Morag Hughson - hughson@uk.ibm.com

IBM UK

THINK GLOBAL – ACT LOCAL

What's new in IBM MQ V8

What's new in IBM MQ V8

N O T E S

- IBM announced MQ V8 on 22 April 2014 and it was generally available in the months that followed.
- This presentation is Part 1 of a two-part presentation all about the new features in IBM MQ V8.
- Part 1 (This part) covers the “Platforms & Standards”, “Security” and “Scalability” columns from the table on the next page.
- Part 2 (afternoon session) covers the “System z exploitation” columns from the table on the next page.

MQ V8 Dates / End of Service

- **Announce: 22 April 2014**
- **Availability:**
  - 23 May 2014 (eGA Distributed)
  - 13 June 2014 (z/OS and pGA Distributed)

- **End of Service for old platforms and versions**
  - MQ V7.0.0 and V7.0.1 for multiplatforms – EOM, EOS effective **September 2015**
    - V7.0 will have had more than 7 years of support
  - MQ V7.0.1 for z/OS – EOM, EOS effective **September 2015**
    - V7.0 .0 already out of service

IBM MQ V8 delivering best in class enterprise messaging

Platforms & Standards	*Security*	*Scalability*	*System z exploitation*
64-bit for all platforms	Userid authentication via OS & LDAP	Multiplexed client performance	64-bit buffer pools in MQ for z/OS means less paging, more performance
Support for JMS 2.0	User-based authorisation for Unix	Queue manager vertical scaling	Performance and capacity
Improved support for .Net and WCF	AMS for IBM i & z/OS	Publish/Subscribe improvements	Performance enhancements for IBM Information Replicator (QRep)
Changes to runmqsc	DNS Hostnames in CHLAUTH records	Routed publish/subscribe	Exploit zEDC compression accelerator
SHA-2 for z, i & NSS	Multiple certificates per queue manager	Multiple Cluster Transmit Queue on all platforms	SMF and shared queue enhancements

IBM MQ V8 delivering best in class enterprise messaging

Platforms & Standards	*Security*	*Scalability*	*System z exploitation*
64-bit for all platforms	Userid authentication via OS & LDAP	Multiplexed client performance	64-bit buffer pools in MQ for z/OS means less paging, more performance
Support for JMS 2.0	User-based authorisation for Unix	Queue manager vertical scaling	Performance and capacity
Improved support for .Net and WCF	AMS for IBM i & z/OS	Publish/Subscribe improvements	Performance enhancements for IBM Information Replicator (QRep)
Changes to runmqsc	DNS Hostnames in CHLAUTH records	Routed publish/subscribe	Exploit zEDC compression accelerator
SHA-2 for z, i & NSS	Multiple certificates per queue manager	Multiple Cluster Transmit Queue on all platforms	SMF and shared queue enhancements

MQ platform consistency

- **64-bit server support for all queue manager platforms**
  - Completion of platform coverage by adding Windows 64-bit engine
  - Applications can still be 32-bit
  - Requires Windows 7 or later
  - Client only package for 32-bit platforms
  - Queue Manager requires 64-bit

MQ platform consistency

N

O

T

E

S

- This release has all the Distributed queue managers fully supporting 64-bit operations. The final remaining platform had been Windows, but now the queue manager runs as 64-bit processes. Existing 32-bit applications continue to work of course, but this should bring additional capacity and scalability to the queue managers on that OS. The lowest level of Windows now supported is Windows 7; older versions are not supported. A client-only package for 32-bit versions of Windows is provided, but the qmgr requires the 64-bit OS.

JMS 2.0

- **Long-awaited update from JMS 1.1 standard**
- **JMS 2.0 – JSR 343 Java Message Service (JMS 2.0)**
  - Final release on 21 May 2013.
  - https://java.net/projects/jms-spec/pages/JMS20FinalRelease
- **New Messaging Features**
  - Delivery Delay
  - Asynchronous Send
  - Subscriptions can be shared across a messaging provider
- **API Changes**
  - Use of java.lang.AutoCloseable
  - Simplified API [combined connection/session]
  - Session doesn't need parameters (for Java EE)
- **Java 7 prereq**
- **Java EE 7 prereq for use of the Resource Adapter in Application Servers**
  - See statement of support here: http://www.ibm.com/support/docview.wss?uid=swg27041968
- **Full presentation can be seen here:**
  - http://www.slideshare.net/calanais/ibm-mq-v8-and-jms-20

JMS 2.0

NOTES

- The JMS 2.0 specification now requires JMS providers to implement both P2P and Pub-Sub.
- The following new messaging features have been added in JMS 2.0:
  - Delivery delay: a message producer can now specify that a message must not be delivered until after a specified time interval.
  - New send methods have been added to allow an application to send messages asynchronously.
  - JMS providers must now set the JMSXDeliveryCount message property.
- The following change has been made to aid scalability:
  - Applications are now permitted to create multiple consumers on the same durable or non-durable topic subscription. In previous versions of JMS only a single consumer was permitted.
- Several changes have been made to the JMS API to make it simpler and easier to use:
  - Connection, Session and other objects with a close method now implement the java.jang.AutoCloseable interface to allow them to be used in a Java SE 7 try-with-resources statement.
  - A new "simplified API" has been added which offers a simpler alternative to the standard API, especially in Java EE applications.
  - New methods have been added to create a session without the need to supply redundant arguments.
  - Although setting client ID remains mandatory when creating an unshared durable subscription, it is optional when creating a shared durable subscription.
  - A new method getBody has been added to allow an application to extract the body directly from a Message without the need to cast it first to an appropriate subtype.
- New methods have been added to Session which return a MessageConsumer on a durable topic subscription. Applications could previously only obtain a domain-specific TopicSubscriber, even though its use was discouraged.
- The specification has been clarified in various places.
- JMS 2.0 implementations require Java 7 for the runtime. They also require Java EE 7 for use of the Resource Adapter in Application Servers. Not all App Servers currently support Java EE 7. However, as with all client implementations, older versions of the RA still work when communicating to an MQ V8 queue manager.

.Net enhancements

- **MQ .Net classes can now use SSL without needing the C client installed**
  - A secure fully-managed .Net implementation
  - Uses Windows native certificate stores
- **For MQ .NET classes (aka Base .NET Classes) SSL properties can be set at**
  - MQEnvironment.cs
  - Hashtable properties (input parameter to MQQueueManager constructor)
- **For XMS .NET, SSL properties can be set as ConnectionFactory properties**

- **WCF interface extended to non-SOAP, non-JMS messages**
  - Making it easier for apps using WCF to communicate with any other MQ application

.Net enhancements: SSL + WCF

NOTES

- MQ .NET in unmanaged mode has supported the use of SSL since MQ v6.0.1. It was based on the C Client (and GSKit).
- In V8 MQ .NET Managed mode now supports SSL based on Microsoft SSLStreams/Security kit.
- There are some limitations when using SSL with Managed .NET.
- CipherSpec setting can be made only at Windows Group policy (gpedit.msc).
  - CipherSpec set on client channel is used only to determine SSLProtocol.
  - Make sure you set same cipherspec on Windows group policy and client channel to make sure correct protocol version is flowed.
- KeyRepository uses Windows Key Store only.
  - Set value as *SYSTEM for accessing certificates under Computer Account
  - Set value as *USER for accessing certificates under User account
- FIPS can be enabled only from Windows group policy.
- KEYRESET is not supported by Microsoft SSLStreams
  - This limitation is overridden by using KEYRESETCOUNT and Client Auto Reconnect feature of MQ.
  - Application can set KEYRESETCOUNT during connection, once the number of bytes sent/received reaches the count, connection will be forcibly broken. If Client Auto Reconnect facility is enabled, connection will be automatically reconnected.
- No managed way to support Cryptographic hardware
- Product samples have been updated to demonstrate SSL Connections. These can be found in WebSphere MQ\tools\dotnet\samples\cs\base\Simple

- Additional model for sending MQ messages from WCF applications
  - Both SOAP/JMS and MQMessaging can be used
- SOAP/JMS
  - Supported since MQ v7.0.1
  - Based on XMS .NET and makes JMS-like calls for MQI
  - Uses JMS nomenclature for URI(jms:\\) and Bindings
  - Now also supports “wmq:\\” style of URI
    - Uses MA93 supportpac specification for URI format/syntax
  - Messages can be delivered only to SOAP enabled Client/Service
- MQMessaging
  - New in MQ V8.0
  - Transmits MQ Messages over the WCF Channel without any SOAP headers
  - Use “wmq:\\” style of URI
    - Uses MA93 supportpac specification for URI format/syntax
  - Messages can be delivered to SOAP or NON-SOAP MQ applications

Changes to runmqsc

- **Can now be run by any user (not just mqm group)**
  - Can take a userid/password for authentication: new "-u" flag
- **Can now connect as a client to remote systems: new "-c" flag**
  - Client channel definitions located by MQSERVER -> MQCHLLIB -> MQCHLTAB
- **Can act as standalone program to create local CCDT: new "-n" flag**
  - Does not connect to queue manager; commands subset to update local channel definition file
- **Ease of use**
  - Customisable prompt using environment variable
  - New "exit" and "quit“ synonyms for "end"

```
$ ls –l runmqsc
-r-xr-xr-x    1 mqm    mqm   25930 06 Mar 04:46 runmqsc

$ export MQPROMPT="MQ +MQ_INSTALLATION_NAME+> "
$ runmqsc –u metaylor QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2014.
Enter password:
******
Starting MQSC for queue manager QM1.

MQ Installation5> DIS QMGR
…
```

Changes to runmqsc

N

O

T

E

S

- This release introduces a number of changes to the runmqsc program
- Firstly, it is now exectuable by any user, not just members of the mqm group. Security controls still apply of course, but the security is checked on each individual command. This makes it easier to have MQ administrators who have been granted full access to objects, but who are not in the mqm group.
- Another reason for making runmqsc world-executable was to make it usable on machines which do not have queue managers installed. It is now installed even on client-only systems, and it can be run either as a client program connecting directly to a remote queue manager, or as a completely standalone program to permit local creation of the Client Channel Definition Table. So you can create and modify a CCDT with no queue manager access at all.
- The program is one of several that have been updated to accept a userid for authentication. If the –u flag is given, then a password is requested. Note that the password is read from stdin so that it can be redirected from a file if necessary. If you also use scripts piped into runmqsc, then you can group commands to avoid having to put the password in the same script as the MQSC commands.
  - Unix: (cat password.stash ; cat script.mqsc) | runmqsc –u userid QM1
  - Windows: (type password.stash & type script.mqsc) | runmqsc –u userid QM1
- There are also a couple of usability enhancements. Firstly, there are some synomyms added to complete an MQSC session – END, QUIT and EXIT can all be used so you don't have to try them all. Different scripting environments for different products typically use one of these commands, and it's annoying to have to remember which goes with which.
- Secondly, you can now make it easier to see that you are in an MQSC command environment and some details of the current environment by setting the MQPROMPT environment variable. Replaceable inserts are recognised such as date and time, and installation-specific details. These are the same variable subsitutions as available for SERVICE objects.

SHA-2 Support

- **Stronger algorithms are now available and recommended**
  - In many cases available pre-V8
  - See technote http://www.ibm.com/support/docview.wss?uid=swg21639606
- **Changes also rolled into V8**
- **CipherSpecs include:-**
  - ECDHE_RSA_AES_128_CBC_SHA256
  - ECDHE_RSA_AES_256_CBC_SHA384
  - TLS_RSA_WITH_AES_128_CBC_SHA256
  - TLS_RSA_WITH_AES_256_CBC_SHA256
  - TLS_RSA_WITH_NULL_SHA256

SHA-2 Support

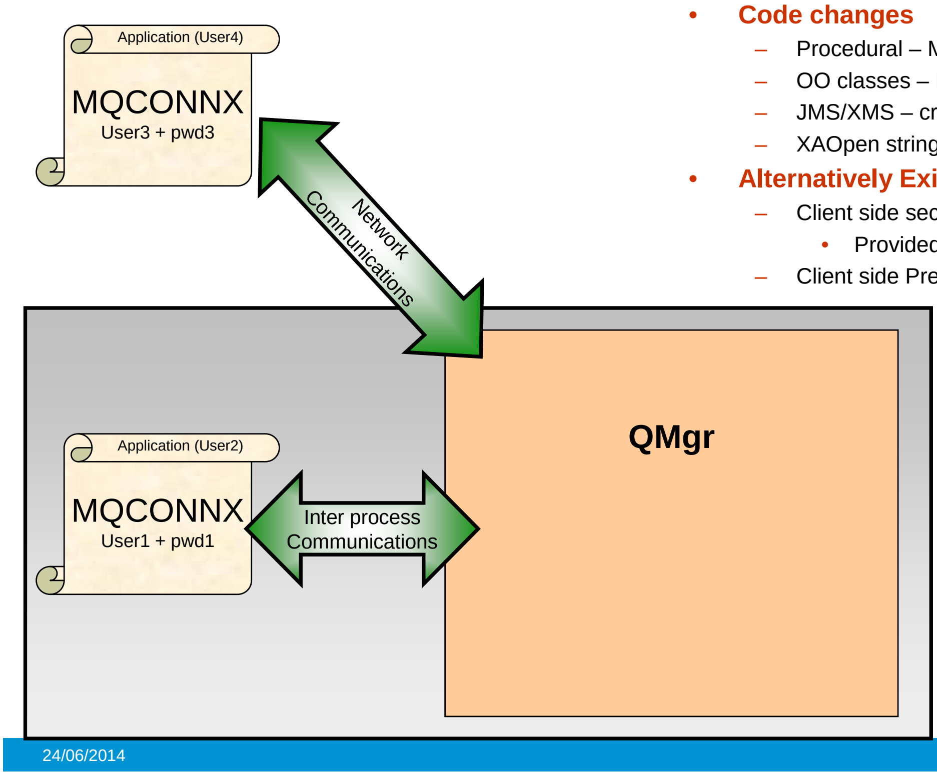
NOTES

- SHA-2 stands for Secure Hash Algorithm 2. This is the generic name for a family of hash functions SHA-224, SHA-256, SHA-384, and SHA-512; the numbers refer to the bit-size of the generated hash value.
- The hashing algorithms historically supported by WebSphere MQ channel cipherSpecs (SSLCIPH) were MD5 and SHA-1. These are no longer regarded as adequately secure due to successful attacks on them which have been widely publicized, and because NIST (the American National Institute of Standards and Technology) has mandated (http://csrc.nist.gov/groups/ST/hash/policy.html) that both MD5 and SHA-1 no longer be used. So we have had considerable customer pressure for provision of “SHA-2 support.”
- This support is in MQ V8, but also in earlier releases by applying appropriate fix pacs for the platform or component on MQ where you are interested in using it. A technote at http://www.ibm.com/support/docview.wss?uid=swg21639606 details exactly which fix pacs or APARs are required in each case.

IBM MQ V8 delivering best in class enterprise messaging

Platforms & Standards	*Security*	*Scalability*	*System z exploitation*
64-bit for all platforms	Userid authentication via OS & LDAP	Multiplexed client performance	64-bit buffer pools in MQ for z/OS means less paging, more performance
Support for JMS 2.0	User-based authorisation for Unix	Queue manager vertical scaling	Performance and capacity
Improved support for .Net and WCF	AMS for IBM i & z/OS	Publish/Subscribe improvements	Performance enhancements for IBM Information Replicator (QRep)
Changes to runmqsc	DNS Hostnames in CHLAUTH records	Routed publish/subscribe	Exploit zEDC compression accelerator
SHA-2 for z, i & NSS	Multiple certificates per queue manager	Multiple Cluster Transmit Queue on all platforms	SMF and shared queue enhancements

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

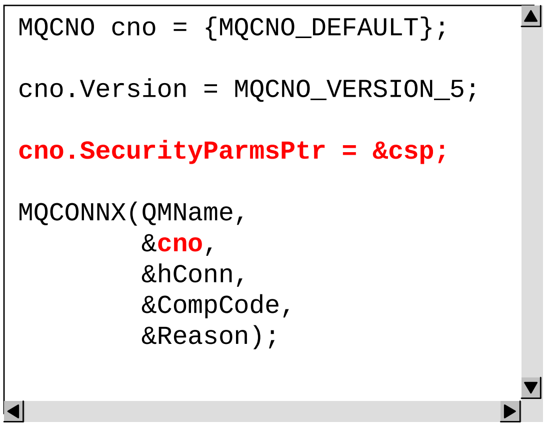
Connection Authentication – Application changes – Notes

N O T E S

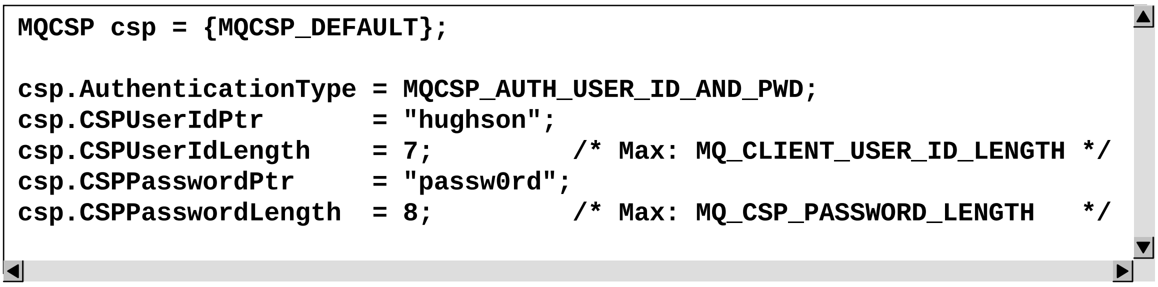
- Since WebSphere MQ V6.0, an application has been able to provide a user ID and password (in the Connection Security Parameters (MQCSP) structure in the MQCNO) at MQCONNX time. These were passed to a user written plug-point in the OAM on distributed to be checked. If the application was running client 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 WebSphere MQ V8 this will be available in all our interfaces listed, even where some of those were not made available in the WebSphere 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
- **WebSphere MQ V6**
  - Passed to OAM (Dist only)
  - Also passed to Security Exit
    - Both z/OS and Distributed
    - MQXR_SEC_PARMS
- **WebSphere MQ V8**
  - Acted upon by the queue manager (all platforms)

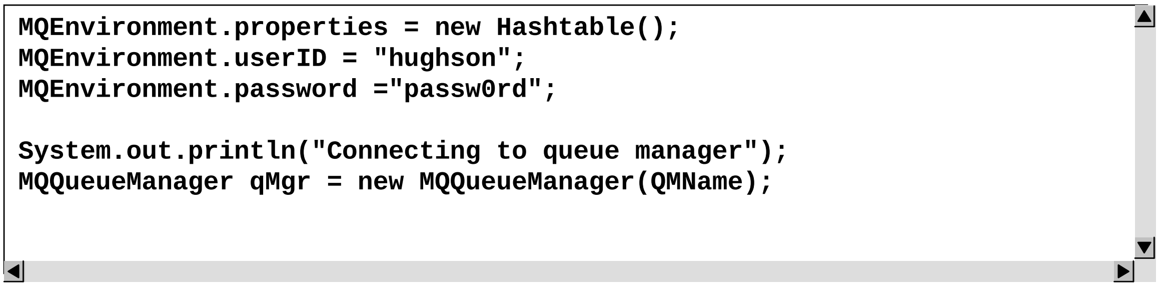


```
MQCNO cno = {MQCNO_DEFAULT};

cno.Version = MQCNO_VERSION_5;

cno.SecurityParmsPtr = &csp;

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



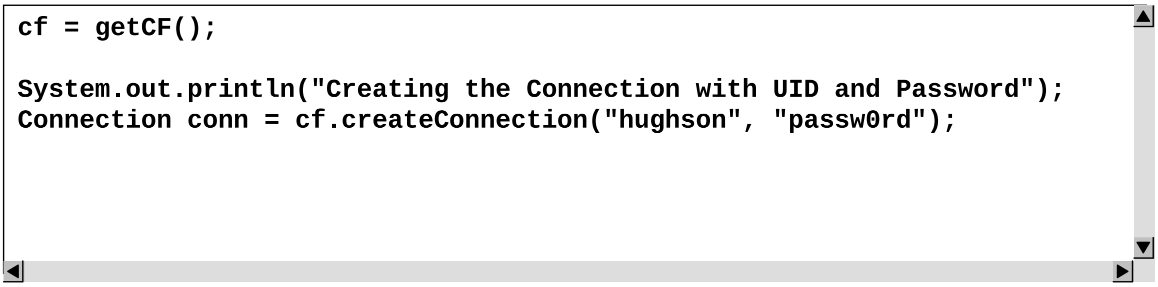
```
MQCSP csp = {MQCSP_DEFAULT};

csp.AuthenticationType = MQCSP_AUTH_USER_ID_AND_PWD;
csp.CSPUserIdPtr       = "hughson";
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 = "hughson";
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("hughson", "passw0rd");
```

Client side Security Exit

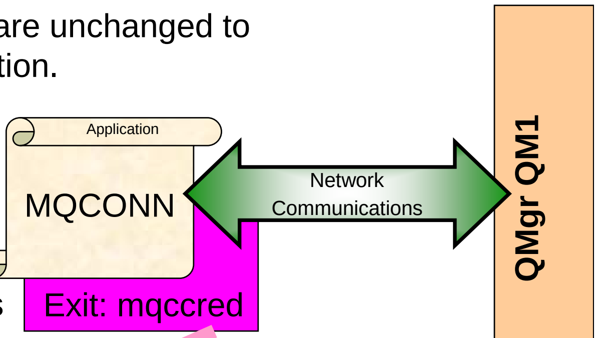
mqccred.ini

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

Assist applications that are unchanged to participate in authentication.

Consists of:

- Client channel security exit to insert uid/passwd
- Command line tool to protect passwords in a config file



Exit can be used by clients from V7.0.1 and later (by copying from a V8 installation)

Tool: runmqccred

mqccred.ini

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

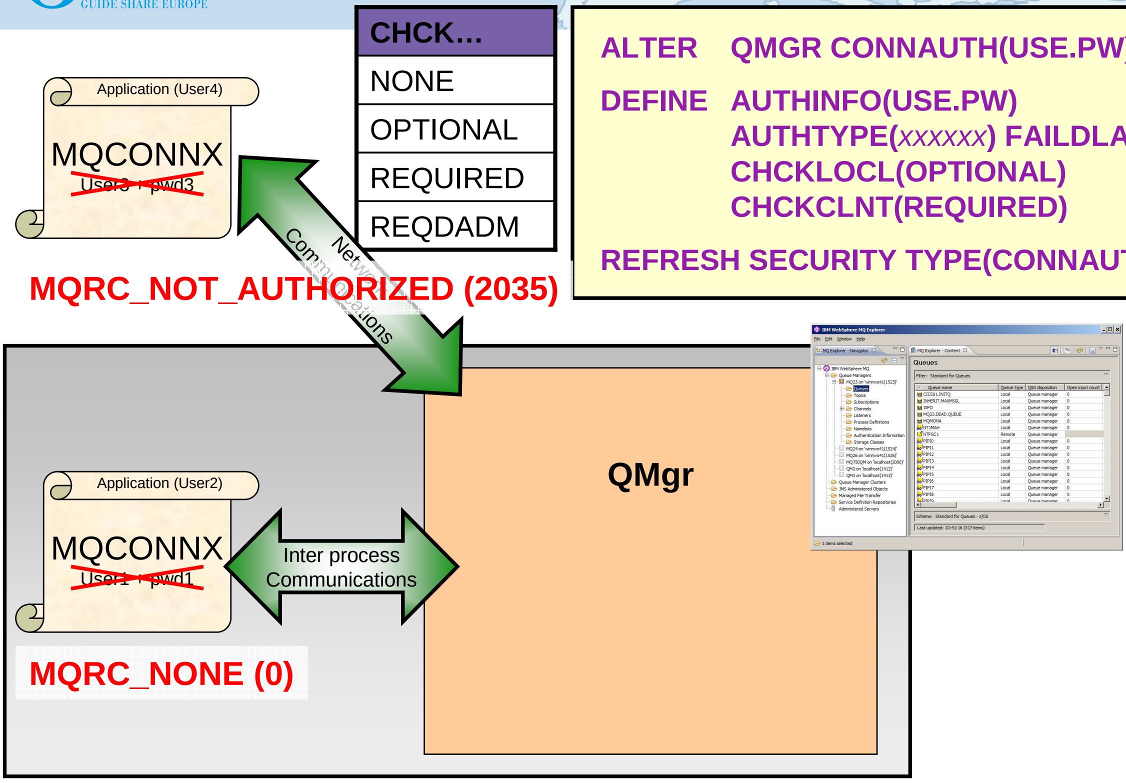
File permissions

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, WebSphere 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.

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

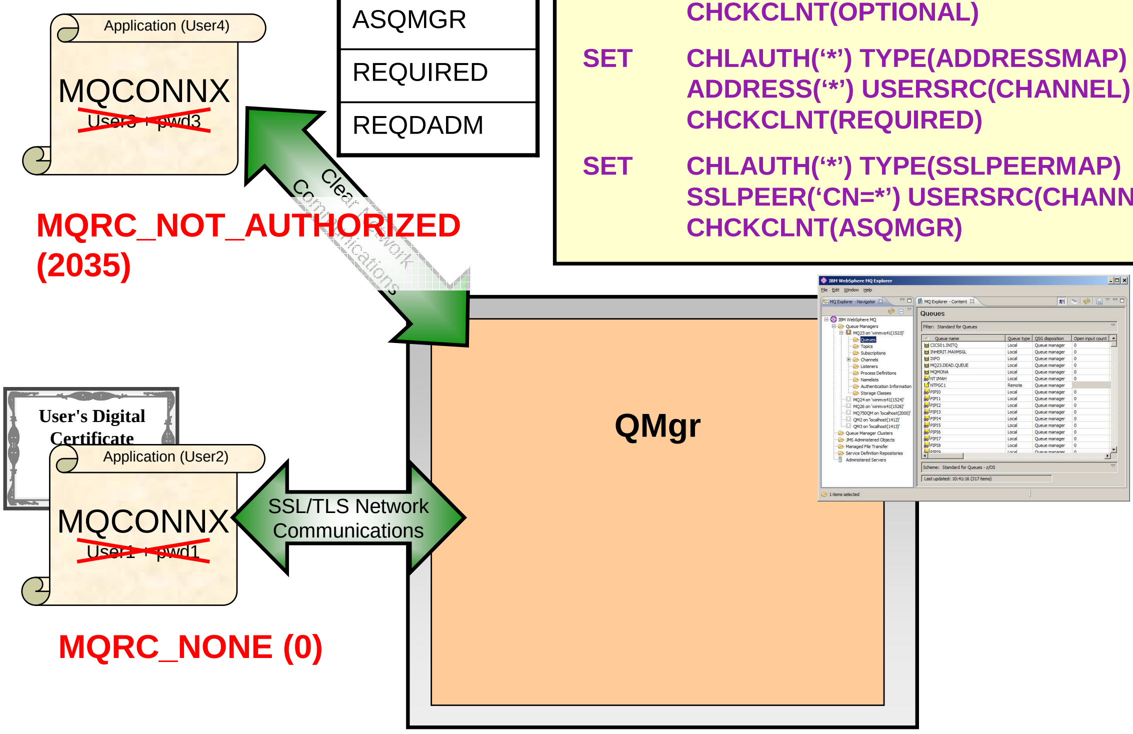
N O T E S

- 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 WebSphere 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.
- 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 – Configuration Granularity

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)
```



MQRC_NOT_AUTHORIZED (2035)

MQRC_NONE (0)

Connection Authentication – Configuration Granularity – Notes

N

O

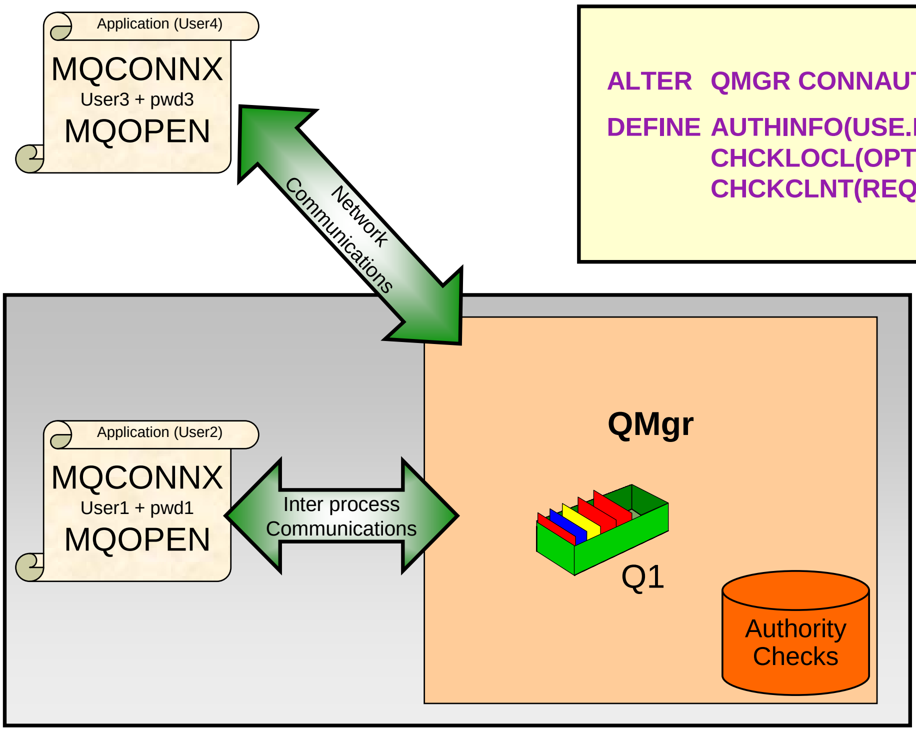
T

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

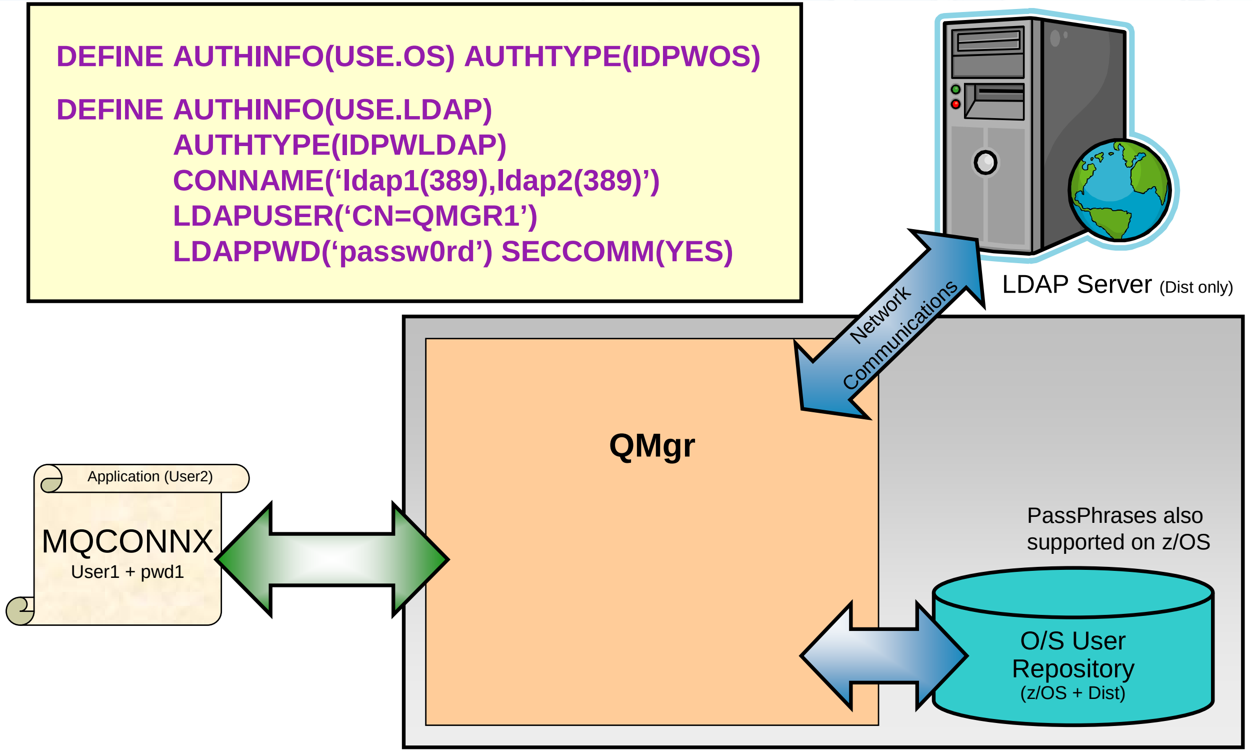
Connection Authentication – Relationship to Authorization – Notes

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 – 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)
```

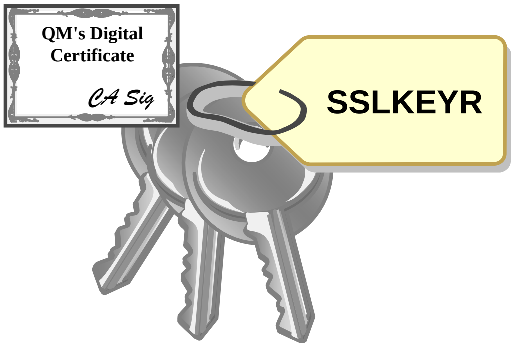


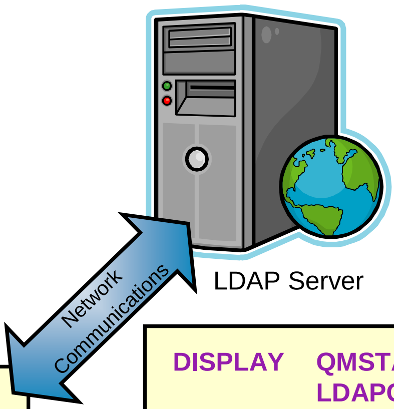
Connection Authentication – User Repositories – Notes

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.

Secure connection to an LDAP Server





```
ALTER  QMGR CONNAUTH(USE.LDAP)
       SSLFIPS(NO) SUITEB(NONE)
       CERTLABL(‘ibmwebspheremqqm1’)
       SSLKEYR('var/mqm/qmgrs/QM1/ssl/key')

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

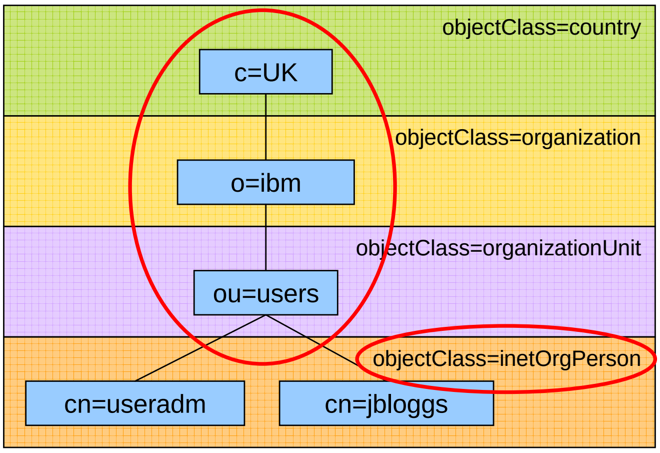
```
DISPLAY  QMSTATUS
         LDAPCONN
```

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 ‘ibmwebspheremq<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



objectClass=country

c=UK

objectClass=organization

o=ibm

objectClass=organizationUnit

ou=users

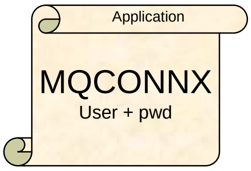
objectClass=inetOrgPerson

cn=useradm

cn=jbloggs

LDAP Server

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



Application

MQCONNX
User + pwd

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

NOTES

- 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.

Connection Authentication – Summary

- **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

MQ Security - Authorisation

- **Make Unix OAM userid-based**
  - Optional configuration
  - Consistent with other platforms
  - Will no longer add primary group to authorities during setmqaut
  - Chosen at queue manager creation or by editing qm.ini
- **Default is still group-based authorisations**

```
$ crtmqm –oa user QMU

----------------
Service:
          Name=AuthorizationService
          EntryPoints=14
          SecurityPolicy=User
```

- **Delete Authority record by SID**
  - Solve problem of orphaned authorities when Windows id is deleted

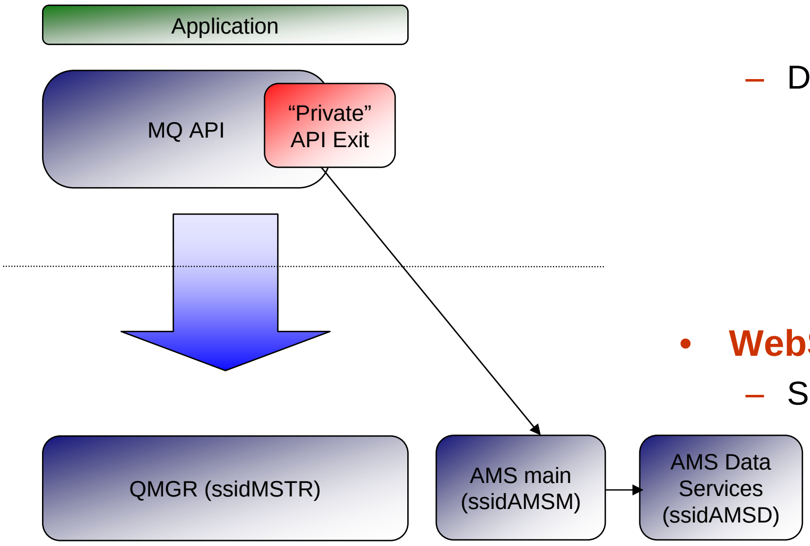
MQ Security - Authorisation

N O T E S

- One further point of consistency in this release is making the Unix authorisation model the same as for Windows – permissions can now be set for individual users, and not just groups. So running a command such as "setmqaut –t qmgr –p usr1 +connect" works for just that user, and not the primary group.
- This is configured by either creating the queue manager with the "-oa user" option or by editing the ini file as shown for existing queue managers – restarting the queue manager sets it to work in the new mode. The change is deliberately not dynamic, and requires the restart, because it is so fundamental to how the queue manager permissions operate, and is not something we would expect to be done frequently.
- All existing permissions are left unchanged when you switch to user-mode authorisations, so the group permissions that have been set remain, but new permissions can be set for the users.
- The user-based model is not the default for new queue managers; to keep compatibility with older systems, the default is still the group-based model.
- One final security feature is specifically for Windows, to deal with situations where users have been deleted from the system but their MQ permissions have not been deleted from the OAM. You can now use the setmqaut command to delete permissions based on SID instead of name, and this will remove the orphaned entries.

Advanced Message Security (AMS) on z/OS

8.0 Interception is built-in



- **Pre-V8.0 (two started tasks)**
  - Main Task: ssidAMSM
    - Runs API interceptor
    - Enforces policies
  - Data Services task: ssidAMSD
    - Performs signature and encryption
    - Calls System SSL PKCS#7 Services (uses SAF keyrings)
- **WebSphere MQ V8**
  - Single task: ssidAMSM
    - Started/stopped with QMgr
    - "Private" API Exit code is now embedded in the product

Advanced Message Security (AMS) on z/OS - Notes

NOTES

- On z/OS before MQ V8, the **MQ Server interceptor** for local (bindings mode) is implemented as a "private" API exit on z/OS.
- In V8, similar to the change made on Distributed in V7.5, AMS is pulled into the base WebSphere MQ product. It's documentation is also pulled into the WebSphere MQ Information Center.
- This provides a better integration with the queue manager including tie-in of the start/stop of the AMS address space with start-up and shut-down of the queue manager. Calling the AMS address space to do the encryption/decryption work is more efficient and due to no longer using the vendor API call intercept method (the "private" API exit), it is less likely to conflict with other OEM products.
- The previous two separate AMS address spaces, ssidAMSM (main) and ssidAMSD (data services) are now combined into a single address space, ssidAMSM. Any authorities that were previously required by ssidAMSD are now needed on ssidAMSM instead. ssidAMSM now consumes the encryption CPU. The utility that is used on z/OS to setup policies is renamed from DRQUTIL to CSQ0UTIL.
- There are no changes to the keyring names, and the hardened version of the policies which are stored as messages on the SYSTEM.PROTECTION.POLICY.QUEUE have the same shape, so existing policies just work.
- AMS is still priced separately as OTC and has a separately installed FMID which is an enablement module for AMS.

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 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="Morag Hughson"')
ADDRESS('9.20.*') MCAUSER(HUGHSON)
```

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="Morag Hughson"')
ADDRESS('s*.ibm.*') MCAUSER(HUGHSON)
```

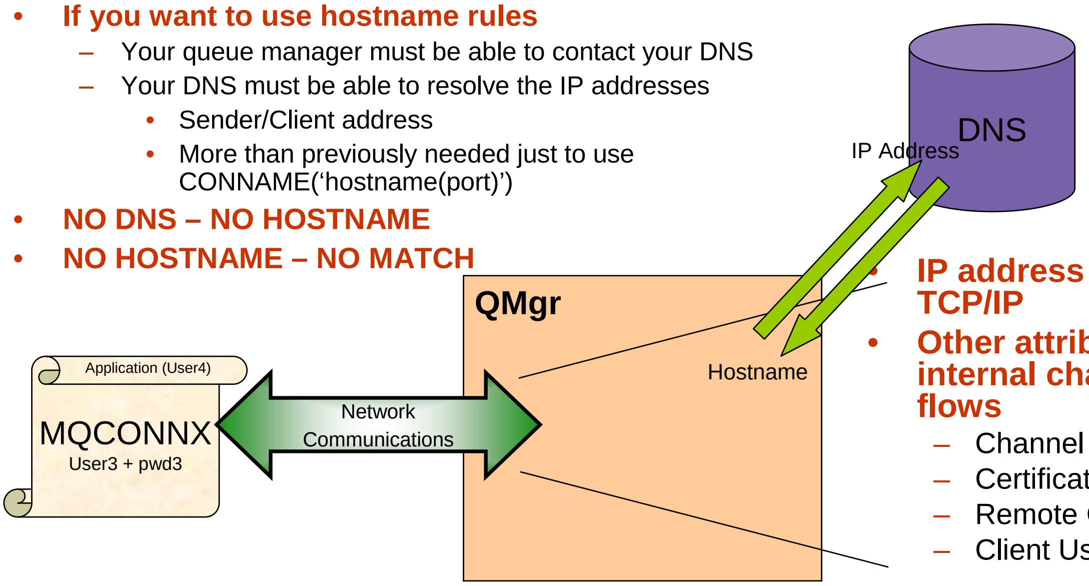
Channel Authentication Rules using Hostnames – Notes

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.

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 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

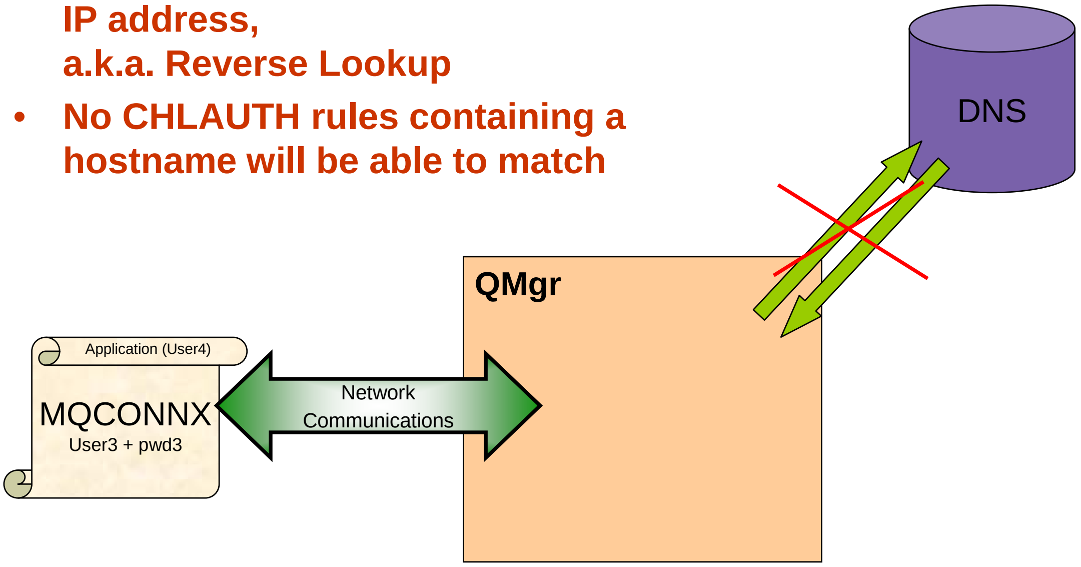
N O T E S

- 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 Server (DNS) 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 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)
```



Avoiding obtaining a hostname – Notes

N O T E S

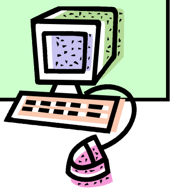
- It is possible that you wish this to always be the case. Some people are more nervous about the potential security hazards of using hostnames than others. When CHLAUTH only used IP addresses to match on, this was not something you had to worry about. Now someone might start to get lazy and use hostname 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.

Using MATCH(RUNCHECK) with hostnames

```
DISPLAY  CHLAUTH(SYSTEM.ADMIN.SVRCONN) MATCH(RUNCHECK)
         SSLPEER(‘CN=“Morag Hughson”, O=“IBM”’)
         CLNTUSER(‘mhughson’) ADDRESS(‘9.180.165.163’)
returns ===>
              CHLAUTH(SYSTEM.ADMIN.SVRCONN)
              TYPE(ADDRESSMAP)
              ADDRESS(‘*.ibm.com’) MCAUSER(HUGHSON)
```

- 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=Morag Hughson.O=IBM
UID: mhughson
IP: 9.180.165.163



Using MATCH(RUNCHECK) with hostnames – Notes

N

O

T

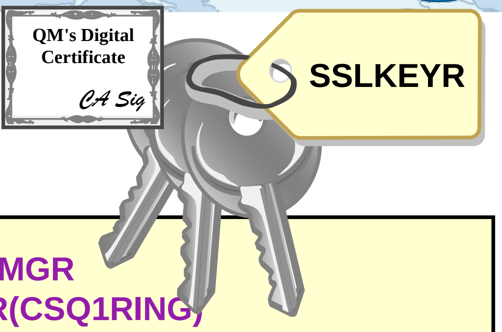
E

S

- 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.

Single Queue Manager Certificate

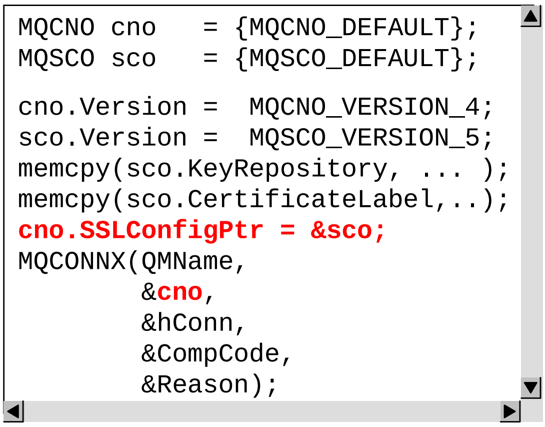
- **Name Queue Manager Certificate**
  - Using CERTLABL attribute
- **Name Client Certificate**
  - mqclient.ini file SSL Stanza
    - CertificateLabel
  - MQCONNX (MQSCO structure)
    - CertificateLabel
- **Environment variable**
  - export MQCERTLABL=MyCert



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



```
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);
```

Single Queue Manager Certificate

N O T E S

- Before WebSphere 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 WebSphere MQ required it, in order for the certificate to be found. This doesn't always meet customer standards of certificate labelling.
- In WebSphere 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 ibmWebSphereMQ<*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.

Helpful for migration of certificates

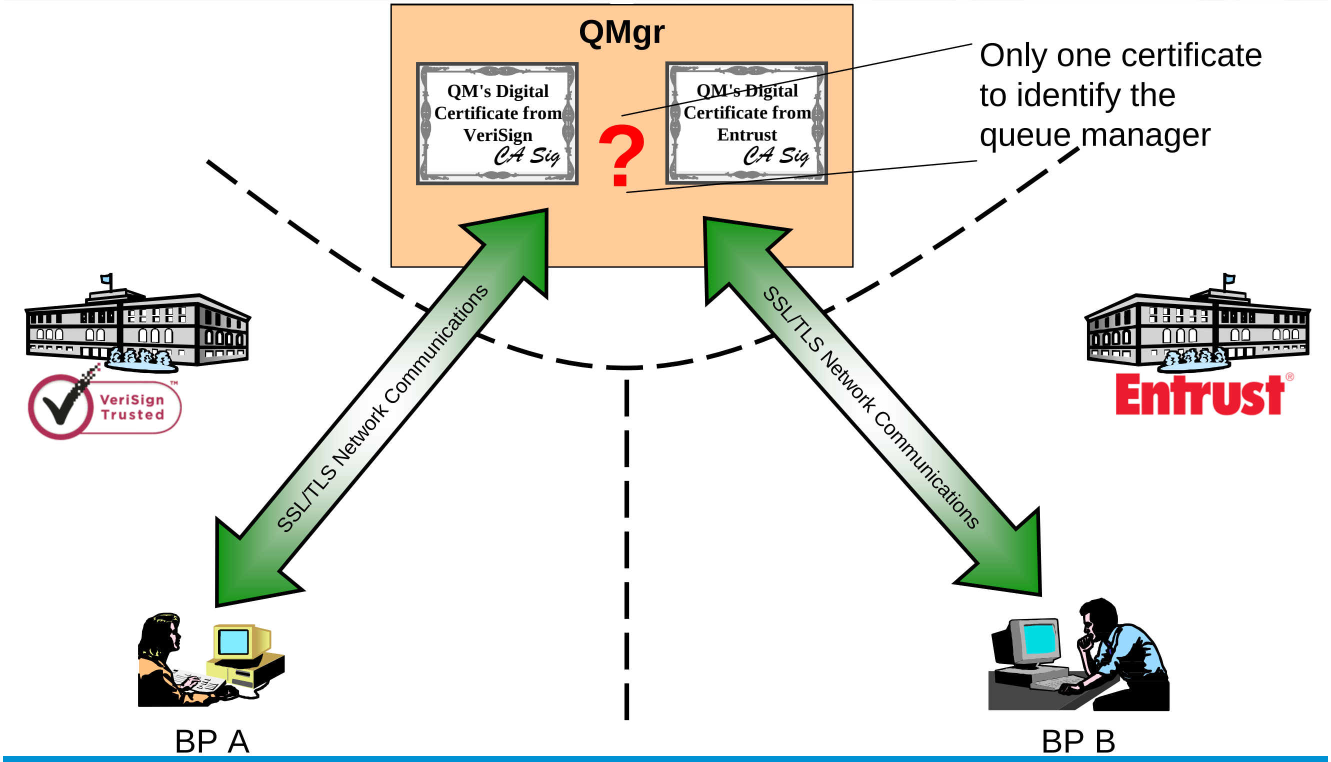
- **Migrating over to a new certificate when main certificate is ready to expire**
  - Used to have to issue GSKit/RACF commands to rename certificate
    - ibmwebspheremqqm1 -> ibmwebspheremqqm1old
    - ibmwebspheremqqm1new -> ibmwebspheremqqm1
    - REFRESH SECURITY TYPE(SSL)
  - Now just MQ commands when the time comes
    - Current label is ‘QM1 Cert 2013’
    - ALTER QMGR CERTLABL(‘QM1 Cert 2014’)
    - REFRESH SECURITY TYPE(SSL)

Helpful for migration of certificates

NOTES

- 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
VeriSign Trusted
Entrust
SSL/TLS Network Communications
SSL/TLS Network Communications
BP A
BP B

Only one certificate to identify the queue manager

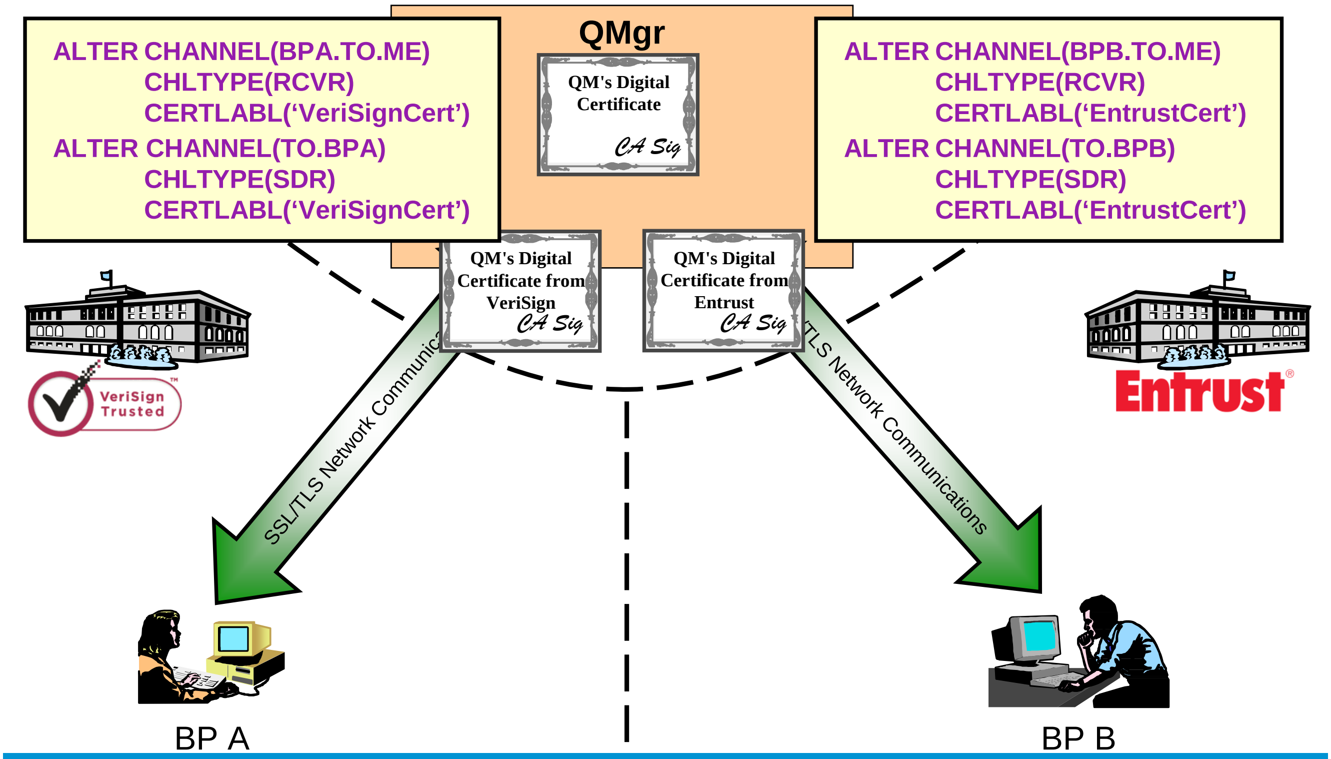
Business Partners with different CA requirements – Notes

NOTES

- 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 WebSphere 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.

GUIDE SHARE EUROPE

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')
```

QMgr
QM's Digital Certificate
CA Sig
QM's Digital Certificate from VeriSign
CA Sig
QM's Digital Certificate from Entrust
CA Sig
VeriSign Trusted
Entrust
SSL/TLS Network Communications
TLS Network Communications
BP A
BP B

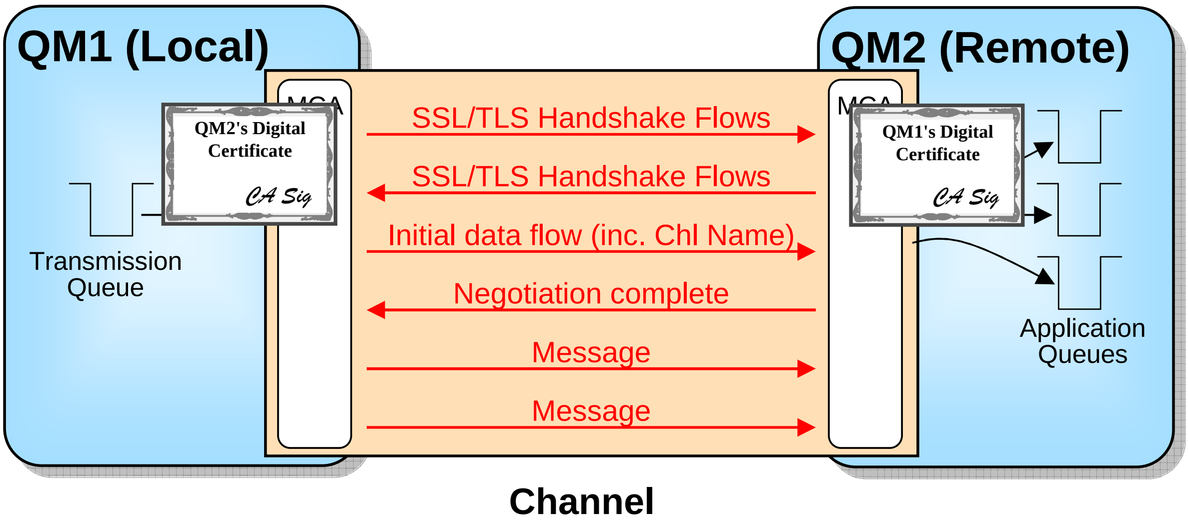
GUIDE SHARE EUROPE

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 WebSphere 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?

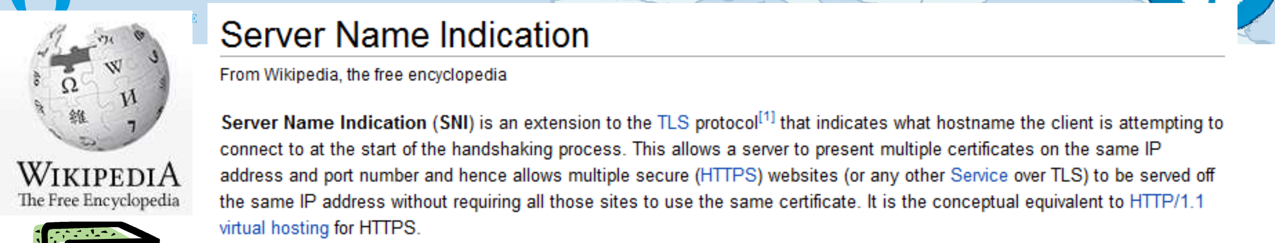


Why haven’t we always done this? – Notes

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).

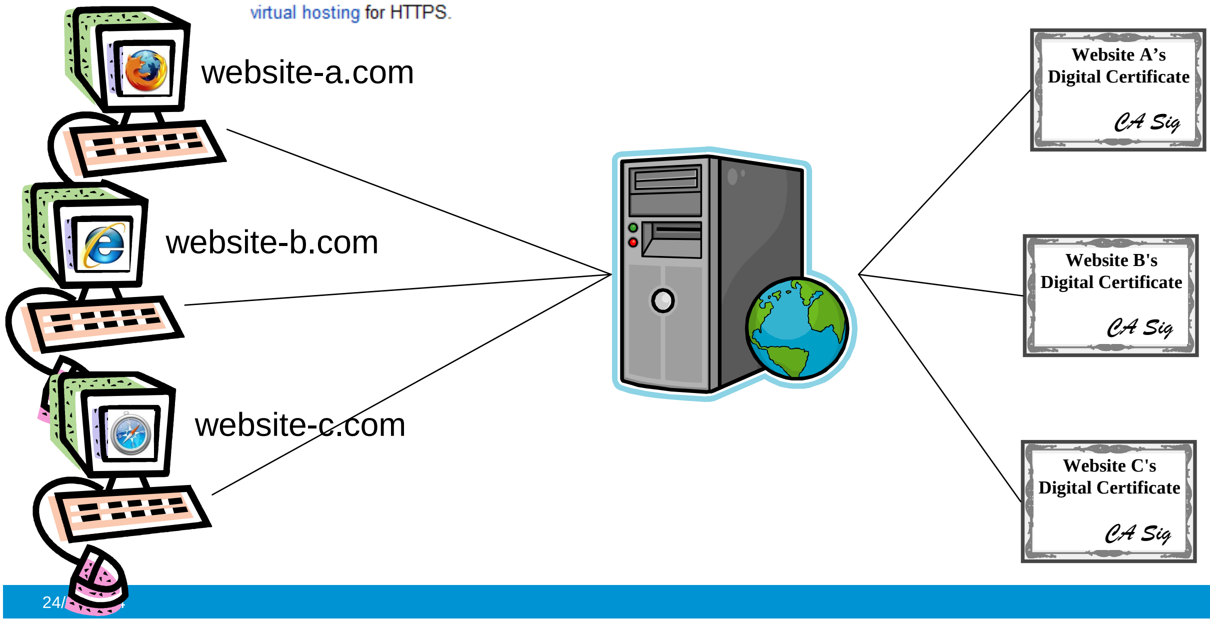
Server Name Indication



Server Name Indication

From Wikipedia, the free encyclopedia

Server Name Indication (**SNI**) is an extension to the TLS protocol[1] that indicates what hostname the client is attempting to connect to at the start of the handshaking process. This allows a server to present multiple certificates on the same IP address and port number and hence allows multiple secure (HTTPS) websites (or any other Service over TLS) to be served off the same IP address without requiring all those sites to use the same certificate. It is the conceptual equivalent to HTTP/1.1 virtual hosting for HTTPS.

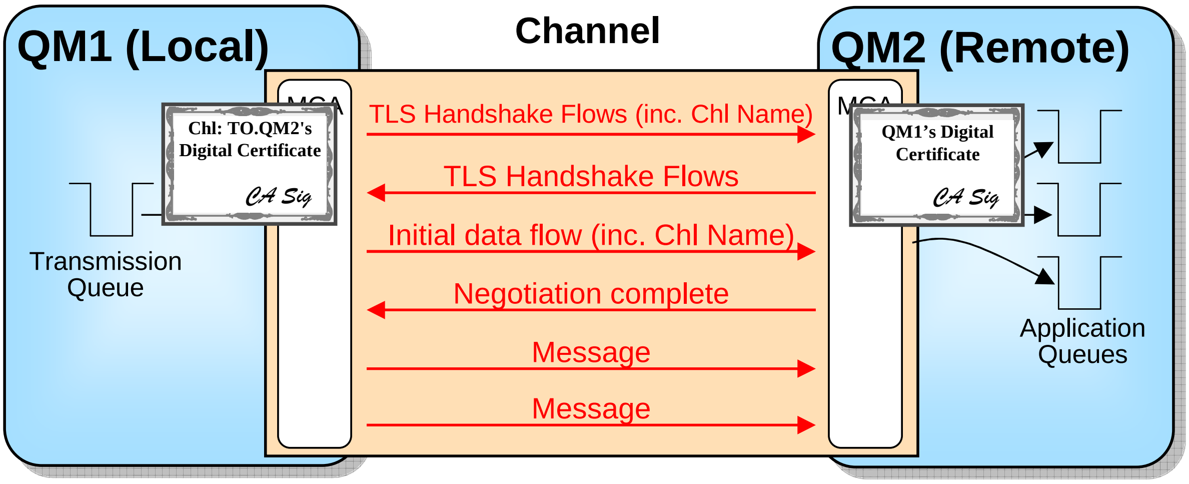

website-a.com
website-b.com
website-c.com
Website A's Digital Certificate
CA Sig
Website B's Digital Certificate
CA Sig
Website C's Digital Certificate
CA Sig

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



QM1 (Local)

Channel

QM2 (Remote)

MCA

Chl: TO.QM2's Digital Certificate

CA Sig

Transmission Queue

TLS Handshake Flows (inc. Chl Name)

TLS Handshake Flows

Initial data flow (inc. Chl Name)

Negotiation complete

Message

Message

MCA

QM1's Digital Certificate

CA Sig

Application Queues

- **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 used, we'd only detect that we needed to supply a different certificate after completion of the handshake and will fail the connection, if it hasn't already failed due to using the wrong certificate!**

Using Server Name Indication (SNI) with a channel name

N

O

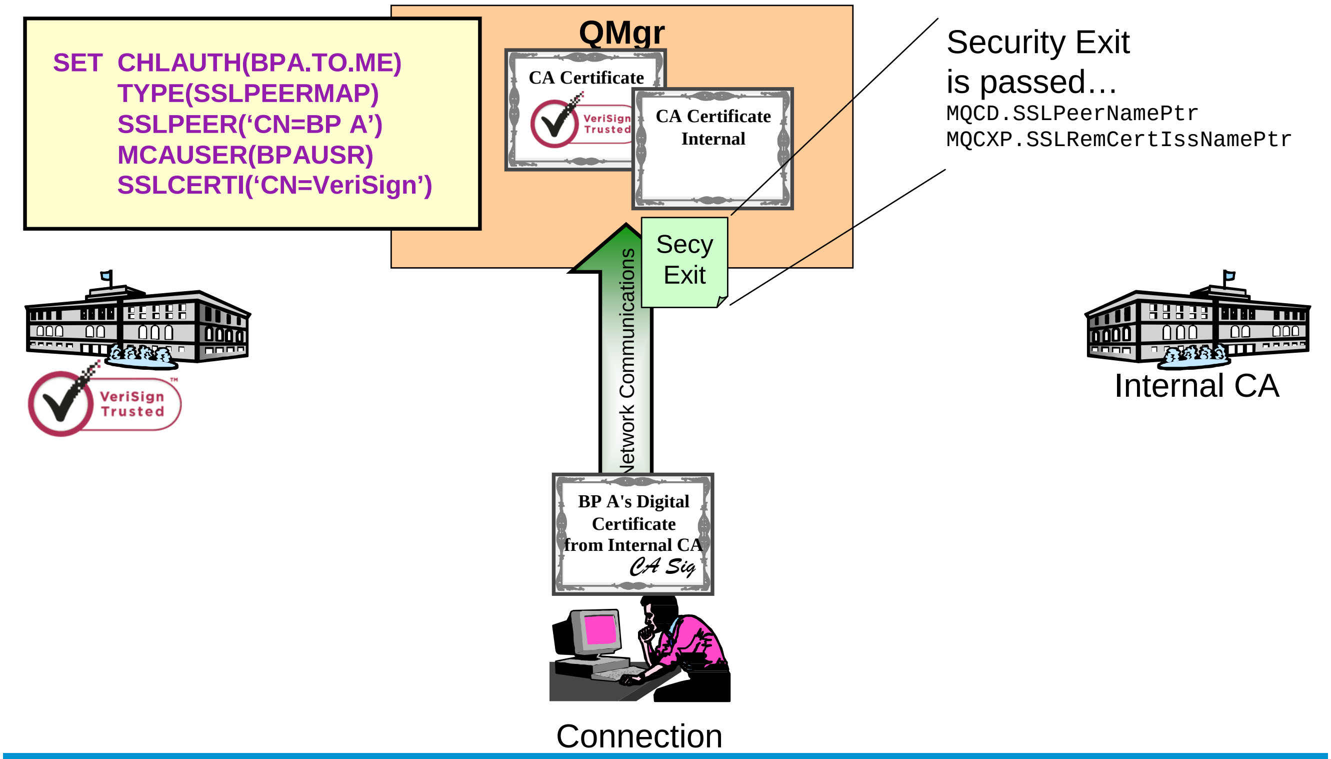
T

E

S

- WebSphere 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.

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

N O T E S

- 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 WebSphere 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).

Changes for Channels using SSL/TLS Certificates

- **Single Queue Manager Certificate**
  - `ALTER QMGR CERTLABL('My certificate name')`
- **Per Channel Certificate**
  - `ALTER CHANNEL … CERTLABL('This chl certificate')`
- **Certificate Matching**

```
– SET CHLAUTH('*')
      TYPE(SSLPEERMAP)
      SSLPEER('CN=Morag Hughson')
      SSLCERTI('CN=IBM CA')
      MCAUSER('hughson')
```

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 WebSphere 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).

IBM MQ V8 delivering best in class enterprise messaging

Platforms & Standards	*Security*	*Scalability*	*System z exploitation*
64-bit for all platforms	Userid authentication via OS & LDAP	Multiplexed client performance	64-bit buffer pools in MQ for z/OS means less paging, more performance
Support for JMS 2.0	User-based authorisation for Unix	Queue manager vertical scaling	Performance and capacity
Improved support for .NET and WCF	AMS for IBM i & z/OS	Publish/Subscribe improvements	Performance enhancements for IBM Information Replicator (QRep)
Changes to runmqsc	DNS Hostnames in CHLAUTH records	Routed publish/subscribe	Exploit zEDC compression accelerator
SHA-2 for z, i & NSS	Multiple certificates per queue manager	Multiple cluster transmission queues on all platforms	SMF and shared queue enhancements

Multiplexed client performance

- **Version 7 introduced support for `SHARECNV`**
  - Multiple client conversations (e.g. threads) can use the same TCP/IP socket (channel instance)
- **`SHARECNV(0)`**
  - No conversation sharing, behaviour as per version 6
- **`SHARECNV(1)`**
  - No conversation sharing
  - Heartbeats, asynchronous message consumption and read-ahead support
- **`SHARECNV(n>1)`**
  - Up to n conversations per channel instance - reduces number of sockets and channel instances
- **Performance improvements**
  - On distributed, `SHARECNV(n>1)` can impact performance if multiple conversations are busy due to contention for the socket
  - In version 8, `SHARECNV(1)` optimized for parity with `SHARECNV(0)`

Multiplexed client performance

NOTES

- Prior to version 7 each client conversation uses a separate socket and channel instance. If clients establish multiple conversations using different threads this can create a large demand for sockets on the server. Each channel instance also requires storage on the queue manager.
- In version 7 support was added to allow client conversations to share the same socket/channel instance. This can significantly reduce the resource overhead associated with these connections. Support for bi-directional communication was also introduced that supports new capabilities, such as heartbeats, read-ahead and asynchronous message consumption.
- However, if multiple conversations share the same socket, contention can arise if the total workload for the conversations exceeds the capacity of the socket. Additionally, an overhead is also introduced to serialize access when sending and receiving TCP/IP data.
- `SHARECNV(0)` disables all version 7 enhancements. `SHARECNV(1)` can be used to disable only conversation sharing. Unfortunately, in version 7 the latter incurs a notable performance overhead as a result of enabling the other version 7 capabilities. In version 8, support for `SHARECNV(1)` has been optimized to achieve parity with `SHARECNV(0)`.

Distributed vertical scaling

- **Vertical scaling of distributed queue managers has been enhanced**
  - Various efficiency improvements, including
    - Better cache alignment
    - Extended 64-bit exploitation for locking primitives
    - Better compiler optimisations
    - Faster data conversion, especially for UTF-8
    - Object catalogue restructured
  - Better exploitation of SMP machines
  - Less targeted at internal benchmarks – hopefully more realistic scenarios

Distributed vertical scaling

NOTES

- A number of enhancements have been implemented in version 8 to improve the vertical scaling of queue managers on the distributed platforms. Some of these improvements are listed on the previous slide.
- We've also updated the focus of our performance testing to be less reliant on some internal benchmarks that do not represent realistic customer workloads. Although each customer workload is different, so guarantees are not possible, the performance improvements in version 8 are more likely to be relevant to real-world scenarios.

Publish/subscribe improvements

- **Improved PROXYSUB(FORCE) behaviour**
  - Version 7 uses individual proxy subscriptions
  - Version 8 uses wildcards where appropriate to reduce flows
- **Improved error handling in multi-queue manager environments**
- **Improved scaling for large topic trees**
  - Linear scaling to at least a million topics
- **Improved DISPLAY PUBSUB**
  - Allows detection of unexpected growth in topics/subscriptions

```
AMQ8723: Display pub/sub status details.
   QMNAME(QMGR3)          TYPE(LOCAL)
   STATUS(ACTIVE)         SUBCOUNT(241)
   TPCOUNT(105)
```

Publish/subscribe improvements

N

O

T

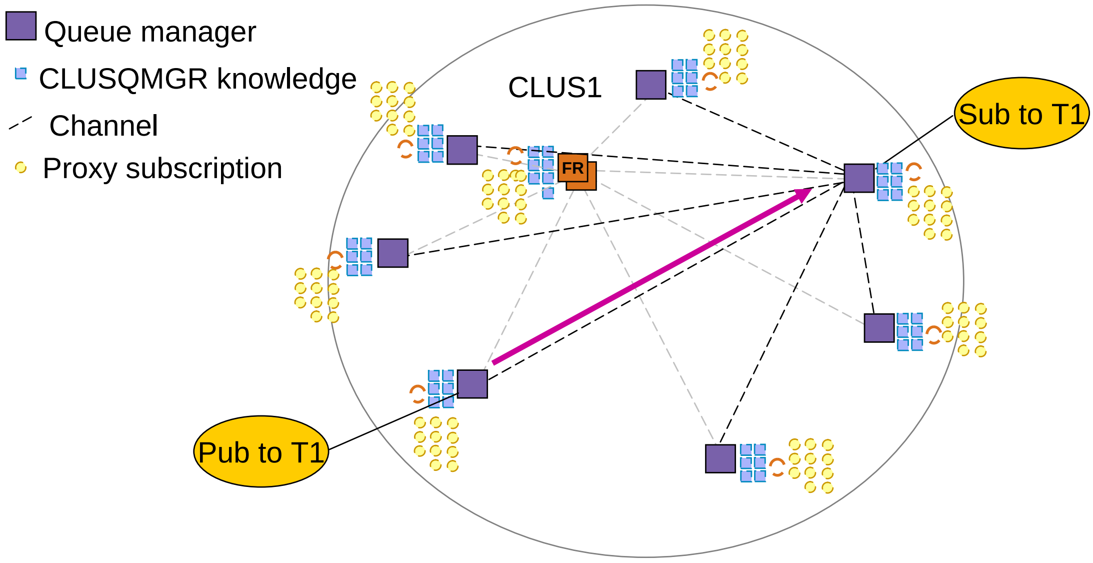
E

S

- In a publish/subscribe hierarchy, or cluster, proxy subscriptions are used to indicate which queue managers in the topology publications need to be propagated to for remote subscribers. If applications have short-lived subscriptions the delay incurred propagating the proxy subscriptions to all queue managers can result in some remote publications not being delivered. PROXYSUB(FORCE) can be specified on a topic to force all publications to be propagated to all remote queue managers. This avoids this problem at the expense of propagating messages even when there are no subscribers connected to a queue manager. In version 8 using PROXYSUB(FORCE) also significantly reduces the overhead of propagating proxy subscriptions for a given topic node.
- Internal topic tree management has been enhanced for large numbers of topics. Performance should scale linearly up to at least a million topics.
- The DISPLAY PUBSUB command has also been enhanced to report the total number of topics and subscribers. This can be used to detect growth in these and help determine the potential impact of a wildcard in a DISPLAY TOPIC/SUB/TPSTATUS/SBSTATUS command.

Routed publish/subscribe

- In version 7, all queue managers in a cluster know everything and need to be able to connect to anyone



TOPIC(T1) CLUSTER(CLUS1)

Routed publish/subscribe

N O T E S

- In previous releases, if publish/subscribe is used within a cluster, all queue managers require knowledge of all subscriptions throughout the cluster for publications to locally connected applications.
- Similarly, every queue manager requires connectivity to every other queue manager to establish proxy-subscriptions and/or deliver publications throughout the cluster.

Routed publish/subscribe

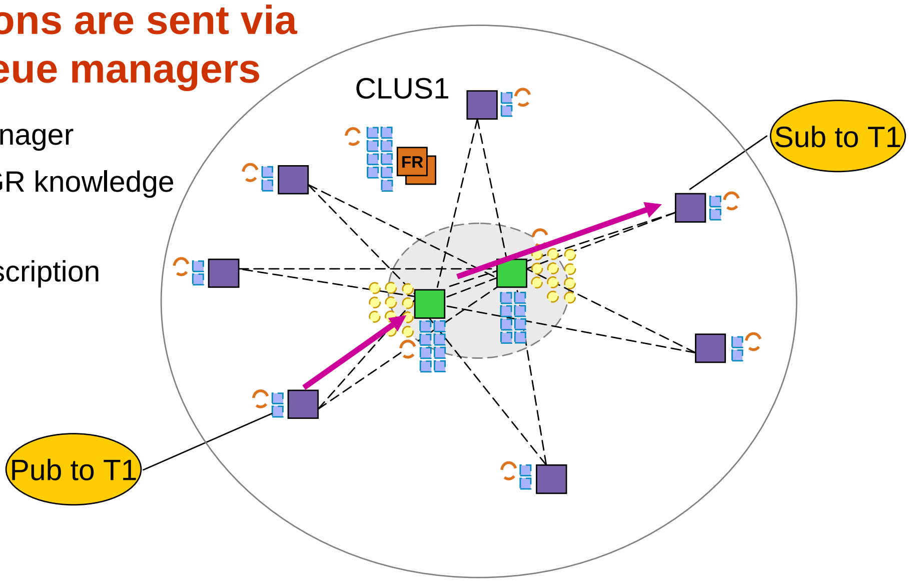
- In version 8 you can configure a subset of queue managers to know everything and connect to everyone
- Publications are sent via these queue managers

▪ Queue manager

▪ CLUSQMGR knowledge

⟋ Channel

○ Proxy subscription



TOPIC(T1) CLUSTER(CLUS1)

Routed publish/subscribe

N

O

T

E

S

- In version 8 you can optionally designate, on a per-topic basis, a subset of queue managers within the cluster as gateways for publish/subscribe.
- This subset of queue managers need to know everything and be able to connect to every other queue manager. However, other queue managers in the cluster only need to know about, and be able to connect to, the subset.
- All publications are routed through one of the gateway queue managers, who then fan them out as required.
- This improves scalability and isolation by reducing traffic within the cluster.

Multiple cluster transmission queues

- **Multiple cluster transmission queues added in V7.5**
  - Support for z/OS and IBM i added in V8
- **Benefits of using multiple transmission queues**
  - Separation of message traffic
    - With a single transmission queue, pending messages for one channel can interfere with those for another, especially when messages build up on the queue
  - Management of messages
    - Use of queue concepts such as `MAXDEPTH` are not useful when using a single transmission queue for all cluster-sender channels
  - System monitoring
    - Tracking the number of messages processed by a cluster-sender channel is not possible using queue monitoring if a single transmission queue is shared by multiple channels, although some information is available using channel status

Multiple cluster transmission queues

N O T E S

- Support for multiple cluster transmission queues was added on the distributed platforms in version 7.5. Version 8 adds support for this capability on z/OS and IBM i.
- A different transmission queue can now be used by each cluster-sender channel, or a subset of cluster-sender channels, instead of all channels using `SYSTEM.CLUSTER.TRANSMIT.QUEUE`.
- The introduction of multiple cluster transmission queues is not designed to improve performance, but to provide a capability to isolate message traffic.
  - It is quicker to assess the impact of an issue if traffic is separated for different applications.
  - If a queue manager is a member of multiple clusters, a different transmission queue can be used for each cluster. This can prevent a build up of messages for one cluster, that results in a full page set or `MAXDEPTH` being reached, impacting the other clusters.
  - If messages for different applications are sent over different cluster channels* a problem for one application can be prevented from impacting others.

* This can be achieved by using a separate cluster for each application, which may overlap with other clusters, or hosting the target queues for each application on different cluster queue managers.

Configuring cluster transmission queues

- **`DEFCLXQ` queue manager attribute**
  - Default transmission queue for cluster-sender channels
  - `SCTQ`
    - Use `SYSTEM.CLUSTER.TRANSMIT.QUEUE`
  - `CHANNEL`
    - Create a permanent-dynamic transmission queue per cluster-sender channel called `SYSTEM.CLUSTER.TRANSMIT.<channel name>`
- **`CLCHNAME` queue attribute**
  - Set on a manually defined transmission queue
  - Generic name for channels that should use it
    - `DEFINE QLOCAL(CLUSTER.XMITQ1) USAGE(XMITQ) CLCHNAME('AAA.*') …`
  - Most specific match is used by a channel

Configuring cluster transmission queues

N O T E S

- The transmission queue to use for each channel can be configured in two ways:
- A manually defined transmission queue can be configured using the `CLCHNAME` attribute. The value of this attribute is a generic, or specific, name for the channels that should use it. The mapping is defined on the transmission queue instead of the channel because the channel's definition is determined from the cluster-receiver definition on the remote queue manager. If a channel's name matches multiple `CLCHNAME` values the most specific match is used to determine which transmission queue to use.
- If no matching queue is found the queue manager uses the default transmission queue setting, specified by the `DEFCLXQ` queue manager attribute, to determine which transmission queue to use. This attribute indicates whether to use `SYSTEM.CLUSTER.TRANSMIT.QUEUE` or to create a permanent-dynamic transmission queue called `SYSTEM.CLUSTER.TRANSMIT.<channel name>`.

GSE GUIDE SHARE EUROPE

Configuring cluster transmission queues

Consider the following definitions …

```
DEFINE QLOCAL(CLUSTER.XMITQ1) USAGE(XMITQ) CLCHNAME(‘AAA.*’) …
DEFINE QLOCAL(CLUSTER.XMITQ2) USAGE(XMITQ) CLCHNAME(‘AAA.BBB’) …
```

What transmission queues are used by the following channels?

– Channel AAA.BBB

> CLUSTER.XMITQ2 because the transmission queue has a specific CLCHNAME value that matches the channel name

– Channel AAA.CCC

> CLUSTER.XMITQ1 because the transmission queue has a generic CLCHNAME value that matches the channel name and there isn’t a more specific match

– Channel XXX.YYY

> It will use either SYSTEM.CLUSTER.TRANSMIT.QUEUE or a permanent-dynamic transmission queue called SYSTEM.CLUSTER.TRANSMIT.XXX.YYY depending on the value of the DEFCLXQ queue manager attribute

GSE GUIDE SHARE EUROPE

Configuring cluster transmission queues

NOTES

- The previous slide illustrates the preferences used by the queue manager to determine which transmission queue should be used by each channel. Given the two manually defined transmission queues shown at the top of the slide, the first channel, AAA.BBB, uses the transmission queue CLUSTER.XMITQ2. This is because this queue has a CLCHNAME value that exactly matches the name of the channel. Although the channel name also matches the generic pattern specified for the transmission queue CLUSTER.XMITQ1, the more specific match takes precedence.
- The second channel, AAA.CCC, only matches the generic CLCHNAME pattern configured for the transmission queue CLUSTER.XMITQ1, so this transmission queue is used.
- The third channel, XXX.YYY, does not match either of the CLCHNAME patterns. Therefore, the queue manager uses the DEFCLXQ queue manager attribute to determine whether to use SYSTEM.CLUSTER.TRANSMIT.QUEUE or a permanent-dynamic transmission queue for the channel called SYSTEM.CLUSTER.TRANSMIT.XXX.YYY. If the latter is used, the transmission queue is created automatically by the queue manager using the model queue SYSTEM.CLUSTER.TRANSMIT.MODEL.QUEUE.

Switching transmission queue

- **A channel switches transmission queue in one of two ways:**
  - Automatically when the channel next starts
    - Changes do not take effect while a channel is running
  - Manually using `CSQUTIL` and the `SWITCH CHANNEL` function
    - This is the equivalent of `runswchl` on distributed platforms
- **Switching sequence**
  1. Channel starts and resolves in-doubt status
  2. Channel initiates switch
  3. Channel switches to get messages from new transmission queue
     - New messages continue to be put to the old transmission queue
  4. Queue manager starts moving messages for the channel from the old transmission queue to the new transmission queue
  5. Switch completes when no committed or uncommitted messages for the channel remain on the old transmission queue
     - New messages now put to the new transmission queue

Switching transmission queue

NOTES

- Administrative changes to the transmission queue each cluster-sender channel uses do not take effect until the channel next starts. When a cluster-sender channel starts it checks for a pending switch of transmission queue. If applicable, the switch is initiated after resolving any in-doubt channel status.
- The switching process completes in two phases. The first phase persists the switch of transmission queue and the channel changes to get messages from the new transmission queue, while messages continue to be put to the old transmission queue. The second phase, which can be long running, involves the queue manager moving messages in the background from the old transmission queue to the new transmission queue. Only when no committed or uncommitted messages for the channel remain on the old transmission queue does the switch complete. Once a switch has completed new messages are put directly to the new transmission queue. Messages for other channels on both the old and new transmission queues are not affected by the switching process.
- An administrator can use `CSQUTIL` to manually switch the transmission queue for a cluster-sender channel while it is not active. This allows for configuration updates to be performed during maintenance windows.

Administering cluster transmission queues

- **Version 8 new function must be enabled**
- **Various console messages output during the switching process**
  - `DISPLAY CHSTATUS` and `DISPLAY CLUSQMGR` can be used to view the transmission queue a cluster-sender channel is using

```
CSQM439I !QM02 CLUSQMGR(QM01)
CLUSTER(CL01)
CHANNEL(CL01.TO.QM01)
STATUS(INACTIVE)
XMITQ(SYSTEM.CLUSTER.TRANSMIT.CL01.TO.QM01)
```

- **`CSQUTIL` can report the following for each cluster-sender channel**
  - Transmission queue currently in use
  - Pending / in-progress switch information
    - The ‘old’ and ‘new’ transmission queue names
    - The number of messages for the channel on the old transmission queue

```
SWITCH CHANNEL(*) STATUS
```

Administering cluster transmission queues

NOTES

- Previous releases of WebSphere MQ on z/OS do not support multiple cluster transmission queues, so this capability cannot be used until version 8 new function has been enabled using the `OPMODE` system parameter, at which time backwards migration is no longer permitted.
- During the switching process various console messages are output to indicate the progress of this operation. The queue manager is responsible for moving messages for the channel to the new transmission queue, so most of the console messages are output in the queue manager job log.
- The display commands for channel status and cluster queue manager information have been enhanced to allow an administrator to view the transmission queue each cluster-sender channel is using. Administrative changes that have not taken effect are not reported by these commands, but `CSQUTIL` can be used to view the transmission queue associated with each channel. If a switch is pending, or in progress, this utility reports the old and new transmission queues, plus the number of messages that remain on the old transmission queue that have yet to be moved.

Summary

Afternoon Session

Platforms & Standards	*Security*	*Scalability*	*System z exploitation*
64-bit for all platforms	Userid authentication via OS & LDAP	Multiplexed client performance	64-bit buffer pools in MQ for z/OS means less paging, more performance
Support for JMS 2.0	User-based authorisation for Unix	Queue manager vertical scaling	Performance and capacity
Improved support for .Net and WCF	AMS for IBM i & z/OS	Publish/Subscribe improvements	Performance enhancements for IBM Information Replicator (QRep)
Changes to runmqsc	DNS Hostnames in CHLAUTH records	Routed publish/subscribe	Exploit zEDC compression accelerator
SHA-2 for z, i & NSS	Multiple certificates per queue manager	Multiple Cluster Transmit Queue on all platforms	SMF and shared queue enhancements