Jonathan Rumsey - jrumsey@uk.ibm.com

IBM UK

# What's new in IBM MQ V8

# 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 V8 Dates / End of Service

- Announce:        22 April 2014
- Availability:
  - 23 May 2014 (eGA Distributed)
  - 13 June 2014 (z/OS and pGA Distributed)
  - 24 September 2014 (8.0.0.1 fixpack)

- End of Service for old platforms and versions
  - MQ V7.0.x 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

## 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 / Windows Server 2008 R2 or later
  - Client only package for 32-bit platforms
  - Queue Manager now requires 64-bit
- AMS available on all V8 server platforms
  - First time AMS is available on IBM i
  - Closer queue manager integration on z/OS
- Client Attach Feature (CAF) no longer required on z/OS
  - Single price includes support for clients
- MFT Integration on IBM i
  - LICPGM and PTF maintenance

## 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

## .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

- Windows Communication Foundation (WCF) interface extended to non-SOAP, non-JMS messages
  - MA93 SupportPac specification for “wmq:\\” URI
  - Making it easier for apps using WCF to communicate with any other MQ application

## runmqsc enhancements

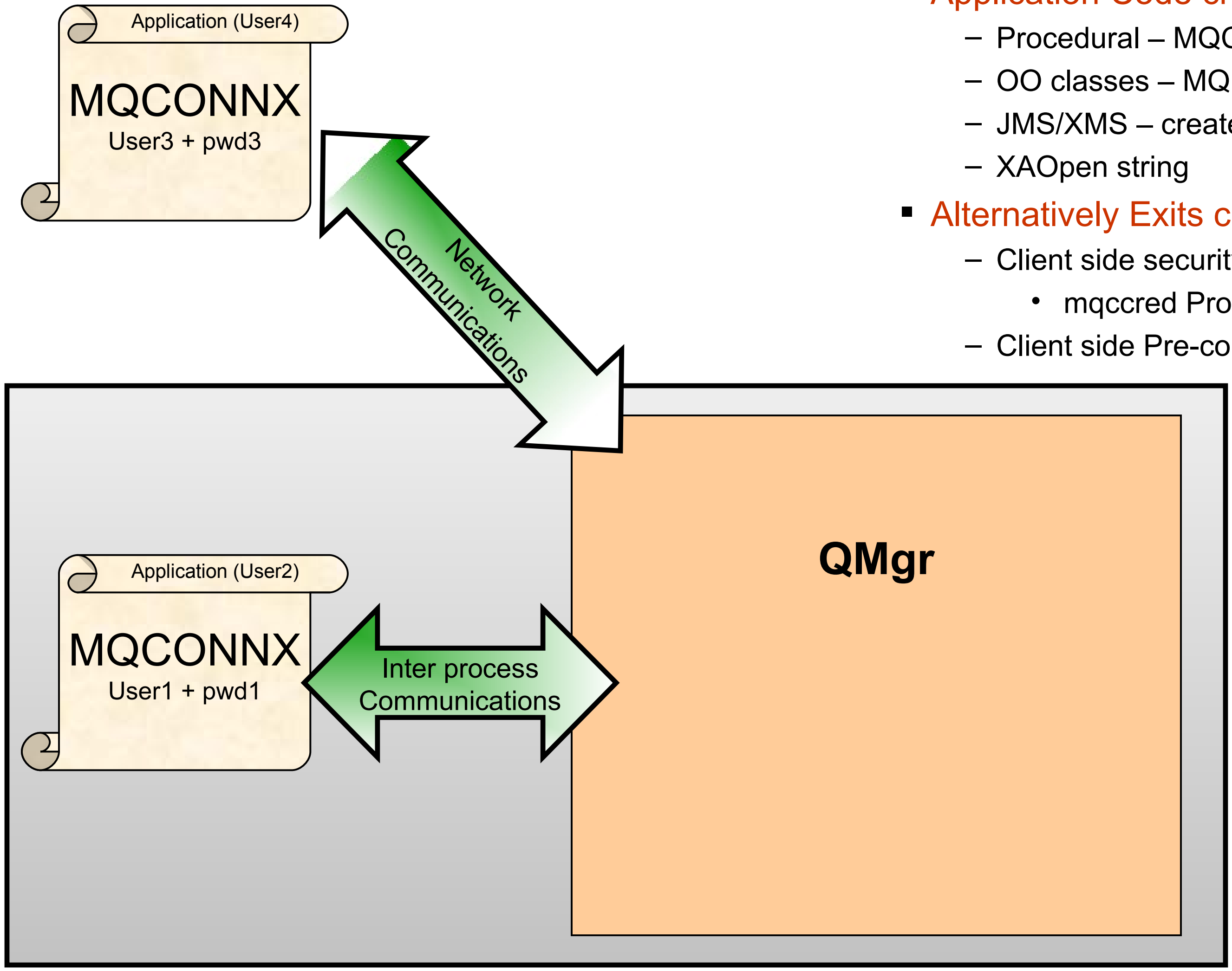
- Can now be run by any user (not just 'mqm')
  - Can take a userid/password for authentication: new "-u" flag
- Can now connect as a client to remote systems: new "-c" flag
  - Client channel definitions located by MQSERVER -> MQCHLLIB -> MQCHLTAB
- Can act as standalone program to create local CCDT: 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
  - Disconnects when queue manager quiescing
  - 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 jrumsey QM1
5724-H72 (C) Copyright IBM Corp. 1994, 2014.
Enter password:
******
Starting MQSC for queue manager QM1.

MQ Installation5> DIS QMGR
…
```

## SHA-2 Support

- Stronger algorithms are now available and recommended
  - In many cases also 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

# Connection Authentication – Application changes



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

## Connection Authentication - Procedural MQI

- 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       = "jrumsey";
csp.CSPUserIdLength    = 7;         /* Max: MQ_CLIENT_USER_ID_LENGTH */
csp.CSPPasswordPtr     = "passw0rd";
csp.CSPPasswordLength  = 8;         /* Max: MQ_CSP_PASSWORD_LENGTH   */
```

## Connection Authentication - OO MQ Classes

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

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

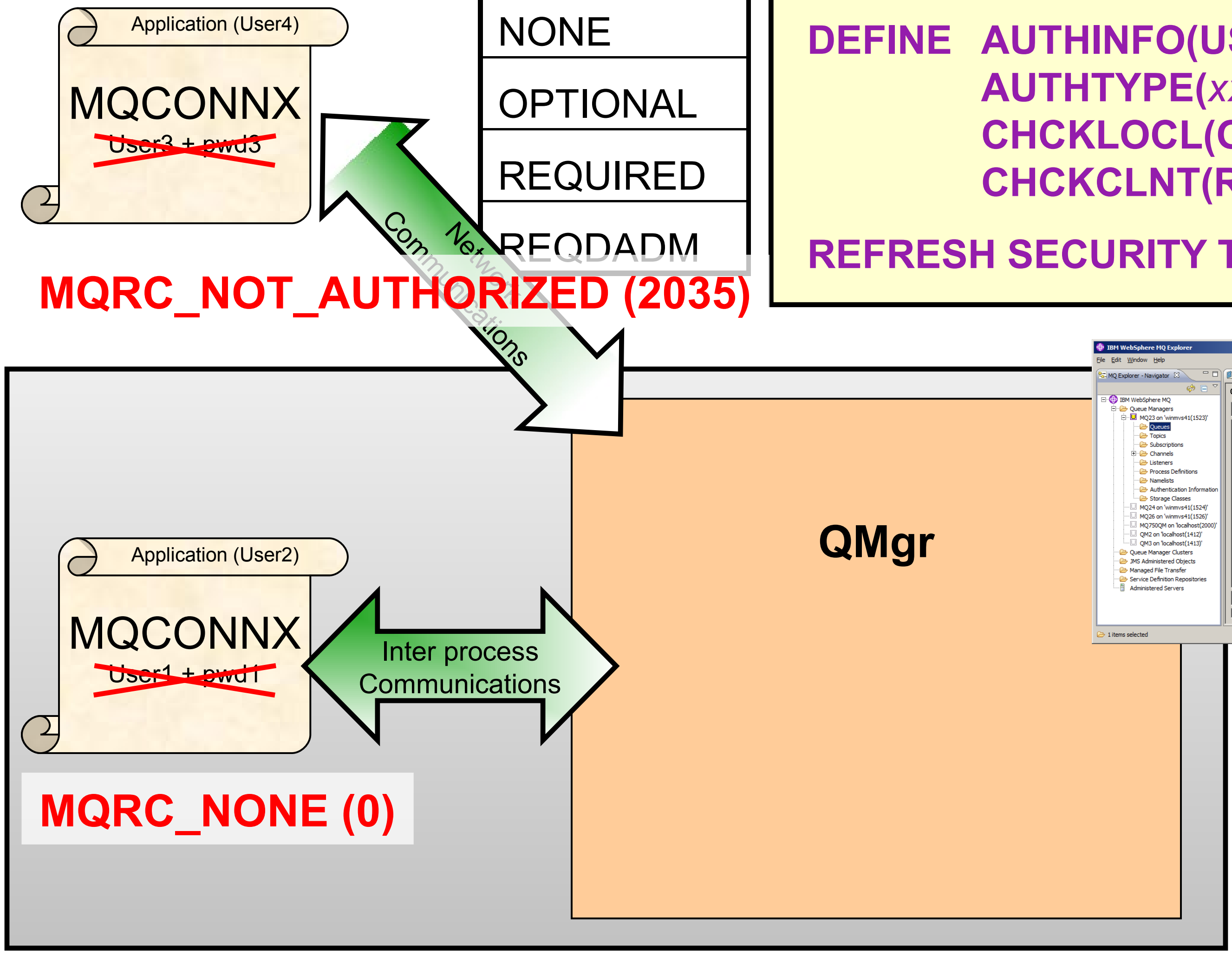
## JMS/XMS classes changes

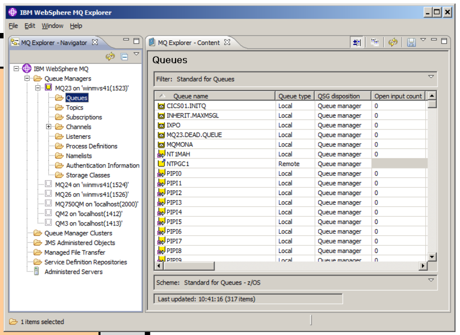
```
cf = getCF();

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

# 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)
```





Queues

Filter: Standard for Queues

| Queue name | Queue type | QSG disposition | Open input count |
|---|---|---|---|
| CICS01.INITQ | Local | Queue manager | 0 |
| INHERIT.MAXMSGL | Local | Queue manager | 0 |
| IXPO | Local | Queue manager | 0 |
| MQ23.DEAD.QUEUE | Local | Queue manager | 0 |
| MQMONA | Local | Queue manager | 0 |
| NT1MAH | Local | Queue manager | 0 |
| NTPGC1 | Remote | Queue manager | |
| PIPI0 | Local | Queue manager | 0 |
| PIPI1 | Local | Queue manager | 0 |
| PIPI2 | Local | Queue manager | 0 |
| PIPI3 | Local | Queue manager | 0 |
| PIPI4 | Local | Queue manager | 0 |
| PIPI5 | Local | Queue manager | 0 |
| PIPI6 | Local | Queue manager | 0 |
| PIPI7 | Local | Queue manager | 0 |
| PIPI8 | Local | Queue manager | 0 |
| PIPI9 | Local | Queue manager | 0 |

Scheme: Standard for Queues - z/OS

Last updated: 10:41:16 (317 items)

1 items selected

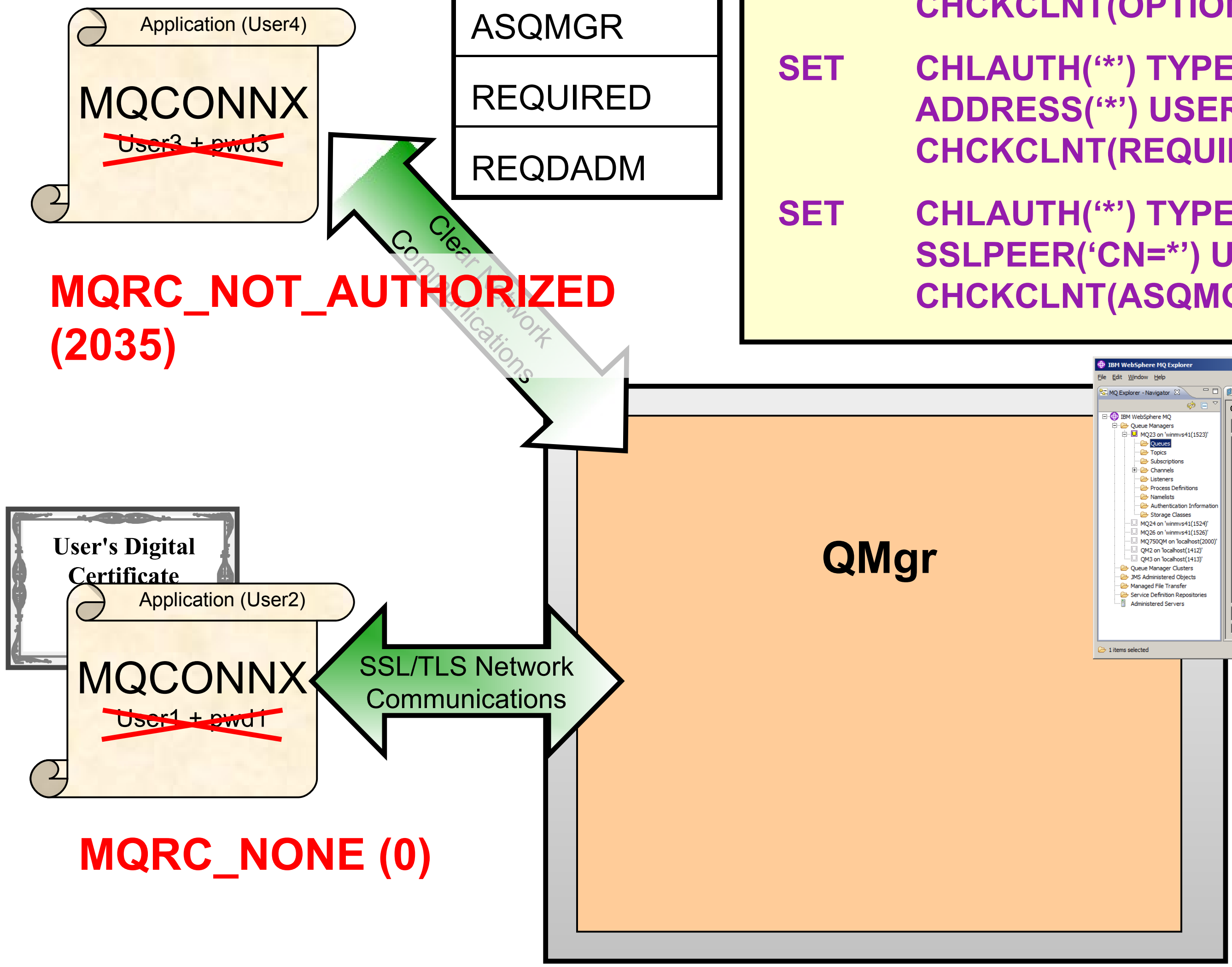
## Connection Authentication – Channel Authentication CHCKCLNT Upgrade

| CHCKCLNT |
|---|
| ASQMGR |
| REQUIRED |
| REQDADM |

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

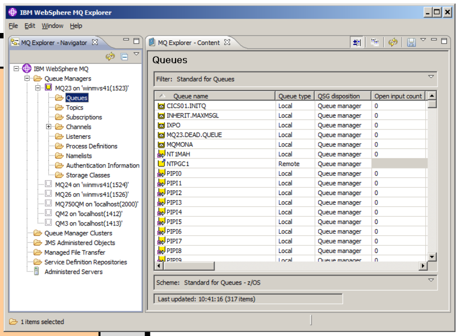
**SET CHLAUTH('*') TYPE(ADDRESSMAP)
ADDRESS('*') USERSRC(CHANNEL)
CHCKCLNT(REQUIRED)**

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



**MQRC_NOT_AUTHORIZED (2035)**

**MQRC_NONE (0)**



Queues

Filter: Standard for Queues

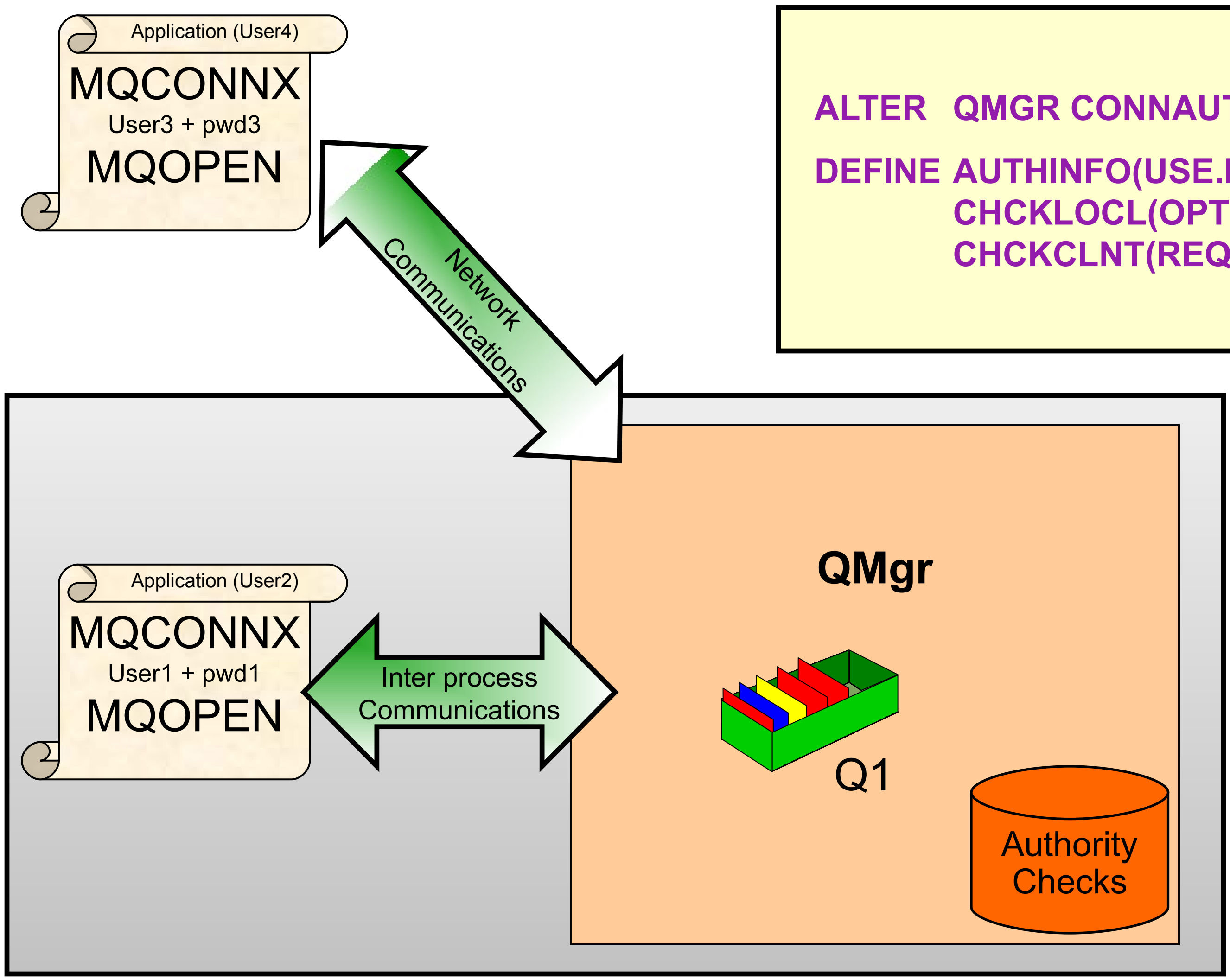
| Queue name | Queue type | QSG disposition | Open input count |
|---|---|---|---|
| CICS01.INITQ | Local | Queue manager | 0 |
| INHERIT.MAXMSGL | Local | Queue manager | 0 |
| IXPO | Local | Queue manager | 0 |
| MQ23.DEAD.QUEUE | Local | Queue manager | 0 |
| MQMONA | Local | Queue manager | 0 |
| NT1MAH | Local | Queue manager | 0 |
| NTPGC1 | Remote | Queue manager | |
| PIPI0 | Local | Queue manager | 0 |
| PIPI1 | Local | Queue manager | 0 |
| PIPI2 | Local | Queue manager | 0 |
| PIPI3 | Local | Queue manager | 0 |
| PIPI4 | Local | Queue manager | 0 |
| PIPI5 | Local | Queue manager | 0 |
| PIPI6 | Local | Queue manager | 0 |
| PIPI7 | Local | Queue manager | 0 |
| PIPI8 | Local | Queue manager | 0 |
| PIPI9 | Local | Queue manager | 0 |

Scheme: Standard for Queues - z/OS

Last updated: 10:41:16 (317 items)

1 items selected

## Connection Authentication – Relationship to Authorization

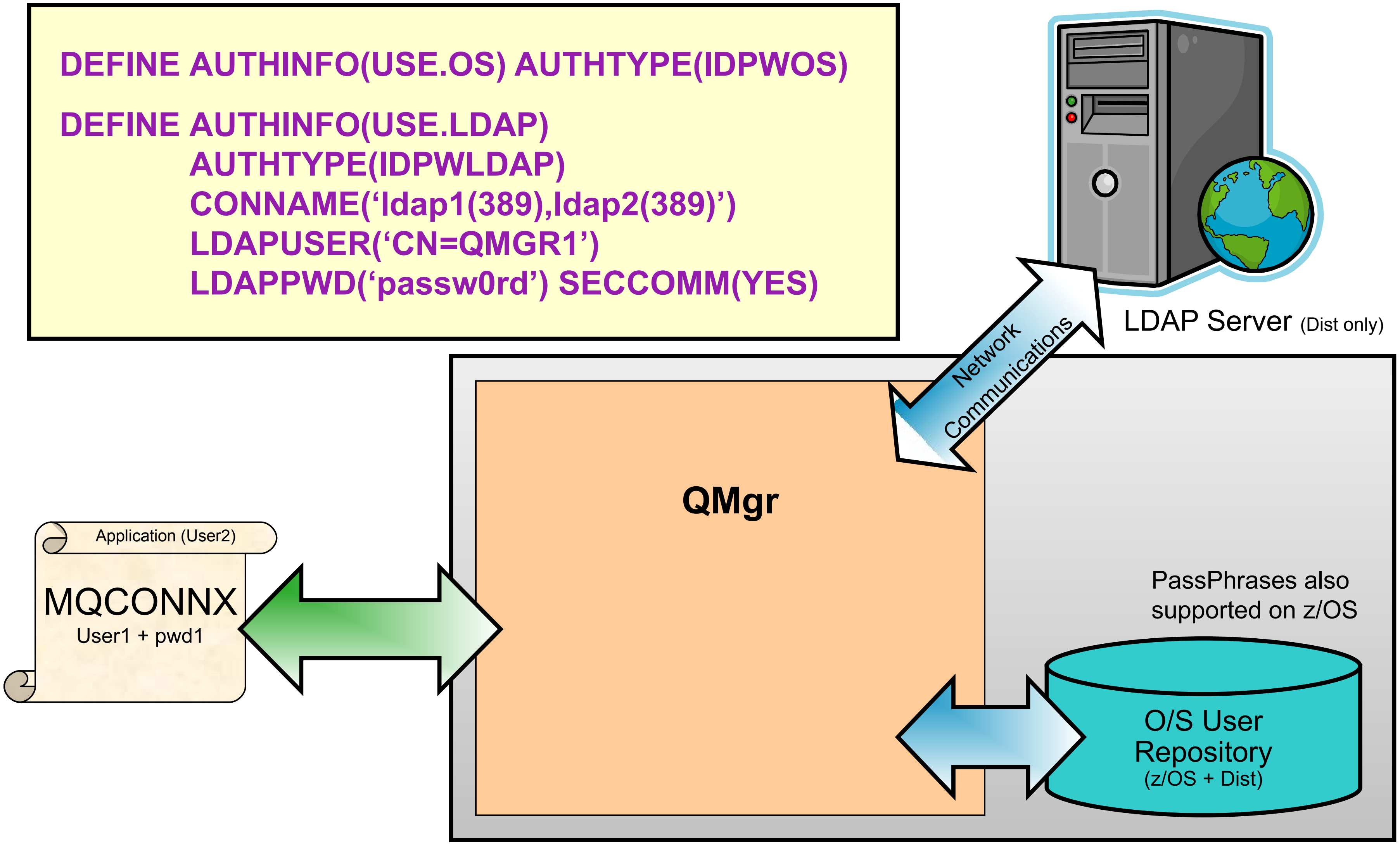


```
ALTER   QMGR CONNAUTH(USE.PWD)
DEFINE  AUTHINFO(USE.PWD) AUTHTYPE(xxxxxx)
        CHCKLOCL(OPTIONAL)
        CHCKCLNT(REQUIRED) ADOPTCTX(YES)
```

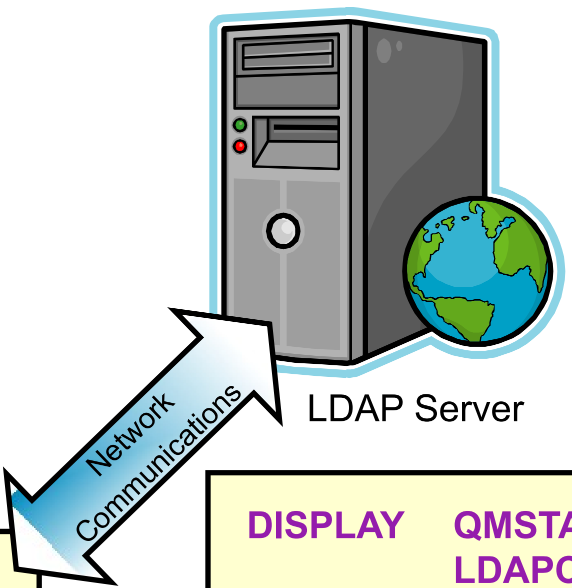
| Authority Records |
|---|
| Q1: User1 +put |
| Q1: User2 +none |
| Q1: User3 +get |
| Q1: User4 +none |

# Connection Authentication – User Repository Choices

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

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



# Secure connection to an LDAP Server

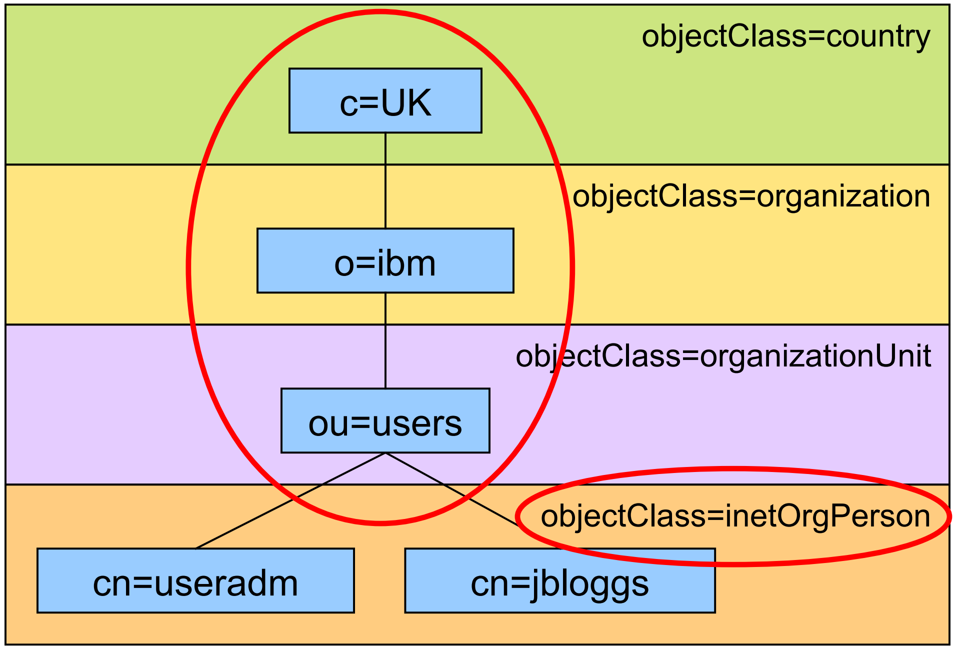


```
ALTER   QMGR CONNAUTH(USE.LDAP)
        SSLFIPS(NO) SUITEB(NONE)
        CERTLABL('ibmwebspheremqqm1')
        SSLKEYR('var/mqm/qmgrs/QM1/ssl/key')

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

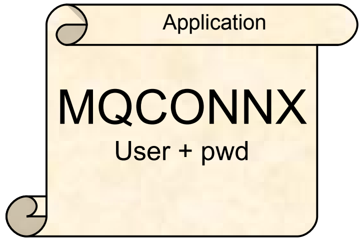
```
DISPLAY    QMSTATUS
           LDAPCONN
```

```
qm.ini
SSL:
   OCSPCheckExtension=YES
```

# LDAP User Repository



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



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

## 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

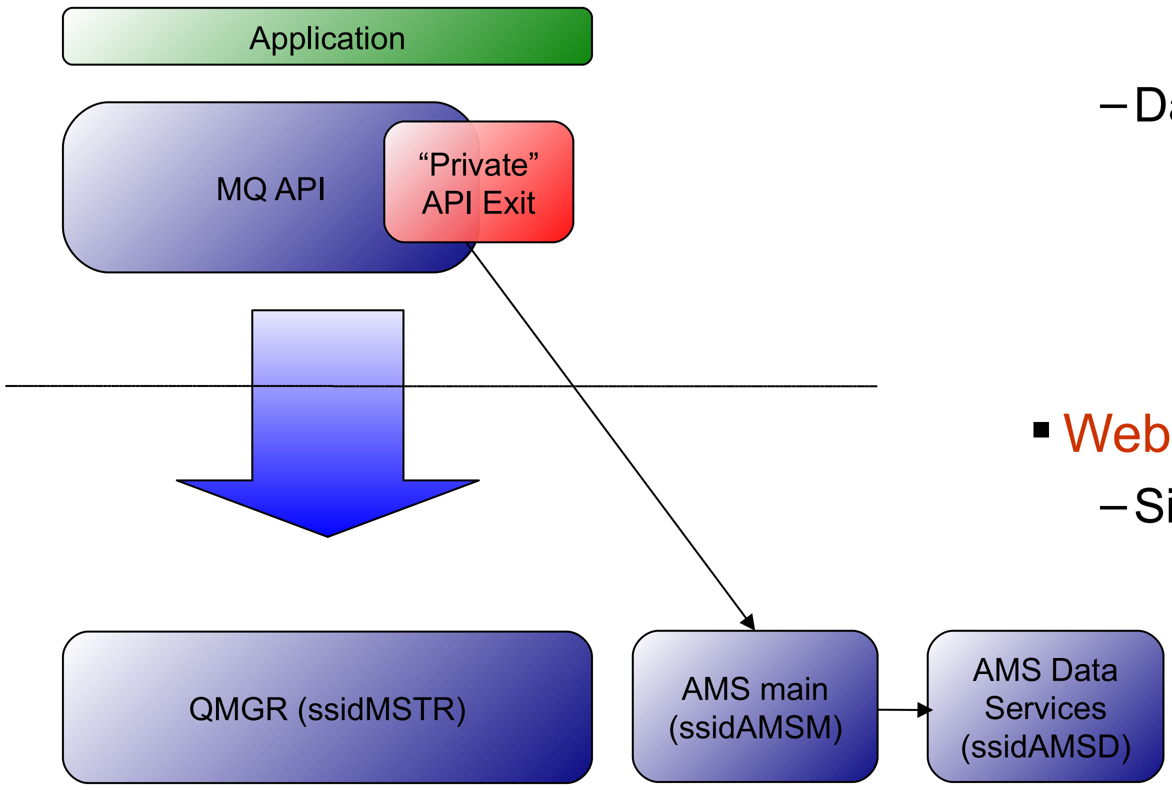
- Make Unix OAM userid-based
  - Optional configuration
  - Consistent with other platforms
  - Will no longer add primary group to authorities during setmqaut
  - Chosen at queue manager creation or by editing qm.ini
- Default is still group-based authorisations

```
$ 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

# 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

## 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
- 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 Rules - 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="Jon Rumsey"')
ADDRESS('9.20.*') MCAUSER(JRUMSEY)
```

# Channel Authentication Rules - 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

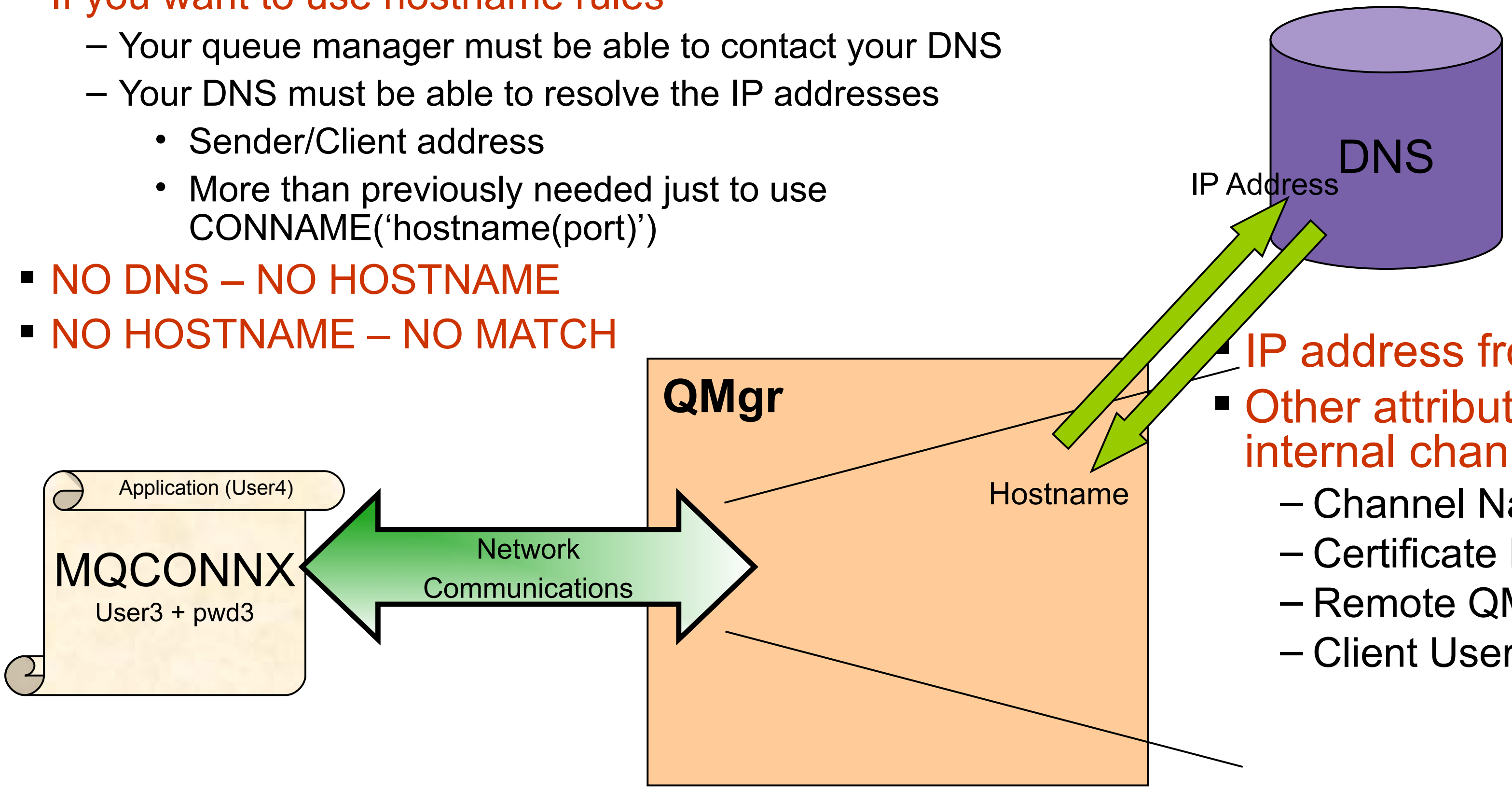
```
SET CHLAUTH(‘*.SVRCONN’)
TYPE(ADDRESSMAP)
ADDRESS(‘mach123.ibm.com’) MCAUSER(HUSER)
```

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

```
SET CHLAUTH(‘*’) TYPE(SSLPEERMAP)
SSLPEER(‘CN=“Jon Rumsey”’)
ADDRESS(‘s*.ibm.*’) MCAUSER(JRUMSEY)
```

# Obtaining a hostname

- Hostname is not 'sent' from the other end of the channel
- IP address is obtained from TCP/IP socket
- MQ 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

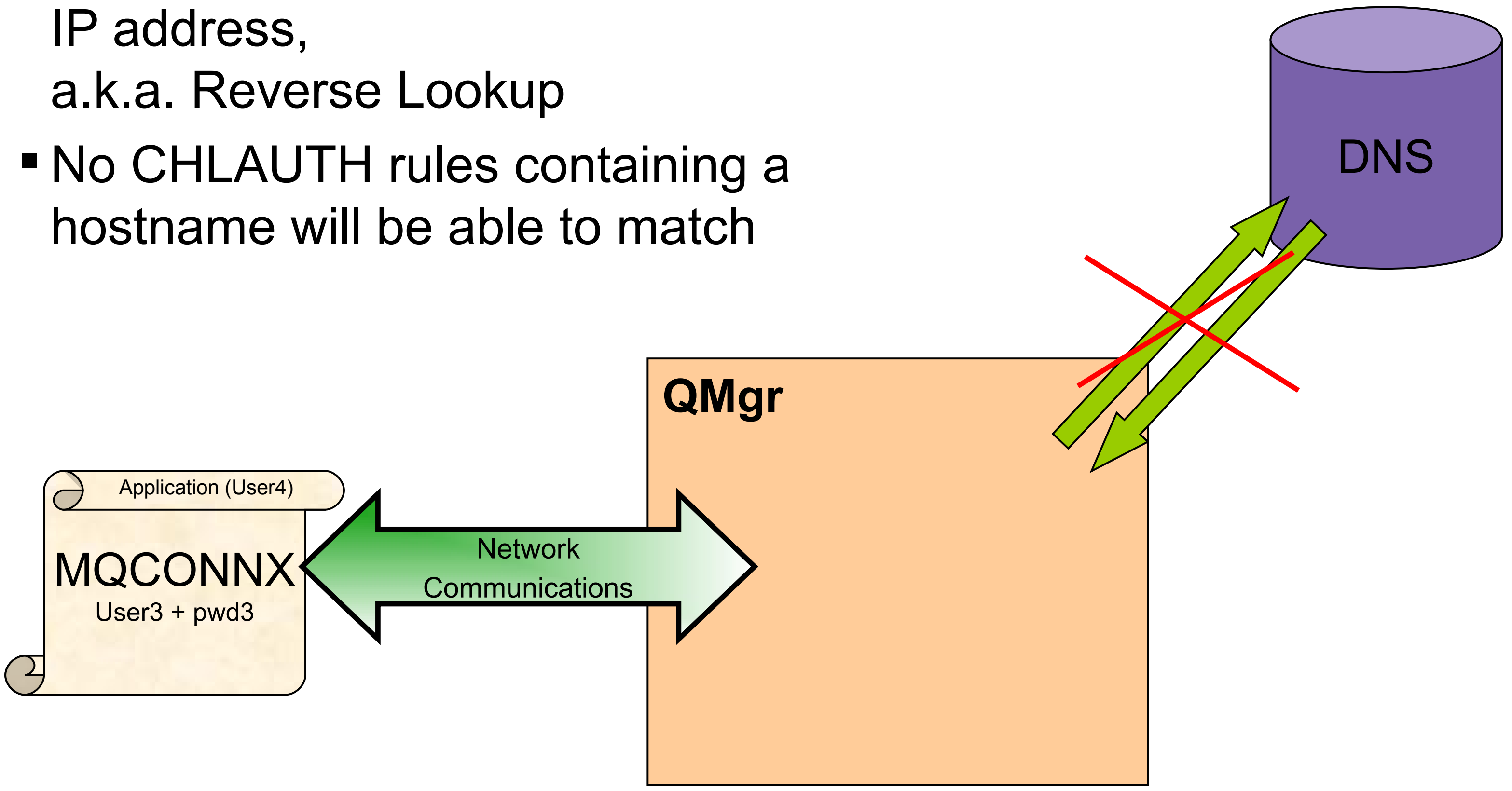


- IP address from TCP/IP
- Other attributes from internal channel flows
  - Channel Name
  - Certificate DN
  - Remote QMgr Name
  - Client User ID

## 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)
```

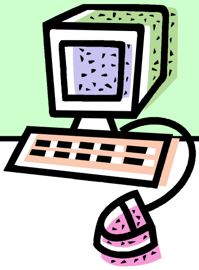


# Using CHLAUTH MATCH(RUNCHECK) with hostnames

```
DISPLAY   CHLAUTH(SYSTEM.ADMIN.SVRCONN) MATCH(RUNCHECK)
          SSLPEER('CN="Jon Rumsey", O="IBM"')
          CLNTUSER('jrumsey') ADDRESS('9.180.165.163')
returns ===>
               CHLAUTH(SYSTEM.ADMIN.SVRCONN)
               TYPE(ADDRESSMAP)
               ADDRESS('*.ibm.com') MCAUSER(JRUMSEY)
```

- 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.

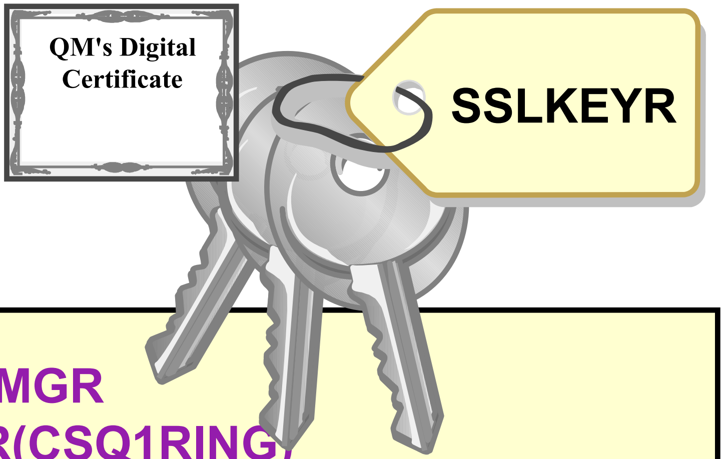
**Chl: SYSTEM.ADMIN.SVRCONN**
**DN: CN=Jon Rumsey.O=IBM**
**UID: jrumsey**
**IP: 9.180.165.163**



# Certificate labelling

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

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

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



**ALTER QMGR
SSLKEYR(CSQ1RING)
CERTLABL(‘CSQ1Certificate’)
CERTQSGL(‘SharedCert’)**

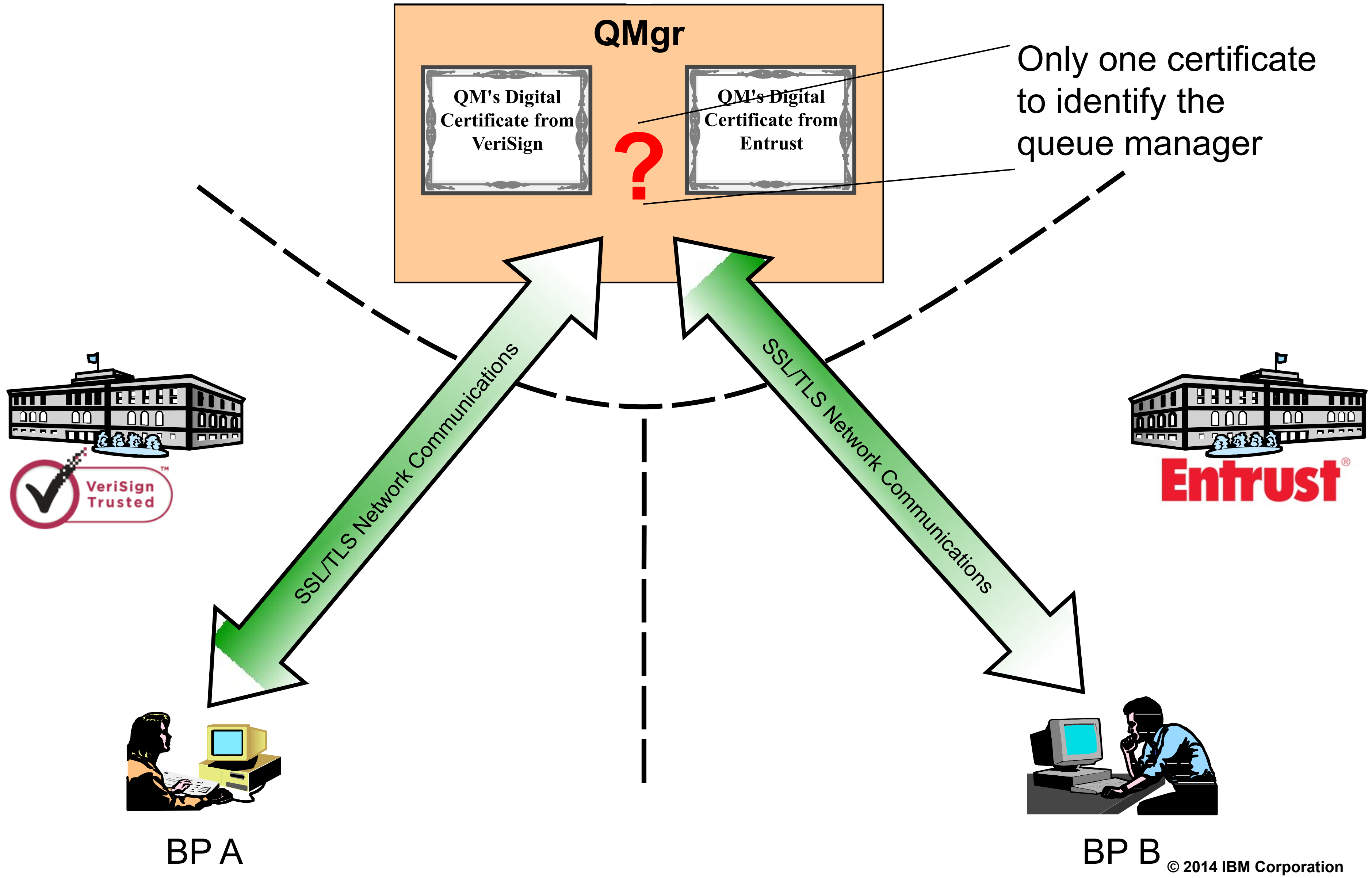
**ALTER QMGR
SSLKEYR('var/mqm/qmgrs/QM1/ssl/key') CERTLABL(‘QM1Certificate’)**

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

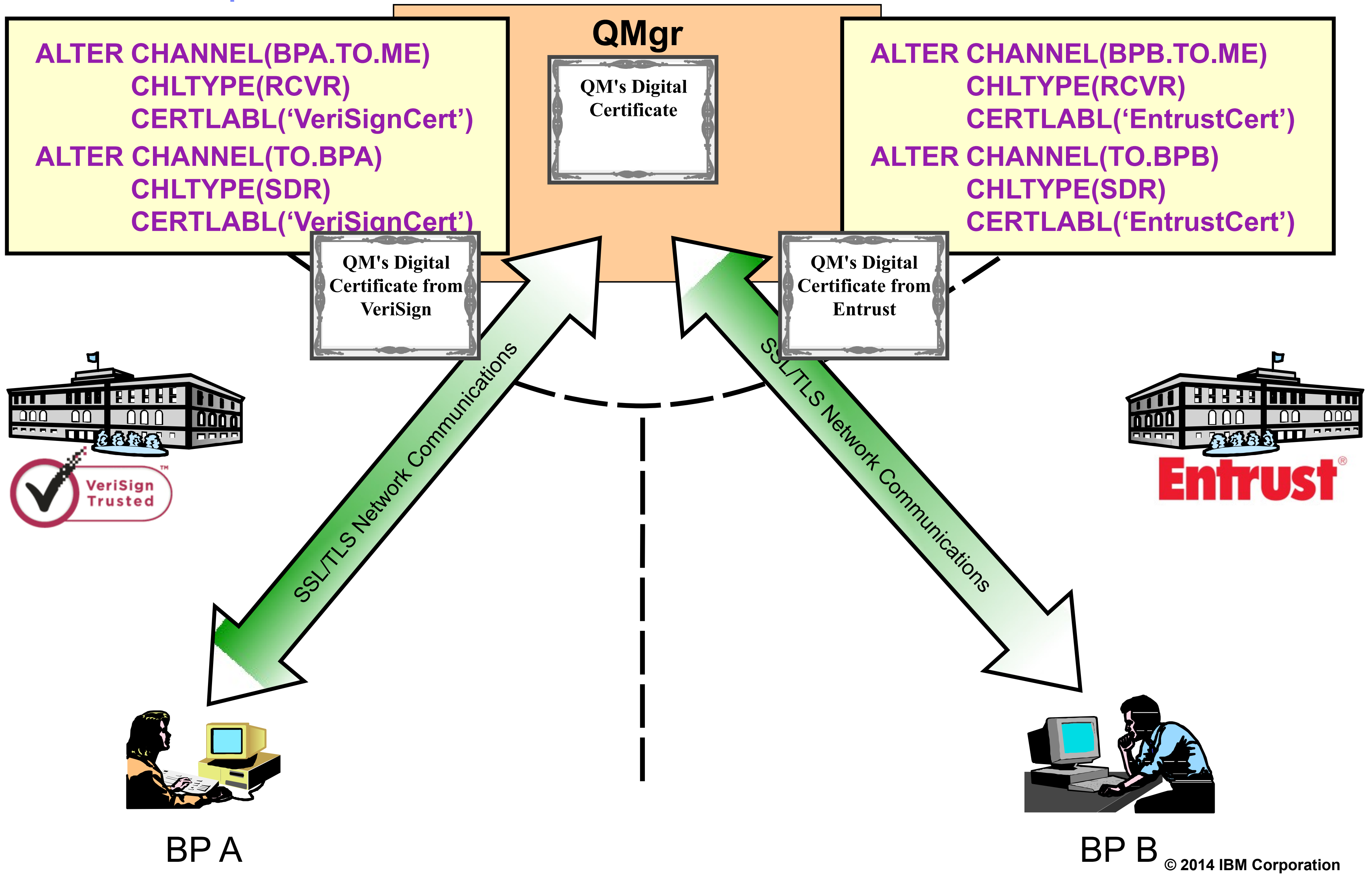
## Certificate labelling - Helpful for migration of certificates

- Migrating over to a new certificate when main certificate is ready to expire
  - Used to have to issue GSKit/RACF commands to rename certificate
  - Certificate labels need to be carefully co-ordinated
    - 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)

## Business Partners with different CA requirements


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

## Certificate per Channel

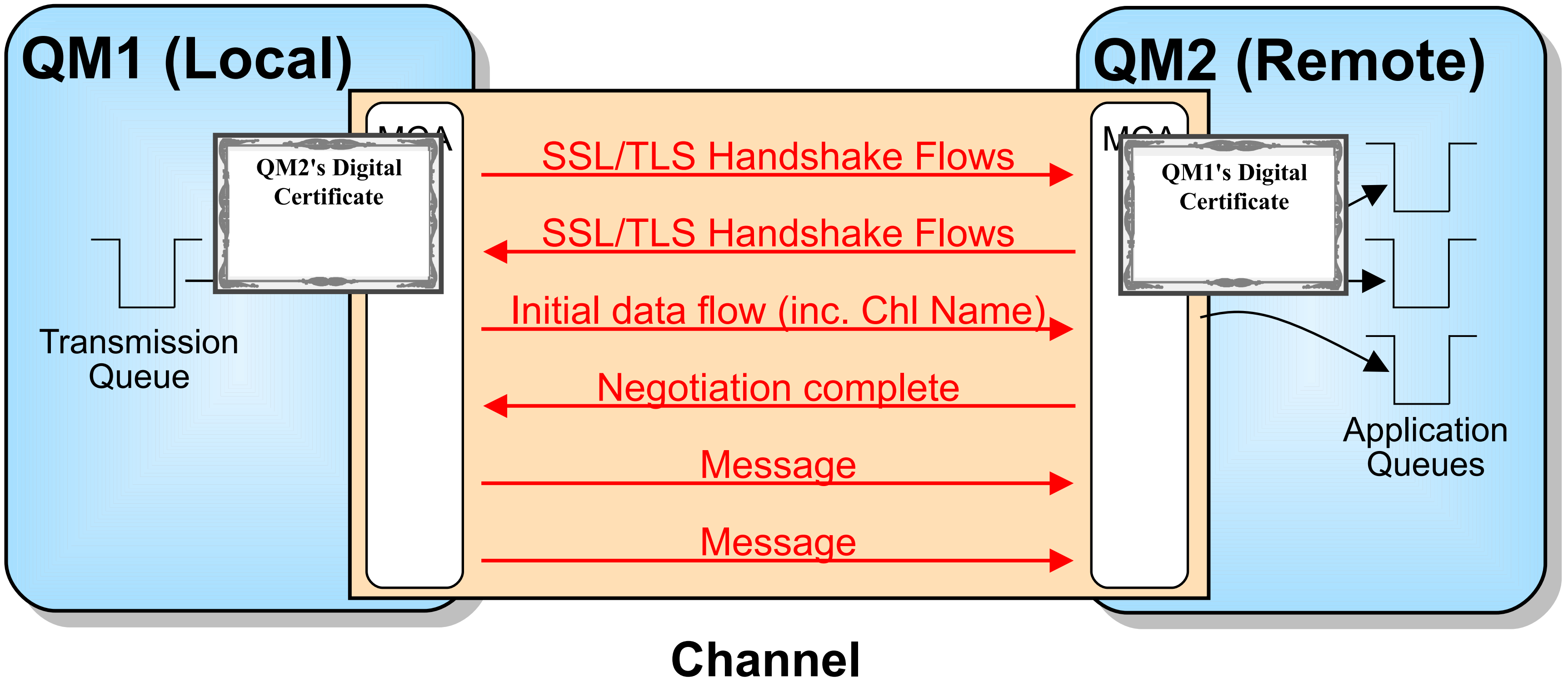


```
ALTER CHANNEL(BPA.TO.ME)
      CHLTYPE(RCVR)
      CERTLABL('VeriSignCert')
ALTER CHANNEL(TO.BPA)
      CHLTYPE(SDR)
      CERTLABL('VeriSignCert')
```

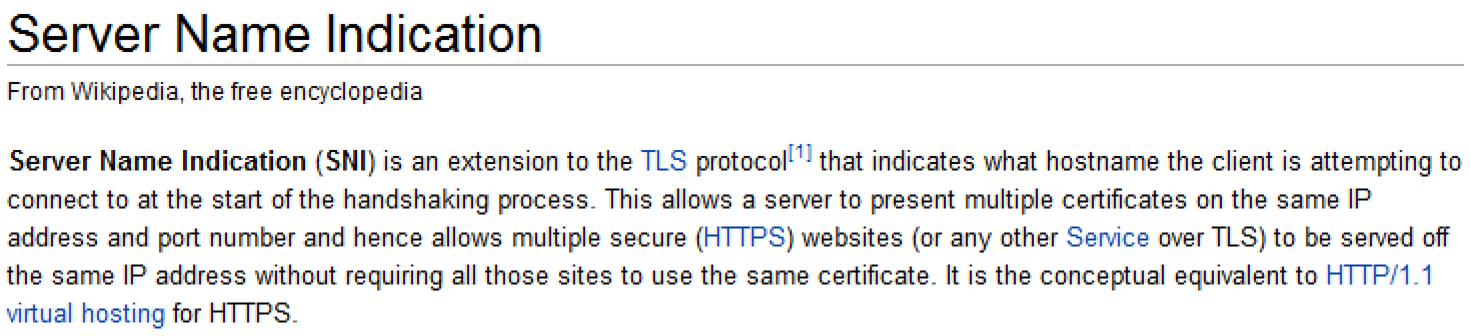
```
ALTER CHANNEL(BPB.TO.ME)
      CHLTYPE(RCVR)
      CERTLABL('EntrustCert')
ALTER CHANNEL(TO.BPB)
      CHLTYPE(SDR)
      CERTLABL('EntrustCert')
```

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

## Why did MQ not always support certificate per channel ?



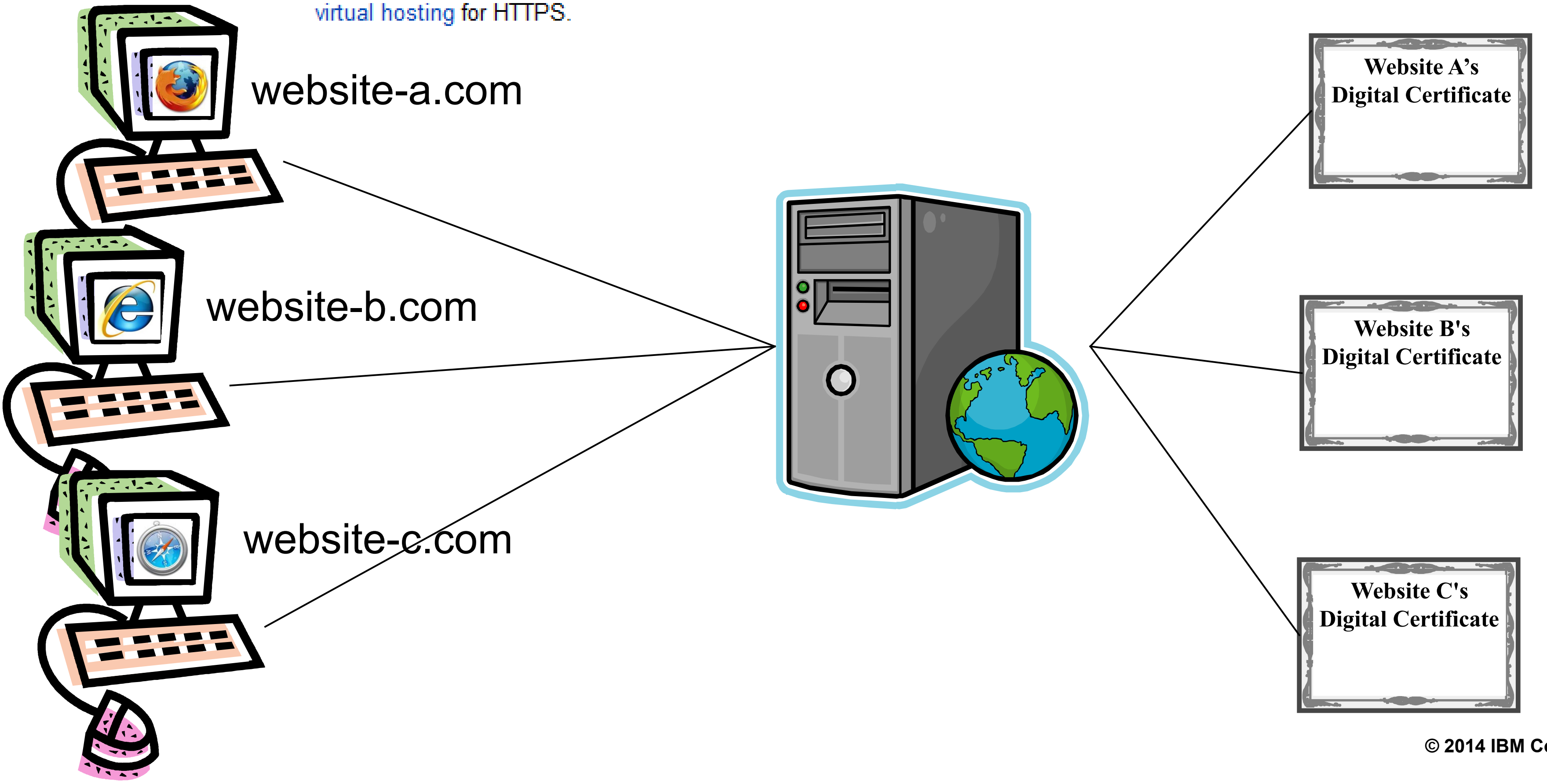
## Server Name Indication



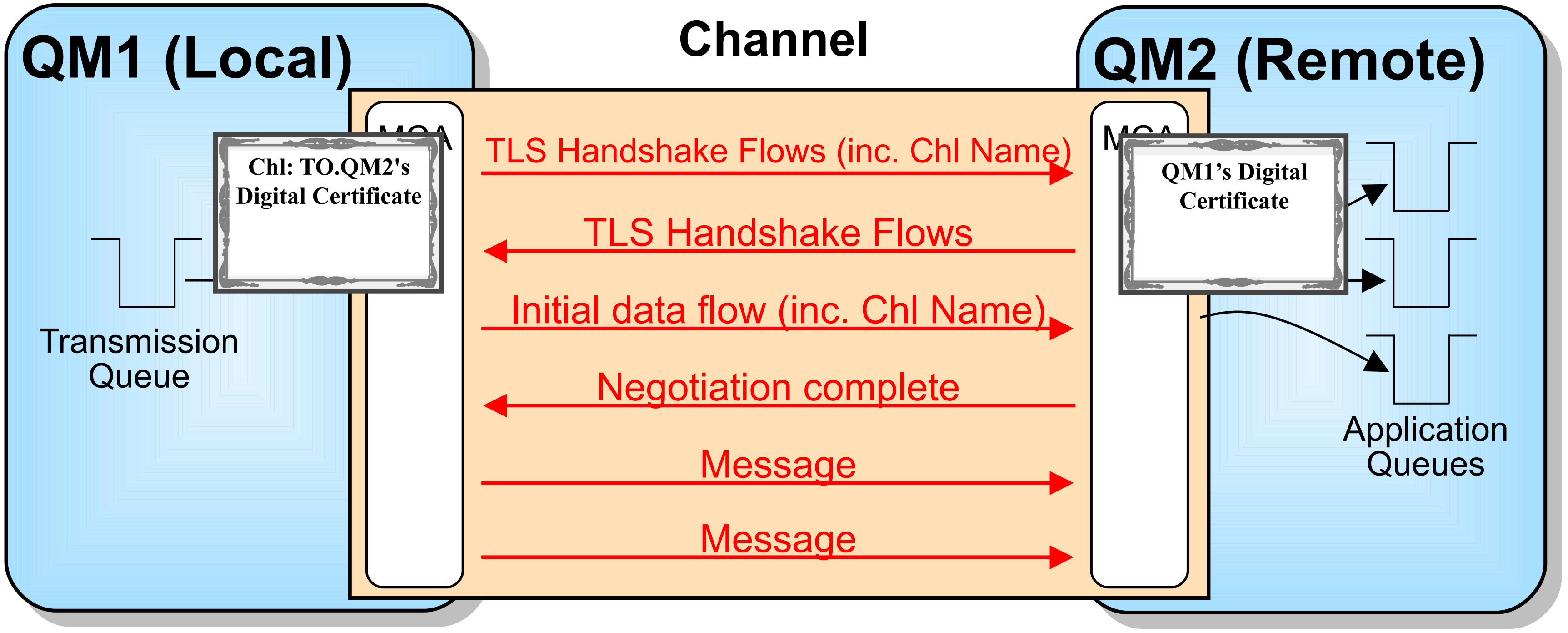
Server Name Indication

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
Website B's Digital Certificate
Website C's Digital Certificate

# Using SNI with a channel name



QM1 (Local)
Channel
QM2 (Remote)
MCA
MCA
Chl: TO.QM2's Digital Certificate
QM1's Digital Certificate
TLS Handshake Flows (inc. Chl Name)
TLS Handshake Flows
Initial data flow (inc. Chl Name)
Negotiation complete
Message
Message
Transmission Queue
Application Queues

- Both ends of the channel must be at the new release
- Only TLS can be used, no SSL
  - Only certain TLS cipherspecs will be able to enable 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!

## Changes for Channels using SSL/TLS Certificates

- Single Queue Manager Certificate
  - `ALTER QMGR CERTLABL('My certificate name')`
- Per Channel Certificate
  - `ALTER CHANNEL … CERTLABL('This channel certificate')`
- Certificate Matching, certificate issuer DN

  -
    ```
    SET CHLAUTH('*')
        TYPE(SSLPEERMAP)
        SSLPEER('CN=Jon Rumsey')
        SSLCERTI('CN=IBM CA')
        MCAUSER('jrumsey')
    ```

## 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)`

## TCP Buffer Autotuning – Distributed Queueing Throughput

- MQ traditionally sets the TCP buffers for channels to communicate efficiently
  - Defaults to 32Kb, but can be manually tuned in qm.ini under TCP stanza

- Modern Operating Systems typically do a better job of determining what TCP buffer sizes to use
  - New queue managers are created with qm.ini TCP buffer sizes all set to 0
  - Buffer size 0 intructs MQ to leave TCP buffer tuning to OS
  - Typically results in significant gains in throughput for DQ channels
  - Can revert to pre-V8 behaviour by removing stanza values from qm.ini

## 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 optimizations
    - Faster data conversion, especially for UTF-8
    - Object catalogue restructured
  - Better exploitation of SMP machines

  - Channel status table restructure

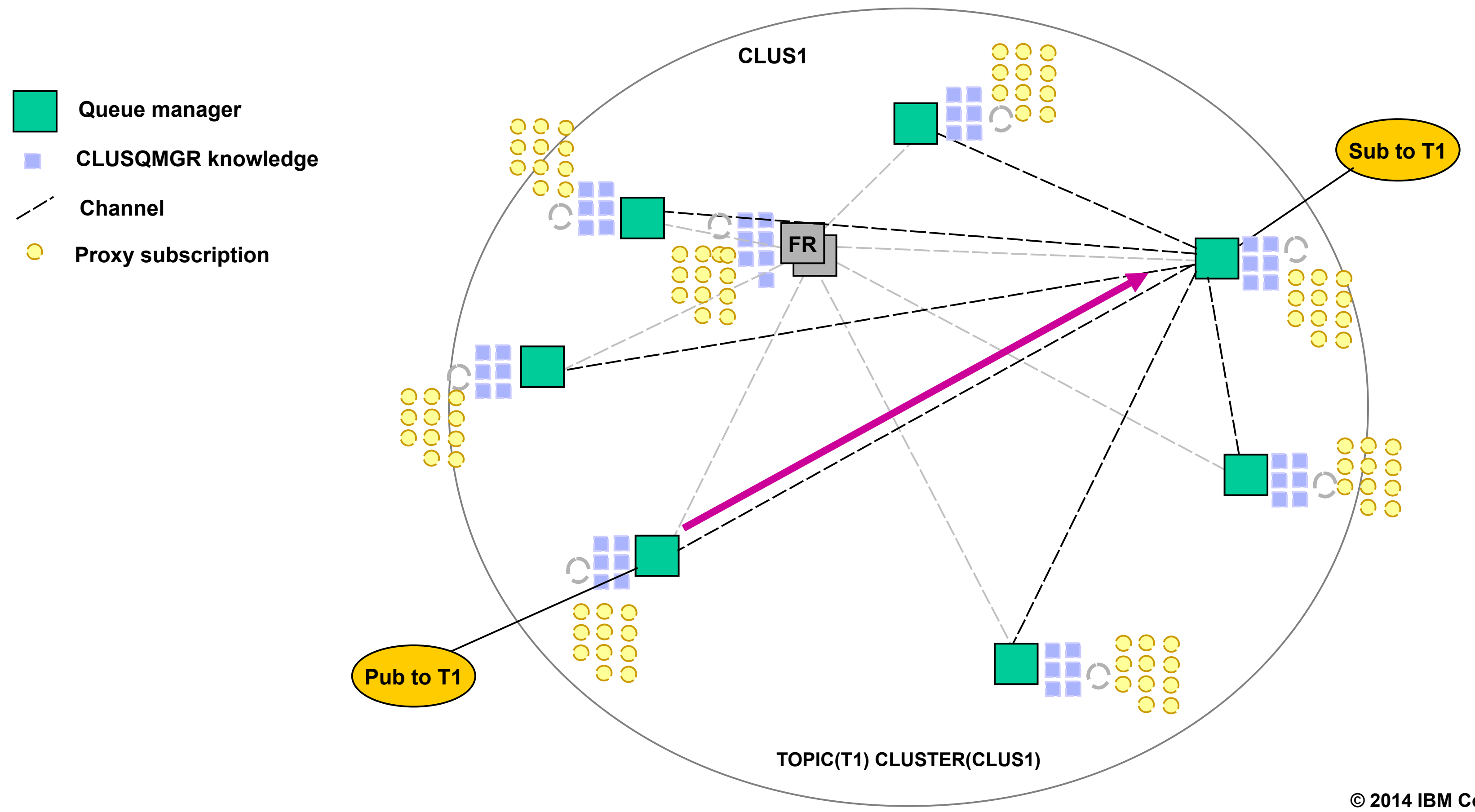
## Publish/subscribe improvements

- Improved `PROXYSUB(FORCE)` behaviour for publish everywhere
  - Version 7 uses individual proxy subscriptions
  - Version 8 uses wildcards where appropriate to reduce flows
- Improved scaling for large topic trees
  - Linear scaling to at least a million topics
- Improved `DISPLAY PUBSUB`

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

# Routed Publish/Subscribe

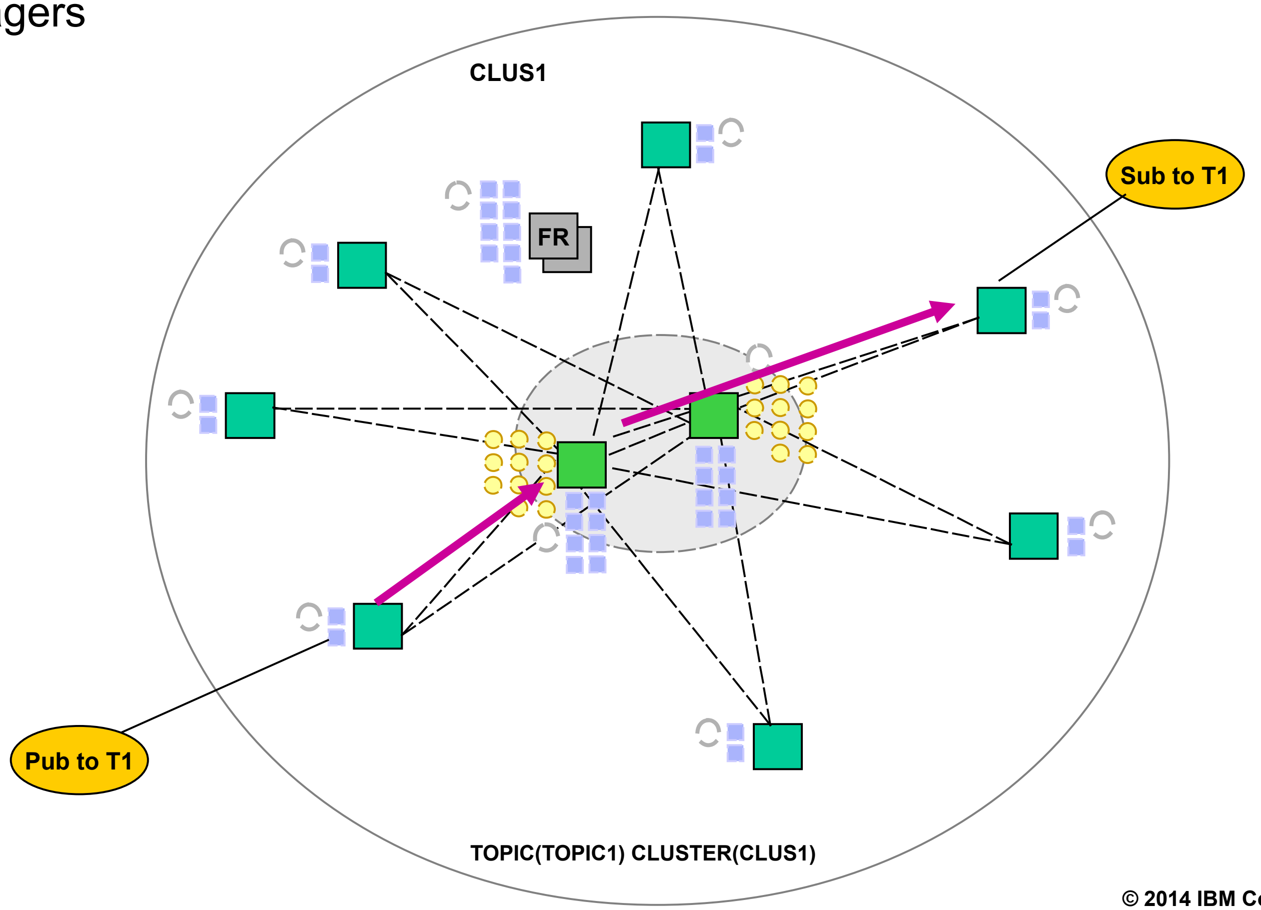
- In version 7, all queue managers in a cluster know everything and need to be able to connect to anyone



## 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 |



## Configuration of routed topics

- Topic routing is configured in the TOPIC object definition
  - Uses the CLROUTE property:

```
DEFINE TOPIC(topic)
  CLUSTER(clustername)
  CLROUTE(DIRECT|TOPICHOST)
```

  - DIRECT: Provides V7 behaviour, publications travel directly from publishing queue managers to subscribing queue managers
  - TOPICHOST: publications travel via a topic hosting queue manager

- New function in MQ V8
  - Only V8 queue managers can participate in routed clustered publish/subscribe
  - Full repositories must also be at V8
  - Older queue managers are not told of the routed topic definitions and therefore behave as if the topic was not clustered.
  - All V7 queue managers can continue to participate in direct clustered pub/sub

## 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

## 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

## MQ for z/OS: 64-bit bufferpools

- 64-bit buffer pools in MQ for z/OS
  - Allows large numbers of messages to be cached before writing to pagesets
  - Allows MQ to exploit the vast amount of storage on today's machines
- Improves performance of putting/getting messages by minimizing disk I/O
- Minimizes administrative overhead of managing buffer pools
- Buffer pool LOCATION attribute says where it is located relative to the bar
  - BELOW: The default. Buffer pool is located below the bar in 31 bit storage
  - ABOVE: Buffer pool is located above the bar in 64 bit storage
  - This can be altered dynamically
- Storage can be pinned based on pageclass attribute

## MQ for z/OS: Performance and  Capacity

- Log RBA constraint relief
  - Already improved messages to warn of approaching RBA
  - Now widening RBA field from 6 to 8 bytes
  - At 100MB/sec this will now take about 5578 years to fill, whereas with 6 byte RBA some customers reach the limit in 12-18 months

- Support for LP64 batch/RRS C applications
- Performance enhancements for IBM Information Replicator (QRep) and similar application patterns
  - Read-ahead and changes to deferred write processing allows MQ  to increase sustainable data rates

- General improvements to channel performance

## Summary

| *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 |

## Questions and Answers