Whether z/OS or distributed, the component that deals with the user IDs and passwords will call out to a facility outside of MQ to do the check – more on that later.
- In MQ V8 this will be available in all our interfaces listed, even where some of those were not made available in the MQ V6 timeframe when the programming interface was originally provided.
- In prior releases the MQCSP had no architected limits on the user ID and password strings that were provided by the application. When using them with these MQ provided features there are limits which apply to the use of these features, but if you are only passing them to your own exits, those limits do not apply.
- The XAOpen string has also been updated to allow the provision of a user ID and password.
- Sometimes of course, it can be hard to get changes into applications, so the user ID and password can be provided using an exit instead of changing the code. Client side security exits or the pre-connect exit, can make changes to the MQCONN before it is sent to the queue manager, and the security exit in fact is designed to allow the setting of the MQCSP since V6 (so clients do not need to be updated to the new version in order to use this).

Procedural MQI changes

- **MQCSP structure**
  - Connection Security Parameters
  - User ID and password
- **MQCNO structure**
  - Connection Options
- **MQ V6**
  - Passed to OAM (Dist only)
  - Also passed to Security Exit
    - Both z/OS and Distributed
    - MQXR_SEC_PARMS
- **MQ V8**
  - Acted upon by the queue manager (all platforms)

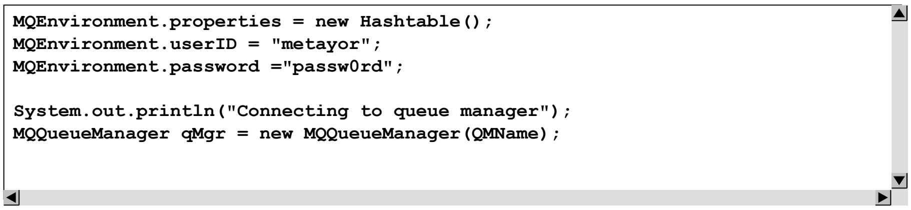


```
MQCNO cno = {MQCNO_DEFAULT};

cno.Version = MQCNO_VERSION_5;

cno.SecurityParmsPtr = &csp;

MQCONNX(QMName,
        &cno,
        &hConn,
        &CompCode,
        &Reason);
```



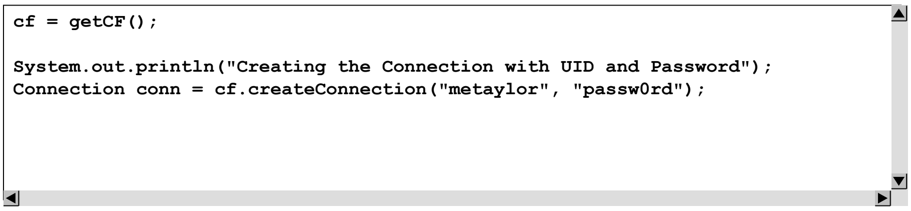
```
MQCSP csp = {MQCSP_DEFAULT};

csp.AuthenticationType = MQCSP_AUTH_USER_ID_AND_PWD;
csp.CSPUserIdPtr       = "metaylor";
csp.CSPUserIdLength    = 7;          /* Max: MQ_CLIENT_USER_ID_LENGTH */
csp.CSPPasswordPtr     = "passw0rd";
csp.CSPPasswordLength  = 8;          /* Max: MQ_CSP_PASSWORD_LENGTH   */
```

Object Oriented MQ classes changes



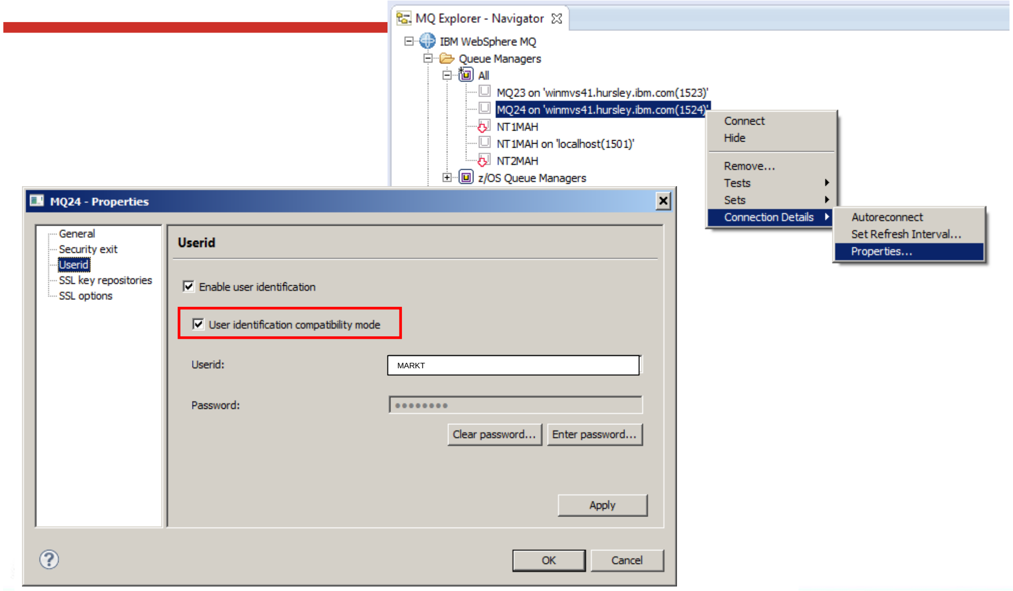
```
MQEnvironment.properties = new Hashtable();
MQEnvironment.userID = "metayor";
MQEnvironment.password ="passw0rd";

System.out.println("Connecting to queue manager");
MQQueueManager qMgr = new MQQueueManager(QMName);
```

JMS/XMS classes changes



```
cf = getCF();

System.out.println("Creating the Connection with UID and Password");
Connection conn = cf.createConnection("metaylor", "passw0rd");
```

Using it from the MQ Explorer GUI



Userid

☑ Enable user identification

☑ User identification compatibility mode

Userid: MARKT

Password: ••••••••

Using it from the MQ Explorer GUI – Notes

N O T E S

- The MQ Explorer GUI is an MQ Java™ application, so since there is a programming interface for MQ Java to supply a user ID and password, the Explorer GUI can use this.
- To configure the Explorer to use a user ID and password on a connection to a queue manager (whether local or client connection), select Connection Details->Properties… from the right-mouse context menu on the queue manager. In the dialog that appears, choose UserId. This panel is the same for both local or client connections in MQ V8, although the Properties dialog will have less selections for other things in the local case.
- Explorer has a password cache which will need to be enabled in order to use passwords. If you have never used it before there will be a link on this panel to take you through it.
- The other interesting item here is the “User identification compatibility mode” check box. This is for those of you who have been using Security exits with the Explorer in the past. The Java client previously did not use the MQCSP structure to supply its user ID and password in previous releases, and there are many exits written that have discovered where the user ID and password were provided instead. In order to retain compatibility for this, the Java client has two modes. It can run in compatibility mode and maintain what you had before, or it can run with the V8 mode and use the MQCSP. The check box shown is how you set that property in the Explorer GUI. For other Java applications, you need to set property to indicate you are happy to use the MQCSP method.
- At the queue manager, if no MQCSP is sent by a client, but the user ID and password are provided in this alternate method that was utilised by Java Clients, the V8 queue manager will accept this and drive the same password check as is used for the MQCSP provided passwords.

Using MQCSP from Java Client

- **Java client (not local bindings) has two ways to send password**
  - FAP Flow
  - MQCSP structure

- **FAP Flow**
  - Mechanism used by many customer security exits
  - Retained as default
  - Restricted to 8 characters user IDs and passwords
  - Not protection by password protection algorithm
  - Used by Connection Authentication if seen and no MQCSP found

- **MQCSP structure**
  - Used by Java Client when property set
  - Non-default
  - Allows longer user IDs and passwords
  - Can be protection by password protection algorithm

MQ Classes for Java

set the property **MQConstants.USE_MQCSP_AUTHENTICATION_PROPERTY** to true in the properties hashtable passed to the com.ibm.mq.MQQueueManager constructor.

MQ Classes for JMS

set the property **JMSConstants.USER_AUTHENTICATION_MQCSP** to true on the appropriate connection factory prior to creating the connection

Globally

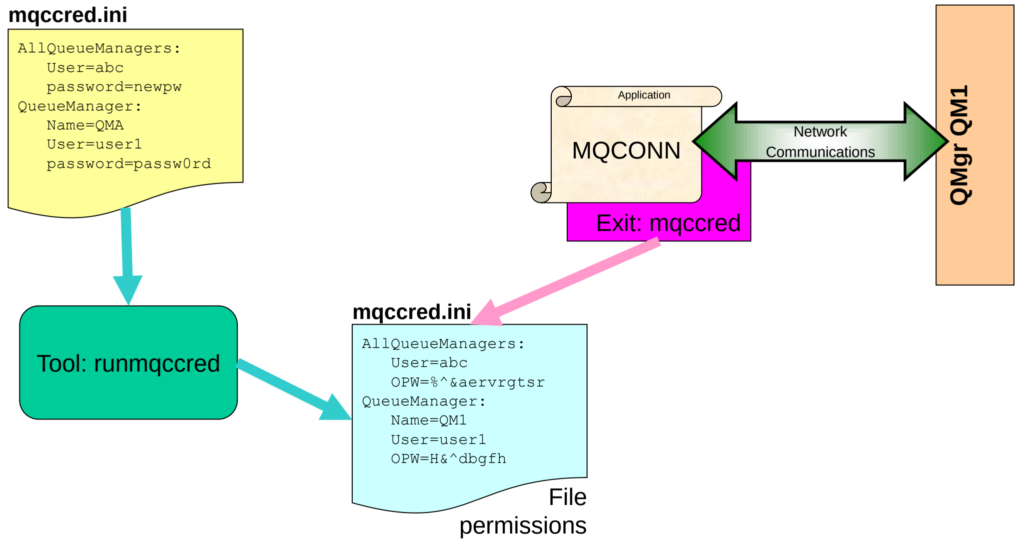
set the **System Property "com.ibm.mq.cfg.jmqi.useMQCSPauthentication"** to a value indicating true, for example by adding **"-Dcom.ibm.mq.cfg.jmqi.useMQCSPauthentication=Y"** to the command line

Using MQCSP from Java - Notes

N O T E S

- We saw on a previous page the example code you might use to provide the user ID and password from a Java classes application or a JMS application. This is actually nothing new. Java clients have been able to send a user ID and password across the channel FAP before. This part of the FAP was very restrictive though, it only allowed or 8 character user IDs and 8 character passwords. And, of course, it was only for clients. The MQCSP interface was designed not to have such limitations.
- There are quite a number of customers pre-V8 who have security exits written to pull the user ID and password sent by Java clients in this way. Because of this, we could not change the default of the Java clients over to use the MQCSP or all these security exits would have to be changed. So by default, Java clients continue to send the user ID and password as this restrictive FAP flow.
- On the queue manager end, if we receive a user ID and password in this FAP flow, and no MQCSP structure, we will use the user ID and password in the FAP flow for Connection Authentication, so you don’t have to make any changes in order to remove a security exit that is checking the user ID and password in this way.
- However, there are benefits to using the MQCSP structure, including password protection and the increased length of the fields, so when you are ready to change over to use MQCSP instead of the FAP flow in a Java client, you need to set the system property.

Client side Security Exit



```
AllQueueManagers:
   User=abc
   password=newpw
QueueManager:
   Name=QMA
   User=user1
   password=passw0rd
```

```
AllQueueManagers:
   User=abc
   OPW=%^&aervrgtsr
QueueManager:
   Name=QM1
   User=user1
   OPW=H&^dbgfh
```

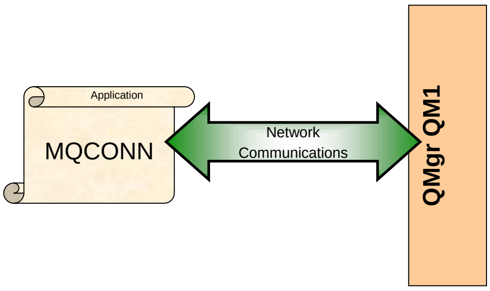
Client side Security Exit – Notes

N O T E S

- To make changes to applications, especially the very prevalent client attached applications where we see the strongest use case for using user ID and password, is difficult for customers. To aid with this issue, MQ V8 provides a client side security exit which can set the user ID and password instead of making changes in the application to do this.
- The exit runs at the CLNTCONN end of the channel and pulls the user ID and the password from a file. This file is controlled by means of OS file permissions. If the exit discovers that the file permissions are too open, it will cause a failure thus ensuring that this important part of protecting the passwords does not go unnoticed.
- The file is additionally obfuscated from casual browsers. The algorithm for this obfuscation is not published, and neither is the source of the exit.
- The exit will be built in such a way that it can be picked up from a V8 installation and copied to a V7.0.1 client installation (or later). Note that using a client installation of < V8 will mean you have the password flowed in the clear. Only V8 and later at both ends will provide the ability to protect the flowed password without the need to use SSL/TLS.
- Along with the exit, we also supply a tool which is used to obfuscate the file containing the passwords.

Protecting your password across a network

- **Use SSL/TLS**
  - Perhaps with anonymous clients



- **If no SSL/TLS**
  - If both ends are V8
  - MQ Code will protect the password – so not sent in the clear
- **If client is < V8**
  - No MQ password protection
  - Consider SSL/TLS

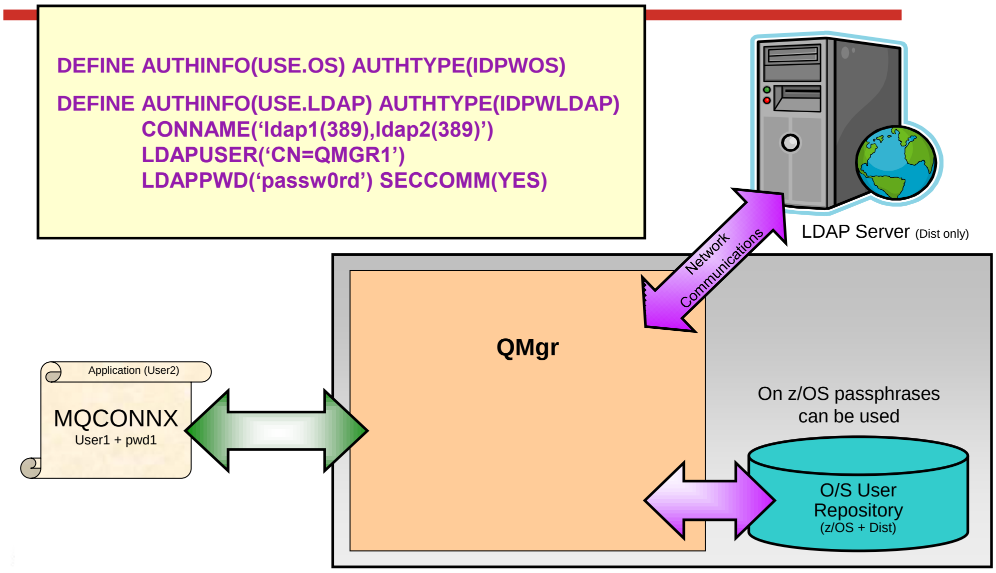
Protecting your password across a network – Notes

NOTES

- When an application connects to a MQ V8 queue manager across the network, i.e. making a client connection, the password it sends for connection authentication purposes travels across the network from the client application to the queue manager for checking. This password should be protected as it does so, so that network sniffers cannot obtain your password.
- For best possible protection, you can of course use SSL/TLS. You might imagine using anonymous SSL/TLS, i.e. the client does not have a certificate, since you are using user ID and password as the means by which to verify the identity of the client application.
- If you do not use SSL/TLS, and your client is at V8.0 or later, the MQ product code will protect your password so that it is not sent in the clear. A good reason to get your clients upgraded to V8!
- If your MQ Client is at a version earlier than V8.0, it can still send user ID and passwords (since the MQCSP structure has been around since V6) but the password will not be protected, so you should consider using SSL/TLS.

Connection Authentication – User Repositories

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



Connection Authentication – User Repositories – Notes

N
O
T
E
S

- So far we have spoken about user ID and password authentication without mentioning what is actually doing the authentication. We’ve also shown that there is a new type of authentication information object without showing you the object type. Here we introduce two new object types of authentication information objects.
- The first type is used to indicate that the queue manager is going to use the local O/S to authentication the user ID and password. This type is IDPWOS.
- The second type is used to indicate that the queue manager is going to use an LDAP server to authenticate the user ID and password. This type is IDPWLDAP and is not applicable on z/OS.
- Only one type can be chosen for the queue manager to use by naming the appropriate authentication information object in the queue manager’s CONNAUTH attribute.
- We have already covered everything there is to say about the configuration of the O/S as the user repository as the common attributes are all there is for the O/S. There is more to say about the LDAP server as an option though.
- Some of the LDAP server configuration attributes are probably fairly obvious. The CONNAME is how the queue manager knows where the LDAP server is, and SECCOMM controls whether connectivity to the LDAP server will be done using SSL/TLS or not. The LDAPUSER and LDAPPWD attributes are how the queue manager binds to the LDAP server so that it can look-up information about user records. It is likely this may be a public area of an LDAP server, so these attributes may not be needed.
- It is worth highlighting that the CONNAME field can be used to provide additional addresses to connect to for the LDAP server in a comma-separated list. This can aid with redundancy if the LDAP server does not provide such itself.

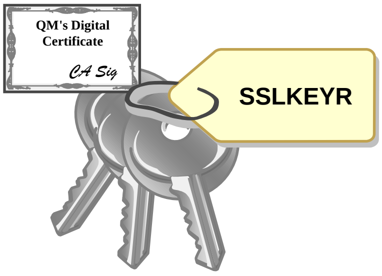
Support for PAM on Unix platforms

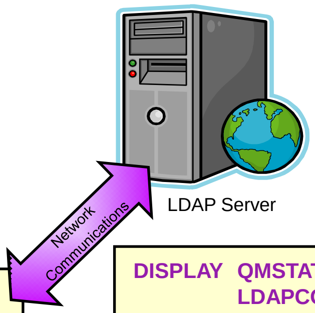
- **V8.0.0.3 extends the OS authentication to call PAM**
  - Allows range of authentication mechanisms to be hidden behind common API
  - Lots of customer requests for it as an enhancement since GA
- **PAM is set up by root in either /etc/pam.conf or files in /etc/pam.d**
  - MQ is known as the "ibmmq" service in PAM configuration
- **AUTHINFO(IDPWOS) objects extended with AUTHENMD attribute**
  - Can be set to **OS** (GA capability) or **PAM** – REFRESH SECURITY to activate
- **Requires updated CMDLEVEL=802 (ie V8.0.0.3) to set AUTHENMD**
- **More on youtube at https://youtu.be/3VW4Op5QQfk**

NOTES

- intentionally blank

Secure connection to an LDAP Server





```
ALTER   QMGR CONNAUTH(USE.LDAP)
        SSLFIPS(NO) SUITEB(NONE)
        CERTLABL('ibmmqqm1')
        SSLKEYR('/var/mqm/qmgrs/QM1/ssl/key')

DEFINE  AUTHINFO(USE.LDAP)
        AUTHTYPE(IDPWLDAP)
        SECCOMM(YES)
        CONNAME('ldapserver(389)')
```

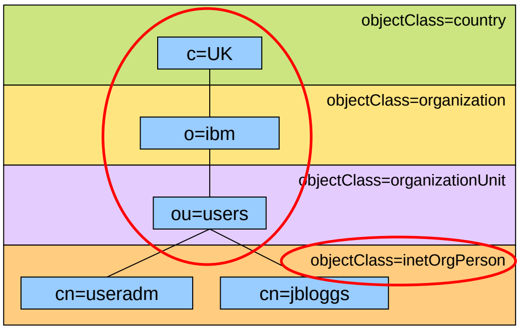
```
DISPLAY  QMSTATUS
         LDAPCONN
```

Secure connection to an LDAP Server – Notes

N O T E S

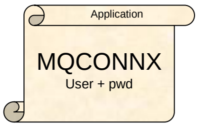
- Unlike on channels, there is no SSLCIPH parameter to turn on the use of SSL/TLS for the communication with the LDAP server. In this case MQ is acting as a client to the LDAP server so much of the configuration will be done at the LDAP server. Some existing parameters in MQ will be used to configure how that connection will work as shown on this slide.
- The overall switch to choose SSL/TLS communication or not, we already saw on the previous page – SECCOMM.
- In addition to this attribute, we will also pay attention to the queue manager attributes SSLFIPS and SUITEB to restrict the set of cipher specs that will be chosen. The certificate that will be used to identify the queue manager to the LDAP server will be the queue manager certificate, either 'ibmmq<qmgr-name>' or the newly added CERTLABL attribute which we'll talked about in an earlier section of this presentation.
- Certificate revocation will be checked by using the OCSP servers that are named in the AuthorityInfoAccess (AIA) certificate extensions. This can be turned off by using the qm.ini SSL stanza attribute OCSPCheckExtensions.
- Connection to an LDAP Server is made as a network connection (which is why you may wish to consider using a secure connection). The status of this connection from the queue manager to the LDAP server is shown in DISPLAY QMSTATUS.

LDAP User Repository



LDAP Server

```
DEFINE  AUTHINFO(USE.LDAP)
        AUTHTYPE(IDPWLDAP)
        CONNAME('ldapserver(389)')
        CLASSUSR('inetOrgPerson')
        BASEDNU('ou=users,o=ibm,c=uk')
        USRFIELD('cn')
```



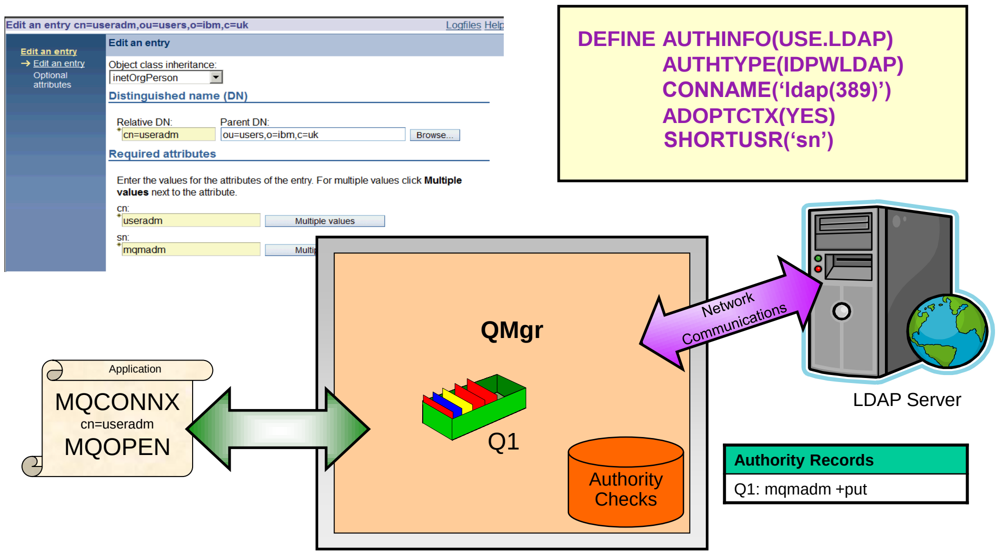
Application provides	USRFIELD	BASEDNU
cn=useradm,ou=users,o=ibm,c=uk		
cn=useradm		**Adds ou=users,o=ibm,c=uk**
useradm	**Adds cn=**	**Adds ou=users,o=ibm,c=uk**

LDAP User Repository – Notes

N O T E S

- When using an LDAP user repository there is some more configuration to be done on the queue manager other than just to tell the queue manager where the LDAP repository resides.
- User IDs records defined in an LDAP server have a hierarchical structure in order to uniquely identify them. So an application could connect to the queue manager and present its user ID as being the fully qualified hierarchical user ID. This however is a lot to provide and it would be simpler if we could configure the queue manager to say, assume all user IDs that are presented are found in this area of the LDAP server and add that qualification onto anything you see. This is what the BASEDNU attribute is for. It identifies the area in the LDAP hierarchy that all the user IDs are to be found. Or to look at it another way, the queue manager will add the BASEDNU value to the user ID presented by an application to fully qualify it before looking it up in the LDAP server.
- Additionally, your application may only want to present the user ID without providing the LDAP attribute name, e.g. CN=. This is what the USRFIELD is for. Any user ID presented to a queue manager without an equals sign (=) will have the attribute and the equals sign pre-pended to it, and the BASEDNU value post-pended to it before looking it up in the LDAP server. This may be a useful migratory aid when moving from O/S user IDs to LDAP user IDs as the application could very well be presenting the same string in both cases, thus avoiding any change to the application.

Relationship to Authorization – LDAP



Edit an entry cn=useradm,ou=users,o=ibm,c=uk

Edit an entry

Object class inheritance: inetOrgPerson

Distinguished name (DN)

Relative DN: * cn=useradm

Parent DN: ou=users,o=ibm,c=uk

Required attributes

Enter the values for the attributes of the entry. For multiple values click **Multiple values** next to the attribute.

cn: * useradm

sn: * mqmadm

```
DEFINE AUTHINFO(USE.LDAP)
       AUTHTYPE(IDPWLDAP)
       CONNAME('ldap(389)')
       ADOPTCTX(YES)
       SHORTUSR('sn')
```

Authority Records

Q1: mqmadm +put

Relationship to Authorization – LDAP - Notes

N O T E S

- We spoke earlier about the ability to adopt the authenticated user ID as the context for this connection. So how does this work if you are using LDAP as the user repository but your authorization is being done using O/S user IDs?
- We need to get a user to represent the LDAP user that has been presented, as an O/S user ID. We find this from the LDAP user record. This can be any field that is defined in the user record, perhaps something like the short name field (sn=) that is a mandatory part of the definition of the inetOrgPerson class, or perhaps something defined more specifically for the purpose such as a user ID (uid=) field.
- The queue manager will use that information to determine what O/S user ID will be used as the context for this connection. You configure it using SHORTUSR to say what the field to locate in the user record is.

Authorization using LDAP credentials

FixPac 8.0.0.2

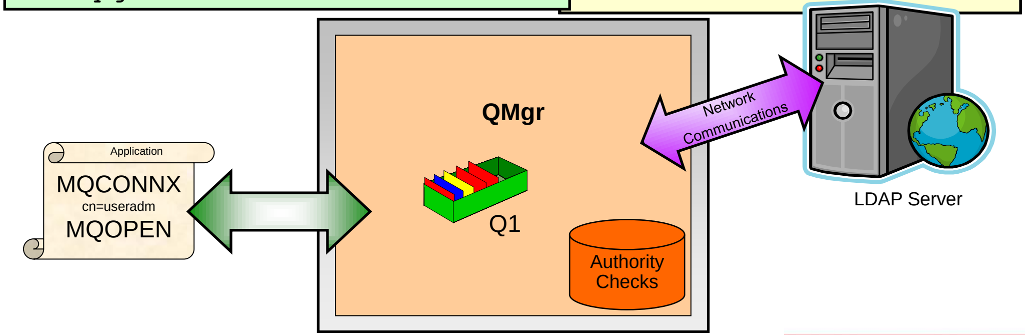
```
setmqaut -g admin
  -t qmgr +connect
```

Equivalent attributes

Users	Groups
CLASSUSR	CLASSGRP
BASEDNU	BASEDNG
USRFIELD	GRPFIELD

```
setmqaut -g cn=admin,ou=groups,o=ibm,c=uk
  -t qmgr +connect
```

```
ALTER  AUTHINFO(USE.LDAP)
       AUTHTYPE(IDPWLDAP)
       AUTHORMD(SEARCHGRP)
       FINDGRP('member')
       CLASSGRP('groupOfNames')
       BASEDNG('ou=groups,o=ibm,c=uk')
       GRPFIELD('cn')
       NESTGRP(NO)
```



Authorization using LDAP credentials - Notes

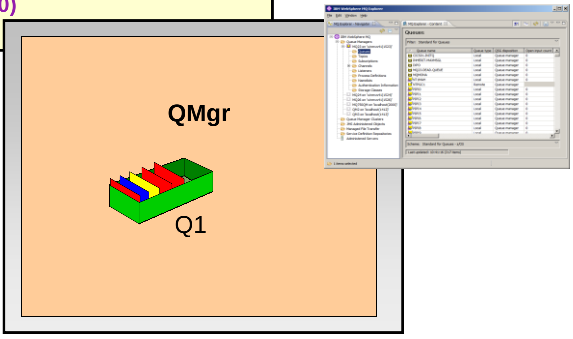
N O T E S

- In FixPac 8.0.0.2 and the MQ Appliance, there is now the option, on UNIX queue managers, to choose to have the authorization checks done using the presented LDAP credentials, instead of the behaviour on the previous page where they are mapped to an OS user for authorization checks.
- In order to use this feature, you need to have your queue manager running with a command level (CMDLEVEL) of 801 which is an explicit action to increase, due to the function being delivered in a FixPac.
- Then we need to know a few more things about the shape of your LDAP user repository; i.e. where the groups live in the hierarchy.

Migration / Defaults

```
AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)
         AUTHTYPE(IDPWOS)
         CHCKLOCL(OPTIONAL)
         CHCKCLNT(REQDADM)
         FAILDLAY(1)
         DESCR( )
         ALTDATE(2013-12-25)
         ALTTIME(12.00.00)
```

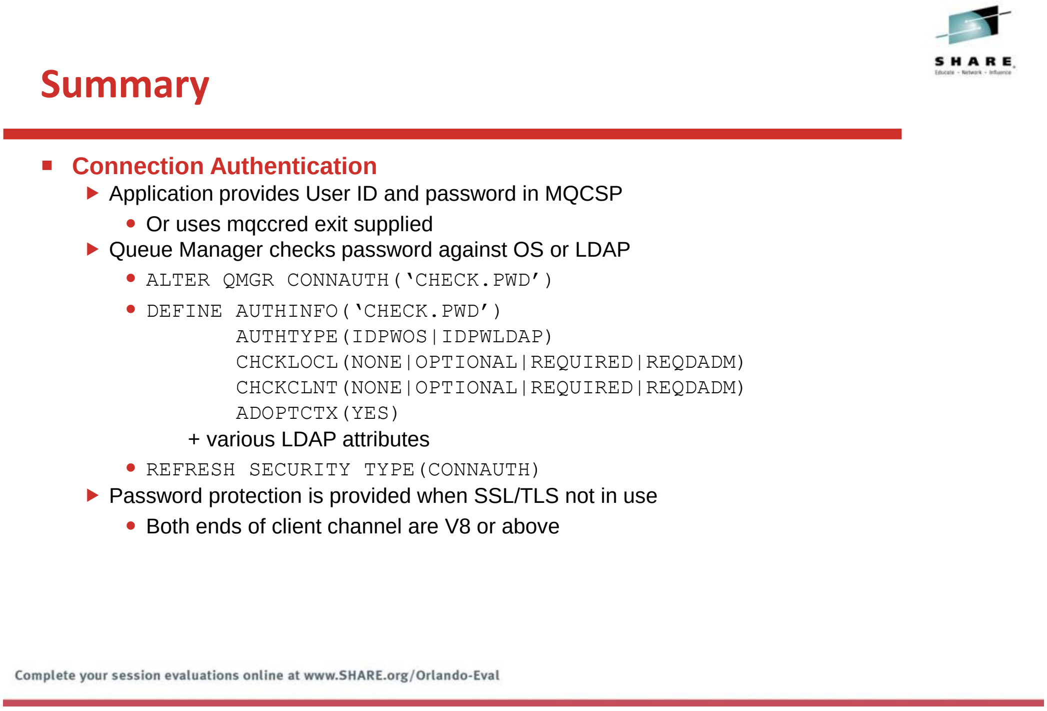
- **Defaults**
  - Migrated queue manager
    - CONNAUTH(' ')
  - New queue manager
    - CONNAUTH(⇦)



Migration / Defaults – Notes

N O T E S

- By default, a migrated queue manager will find that CONNAUTH is blank – and therefore connection authentication is switched off.
- A brand new queue manager created with the MQ V8 binaries will find that the CONNAUTH field points to the SYSTEM.DEFAULT.AUTHINFO.IDPWOS authentication information object.



Summary

- **Connection Authentication**
  - Application provides User ID and password in MQCSP
    - Or uses mqccred exit supplied
  - Queue Manager checks password against OS or LDAP
    - `ALTER QMGR CONNAUTH('CHECK.PWD')`

    -
      ```
      DEFINE AUTHINFO('CHECK.PWD')
             AUTHTYPE(IDPWOS|IDPWLDAP)
             CHCKLOCL(NONE|OPTIONAL|REQUIRED|REQDADM)
             CHCKCLNT(NONE|OPTIONAL|REQUIRED|REQDADM)
             ADOPTCTX(YES)
      ```
      + various LDAP attributes
    - `REFRESH SECURITY TYPE(CONNAUTH)`
  - Password protection is provided when SSL/TLS not in use
    - Both ends of client channel are V8 or above

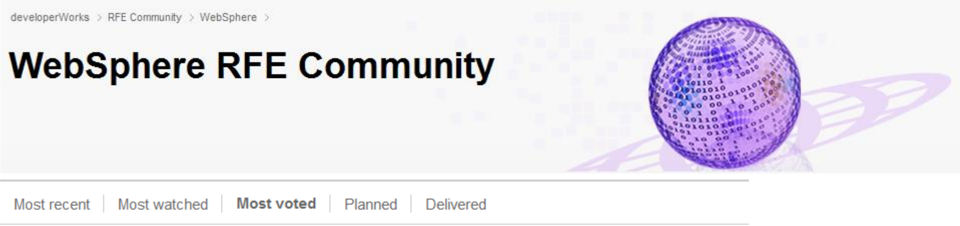
Hostnames in CHLAUTH

Agenda

- **Requests for Enhancement**
- **Channel Authentication Records**
  - Recap
  - Rules which use IP addresses
  - Hostnames
  - Precedence Order
  - Reverse Look-up of IP address
  - MATCH(RUNCHECK)

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

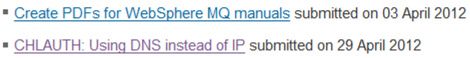
Request for Enhancement



developerWorks > RFE Community > WebSphere >

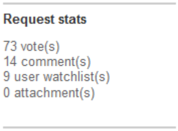
WebSphere RFE Community

Most recent | Most watched | **Most voted** | Planned | Delivered



- Create PDFs for WebSphere MQ manuals submitted on 03 April 2012
- CHLAUTH: Using DNS instead of IP submitted on 29 April 2012

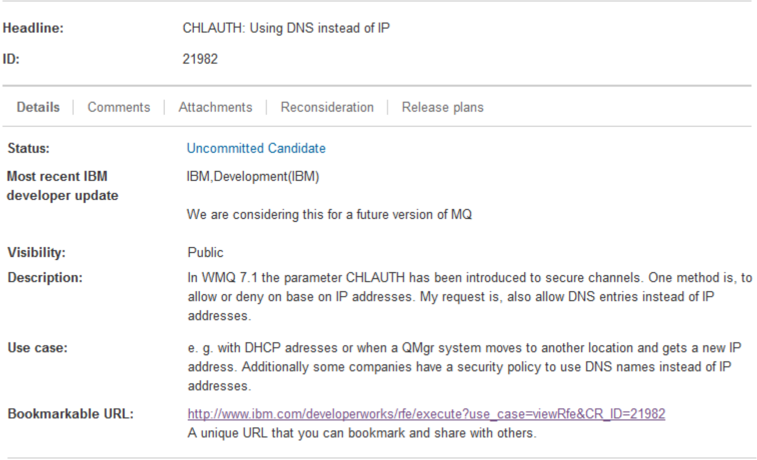
- Second in the Most voted list!



Request stats

73 vote(s)
14 comment(s)
9 user watchlist(s)
0 attachment(s)

Request for Enhancement (21892)



Headline:	CHLAUTH: Using DNS instead of IP
ID:	21982

Details | Comments | Attachments | Reconsideration | Release plans

Status:	Uncommitted Candidate
Most recent IBM developer update	IBM,Development(IBM) We are considering this for a future version of MQ
Visibility:	Public
Description:	In WMQ 7.1 the parameter CHLAUTH has been introduced to secure channels. One method is, to allow or deny on base on IP addresses. My request is, also allow DNS entries instead of IP addresses.
Use case:	e. g. with DHCP adresses or when a QMgr system moves to another location and gets a new IP address. Additionally some companies have a security policy to use DNS names instead of IP addresses.
Bookmarkable URL:	http://www.ibm.com/developerworks/rfe/execute?use_case=viewRfe&CR_ID=21982 A unique URL that you can bookmark and share with others.

Channel Authentication Records – Recap

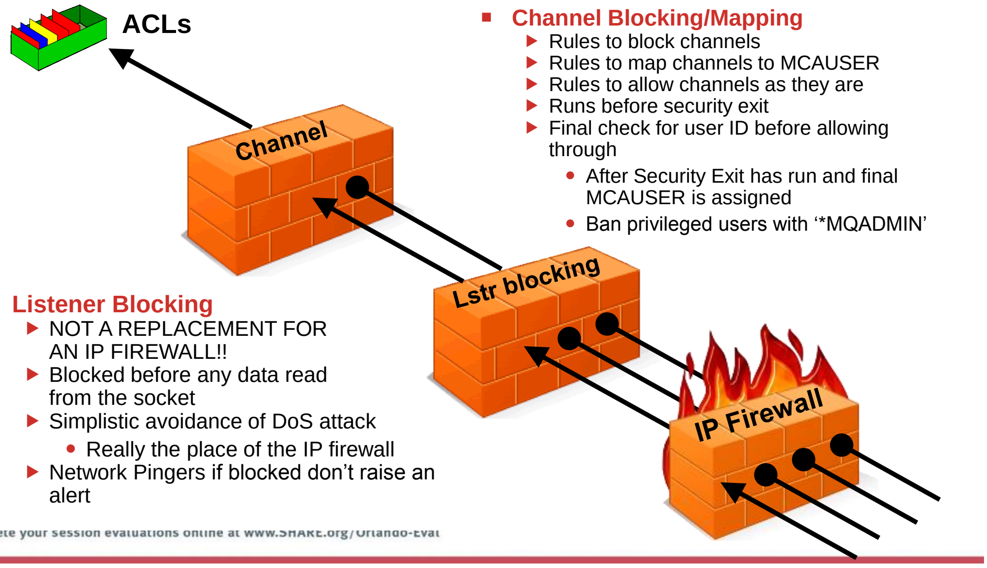
- **Set rules to control how inbound connections are treated**
  - Inbound Clients
  - Inbound QMgr to QMgr channels
  - Other rogue connections causing FDCs
- **Rules can be set to**
  - Allow a connection
  - Allow a connection and assign an MCAUSER
  - Block a connection
  - Ban privileged access
  - Provide multiple positive or negative SSL Peer Name matching
- **Rules can use any of the following identifying characteristics of the inbound connection**
  - IP Address
  - SSL/TLS Subject's Distinguished Name
  - Client asserted user ID
  - Remote queue manager name

Channel Authentication Records – Notes

N O T E S

- Channel Authentication records allow you to define rules about how inbound connections into the queue manager should be treated. Inbound connections might be client channels or queue manager to queue manager channels. These rules can specify whether connections are allowed or blocked. If the connection in question is allowed, the rules can provide a user ID that the channel should run with or indicate that the user ID provided by the channel (flowed from the client or defined on the channel definition) is to be used.
- These rules can therefore be used to
  - Set up appropriate identities for channels to use when they run against the queue manager
  - Block unwanted connections
  - Ban privileged users
- Which users are considered privileged users is slightly different depending on which platform you are running your queue manager on. There is a special value '*MQADMIN' which has been defined to mean "any user that would be privileged on this platform". This special value can be used in the rules that check against the final user ID to be used by the channel – TYPE(USERLIST) rules – to ban any connection that is about to run as a privileged user. This catches any blank user IDs flowed from clients for example.

Channel Access Blocking Points



- **Channel Blocking/Mapping**
  - Rules to block channels
  - Rules to map channels to MCAUSER
  - Rules to allow channels as they are
  - Runs before security exit
  - Final check for user ID before allowing through
    - After Security Exit has run and final MCAUSER is assigned
    - Ban privileged users with '*MQADMIN'

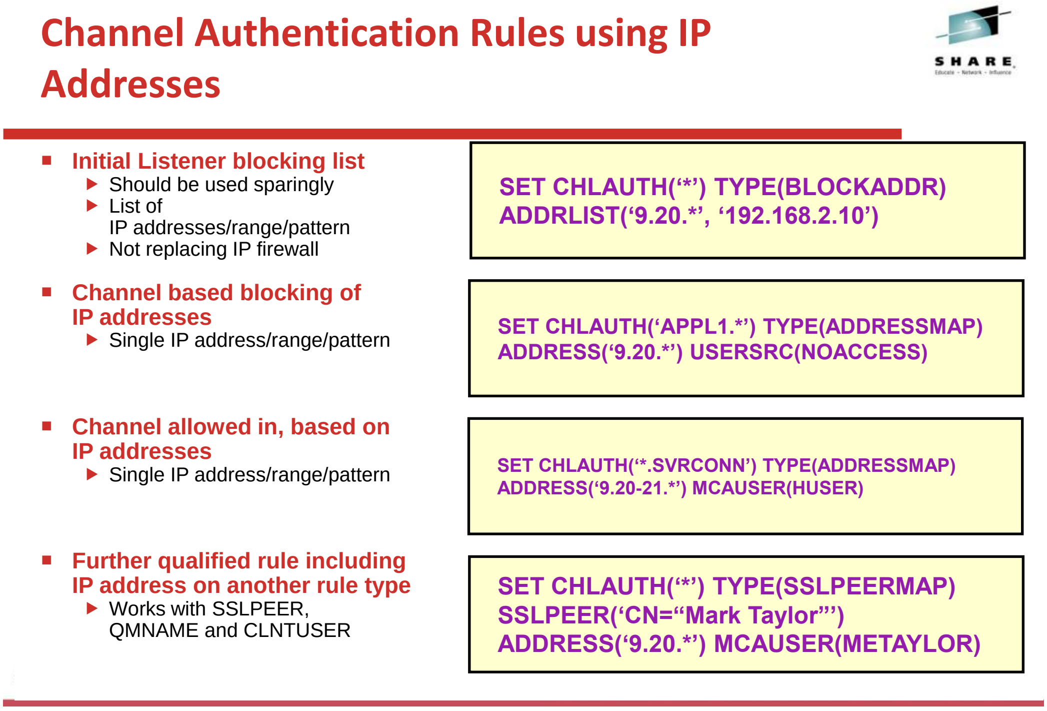
- **Listener Blocking**
  - NOT A REPLACEMENT FOR AN IP FIREWALL!!
  - Blocked before any data read from the socket
  - Simplistic avoidance of DoS attack
    - Really the place of the IP firewall
  - Network Pingers if blocked don't raise an alert

Channel Access Blocking Points – Notes

N O T E S

- In this picture we illustrate that there are a number of points that an inbound connection must get through in order to actually make use of an MQ queue.
- First, we remind you that your IP firewall is included in this set of blocking points and should not be forgotten, and is not superseded by this feature in MQ.

- One point of note, the inbound connections can be from any version of MQ. There is no requirement that the clients or remote queue managers also be on MQ V7.1 to be blocked or mapped by these rules.



Channel Authentication Rules using IP Addresses

- **Initial Listener blocking list**
  - Should be used sparingly
  - List of IP addresses/range/pattern
  - Not replacing IP firewall

```
SET CHLAUTH('*') TYPE(BLOCKADDR)
ADDRLIST('9.20.*', '192.168.2.10')
```

- **Channel based blocking of IP addresses**
  - Single IP address/range/pattern

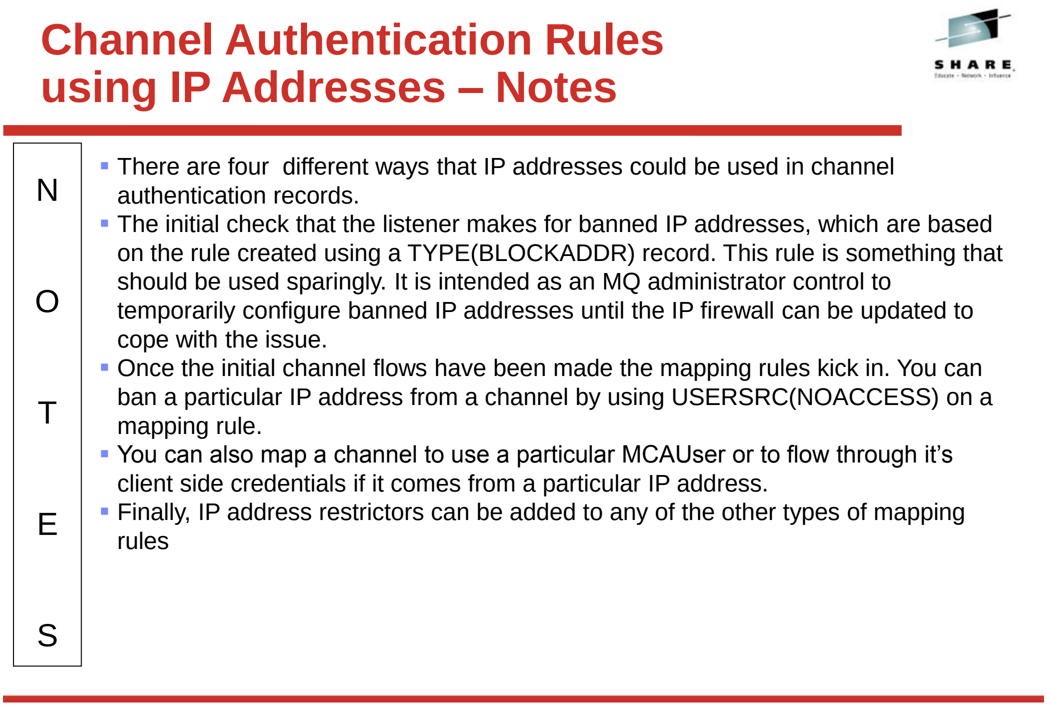
```
SET CHLAUTH('APPL1.*') TYPE(ADDRESSMAP)
ADDRESS('9.20.*') USERSRC(NOACCESS)
```

- **Channel allowed in, based on IP addresses**
  - Single IP address/range/pattern

```
SET CHLAUTH('*.SVRCONN') TYPE(ADDRESSMAP)
ADDRESS('9.20-21.*') MCAUSER(HUSER)
```

- **Further qualified rule including IP address on another rule type**
  - Works with SSLPEER, QMNAME and CLNTUSER

```
SET CHLAUTH('*') TYPE(SSLPEERMAP)
SSLPEER('CN="Mark Taylor"')
ADDRESS('9.20.*') MCAUSER(METAYLOR)
```



Channel Authentication Rules using IP Addresses – Notes

N O T E S

- There are four different ways that IP addresses could be used in channel authentication records.
- The initial check that the listener makes for banned IP addresses, which are based on the rule created using a TYPE(BLOCKADDR) record. This rule is something that should be used sparingly. It is intended as an MQ administrator control to temporarily configure banned IP addresses until the IP firewall can be updated to cope with the issue.
- Once the initial channel flows have been made the mapping rules kick in. You can ban a particular IP address from a channel by using USERSRC(NOACCESS) on a mapping rule.
- You can also map a channel to use a particular MCAUser or to flow through it's client side credentials if it comes from a particular IP address.
- Finally, IP address restrictors can be added to any of the other types of mapping rules

Channel Authentication Rules using Hostnames

- **Initial Listener blocking list**
  - Hostnames not allowed

~~SET CHLAUTH(‘*’) TYPE(BLOCKADDR) ADDRLIST( )~~

- **Channel based blocking of Hostnames**
  - Single IP address/range/pattern or hostname/pattern

```
SET CHLAUTH(‘APPL1.*’) TYPE(ADDRESSMAP)
ADDRESS(‘*.ibm.com’) USERSRC(NOACCESS)
```

- **Channel allowed in, based on Hostnames**
  - Single IP address/range/pattern or hostname/pattern

```
SET CHLAUTH(‘*.SVRCONN’) TYPE(ADDRESSMAP)
ADDRESS(‘mach123.ibm.com’) MCAUSER(HUSER)
```

- **Further qualified rule including hostname on another rule type**
  - Works with SSLPEER, QMNAME and CLNTUSER

```
SET CHLAUTH(‘*’) TYPE(SSLPEERMAP)
SSLPEER(‘CN=“Mark Taylor”’)
ADDRESS(‘s*.ibm.*’) MCAUSER(METAYLOR)
```

Channel Authentication Rules using Hostnames – Notes

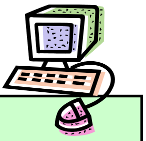
N O T E S

- Hostnames can be used in almost all places in channel authentication records that IP address could be used. The one exception to this is the TYPE(BLOCKADDR) record. This is only going to accept IP addresses.
- If you want to block IP addresses with CHLAUTH rules permanently in MQ, rather than via your IP firewall, you should be doing it using the TYPE(ADDRESSMAP) record and specifying USERSRC(NOACCESS). This type of rules will allow hostnames as well.
- Additionally, positive mapping records allow hostnames, and address restrictors can also use hostnames.
- Channel Authentication rules utilise pattern matching to allow the most flexible control. IP Addresses have a special form of pattern matching that includes ranges and wildcards within each ‘.’ (or ‘:’ for IPv6) section of an IP address. Other pattern matching which is done on channel names, and queue manager names is simpler with just wild-carded string matching (in other words dots are not considered special).
- Hostnames also have pattern matching applied to them – as for channel names and queue manager names. That is it is just a wild-carded string matching and separators such as dots are not considered special.

Precedence Order

```
DISPLAY CHLAUTH(APPL1.*)
returns ===>
  CHLAUTH(APPL1.*)
  TYPE(SSLPEERMAP)
  SSLPEER(‘O=”IBM UK”’) MCAUSER(UKUSER)

  CHLAUTH(APPL1.*)
  TYPE(USERMAP)
  CLNTUSER(‘metaylor’) MCAUSER(METAYLOR)

  CHLAUTH(APPL1.*)
  TYPE(ADDRESSMAP)
  ADDRESS(‘9.180.165.163’) MCAUSER(METAYLOR)

  CHLAUTH(APPL1.*)
  TYPE(ADDRESSMAP)
  ADDRESS(‘*.ibm.com’) MCAUSER(IBMUSER)
```

Order	Identity mechanism	Notes
0	Channel Name	
1	SSL Distinguished Name	
2=	Client asserted User ID	Clearly several different user IDs can be running on the same IP address.
2=	Queue Manager Name	Clearly several different queue managers can be running on the same IP address
4	IP address	
5	Hostname	One IP address can have multiple hostnames



Chl: APPL1.SVRCONN
DN: CN=M Taylor,O=IBM UK
UID: metaylor
IP: 9.180.165.163

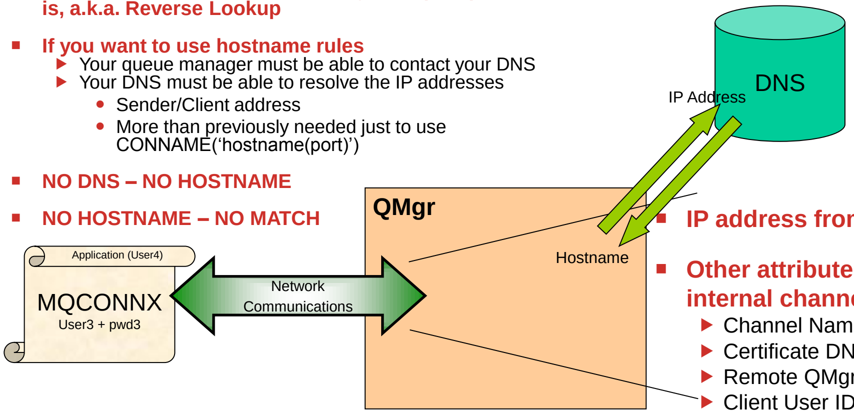
Precedence Order – Notes

N O T E S

- Rules created using Channel Authentication Records follow a precedence order so that it is clear which rule will be used when an inbound connection could have match multiple rules.
- Hostnames are added to the precedence order at the very bottom. They are considered to be less specific than an IP address because a single IP address can have multiple hostnames.
- If you have an IP address rule and a hostname rule that could both match an inbound connection, then the IP address rule will be the one that is used, as it is considered to be more specific.

Obtaining a hostname

- Hostname is not 'sent' from the other end of the channel
- IP address is obtained from TCP/IP socket
- We must ask the Domain Name System (DNS) Server what the hostname is, a.k.a. Reverse Lookup
- If you want to use hostname rules
  - Your queue manager must be able to contact your DNS
  - Your DNS must be able to resolve the IP addresses
    - Sender/Client address
    - More than previously needed just to use CONNAME('hostname(port)')
- NO DNS – NO HOSTNAME
- NO HOSTNAME – NO MATCH



- IP address from TCP/IP
- Other attributes from internal channel flows
  - Channel Name
  - Certificate DN
  - Remote QMgr Name
  - Client User ID

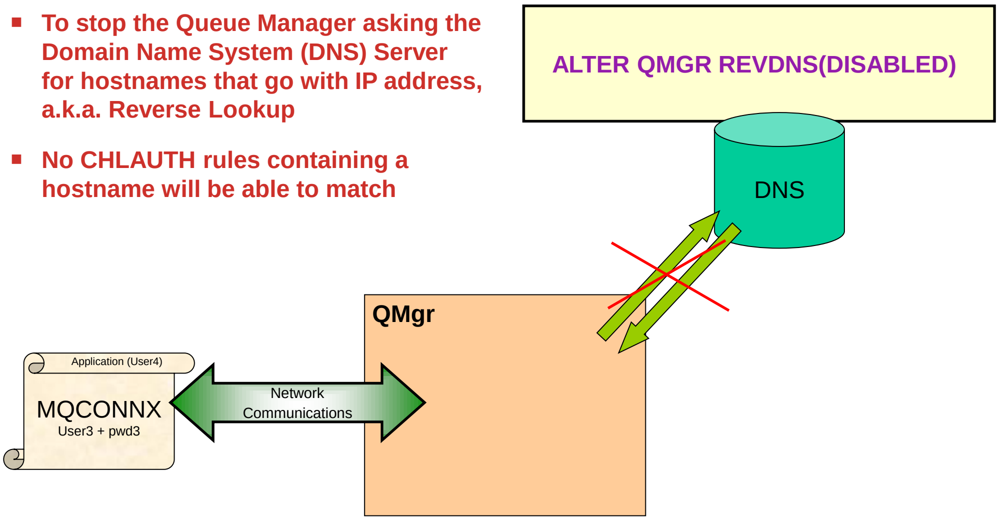
Obtaining a hostname – Notes

N O T E S

- In order to be able to process channel authentication records that contain rules using hostnames we need to be able to obtain the hostname that represents the IP address of the socket. The hostname is not 'sent' to us by the channel or by TCP/IP. We get the IP address from the socket. We get the other attributes that channel authentication records use from the various internal flows across the socket.
- To get the hostname we must ask the Domain Name System (DNS) Server what hostname goes with the IP address we are currently looking at. In order for this to be successful our queue manager must be able to use the DNS. This may already be true if you are using hostnames in CONNAME fields for example – which is certainly common-place. Also, the DNS must be able to reverse look-up the IP address and find a hostname for us. This may not be true in your current set up. Are all the sender channel or client application IP addresses currently available in your DNS? In order for hostname rules to be used, this must be the case.
- If you cannot reverse look up the hostname then CHLAUTH hostname rules will not be able to be matched.

Avoiding obtaining a hostname

- To stop the Queue Manager asking the Domain Name System (DNS) Server for hostnames that go with IP address, a.k.a. Reverse Lookup
- No CHLAUTH rules containing a hostname will be able to match



```
ALTER QMGR REVDNS(DISABLED)
```

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

Avoiding obtaining a hostname – Notes

NOTES

- It is possible that you wish this to always be the case. Some people are more nervous about the potential security hazards of using hostnames than others. When CHLAUTH only used IP addresses to match on, this was not something you had to worry about. Now someone might start to get lazy and use hostname rules.
- We have added a control to turn off the reverse look up of hostnames. There were previously undocumented parameters on both z/OS® and distributed to allow this, but as part of this feature we have made an official version of these.
- When REVDNS is ENABLED, the reverse look-up of the IP Address to retrieve the hostname will still only be done when it is required. If you do not use hostnames in CHLAUTH rules, then the only time a reverse look-up will be done is when writing an error message which contains that information. This is the same as the product behaviour pre-V8.

Diagnosing hostname look-up failures

- **MQ V7.1**

```
AMQ9777: Channel was blocked
EXPLANATION:
The inbound channel 'SYSTEM.DEF.SVRCONN' was blocked from address '9.180.165.163'
because the active values of the channel matched a record configured with
USERSRC(NOACCESS). The active values of the channel were 'CLNTUSER(metaylor)'.
```

- **MQ V8**

```
AMQ9777: Channel was blocked
EXPLANATION:
The inbound channel 'SYSTEM.DEF.SVRCONN' was blocked from address
'metaylor.ibm.com(9.180.165.163)' because the active values of the channel matched a
record configured with USERSRC(NOACCESS). The active values of the channel were
'CLNTUSER(metaylor) ADDRESS(metaylor.ibm.com,metaylor.hursley.ibm.com)'.
```

Diagnosing hostname look-up failures – Notes

NOTES

- In MQ V7.1, this was the message you saw when a channel was blocked. It gave you all the pieces of information you needed to work out why the channel was blocked. You can use the information in this error message to create a DISPLAY CHLAUTH MATCH(RUNCHECK) command.
- In MQ V8, this message will also now contain the hostname (possibly several) that go with the IP address, assuming that we have been able to find one. The description of the message will indicate that if a hostname is not shown this implies that either REVDNS is DISABLED or that reverse DNS lookup was unable to obtain a hostname for this IP address.

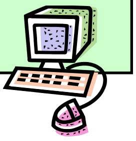
MESSAGE:
Channel was blocked
EXPLANATION:
The inbound channel '<insert one>' was blocked from address '<insert two>' because the active values of the channel matched a record configured with USERSRC(NOACCESS). The active values of the channel were '<insert three>'.
ACTION:
Contact the systems administrator, who should examine the channel authentication records to ensure that the correct settings have been configured. If no hostnames are shown this means that either the queue manager is configured with REVDNS(DISABLED) or the queue manager was unable to find a hostname for this IP address when making a reverse look up call to the Domain Name Server. The ALTER QMGR CHLAUTH switch is used to control whether channel authentication records are used. The command DISPLAY CHLAUTH can be used to query the channel authentication records.

Using MATCH(RUNCHECK) with hostnames

```
DISPLAY   CHLAUTH(SYSTEM.ADMIN.SVRCONN) MATCH(RUNCHECK)
          SSLPEER(‘CN=“Mark Taylor”, O=“IBM UK”’)
          CLNTUSER(‘metaylor’) ADDRESS(‘9.180.165.163’)
returns ===>
               CHLAUTH(SYSTEM.ADMIN.SVRCONN)
               TYPE(ADDRESSMAP)
               ADDRESS(‘*.ibm.com’) MCAUSER(METAYLOR)
```

- Just as before, MATCH(RUNCHECK) mandates an IP address is provided
- Then the queue manager will employ DNS to find the hostname
- MATCH(RUNCHECK) thus also tests whether your DNS is correctly set up.

Chl: SYSTEM.ADMIN.SVRCONN
DN: CN=Mark Taylor, O=IBM UK
UID: metaylor
IP: 9.180.165.163



Using MATCH(RUNCHECK) with hostnames – Notes

NOTES

- The DISPLAY CHLAUTH variant invoked using MATCH(RUNCHECK) allows you to provide all the same pieces of information that an inbound client presents to the queue manager. As we noted earlier, the hostname is not one of those pieces of information, the queue manager has to go and find that information out from the Domain Name Server (DNS).
- So when providing information into the MATCH(RUNCHECK) command, you do the same as before, you provide the IP address. The queue manager will then make the call to DNS as it would if the real inbound connection appeared and find out what the hostname is, then run the matching against the rules. If it was able to find out a hostname then it will match against a hostname rules, but if it was not, then it won’t.
- If you have your queue manager configured to use REVDNS(DISABLED) and you also have some CHLAUTH rules that use hostnames, then a message will appear along with the output of the MATCH(RUNCHECK) display in rather the same way that it warns you that CHLAUTH is DISABLED.
- Thus DISPLAY CHLAUTH MATCH(RUNCHECK) can help you to determine whether your reverse look-up for particular IP addresses is likely to work.

Channel Authentication Records – Summary

- **Set rules to control how inbound connections are treated**
  - Inbound Clients
  - Inbound QMgr to QMgr channels
  - Other rogue connections causing FDCs
- **Rules can be set to**
  - Allow a connection
  - Allow a connection and assign an MCAUSER
  - Block a connection
  - Ban privileged access
  - Provide multiple positive or negative SSL Peer Name matching
  - Mandate user ID & password checking
- **Rules can use any of the following identifying characteristics of the inbound connection**
  - IP Address
  - Hostname
  - SSL/TLS Subject’s Distinguished Name
  - SSL/TLS Issuer’s Distinguished Name
  - Client asserted user ID
  - Remote queue manager name

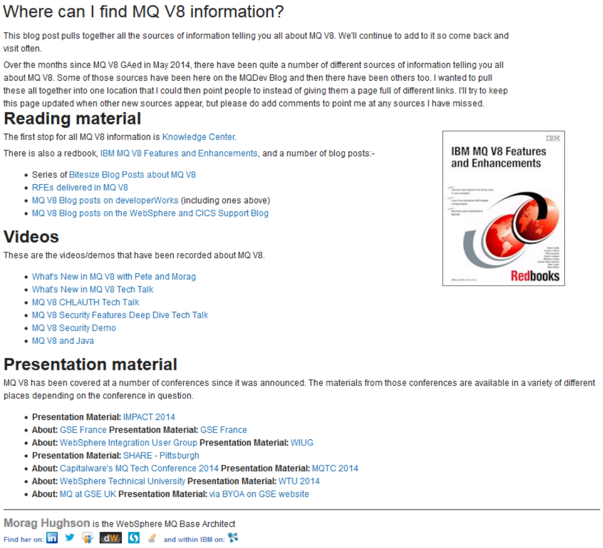
Channel Authentication Records – Summary – Notes

N O T E S

- Here is a repeat of our first slide with some small updates.
- We saw earlier in the presentation that CHLAUTH links into the Connection Authentication feature, and we saw that we can now fully qualify SSL/TLS DN matching in our CHLAUTH rules with Issuer’s DN as well as the Subject’s DN, and now in this last section we’ve seen that we have Hostnames as well.

For Additional Information

https://ibm.biz/MQV8Info



Where can I find MQ V8 information?

This blog post pulls together all the sources of information telling you all about MQ V8. We'll continue to add to it so come back and visit often.

Over the months since MQ V8 GAed in May 2014, there have been quite a number of different sources of information telling you all about MQ V8. Some of those sources have been here on the MQDev Blog and then there have been others too. I wanted to pull these all together into one location that I could then point people to instead of giving them a page full of different links. I'll try to keep this page updated when other new sources appear, but please do add comments to point me at any sources I have missed.

Reading material

The first stop for all MQ V8 information is Knowledge Center.

There is also a redbook, IBM MQ V8 Features and Enhancements, and a number of blog posts:-

- Series of Bitesize Blog Posts about MQ V8
- RFEs delivered in MQ V8
- MQ V8 Blog posts on developerWorks (including ones above)
- MQ V8 Blog posts on the WebSphere and CICS Support Blog

Videos

These are the videos/demos that have been recorded about MQ V8.

- What's New in MQ V8 with Pete and Morag
- What's New in MQ V8 Tech Talk
- MQ V8 CHLAUTH Tech Talk
- MQ V8 Security Features Deep Dive Tech Talk
- MQ V8 Security Demo
- MQ V8 and Java

Presentation material

MQ V8 has been covered at a number of conferences since it was announced. The materials from those conferences are available in a variety of different places depending on the conference in question.

- **Presentation Material:** IMPACT 2014
- **About:** GSE France **Presentation Material:** GSE France
- **About:** WebSphere Integration User Group **Presentation Material:** WIUG
- **Presentation Material:** SHARE - Pittsburgh
- **About:** Capitalware's MQ Tech Conference 2014 **Presentation Material:** MQTC 2014
- **About:** WebSphere Technical University **Presentation Material:** WTU 2014
- **About:** MQ at GSE UK **Presentation Material:** via BYOA on GSE website

Morag Hughson is the WebSphere MQ Base Architect

Find her on: and within IBM on:

This was session 17894 - The rest of the week ……

	Monday	Tuesday	Wednesday	Thursday	Friday
08:30			MQ for z/OS, Using and Abusing New Hardware and the New v8 Features	Nobody Uses Files Any More Do They? New Technologies for Old Technology, File Processing in MQ MFT and IIB	Monitoring and Auditing MQ Securing MQ Initiated CICS Workload
10:00	Introduction to MQ - Can MQ Really Make My Life Easier?	MQ for z/OS: The Insider Story	IBM Integration Bus MQ Flexibility	Common Problems and Problem Determination for MQ z/OS	IBM MQ and IBM Integration Bus - from Migration and Maintenance to Continuous Enhancements, How and Why to Stay Current
11:15	Introduction to IBM Integration Bus on z/OS	Introduction to the New MQ Appliance	MQ V8 Hands-on Labs! MQ V8 with CICS and COBOL! MQ SMF Labs!		
12:15					
1:45	What's New in the Messaging Family - MQ v8 and More		Getting Started with Performance of MQ on z/OS	IBM MQ: Are z/OS & Distributed Platforms Like Oil & Water?	
3:15	What's New in IBM Integration Bus	Live!: End to End Security of My Queue Manager on z/OS Application Programming with MQ Verbs	Digging into the MQ SMF Data	MQ Parallel Sysplex Exploitation, Getting the Best Availability from MQ on z/OS by Using Shared Queues	
4:30	MQ Security: New v8 Features Deep Dive	Live!: What's the Cloud Going to Do to My MQ Network? The Do's and Don'ts of IBM Integration Bus Performance	Giving It the Beans: Using IBM MQ as the Messaging Provider for JEE Applications in IBM Application Server	Challenge the MQ & IIB Experts?	

Any questions?

Please fill in evaluations
(Session # 17894)