Authentication in MQ

Morag Hughson

morag@mqgem.com

Authentication in MQ - Abstract

N O T E S

- Over the last few releases of IBM MQ, there have been a number of security features added that provide authentication features. This session illustrates all those features and how they work together. This session will cover the use of SSL/TLS on channels (and some changes in this area in MQ V8); the security exit facility which is still available although some of the common uses for it are now built into the MQ product; the use of CHLAUTH rules (added in MQ V7.1 and enhanced in MQ V8) and the MQ V8 feature Connection Authentication, which allows the validation of user ID and password from an application at connect time.

Agenda

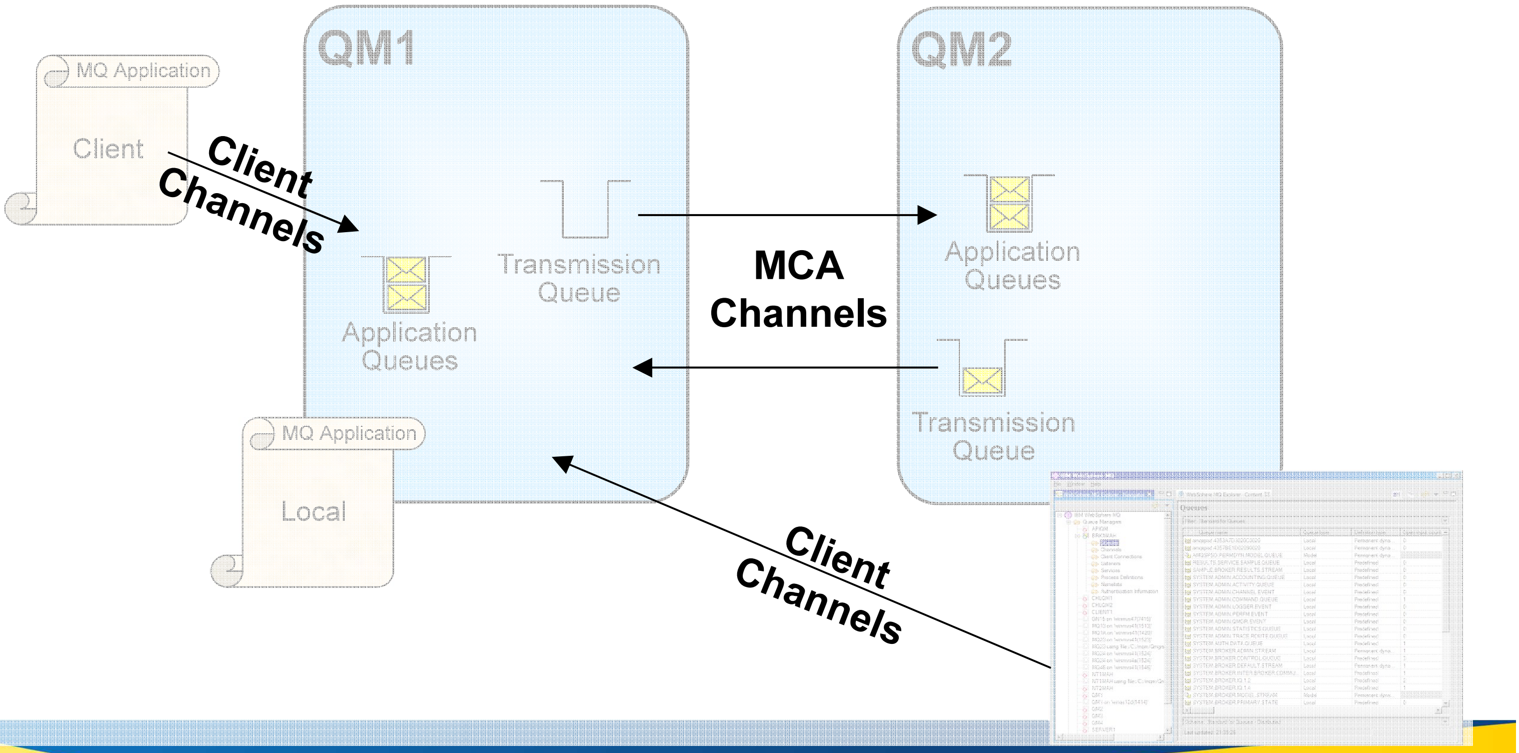
- **Authentication - Channels**
  - SSL/TLS on MQ Channels
    - Some V8 updates
  - Channel Authentication (CHLAUTH)
    - Introduced in V7.1
    - Some V8 updates
  - Security Exits
- **Authentication - Applications**
  - O/S Logon
    - Useful for locally bound applications
    - Not to be relied upon for client applications!
  - Connection Authentication using MQCONNX
    - Introduced in V8
- **Tying Authentication to Authorization**
- **How do they all fit together?**

Agenda

N O T E S

- We'll take a little time to remind ourselves what each of these features is and how it works. Then we'll look at how these features work together and how the authentication features ultimately can set the user ID used for authorization checks.

Authentication - Channels



QM1

QM2

MQ Application

Client

Client Channels

Transmission Queue

Application Queues

MCA Channels

Application Queues

Transmission Queue

MQ Application

Local

Client Channels

Authentication – Channels – Notes

N
O
T
E
S

- First we shall look at authentication of remote partners, whether clients or remote queue managers, which connect into your system.
- It is very important to ensure that you have good authentication mechanisms in place for any remote partners. You must know that the connections coming into your queue manager can be trusted.

Identification

- When an MQ application connects remotely to a queue manager it can assert an identity across the network connection to the queue manager. This identity could be anything and so should not be trusted without some form of queue manager side authentication.

Authentication

- Authentication is the way in which a channel ensures that the other end of the channel is who they say they are. Channels can make use of SSL/TLS to authenticate a digital certificate sent by the partner. In WebSphere MQ V7.1 Channel Authentication Records can be used to do many of the jobs a security exit can do, such as allowing or blocking a channel based on IP address, Certificate DN, Remote Queue Manager Name or Client User ID.
- Once a remote partner has been authenticated, Channel Authentication Records or a security exit can also set the identity that this channel will use for all access control checks.

Confidentiality

- In an ideal environment all channels would be running inside the enterprise with good physical security. However, often there will be cross enterprise channels or channels running on networks where physical security can not be guaranteed. In those cases it is worth considering adding some level of encryption to the data flow. This can either be done in channel exits or by using SSL/TLS on the channels.

Channels – Authentication Facilities

- **Transport Layer Security (SSL/TLS)**
  - Using Digital Certificates
- **Channel Authentication Records**
  - Introduced in WebSphere MQ V7.1
  - Updated in IBM MQ V8
- **Security Exits**
  - Many Vendor exits available

Channels – Authentication Facilities – Notes

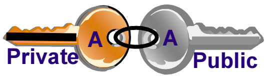
N O T E S

- Over the next few pages we are going to introduce each of the following facilities which allow you to provide some authentication for your client or MCA channels. The strength of the authentication provided varies by each facility so the choice of facility should take that into account when making a business decision as to the level of authentication required.

Using SSL/TLS with WebSphere MQ

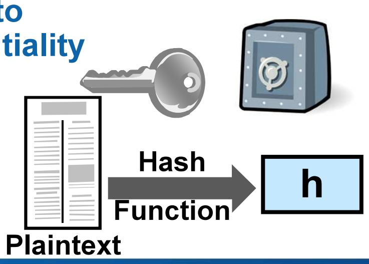
- **Get your certificates for Authentication**
  - Digital Certificates
  - Asymmetric Keys

- **Put your certificates in a place that MQ can use**
  - Label them how you wish



- **Decide if you need revocation status checking (LDAP or OCSP)**

- **Decide if you need cipher spec restriction (FIPS or SUITEB)**
- **Configure your channels to use SSL/TLS for Confidentiality**
  - Symmetric Key Cryptography
- **… and Data Integrity**
  - Hash Function



- **WebSphere MQ SSL Wizard (MO04)**

SSLKEYR(QM1KEYRING)
CERTLABL('QM1Cert')

New in V8

SSLCRLNL(REVOKE.NL)

SSLFIPS(NO)
SUITEB(NONE)

SSLCIPH(RC4_MD5_US)
SSLRKEYC(999 999 999)
SSLCAUTH(REQUIRED)
SSLPEER('O=IBM')
SSLCERTI('CN=MQ CA')

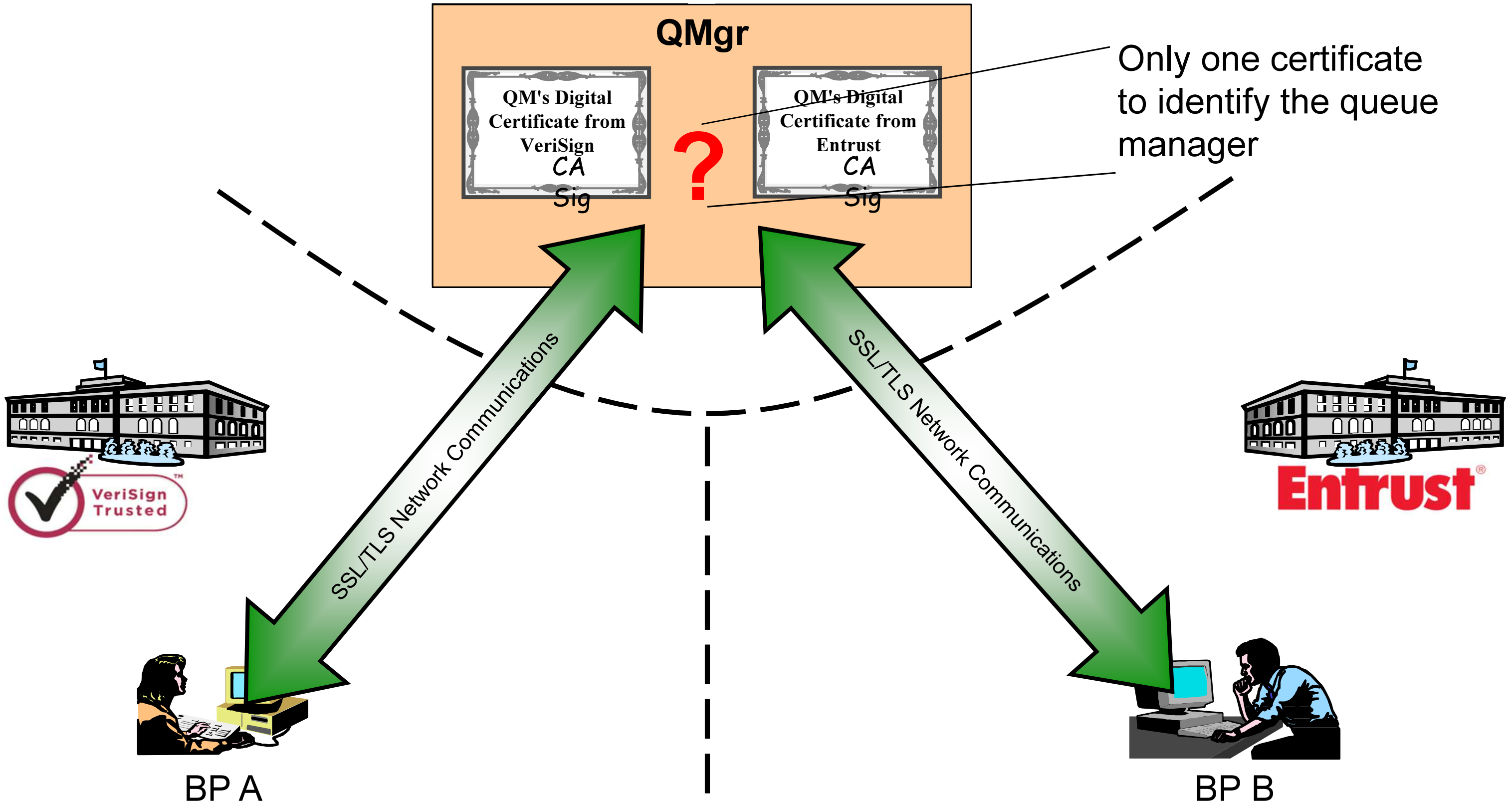
New in V8

Using SSL/TLS with WebSphere MQ – Notes

N O T E S

- The three main issues that Transport Level Security (SSL/TLS) addresses are Confidentiality, Data Integrity and Authentication. The techniques that it uses to address these issues are
  - For Confidentiality, we have symmetric key cryptography with the capability to periodically reset the secret key;
  - For Data Integrity we have the hash function; and
  - For Authentication we have digital certificates, asymmetric keys and certificate revocation lists.
- WebSphere MQ makes use of these techniques to address these security issues.
- Digital Certificates and Public Keys are found in a key repository which can be specified to WebSphere MQ (SSLKEYR on the queue manager). The certificates have a local label which can either be specified using CERTLABL (on the queue manager or on a per channel basis) or will default to ibmwebspheremq<qmgr-name>. CERTLABL was added
- One can specify which symmetric key cryptography algorithm and which hash function to use by providing WebSphere MQ with a SSLCipherSpec (SSLCIPH on a channel). The secret key can be periodically reset by setting an appropriate number of bytes in SSLKeyResetCount (SSLRKEYC on the queue manager).
- The set of cipher specs to be used by the queue manager can be restricted to a set that are compliant to the FIPS 140-2 standard (SSLFIPS on the queue manager) available on both distributed and in WebSphere MQ V7.1 on z/OS; or to the Suite-B standard (SUITEB on the queue manager) available on the distributed platforms in WebSphere MQ V7.1
- We can choose to authenticate both ends of the connection or only the SSL Server end of the connection (SSLCAUTH on a channel). Also we can make choose to do certificate revocation status checking using either LDAP CRLs or OCSP (SSLCRLNL on the queue manager).
- We can also check that we are talking to the partner we expect to be talking to by setting a CHLAUTH rule to check the Subjects DN from the certificate (SSLPEER) and the Issuer's DN (SSLCERTI). SSLCERTI was added in IBM MQ V8.

Business Partners with different CA requirements

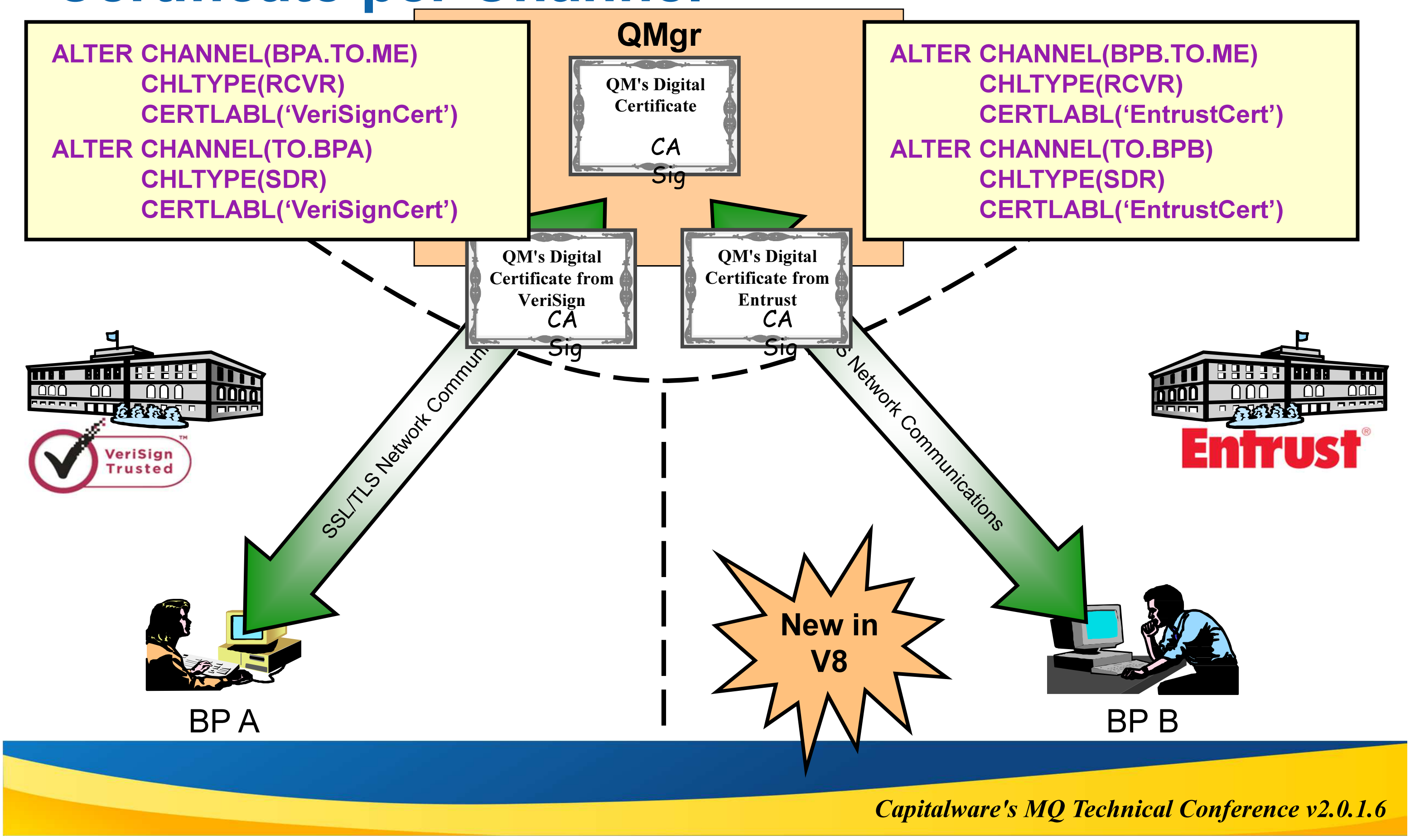

QMgr
QM's Digital Certificate from VeriSign
CA
Sig
?
QM's Digital Certificate from Entrust
CA
Sig
Only one certificate to identify the queue manager
SSL/TLS Network Communications
SSL/TLS Network Communications
VeriSign Trusted
Entrust
BP A
BP B

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 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.

Certificate per Channel



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

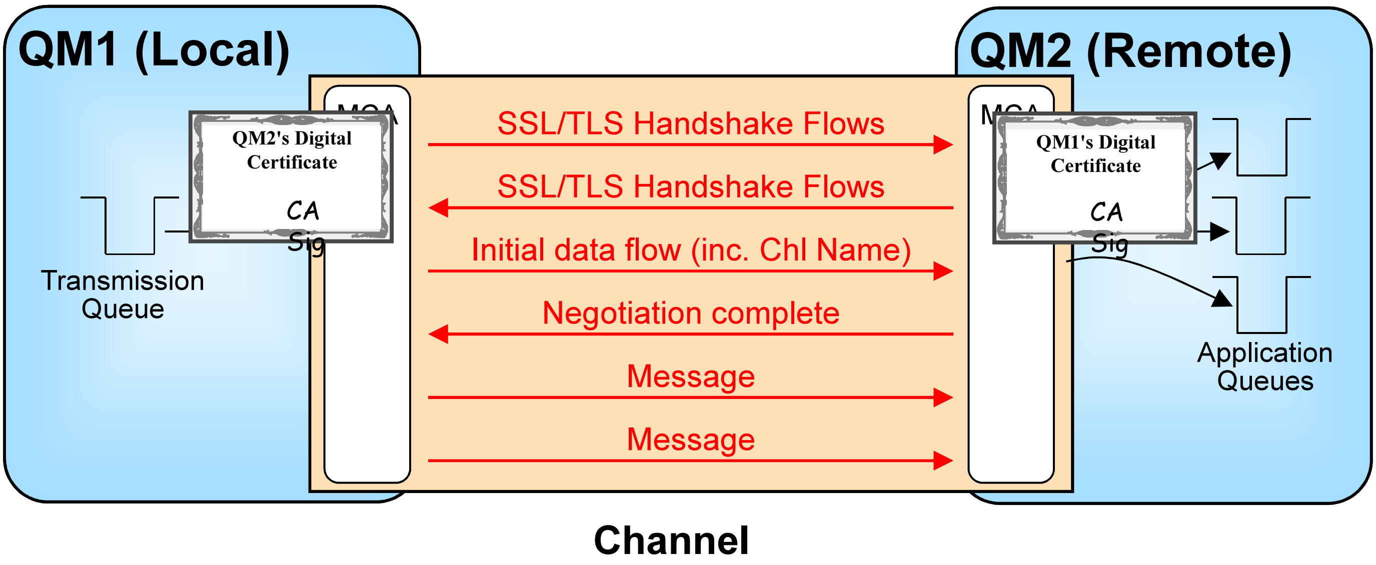
```
ALTER CHANNEL(BPB.TO.ME)
        CHLTYPE(RCVR)
        CERTLABL('EntrustCert')
ALTER CHANNEL(TO.BPB)
        CHLTYPE(SDR)
        CERTLABL('EntrustCert')
```

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 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 hasn't IBM MQ always done this?

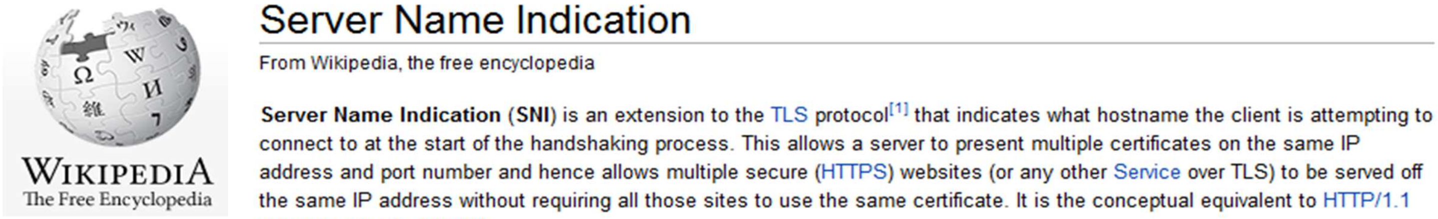


Why hasn't IBM MQ always done this? – Notes

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).

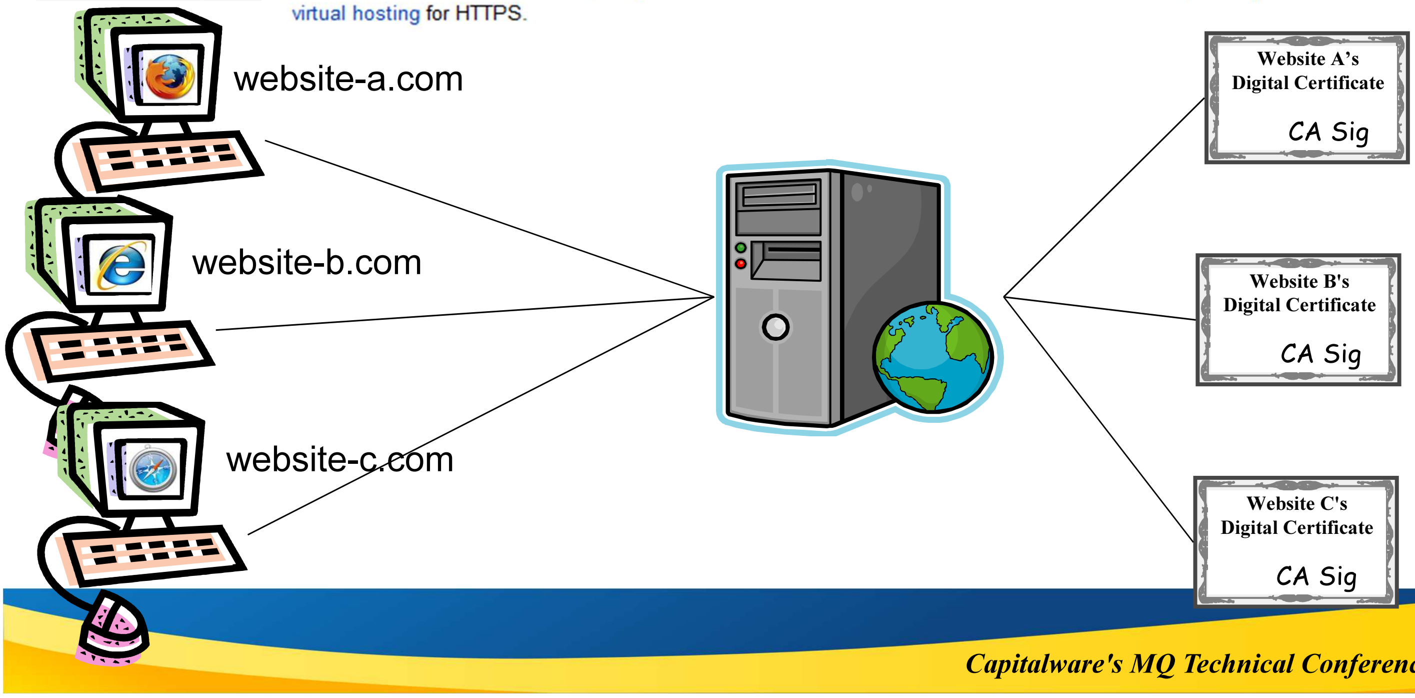
Server Name Indication



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.

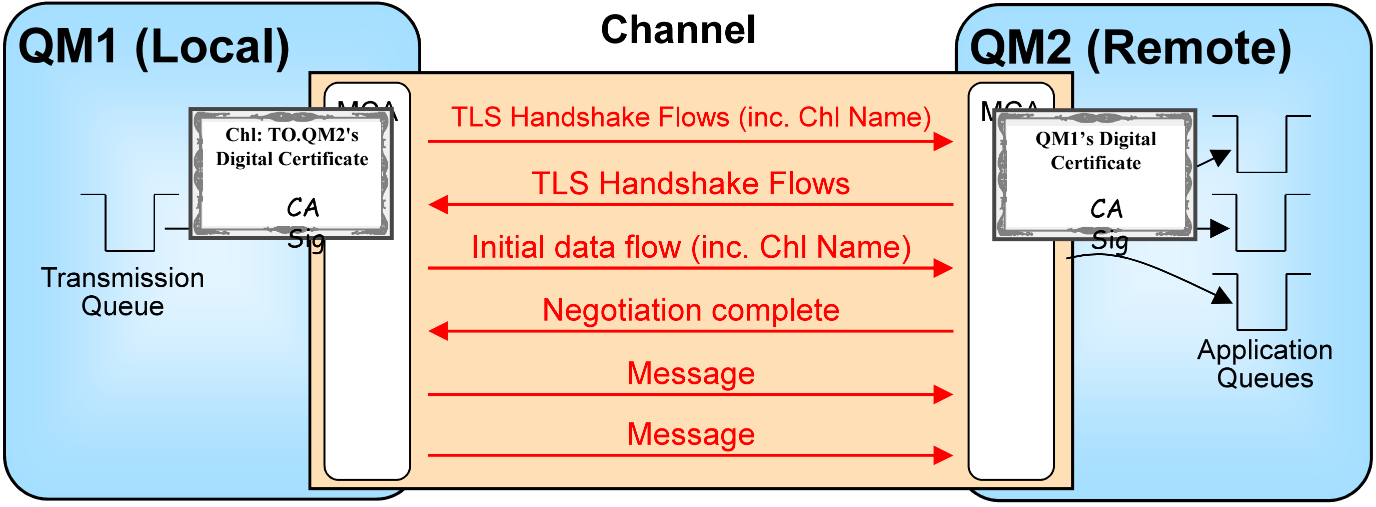

website-a.com
website-b.com
website-c.com
Website A's Digital Certificate
CA Sig
Website B's Digital Certificate
CA Sig
Website C's Digital Certificate
CA Sig

Server Name Indication – Notes

NOTES

- 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



Channel

QM1 (Local)

QM2 (Remote)

TLS Handshake Flows (inc. Chl Name)

TLS Handshake Flows

Initial data flow (inc. Chl Name)

Negotiation complete

Message

Message

- **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**
  - IBM MQ only detects that it 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

- 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.

Channel Authentication Records

- **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/TLS Distinguished Name matching
  - Mandate user ID & password checking
- **Rules can use any of the following identifying characteristics of the inbound connection**
  - IP Address
  - Hostnames
  - SSL/TLS Subject’s Distinguished Name
  - SSL/TLS Issuer’s Distinguished Name
  - Client asserted user ID
  - Remote queue manager name

Red items: New in V8

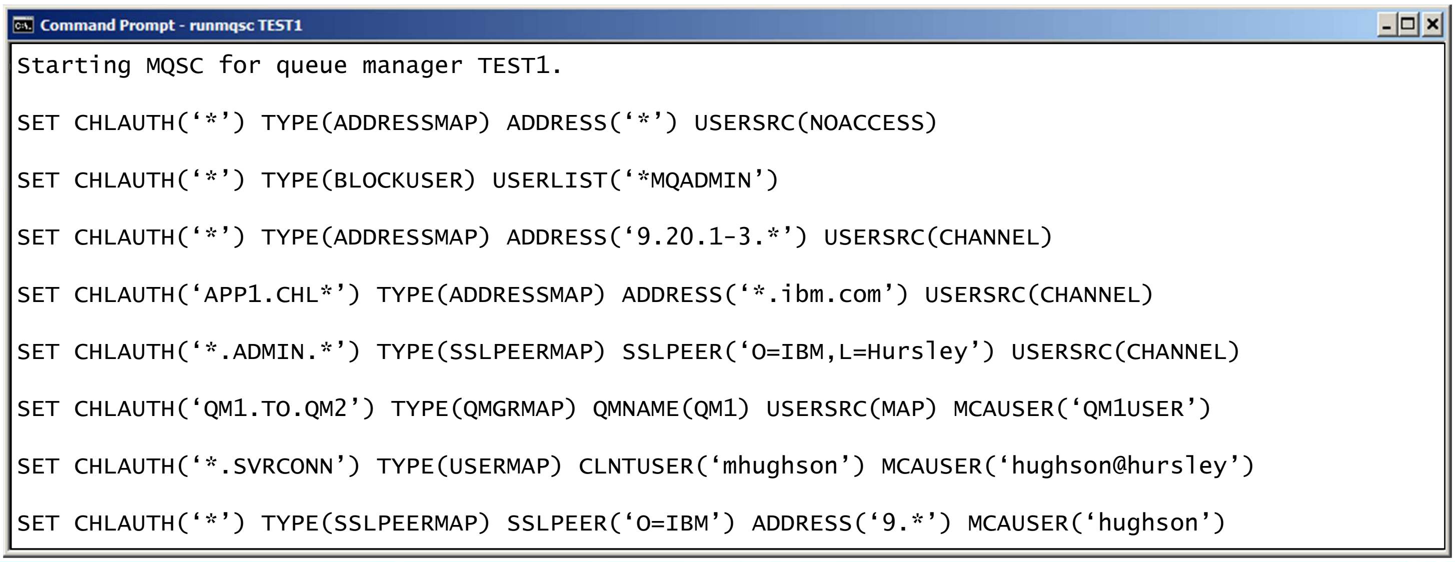
Channel Authentication Records – Notes

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.

CHLAUTH – Configuration

- **Create rules using**
  - MQSC: SET CHLAUTH
  - PCF
- **Pattern matching**
  - Channel Name/QMgr Name/Hostname
    - Beginning, middle, end
  - IP addresses (IPV4 or IPV6)
  - SSL Peer Name (as today)



```
Starting MQSC for queue manager TEST1.

SET CHLAUTH(‘*’) TYPE(ADDRESSMAP) ADDRESS(‘*’) USERSRC(NOACCESS)

SET CHLAUTH(‘*’) TYPE(BLOCKUSER) USERLIST(‘*MQADMIN’)

SET CHLAUTH(‘*’) TYPE(ADDRESSMAP) ADDRESS(‘9.20.1-3.*’) USERSRC(CHANNEL)

SET CHLAUTH(‘APP1.CHL*’) TYPE(ADDRESSMAP) ADDRESS(‘*.ibm.com’) USERSRC(CHANNEL)

SET CHLAUTH(‘*.ADMIN.*’) TYPE(SSLPEERMAP) SSLPEER(‘O=IBM,L=Hursley’) USERSRC(CHANNEL)

SET CHLAUTH(‘QM1.TO.QM2’) TYPE(QMGRMAP) QMNAME(QM1) USERSRC(MAP) MCAUSER(‘QM1USER’)

SET CHLAUTH(‘*.SVRCONN’) TYPE(USERMAP) CLNTUSER(‘mhughson’) MCAUSER(‘hughson@hursley’)

SET CHLAUTH(‘*’) TYPE(SSLPEERMAP) SSLPEER(‘O=IBM’) ADDRESS(‘9.*’) MCAUSER(‘hughson’)
```

CHLAUTH – Configuration – Notes

NOTES

- Here we show some example rules illustrating the commands used for creating the rules. These examples are in MQSC. There is also PCF, and this is used by the MQ Explorer GUI.
- Some of these examples illustrate the pattern matching that can be applied to channel names, IP addresses, Hostnames, SSL/TLS DNs and remote queue manager names. Also we see all three types of rules, blocking channels – USERSRC(NOACCESS); allowing channels to run with the user ID provided by the channel – USERSRC(CHANNEL); and assigning a user ID to a channel – USERSRC(MAP) MCAUSER(user-id). USERSRC(MAP) is the default so we also see in another example that it does not need to be specified on the command.

Restricting the Mappings

- Rules matching on
  - SSL Peer Name
  - Remote QMgr Name
  - Client User ID
- Can add IP address/Hostname
- Restrict where an SSL Certificate can be used from
  - Specific IP address/Hostname
- Restrict where a queue manager or client user ID can come from
  - Specific IP address/Hostname

Mapped \ Restrict By	SSL Peer	QM Name	Client User	IP Address/Hostname
SSL Peer	▨	X	X	8
QM Name		▨	▨	8
Client User		▨	▨	8
IP Address				▨

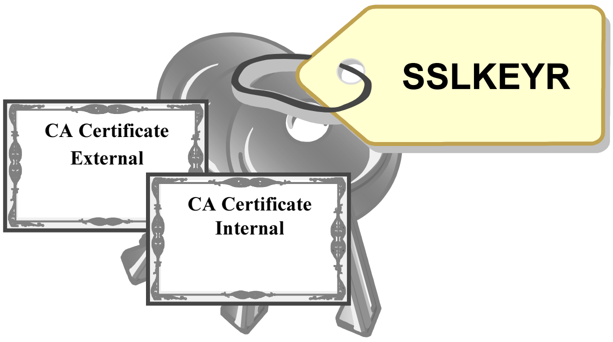
```
SET CHLAUTH(*) TYPE(SSLPEERMAP)
SSLPEER('L="Hursley"') MCAUSER(HURUSER) ADDRESS('9.20.*')

SET CHLAUTH(*) TYPE(QMGRMAP)
QMNAME(CLUSQM*) MCAUSER(CLUSUSR) ADDRESS('*.ibm.com')
```

Restricting the Mappings - Notes

N
O
T
E
S

- When mapping from an SSL certificate DN, you may also want to ensure that certificate is being used from the correct IP address, mitigating what might happen if a certificate is stolen.
- When mapping from a queue manager name, you may also want to ensure that the queue manager is running on the correct IP address to ensure it is not a rogue queue manager with the same name as one in your cluster for example.
- We could imagine using the remote queue manager name or the client user ID as a restrictor on an SSL Peer rule, however feedback from EAP did not suggest anyone needed it so it was not implemented. For the most part, attributes within the X509 DN will contain the same information for most practical uses. For example CN=<Queue Manager Name>.

Fully Qualifying your Peer Name rules



- **Key Repository contains**
  - All CA certs we trust
  - Multiple CAs means possible DN clashes
- **External CAs**
  - Checks and balances
  - Unlikely to have DN clashes
- **Internal CAs**
  - Less rigid
  - May give out certs exactly as requested
  - May end up with clashes
- **Could solved in a Security Exit**
  - MQCD.SSLPeerNamePtr
  - MQCXP.SSLRemCertIssNamePtr
- **CHLAUTH rules extended**
  - Check Subject's DN (SSLPEER)
  - Check Issuer's DN (SSLCERTI)

```
SET  CHLAUTH(BPA.TO.ME)
     TYPE(SSLPEERMAP)
     SSLPEER('O=IBM')
     MCAUSER(BPAUSR)
     SSLCERTI('CN=External')
```

New in V8

Fully Qualifying your Peer Name rules – Notes

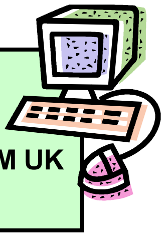
N O T E S

- As we just saw, you can add IP address or hostname restrictors to many of the rule types to further qualify the matching that happens.
- In the case of a Peer name map, you can fully qualify the certificate matching by providing both the Subject's DN (SSLPEER) and the Issuer's DN (SSLCERTI) on a rule. SSLCERTI is new in MQ V8.
- This is especially important if you have more than one Certificate Authority (CA) certificate in your key repository which you may be more likely to do with the introduction of multiple certificates for one queue manager which was a new feature in MQ V8.
- 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 Externals CAs is that they guarantee not to issue the certificates with the same DN as another certificate that they have already issued. 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 non-unique 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 qualified to use both SSLPEER (the subject's DN) and SSLCERTI (the issuer's DN).

CHLAUTH – Precedence

Precedence matching

- Most specific rule is matched



Chl: MY.CHANNEL
IP: 9.20.1.123
DN: CN=Morag Hughson.O=IBM UK
UID: mhughson

- **Identifying attributes are**
  - Channel Name
  - SSL Peer Name pattern
    - Precedence defined for partial patterns
  - Remote queue manager name pattern (MCA channels)
  - Client asserted user ID (MQI channels)
    - No pattern matching on this
  - IP address pattern
  - Hostname pattern (least specific)

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

- **Within SSL Peer Name matching**
  - Most specific substring is matched

CHLAUTH – Precedence – Notes

N

O

T

E

S

- When there is more than one rule that could match the inbound connection in question, then we define which rule will actually be used by defining the precedence order of what is the most specific match. The table shows that SSL Peer Names are considered a more specific match than a queue manager name or client user ID (because there is much more detailed information in an SSL Peer Name); and Hostnames are considered the least specific since clearly more than one queue manager or client can be connecting from the same IP address/Hostname. Hostnames are even less specific than IP addresses because an IP address can have multiple hostnames.

SSL DN Precedence Mapping Example

```
SET CHLAUTH(*) TYPE(SSLPEERMAP) SSLPEER(‘OU=“MQ Devt”’)
MCAUSER(MQUSER)

SET CHLAUTH(*) TYPE(SSLPEERMAP) SSLPEER(‘L=“Hursley”’)
MCAUSER(HURUSER)
```

Order	DN Substring	Name
1	CN=	Common name
2	T=	Title
3	OU=	Organizational unit
4	O=	Organization
5	L=	Locality
6	ST=, SP=, S=	State or province name
7	C=	Country

Most Specific Match

CN=Morag Hughson.OU=MQ Devt.
O=IBM UK.L=Hursley.C=UK

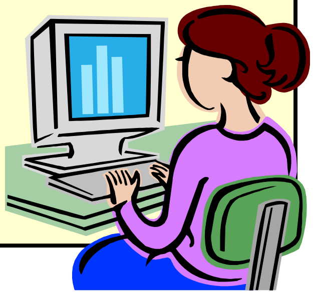
SSL DN Precedence Mapping Example – Notes

N O T E S

- Not only do we define the order of precedence between the various different identifying characteristics of an inbound connection, we also must do a similar job for SSL Peer Name.
- Here is an example to illustrate what happens when two partial patterns could both match an inbound Distinguished Name (DN) from a client.
- We want the most specific match to be used, so we have defined a precedent order of what we mean by the most specific.
- The table shown here that defines the precedence order is a subset of the contents of an SSL Peer Name in WebSphere MQ V7.1. It suffices to describe this example. For the full table of SSL Peer Name attributes, visit Knowledge Centre here:- http://www.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.sec.doc/q009860_.htm

Channel Authentication – How should I use this?

```
SET CHLAUTH(*) TYPE(ADDRESSMAP) ADDRESS(‘*’) USERSRC(NOACCESS)

SET CHLAUTH(BPCHL.*) TYPE(SSLPEERMAP) SSLPEER(‘O=Bank of Shetland’) MCAUSER(BANK123)

SET CHLAUTH(BPCHL.*) TYPE(SSLPEERMAP) SSLPEER(‘O=Bank of Orkney’) MCAUSER(BANK456)

SET CHLAUTH(SYSTEM.ADMIN.SVRCONN) TYPE(ADDRESSMAP)
ADDRESS(‘9.20.1-30.*’) MCAUSER(ADMUSER)

SET CHLAUTH(TO.CLUS.*) TYPE(QMGRMAP)
QMNAME(CLUSQM*) MCAUSER(CLUSUSR) ADDRESS(‘9.30.*’)
```



“Our internal cluster doesn’t use SSL, but we must ensure only the correct queue managers can connect into the cluster”

CHLAUTH – How should I use this? - Notes

N

O

T

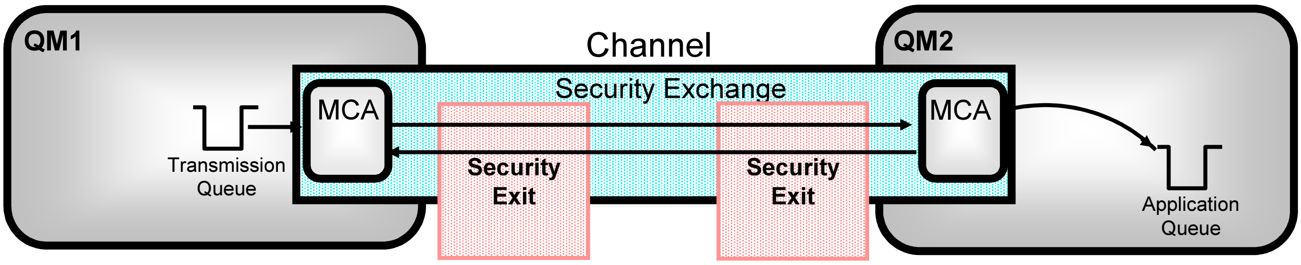
E

S

- Here is an example of how we expect this to be used.
- Our business requires that “We must make sure our system is completely locked down”. So we start off with a rule that blocks everyone. Therefore anyone that doesn’t match a more specific rule will not be allowed in.
- Our business requires that “Our Business Partners must all connect using SSL, so we will map their access from the certificate DNs”. So we have some rules that map specific DNs of our Business Partners to specific user IDs. Previously you might have done this by having separate channel definitions for each BP, now if you wish they can come into the same receiver definition.
- Our business requires that “Our Administrators connect in using MQ Explorer, but don’t use SSL. We will map their access by IP Address”. So we have a rule that gives them all a single administrative access user ID based on a range of IP addresses.
- Our business requires that “Our internal cluster doesn’t use SSL, but we must ensure only the correct queue managers can connect into the cluster”. So we have a rule that gives access to the correctly named queue managers but only if they come from a recognised IP address.

Security Exits

- **Channel 'Gate Keeper'**
  - Indefinite exchange of data between exits
  - No defined format
  - No communications knowledge required
  - Can end channel
  - Can set MCAUSER
- **Many traditional uses now covered by recent product features**
  - Always there for niche use cases

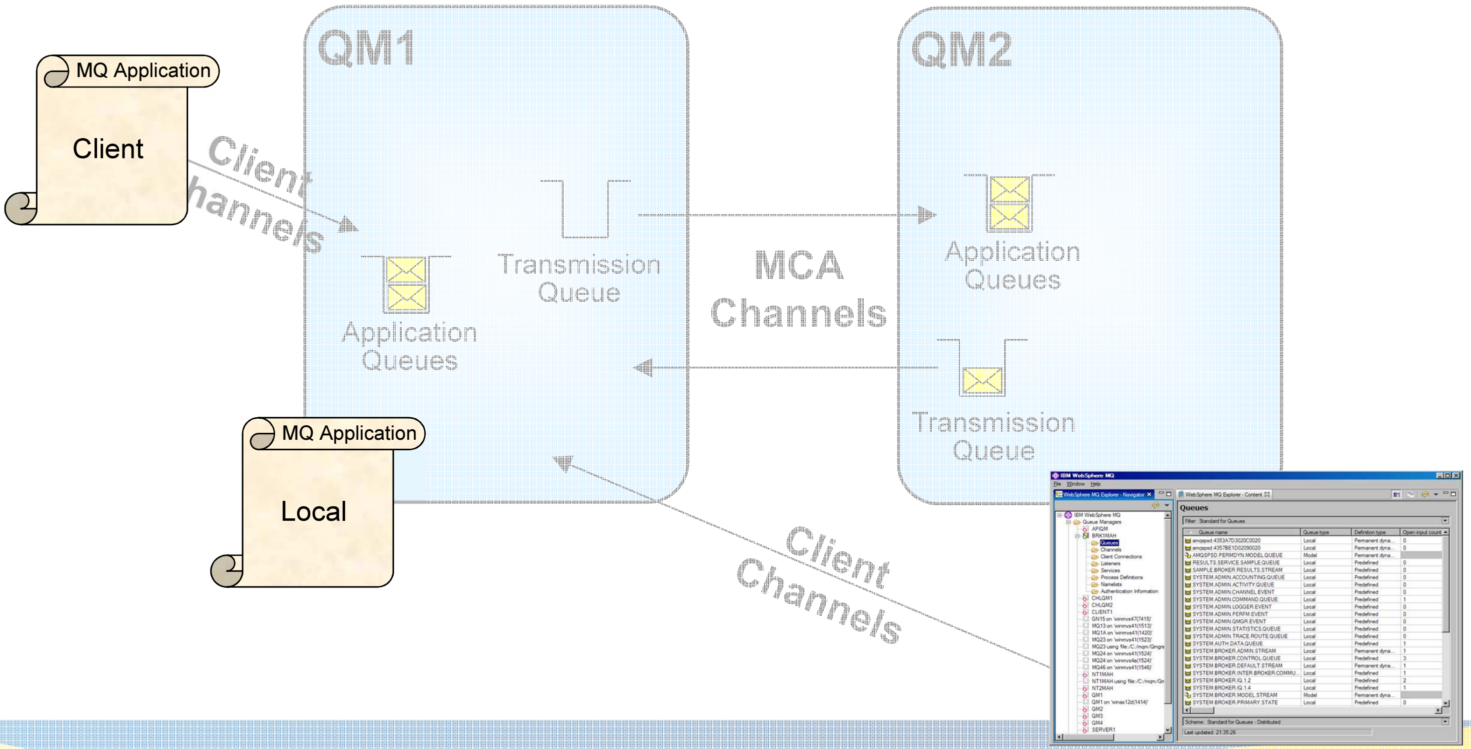


Security Exits – Notes

N O T E S

- One of the problems with authentication is that the industry could not decide how it should be done. Different environments suit different strategies and require different levels of security. The most common approaches seem to be third party authenticators such as Kerberos, SSL/TLS and Public/Private key encryption. Originally, WebSphere MQ decided that the most flexible approach was to make authentication a plug in service. That way each channel could have exactly the level of authentication it needed.
- Authentication can now be done without the use of a security exit, by using SSL and digital certificates and/or by using Channel Authentication Records.
- Security exits are the first channel exits to gain control of the conversation. They can exchange free format data with their remote partner, exchanging passwords, public keys etc to authenticate the remote partners request.
- No knowledge of communications is required. The exit merely passes a buffer of data back to the MCA who then transmits it to the partner machine. The data is received by the other MCA and then passed to the other security exit.
- If the security exit agrees with the authentication then it can change the default userid used for access control, known as the MCAUserid.
- A number of security exits are shipped as samples with the product. There are also some available for download from the supportpac web site. A number of third party products are also available.

Authentication – Applications



IBM WebSphere MQ

File Window Help

WebSphere MQ Explorer - Navigator

- IBM WebSphere MQ
  - Queue Managers
    - APIQM
    - BRK1MAH
      - Queues
      - Channels
      - Client Connections
      - Listeners
      - Services
      - Process Definitions
      - Namelists
      - Authentication Information
    - CHLQM1
    - CHLQM2
    - CLIENT1
    - GN15 on 'winmvs47(7415)'
    - MQ13 on 'winmvs41(1513)'
    - MQ1A on 'winmvs41(1420)'
    - MQ23 on 'winmvs41(1523)'
    - MQ23 using 'file:/C:/mqm/Qmgrs
    - MQ24 on 'winmvs41(1524)'
    - MQ24 on 'winmvs4a(1524)'
    - MQ46 on 'winmvs41(1546)'
    - NT1MAH
    - NT1MAH using 'file:/C:/mqm/Qn
    - NT2MAH
    - QM1
    - QM1 on 'winas12d(1414)'
    - QM2
    - QM3
    - QM4
    - SERVER1

WebSphere MQ Explorer - Content

Queues

Filter: Standard for Queues

Queue name	Queue type	Definition type	Open input count
amqspsd.4353A7D3020C0020	Local	Permanent dyna...	0
amqspsd.4357BE1D02090020	Local	Permanent dyna...	0
AMQSPSD.PERMDYN.MODEL.QUEUE	Model	Permanent dyna...	
RESULTS.SERVICE.SAMPLE.QUEUE	Local	Predefined	0
SAMPLE.BROKER.RESULTS.STREAM	Local	Predefined	0
SYSTEM.ADMIN.ACCOUNTING.QUEUE	Local	Predefined	0
SYSTEM.ADMIN.ACTIVITY.QUEUE	Local	Predefined	0
SYSTEM.ADMIN.CHANNEL.EVENT	Local	Predefined	0
SYSTEM.ADMIN.COMMAND.QUEUE	Local	Predefined	1
SYSTEM.ADMIN.LOGGER.EVENT	Local	Predefined	0
SYSTEM.ADMIN.PERFM.EVENT	Local	Predefined	0
SYSTEM.ADMIN.QMGR.EVENT	Local	Predefined	0
SYSTEM.ADMIN.STATISTICS.QUEUE	Local	Predefined	0
SYSTEM.ADMIN.TRACE.ROUTE.QUEUE	Local	Predefined	0
SYSTEM.AUTH.DATA.QUEUE	Local	Predefined	1
SYSTEM.BROKER.ADMIN.STREAM	Local	Permanent dyna...	1
SYSTEM.BROKER.CONTROL.QUEUE	Local	Predefined	3
SYSTEM.BROKER.DEFAULT.STREAM	Local	Permanent dyna...	1
SYSTEM.BROKER.INTER.BROKER.COMMU...	Local	Predefined	1
SYSTEM.BROKER.IQ.1.2	Local	Predefined	2
SYSTEM.BROKER.IQ.1.4	Local	Predefined	1
SYSTEM.BROKER.MODEL.STREAM	Model	Permanent dyna...	
SYSTEM.BROKER.PRIMARY.STATE	Local	Predefined	0

Scheme: Standard for Queues - Distributed

Last updated: 21:35:26

Authentication – Applications – Notes

N O T E S

- Now we shall look at authentication of applications, whether client connected or locally bound, which connect into your system.
- It is very important to ensure that you have good authentication mechanisms in place for any application. You must know that the connections made to your queue manager can be trusted.
- These applications may be business applications putting and getting messages from application queues, or administrative applications, issuing commands to the queue manager.

Applications – Authentication Facilities

- **O/S Logon**
  - Useful for locally bound applications
  - Not to be relied upon for client applications!
    - Use client channel authentication
- **MQCONNX**
  - Connection Security Parameters
    - User ID and Password

Applications – Authentication Facilities – Notes

N O T E S

- Over the next few pages we are going to introduce each of the following facilities which allow you to provide some authentication for your client or locally bound application.

Identification

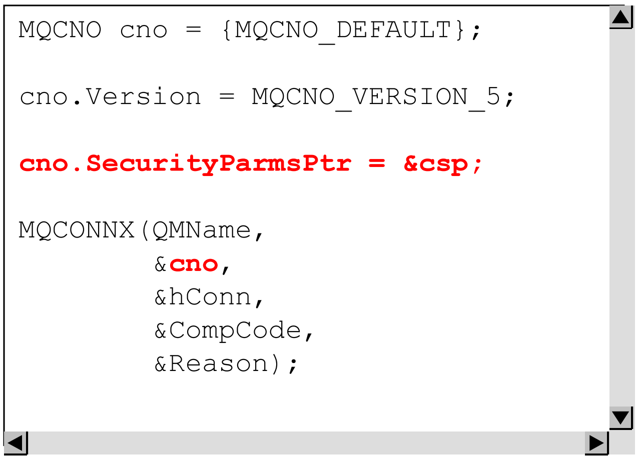
- When an MQ application connects to the queue manager the O/S is interrogated to discover the user ID that it is running under. This is used as the identity. We can see this user ID in the context information of a message.

Authentication

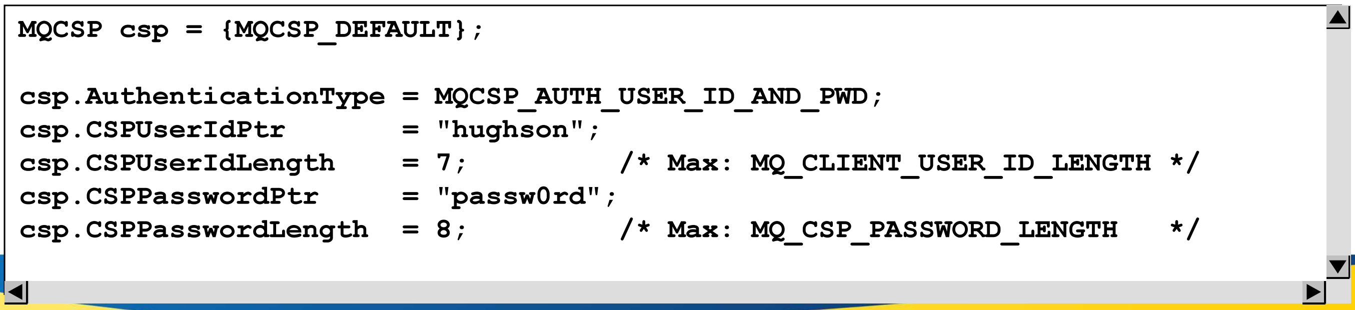
- A locally bound MQ application is running against MQ under an user ID that the O/S has provided and which has been logged onto prior to running the application. This may be enough authentication for a locally bound application for your business purposes, or you may wish more.

Connection Authentication

- **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   */
```

Authentication - MQCONNX - Notes

N

O

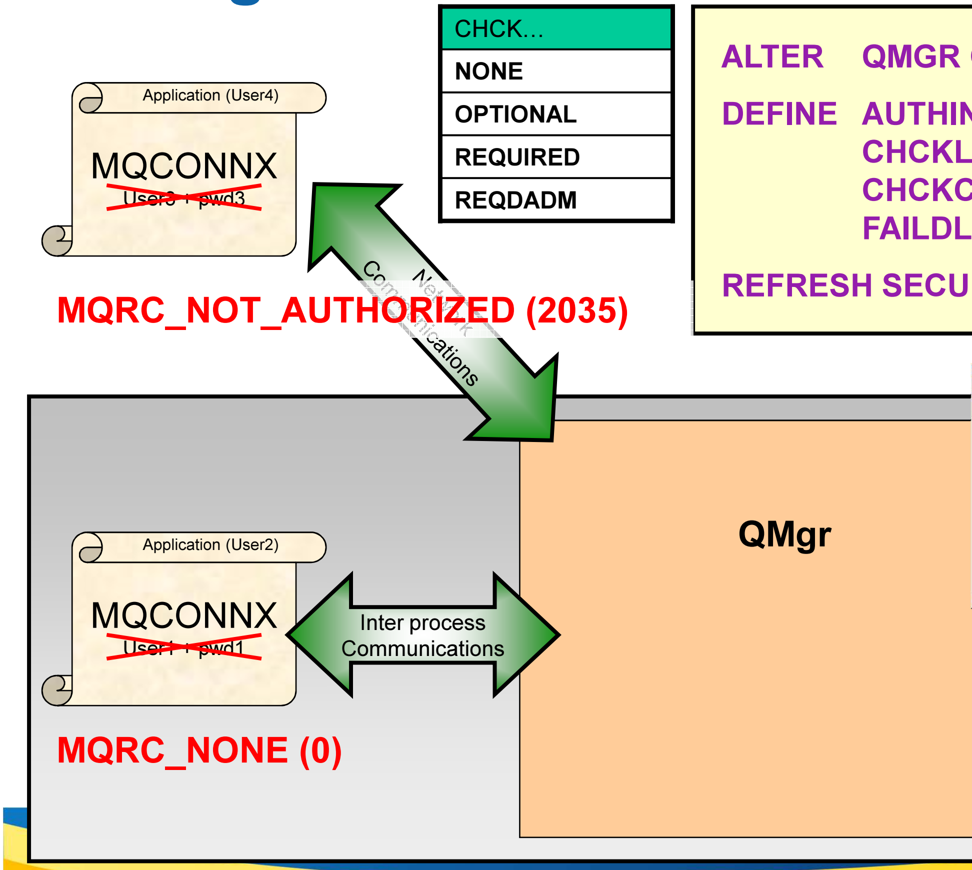
T

E

S

- On MQCONNX an application can provide a user ID and password (in the Connection Security Parameters (MQCSP) structure in the MQCNO), which are passed to the queue manager at V8 to be checked (depending on whether the queue manager is configured to do this).
- The MQCSP structure was available since WebSphere MQ V6, but a security exit or custom OAM needed to be written to check the password. Now this is built into the queue manager in V8.

Configuration

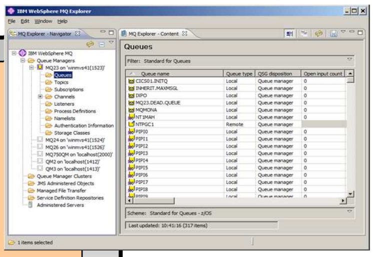


MQRC_NOT_AUTHORIZED (2035)

MQRC_NONE (0)

- CHCK...
- NONE
- OPTIONAL
- REQUIRED
- REQDADM

```
ALTER    QMGR CONNAUTH(USE.PW)
DEFINE   AUTHINFO(USE.PW) AUTHTYPE(xxxxxx)
         CHCKLOCL(OPTIONAL)
         CHCKCLNT(REQUIRED)
         FAILDLAY(1) ADOPTCTX(YES)
REFRESH SECURITY TYPE(CONNAUTH)
```



Configuration - Notes (1)

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.

Configuration - Notes (2)

N O T E S

- 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.

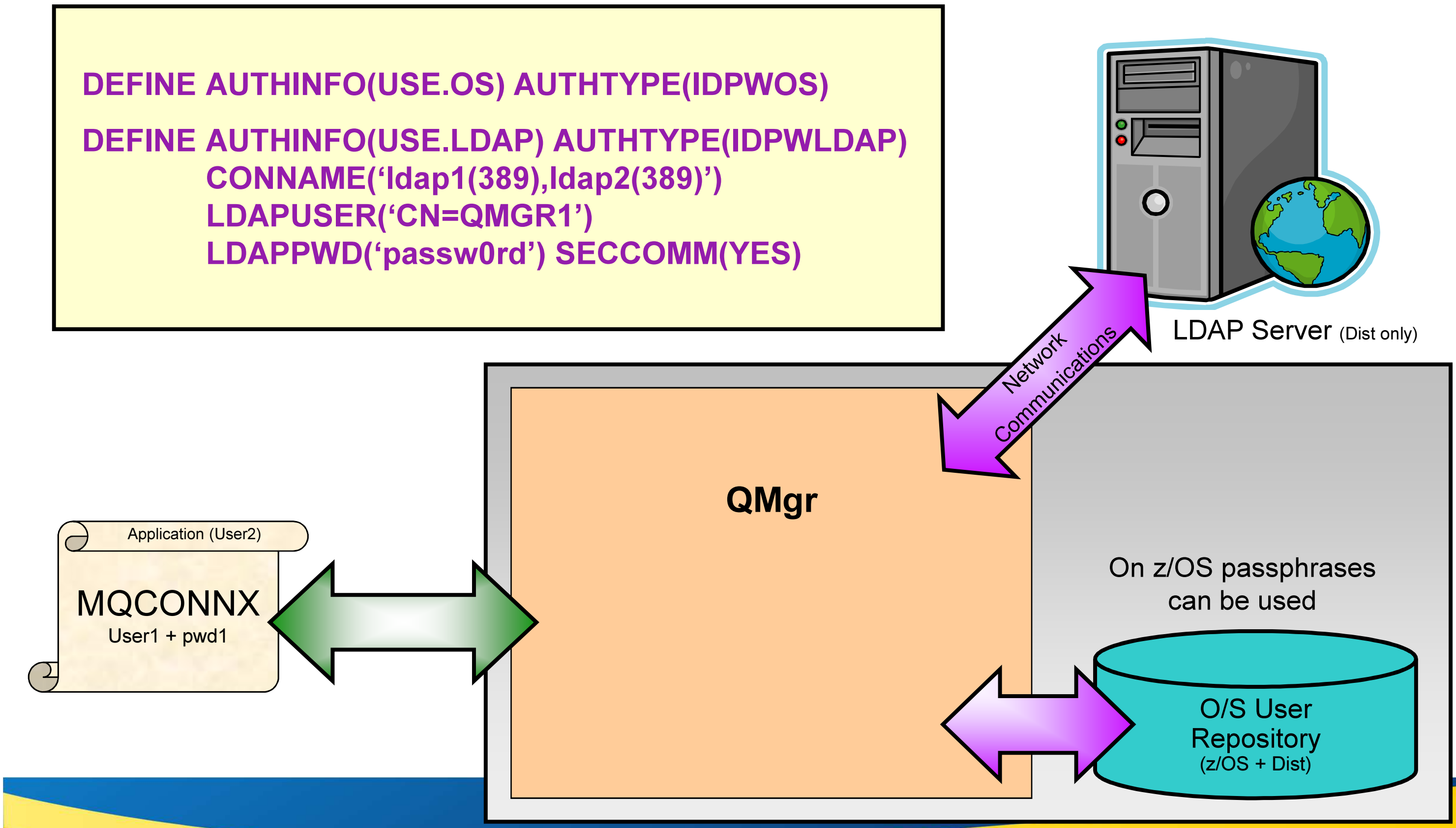
Configuration - Notes (3)

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.

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)
```



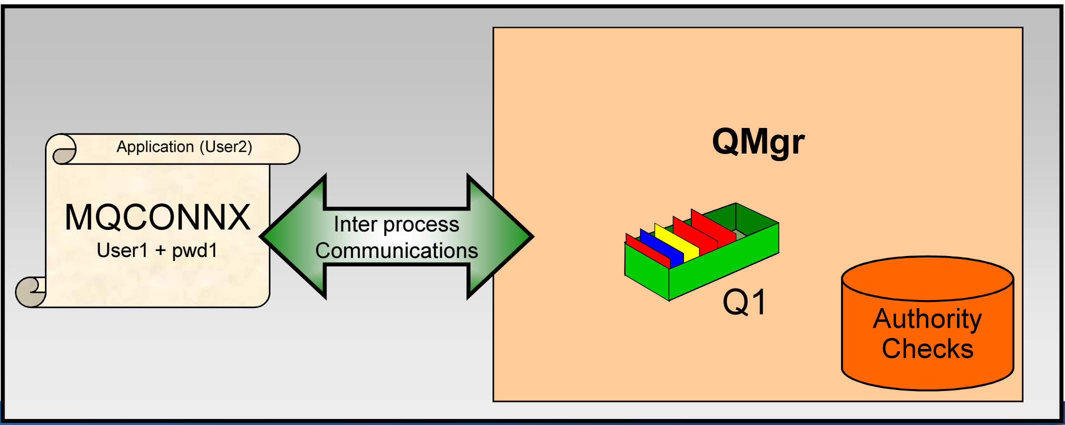
User Repositories – Notes

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. This includes the use of password phrases on z/OS
- 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.

Local Bindings Authorization

Method	Notes
User ID running the application process	Authenticated when the user ID logged onto the machine
User ID set by ADOPTCTX(YES)	The queue manager wide setting to adopt the password authenticated user ID as the user to be used will over-ride the above.



Local Bindings Authorization - Notes

N O T E S

- A locally bound application is one that is running on the same machine as the queue manager and using Inter-process communication to connect and not sockets and a channel. When an application is locally bound, the identity used for authorization checks for all the resources that application wants to use is assigned in one of the following ways:-
- The user ID that the process running the application is running under, will be used for authorization checks. This user ID has been authenticated by means of that user ID logging on to the O/S.
- Alternatively, if the application provided a user ID and password for authentication, and the queue manager is configured to check these, there is a setting called ADOPTCTX (Adopt Context) and if that is set to YES, the authentication user ID will be user for authorization checks. This user ID may well be different to the one the process running the application is running under.

Client Bindings Authorization

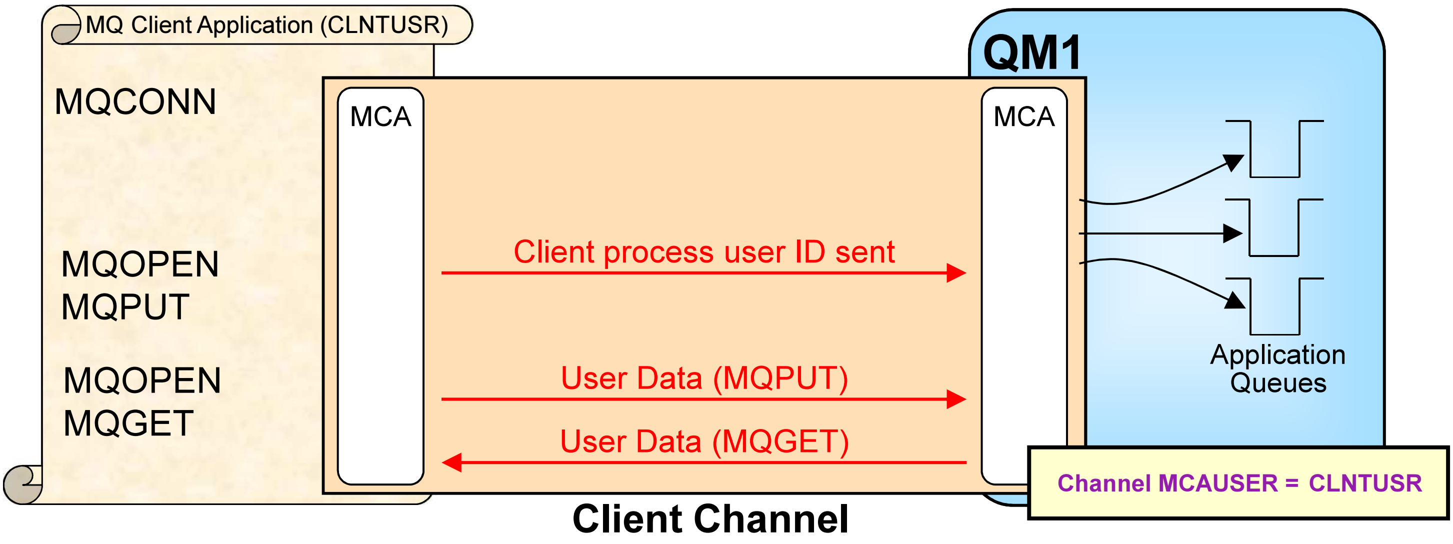
Method	Notes
Client machine user ID flowed to server	This will be over-ridden by anything else. Rarely do you want to trust an unauthenticated client side user ID.
MCAUSER set on SVRCONN channel definition	A handy trick to ensure that the client flowed ID is never used is to define the MCAUSER as 'rubbish'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

Client Bindings Authorization - Notes

NOTES

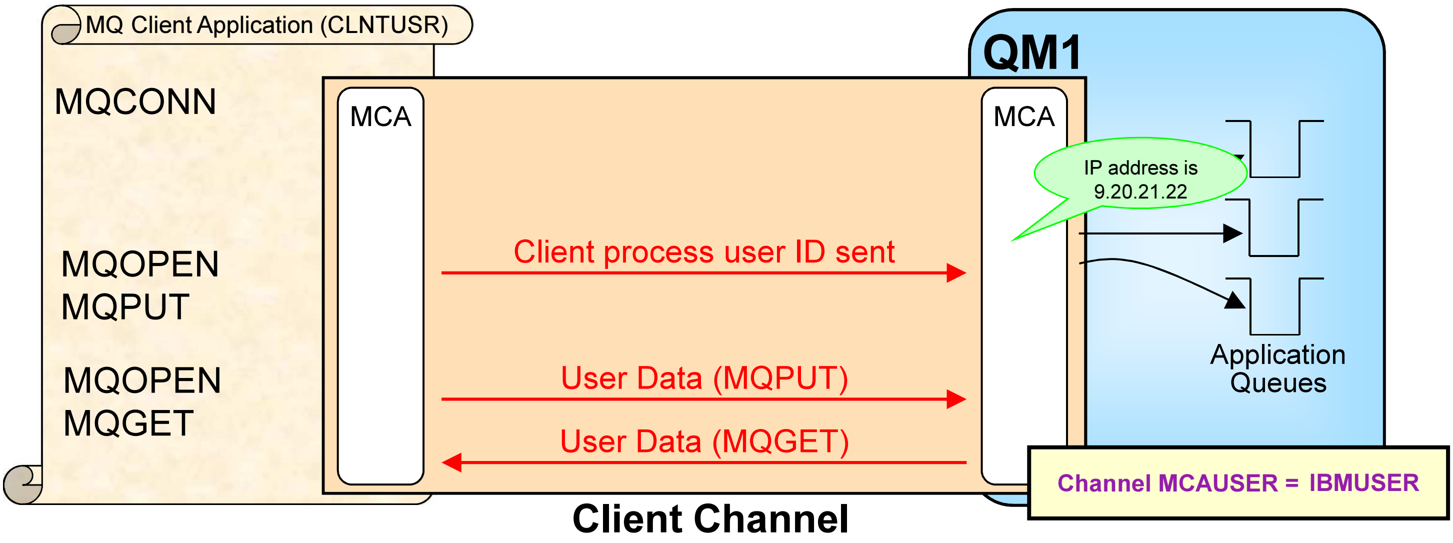
- A client bound application is one that is connecting to the queue manager using a channel and therefore a socket. When an application is running as a client, the identity used for authorization checks for all the resources that application wants to use is assigned in one of the ways shown in the table.
- There are numerous ways that the running user can be set for a SVRCONN channel, i.e. the user which is representing the client application when it is running on the queue manager machine. The ADOPTCTX(YES|NO) attribute that we just saw is yet another one. How do all these different ways of setting the MCAUSER on the SVRCONN interact.
- There is an order of events and certain ways of setting the MCAUSER over-ride others. The table shows the order.

Security Scale - None!!!



- **Client side user is used without any authentication at all**
  - So client machine can choose to assert anything it wants
- **Simple mitigation - add defined MCAUSER of an unknown ID to block this off**
  - …. and pick something even just slightly higher on the scale!

Security Scale - Low

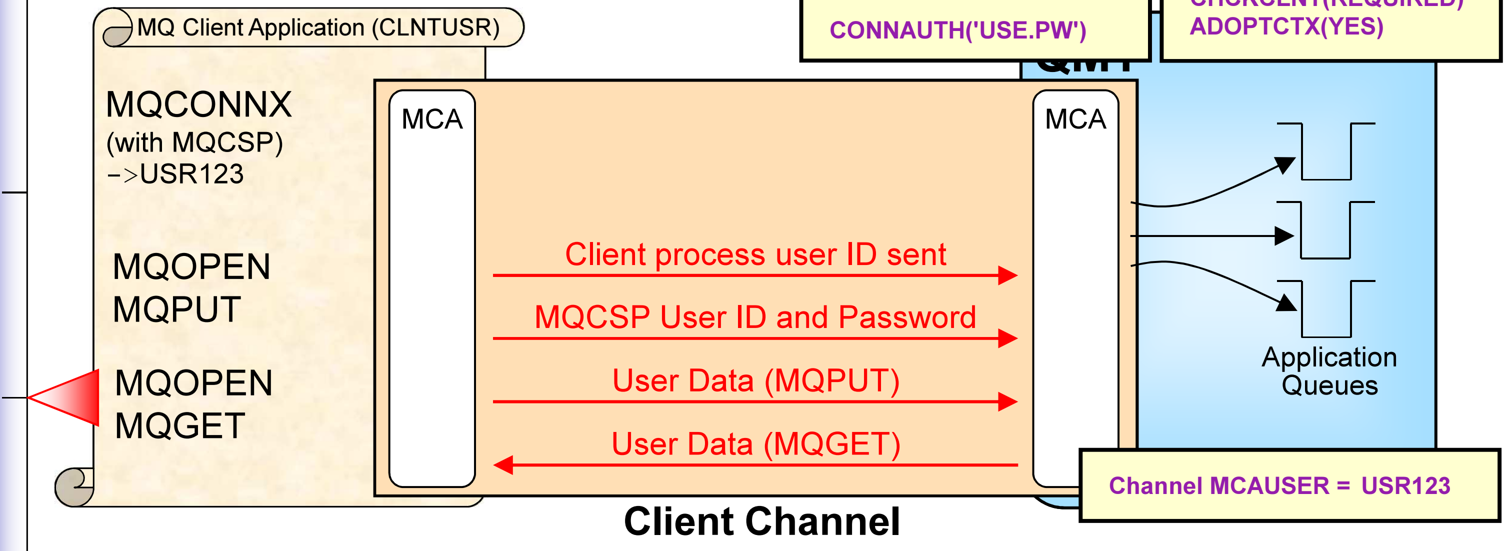


- **Simple IP address filtering**
  - Not really authentication
- **Better than nothing?**

CHLAUTH Rules

```
SET   CHLAUTH(*) TYPE(ADDRESSMAP)
      ADDRESS('9.20.21.22')
      MCAUSER('IBMUSER')
```

Security Scale - Medium



QMgr Settings

CONNAUTH('USE.PW')

'USE.PW' Settings

AUTHTYPE(IDPWOS)
CHCKCLNT(REQUIRED)
ADOPTCTX(YES)

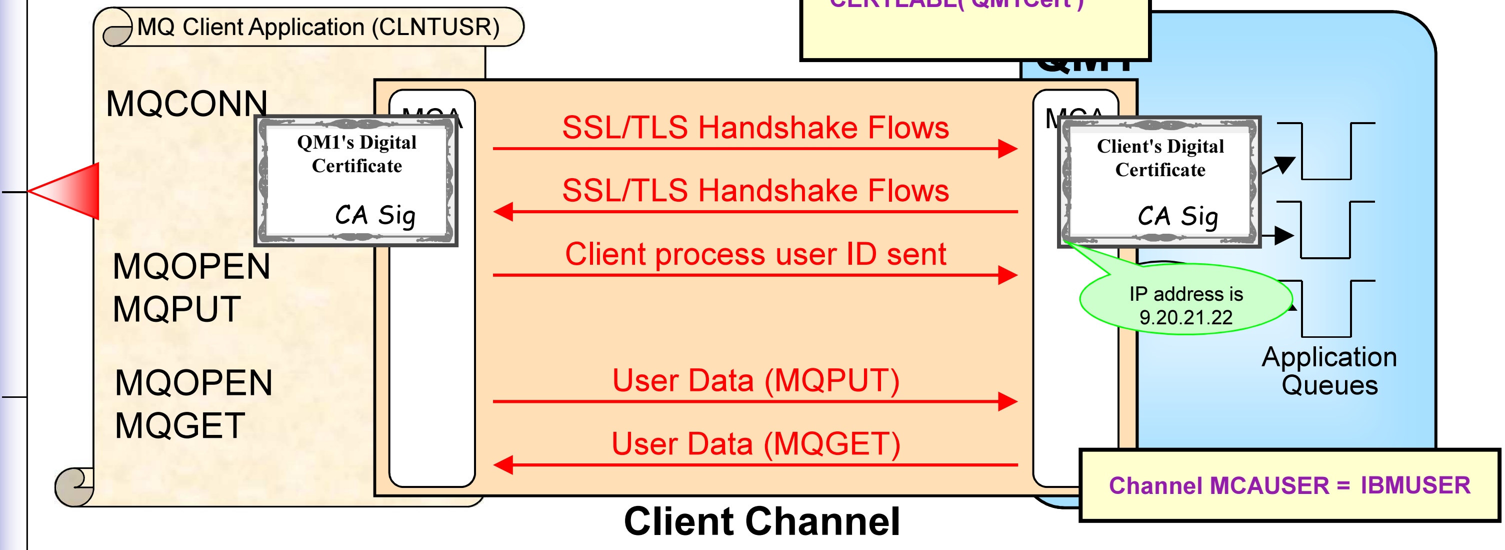
Channel MCAUSER = USR123

- **User ID & Password checking**
- **With IP address filtering**
  - For an extra level of proof

CHLAUTH Rules

```
SET   CHLAUTH(*) TYPE(ADDRESSMAP)
      ADDRESS('9.20.21.22')
      USERSRC(CHANNEL)
```

Security Scale - High



QMgr Settings

SSLKEYR(QM1KEYRING)
CERTLABL('QM1Cert')

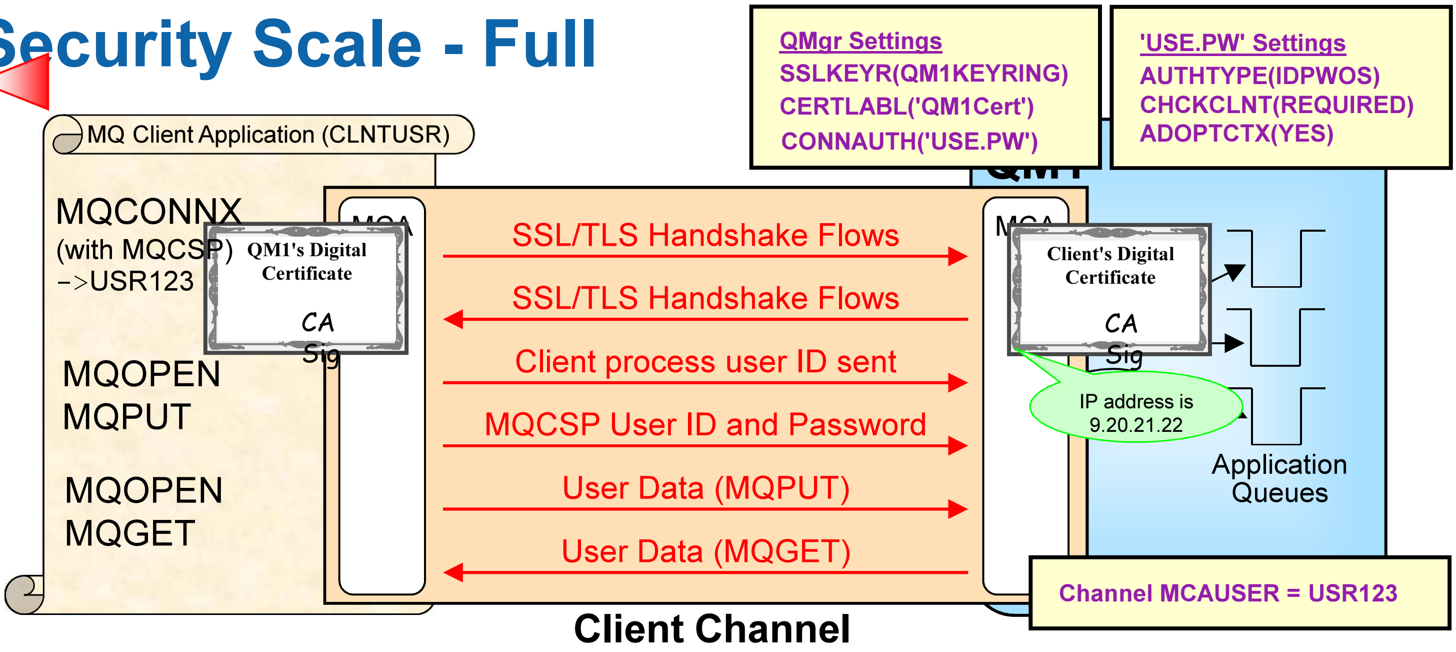
Channel MCAUSER = IBMUSER

- **Digital Certificate for Authentication**
  - Set MCAUSER based on DN
  - Fully qualified DN checking
  - Address restrictor for extra proof

CHLAUTH Rules

```
SET   CHLAUTH(*) TYPE(SSLPEERMAP)
      SSLPEER('O=IBM') SSLCERTI('CN=MQ CA')
      ADDRESS('9.20.21.22')
      MCAUSER('IBMUSER')
```

Security Scale - Full



QMgr Settings
SSLKEYR(QM1KEYRING)
CERTLABL('QM1Cert')
CONNAUTH('USE.PW')

'USE.PW' Settings
AUTHTYPE(IDPWOS)
CHCKCLNT(REQUIRED)
ADOPTCTX(YES)

Channel MCAUSER = USR123

- **Certificates and Password Validation**
- **Set MCAUSER based on**
  - Either DN as before
  - Or validated user ID

CHLAUTH Rules
```
SET   CHLAUTH(*) TYPE(SSLPEERMAP)
      SSLPEER('O=IBM') SSLCERTI('CN=MQ CA')
      ADDRESS('9.20.21.22')
      USERSRC(CHANNEL)
```

Putting it all together - Notes

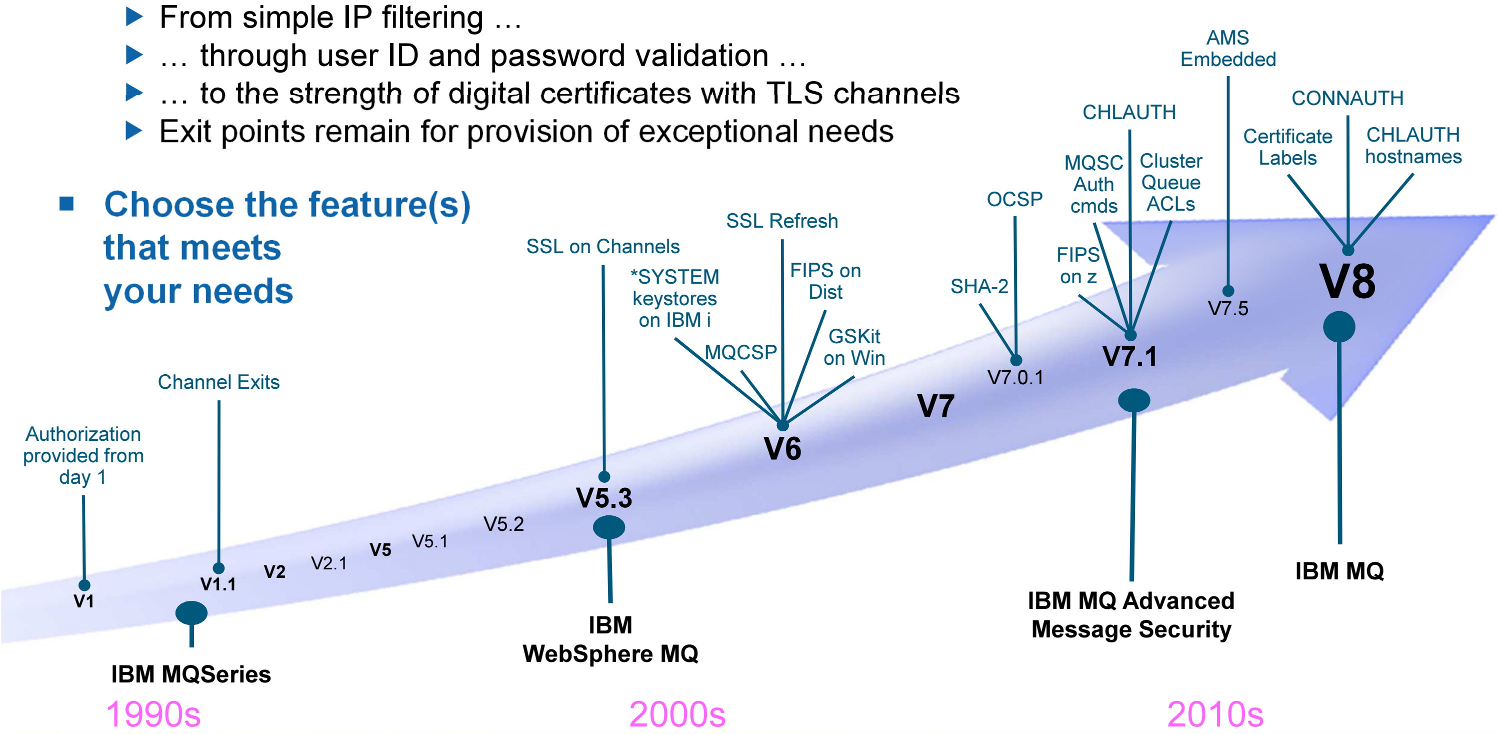
N O T E S

- The process of authenticating a locally bound application is fairly simple as we saw on earlier pages.
- However, the process of authenticating a client application has a number of possible features. Here we show how they can all be used together if you wish, or how you can dial down to only features that meet your business requirements.

MQ Security History and Summary

- **IBM MQ provides a scale of authentication features**
  - From simple IP filtering …
  - … through user ID and password validation …
  - … to the strength of digital certificates with TLS channels
  - Exit points remain for provision of exceptional needs

- **Choose the feature(s) that meets your needs**



Authorization provided from day 1
Channel Exits
V1
V1.1
V2
V2.1
V5
V5.1
V5.2
IBM MQSeries
SSL on Channels
V5.3
IBM WebSphere MQ
*SYSTEM keystores on IBM i
MQCSP
SSL Refresh
FIPS on Dist
GSKit on Win
V6
V7
OCSP
SHA-2
V7.0.1
CHLAUTH
MQSC Auth cmds
Cluster Queue ACLs
FIPS on z
V7.1
IBM MQ Advanced Message Security
AMS Embedded
V7.5
CONNAUTH
Certificate Labels
CHLAUTH hostnames
V8
IBM MQ
1990s
2000s
2010s