

WebSphere MQ Security

Craig Both – bothcr@uk.ibm.com
IBM Hursley

14th March 2012
10697



Agenda

- Objectives and Terminology
- WebSphere MQ Security
 - Identification
 - Authentication
 - Access Control
 - Auditing
 - Confidentiality
 - Data Integrity
- Including new security features in V7.1



WebSphere MQ Security - Notes

- When you start thinking about security, you need to decide exactly what it is you want to achieve, determine what your objectives are.
- We will start with a look at some possible objectives you may have and introduce the terminology used to describe each of these, and exactly what these terms mean. These terms will be used throughout the remainder of the presentation.
 - Objectives - What are you trying to achieve?
 - Terminology - What do we mean by these?
- Then we will take a closer look at WebSphere MQ messages and what attributes in a message are relevant to the security of them.
- Finally, we will look at the security features available in the WebSphere MQ product.

Objectives and Terminology



Objectives and Terminology

- Ensure each user is uniquely identified
 - Identification:- Being able to uniquely identify a user of a system or an application that is running in the system.
- Prove that a user is who they say they are
 - Authentication:- Being able to prove that a user or application is genuinely who that person or what that application claims to be.
- Track who does what to what and when
 - Auditing:- Tracking who has done what to what and when.

Objectives and Terminology

- Limit Access to authorized users only
 - Access Control:- Protects critical resources in a system by limiting access only to authorized users and their applications. It prevents unauthorized use of a resource or the use of a resource in an unauthorized manner.
- Protect your sensitive data from unauthorized viewing
 - Confidentiality:- Protects sensitive information from unauthorized disclosure.

Objectives and Terminology

- Check unauthorized changes have not been made to data
 - Data Integrity:- Detects whether there has been unauthorized modification of data. There are two ways in which this can occur, accidentally, through hardware or transmission errors, or by deliberate attack.
- Ensure a message really is associated with whom it claims
 - 'Non-Repudiation':- The goal is usually to prove that a particular message is associated with a particular individual.

WebSphere MQ Security



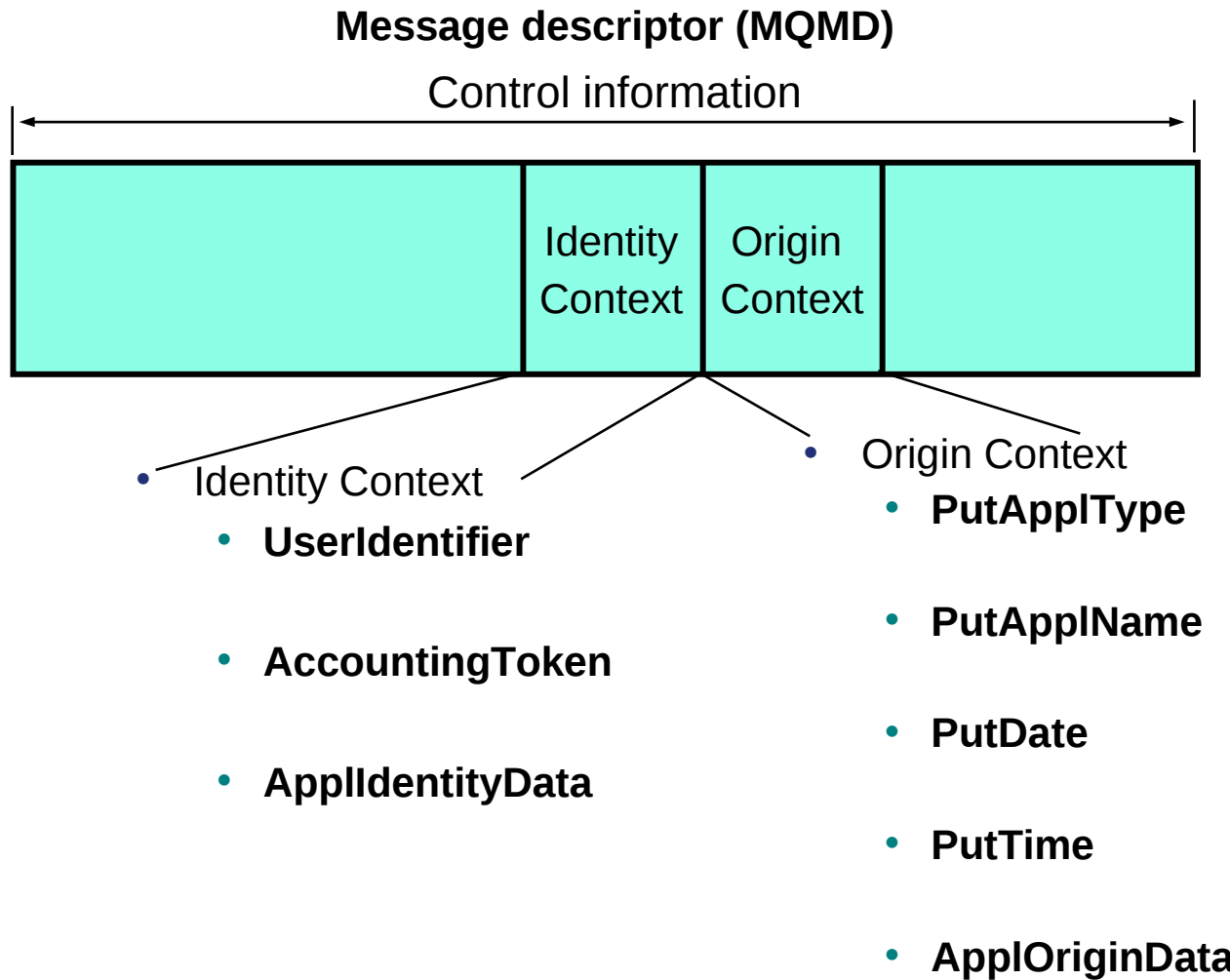
WebSphere MQ Security

- Identification
 - O/S User IDs
 - Context
 - Link-level considerations (later)
- Authentication
- Access Control
- Auditing
- Confidentiality
- Data Integrity
- Non-Repudiation

WebSphere MQ Security - Notes

- We're going to look at what security features are available to you, when you use WebSphere MQ, under each of these sections.
- 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 which we'll look at on the next page. Later in the presentation we'll also see that with appropriate access granted to a user, context information can be specified by the application instead of being generated from the O/S.
- 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. There is also a feature on MQCONNx for authentication purposes that we will look at on a later foil.

What is Context information?



Origin Context Information - Notes

- Origin Context information usually relates to the most recent application that put the message on the queue where it is currently stored. There are exceptions to this, for example the Message Channel Agent, who leaves these fields as they were when received. Origin Context is made up of the following fields :-
- PutApplType: The type of application that put the message, for example CICS, IMS, AIX...
- PutAppName: The name of the application that put the message. The value of this field depends on the PutApplType and if set by the queue manager it is determined by the environment.
- PutDate: The date on which the message was put on the queue. When generated by the queue manager the format is YYYYMMDD
 - YYYY - year (four numeric digits); MM - month (01 through 12); DD - day of month (01 through 31)
- PutTime: The time at which the message was put on the queue. When generated by the queue manager the format is HHMMSSSTH
 - HH - hours (01 through 23); MM - minutes (01 through 59); SS - seconds (01 through 59)
 - T - tenths of seconds (0 through 9); H - hundredths of seconds (0 through 9)
- ApplOriginData: Any other information the application may want to add. For example suitably authorized applications may state whether the identity context information is to be trusted.
- Origin context information is usually supplied by the queue manager and Greenwich Mean Time(GMT) is used for the put time and date

Identity Context Information - Notes

- Identity Context information usually relates to the application that first put the message on the queue and is made up of the following fields :-
- **UserIdentifier:** A 1 -12 character field contains the User Identifier of the application that put the message on the queue. For userids longer than 12 characters the first 12 characters are used. The queue manager fills this in with a name that identifies the user. The way that it does this depends upon the environment in which the application is running. Once the message has been received this can be used in the Alternate Userid field of the object descriptor parameter of the subsequent MQOPEN call.
- **AccountingToken:** Allows work done as a result of the message to be charged correctly, how the queue manager fills this in depends upon the environment. When a message is created under WebSphere MQ for windows a Windows System Security Identifier (SID) is stored in the Accounting token and can be used to supplement the user identifier when establishing credentials.
- **ApplIdentityData :** This field is for use by the application to store any information it wants. When it is generated by the queue manager it is left entirely blank.
- Suitably authorized applications can set these fields if they need to.
- Applications that pass messages on from one queue manager to another should pass on the identity context information so the receiving applications know the identity of the originator of the message.



SHARE
Technology • Connections • Results

WebSphere MQ Security

- Identification
 - O/S User IDs
 - Context
 - Link-level considerations (later)
- Authentication
 - O/S Logon
 - MQCONN
 - Link-level considerations (later)
- Access Control
- Auditing
- Confidentiality
- Data Integrity
- Non-Repudiation

Authentication - MQCONNX

- MQCSP structure
 - Connection Security Parameters
 - User ID and password
- MQCNO structure
 - Connection Options
- Passed to OAM
 - Distributed Queue Manager only
- Also passed to Security Exit
 - Both z/OS and Distributed

```
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;  
csp.CSPPasswordPtr    = "12345";  
csp.CSPPasswordLength = 5;
```


Authentication - MQCONNX - Notes

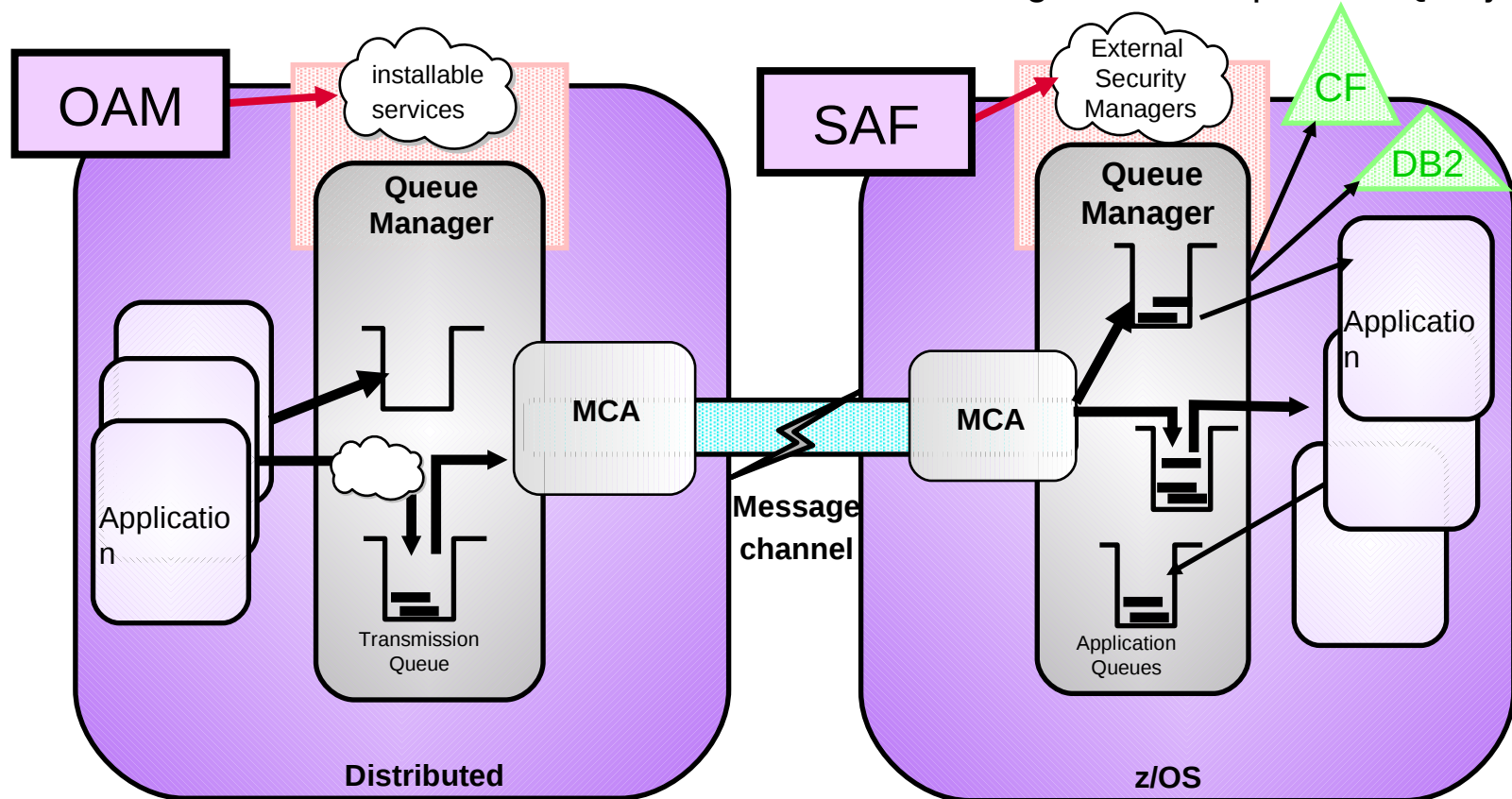
- On MQCONNX an application can provide a user ID and password (in the Connection Security Parameters (MQCSP) structure in the MQCNO), which are passed to a user written plug-point in the OAM on distributed to be checked.
- If the application is running client bound, this user ID and password are also 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 – more on channels later.
- If the queue manager is z/OS, there is no OAM plug-point, and the security exit can be used to call RACF (for example) to check the user ID and password. A sample security exit is provided (CSQ4BCX3) to illustrate how to do this.

WebSphere MQ Security

- Identification
 - O/S User IDs
 - Context
 - Link-level considerations (later)
- Authentication
 - O/S Logon
 - MQCONNX
 - Link-level considerations (later)
- Access Control
 - Link-level considerations (later)
- Auditing
- Confidentiality
- Data Integrity
- Non-Repudiation

Access Control Mechanisms

- Two equivalent mechanisms
 - To secure the following tasks
 - Administering WebSphere MQ
 - Working with WebSphere MQ objects



Access Control Mechanisms - Notes

- All advanced level Queue Managers provide access control facilities to control which users have access to which MQ resources. Note, however, that none of the Queue Managers has an access control component; all Queue Managers make use of an associated security manager to provide access control services.
- MQ for z/OS uses the z/OS standard System Authorization Facility (SAF) interface to an external security manager (ESM). This means that MQ for z/OS can operate with any security manager which conforms to the SAF interface. Examples of such (conforming) security managers are RACF, Top Secret and ACF2.
- The distributed Queue Managers use the Installable Services component of MQ - using the Authorization Service - to provide access control for MQ resources. MQ supplies an Object Authority Manager (OAM) as an authorization service which conforms to the Installable Services interface.
- The OAM provides a full set of access control facilities for MQ including both the access control checking and commands to set, change and inquire on MQ access control information. The OAM, like all Installable Services components, is replaceable by any component - user or vendor supplied - that conforms to the authorization Service interface.
- The set of facilities provided for by the different platforms, although similar provide different levels of granular control of resources and capabilities.
- The next few foils will detail the various commands on the different platforms.

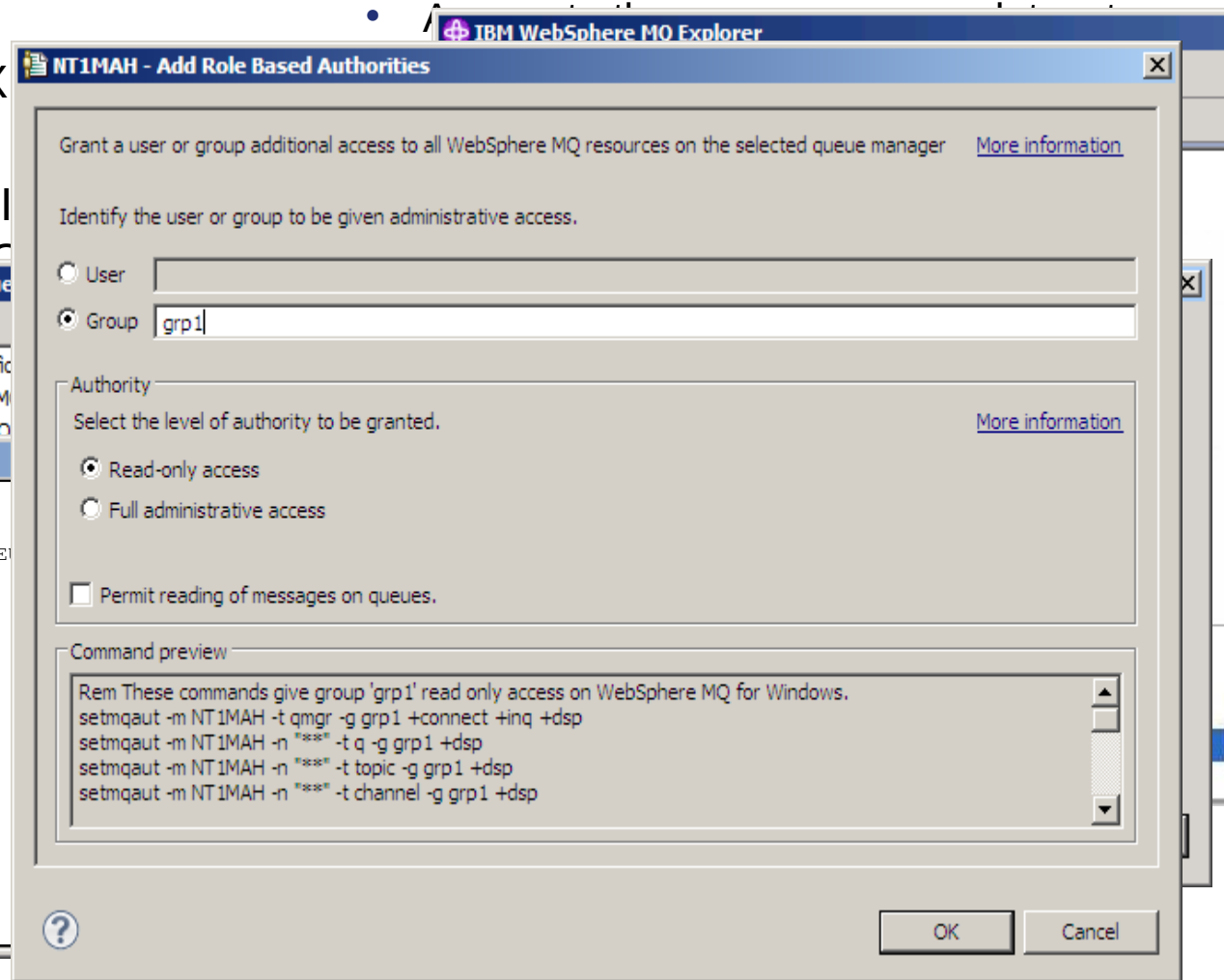
Access Control - Administering WebSphere MQ

- Control Commands
 - e.g. crtmqm (UNIX)
 - CRTMQM (IBM i)
 - e.g. setmqaut (UNIX)
 - Windows) CRTMQM (IBM i)
 - use mqm g
 - secure

```

C:\ Command Prompt - runmqsc TEST1
Starting MQSC for queue manager TEST1.

SET AUTHREC PROFILE ('APP1.**') OBJTYPE (QUE)
AUTHADD (PUT,GET,INQ)
  
```



IBM WebSphere MQ Explorer

NT1MAH - Add Role Based Authorities

Grant a user or group additional access to all WebSphere MQ resources on the selected queue manager [More information](#)

Identify the user or group to be given administrative access.

User

Group

Authority

Select the level of authority to be granted. [More information](#)

Read-only access

Full administrative access

Permit reading of messages on queues.

Command preview

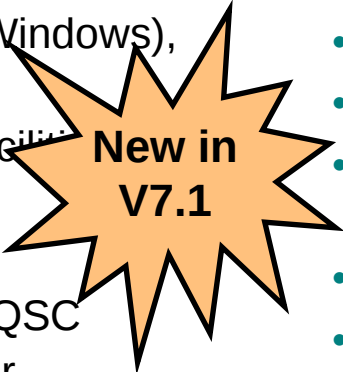
```

Rem These commands give group 'grp1' read only access on WebSphere MQ for Windows.
setmqaut -m NT1MAH -t qmgr -g grp1 +connect +inq +dsp
setmqaut -m NT1MAH -n "*" -t q -g grp1 +dsp
setmqaut -m NT1MAH -n "*" -t topic -g grp1 +dsp
setmqaut -m NT1MAH -n "*" -t channel -g grp1 +dsp
  
```

Buttons: ? OK Cancel

Access Control - Administering WebSphere MQ

- Control Commands
 - e.g. crtmqm (UNIX & Windows), CRTMQM (IBM i)
 - e.g. setmqaut (UNIX & Windows), GRTMQMAUTH (IBM i)
 - use mqm group and OS Facility to secure
- Issue MQ Commands using
 - runmqsc or STRMQMMQSC
 - WebSphere MQ Explorer
 - Ops and Control panels on z/OS
 - WebSphere MQ CSQUTIL Utility program on z/OS
- Access the queue manager datasets on z/OS
- Secure access to MQ Commands
 - use setmqaut (UNIX & Windows)
 - GRTMQMAUTH (IBM i)
 - **SET AUTHREC MQSC Commands (Dist)**
 - Authorities Wizards in MQ Explorer
 - use ESM (z/OS) profiles to secure
 - switches
 - Command Security - controls who is allowed to issue an MQSC command
 - Command Resource security - protects WebSphere MQ resources
- Role Based Authorities Wizard



Administering WebSphere MQ - Notes

- WebSphere MQ Administrators need authority to do these various tasks. On UNIX and Windows, Administrators must be a member of mqm group, and on i5/OS a member of QMQMADM group (or have *ALLOBJ authority). Members of these groups have access to all WebSphere MQ resources on the system, and queue managers running on the system.
- There are several MQ commands available - both for controlling the Queue Manager (such as CRTMQM, STRMQM) and for configuration of the Queue Manager (such as SETMQAUT). Either (or both) the operating system or WebSphere MQ facilities may be used to control which users may access these commands. Base operating system facilities may be used to control access to libraries which contain the commands. Such facilities are outside of the scope of WebSphere MQ. Alternatively, WebSphere MQ provides access control facilities to restrict access.
- Commands can be issued in a number of ways. The control command runmqsc (UNIX & Windows) or STRMQMMQSC (i5/OS) can be secured, but also the user must have authority to issue the WebSphere MQ command and authority to access the WebSphere MQ Object. i5/OS also has Group 2 commands (e.g. CRTMQMQ to create a queue, or CHGMQMPROC to change a process) where in addition to the above you need i5/OS authority to use the command. This is granted using the GRTOBJAUT command.
- Commands can be issued from a remote machine, so controlling the runmqsc control command is clearly not enough. Your WebSphere MQ Commands and Objects must also be controlled.

Administering .. on z/OS - Notes

- WebSphere MQ on z/OS uses the ESM to maintain the security control for access to commands and objects. This is the equivalent to the setmqaut / CRTMQMAUTH commands we've just seen on the previous foils.
- Security overall, and the different types of security checking, for example, connection security or queue security, are controlled by a set of switches. These switches are RACF profiles that have a special meaning to WebSphere MQ. For example, the existence of a profile called hlq.NO.CONNECT.CHECKS tells WebSphere MQ not to do any connection security checking. If it does not exist then WebSphere MQ will do connection security checking.
- In addition to the security set up for command and resource checking, the other considerations are the security of the various data sets. Base O/S facilities are used to control access to these. Some points to note are:
 - You should control who has access to the started task procedures for the Queue Manager and Channel Initiator.
 - Commands can be issued in the CSQINP1 and CSQINP2 data sets. There are no security checks done on these commands because they happen before the security manager has started on the queue manager. Therefore these data sets must be secured to ensure no unauthorized commands can be added to them.
 - Commands issued from the Ops and Control panels, or via CSQUTIL, do have security checking done on the user ID running them. You may also want to control who has access to use them overall though.
 - In order to set up Queue Sharing Groups, the queue manager requires access to Coupling Facility and DB2 data sets. Also think about access for Qmgrs that will use DB2 and IMS.

Secure Access to MQ Commands - Notes

- setmqaut (UNIX & Windows) or GRTMQMAUTH (i5/OS) are control commands used to grant authorities to other users to access WebSphere MQ resources. There are also alternative ways to grant authorities. There are PCF commands to do the equivalent job and these are used by the MQ Explorer GUI to provide an Authorities Wizards allowing easy display and setting of authorities.
- In WebSphere MQ V7.1 MQSC commands to do the equivalent job were also added allowing an MQSC script of object definitions and authorities to be created which is very useful for recreating your queue manager, instead of needing two separate scripts.
- Originally part of SupportPac MD05, the Role Based Authorities Wizard in the MQ Explorer GUI is now part of the product too. This wizard allows a very quick way to permit a user read-only access to all objects, or admin access (without mqm group membership) to all objects. The commands generated by the wizard are shown in the bottom pane and can be cut'n'pasted into a script for future use.

Access Control - API Security

- Queue Managers
 - Using MQCONN or MQCONNX (see notes for reference)
- Working with WebSphere MQ objects
 - Using MQOPEN (or MQSUB)
 - Namelists (see notes for reference)
 - Processes (see notes for reference)
 - Queues
 - Topics
- Alternate Userid
 - (see notes for reference)
- Message Context
 - (see notes for reference)

Access Control - API Security - Notes

- We have looked at the various different platform specific ways to control which users have access to which objects. We are now going to look at exactly what can be secured using these different mechanisms as we discuss API Security.
- Access to MQ applications may be controlled by restricting access to the link libraries (used when link-editing MQ applications) and then by restricting access to the compiled and linked executable. Both of these controls are base operating system facilities and are outside the scope of MQ. Again, for the supremely cautious, placing of the MQ link libraries onto diskette will certainly restrict access to diskette holders only !!
- MQ provides access control facilities to control which users may run applications which issue MQCONN API calls. This will control which users may access the running Queue Manager, even though they may have access to the application libraries.
- Once a program is connected to the queue manager, it is very likely that MQ resources will be used. The queue manager will control which users have access to which resources and in which way. Note that all (well, most) access control checks are made when a resource is opened. There are no resource checks made at GET and PUT time.

API Security

- Queue Managers
 - When an application connects to a Queue manager using an MQCONN or MQCONNX

- Working with WebSphere MQ objects
 - Namelists
 - When an application opens a Namelist using an MQOPEN
 - Processes
 - When an application opens a Process using an MQOPEN

API Security - Notes

- Checks are carried out for the following API calls.

MQCONN, MQCONNX

- When an application tries to connect to a Queue manager using either MQCONN or MQCONNX, the Queue manager asks the operating system for a userid associated with the application and then checks to see if that application is authorized to connect to the Queue manager.

MQOPEN

- When an application tries to access a WebSphere MQ resource checks will be performed to see whether the userid(s) associated with that application have the correct authority to do what they are requesting. For Namelists and Processes these checks are carried out when an MQOPEN is issued.

API Security – Queues (next page)

MQOPEN, MQPUT1, MQCLOSE

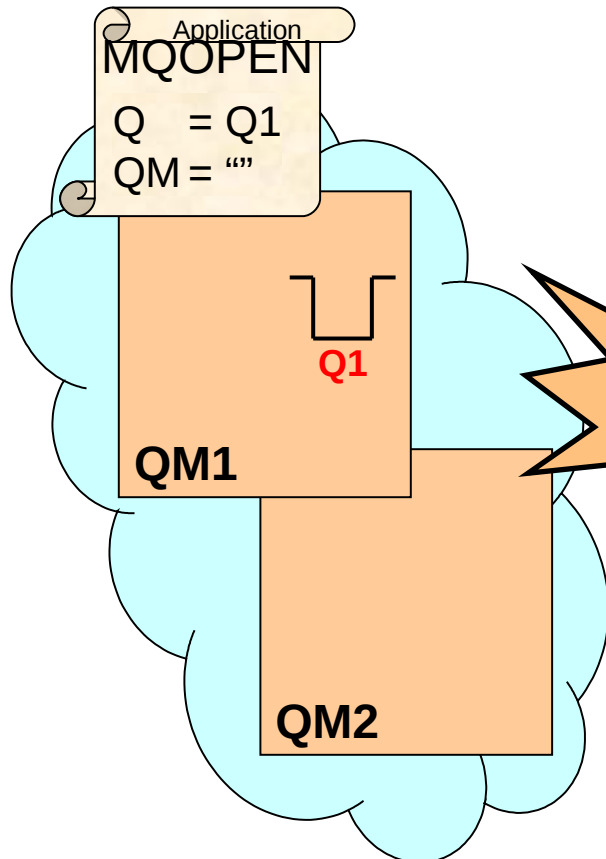
- When an application tries to access a WebSphere MQ resource checks will be performed to see whether the userid(s) associated with that application have the correct authority to do what they are requesting. These checks are usually carried out when an MQOPEN or MQPUT1 is issued.

MQOPEN

- when opening an ALIASQ that resolves to a topic two checks will be performed, one against the aliasq name and one against the topic it resolves to. (this is different to aliasq-> queue resolution)

API Security - Queues

- Working with WebSphere MQ Queues
 - When an application opens a Queue using an MQOPEN or MQPUT1.
 - Named resource that is checked



Changed in V7.1

- Alias queues that resolve to a topic
 - Check on named alias queue
 - Check on topic
- Creation of a dynamic queue
 - Check on named model queue
 - Check on dynamic queue name (generic profile!!)
- When an MQSUB or DEFINE SUB specifies a destination queue for publications
- When an application close deletes a permanent dynamic queue using an MQCLOSE
- Fully qualified Remote queues
 - Check on XMITQ actually used (Dist)
 - Check on ToQmgrName (z/OS)
- **Remotely hosted cluster queues**
 - **Check on named cluster queue (All)**
 - On Dist prior to MQ V7.1 check was on CLUSTER XMITQ
- **Fully qualified cluster queues**
 - **Check on ToQmgrName profile (All)**
 - On Dist prior to MQ V7.1 check was on CLUSTER XMITQ

API Security - Queues - Notes

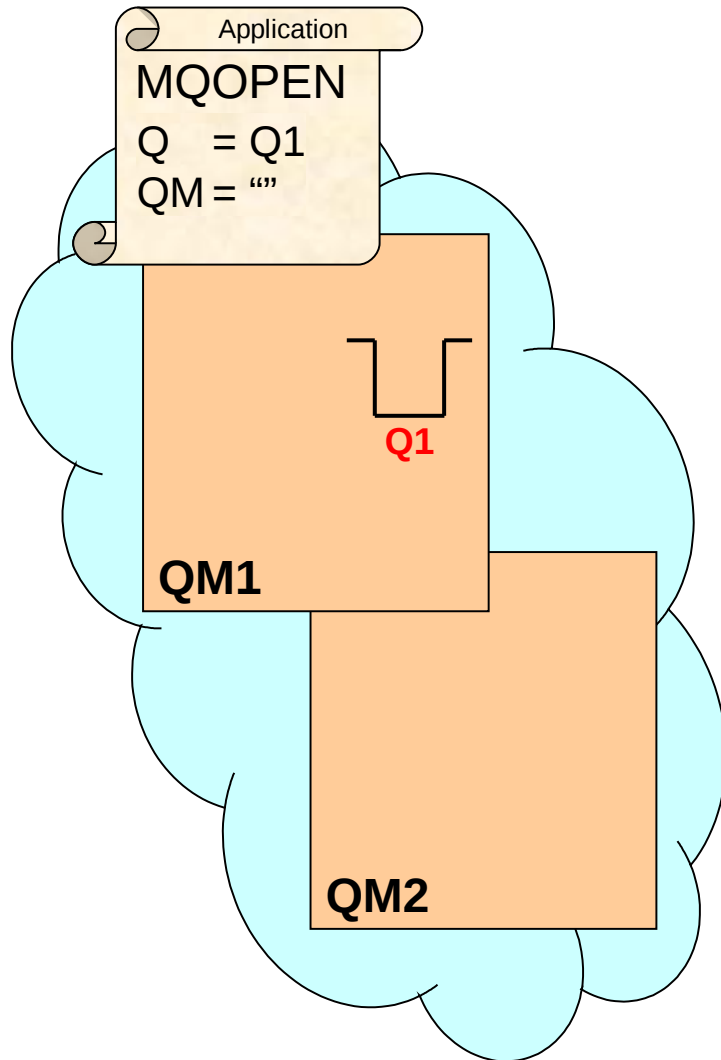
MQSUB or DEFINE SUB command

- When an application performs an MQSUB or a DEFINE SUB command supplying a destination queue for the publications to be sent to a check is carried out to ensure the subscriber has authority to PUT messages on that queue.

MQCLOSE

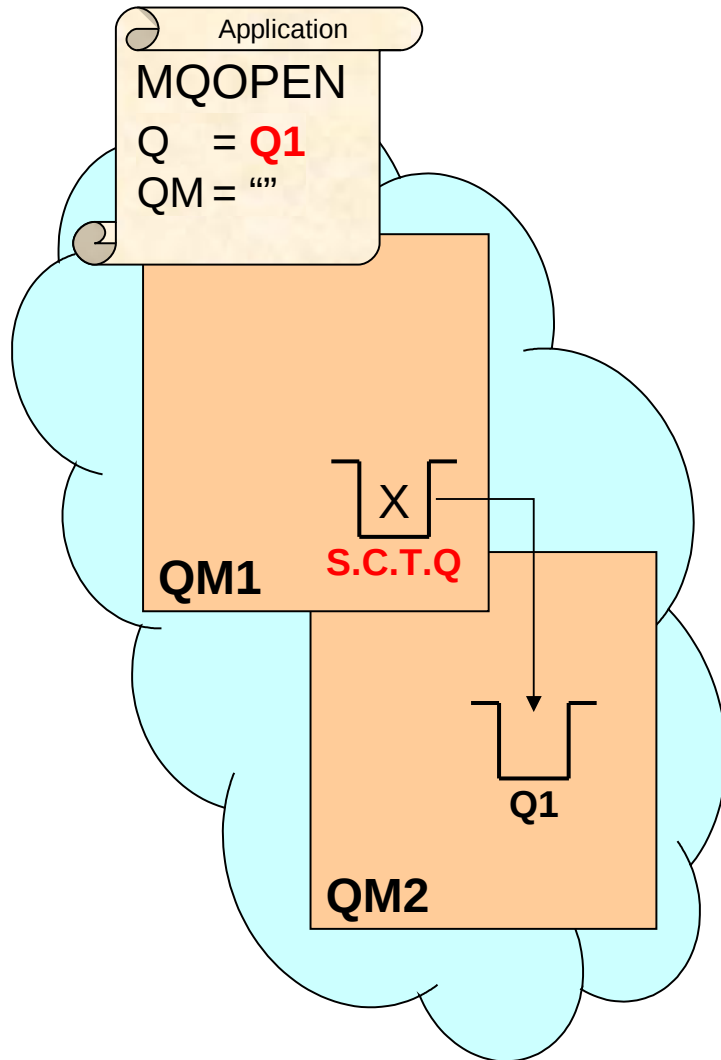
- A check can also be performed when an MQCLOSE, with the DELETE option, is issued for a permanent dynamic queue.
- MQ checks to see if a user is permitted to access a particular resource, it is the name specified in the MQ API call which is used for the check. For the case of an Alias or Remote queue definition, it is still the name of the queue specified in the MQ API call and not the resolved-to name. Thus, a user needs access to the first named resource and not the resolved-to resource.
- For dynamic queues, there may be instances where MQ will generate the name from the model queue. In this case it is recommended that generic named profiles are used for access control.
- If your application is opening queues using the fully-qualified technique (where the queue manager name is also specified in the MQOD.ObjectQMgrName), then there is no local object (e.g. a name QREMOTE definition) that can be checked. This is particularly relevant for the ReplyToQueue/ReplyToQMgr which may be being used for sending responses from a server application.
- In this case the profile checked will be
 - the transmission queue which will be used to send messages to that remote queue manager if you are using a distributed platform
 - a profile naming the 'ToQMgr' (instead of the transmission queue) if you are using z/OS
- Often times you transmission queue will be named to match your remote queue manager so these will amount to the same name.

Cluster Queue Security in WebSphere MQ V7.1



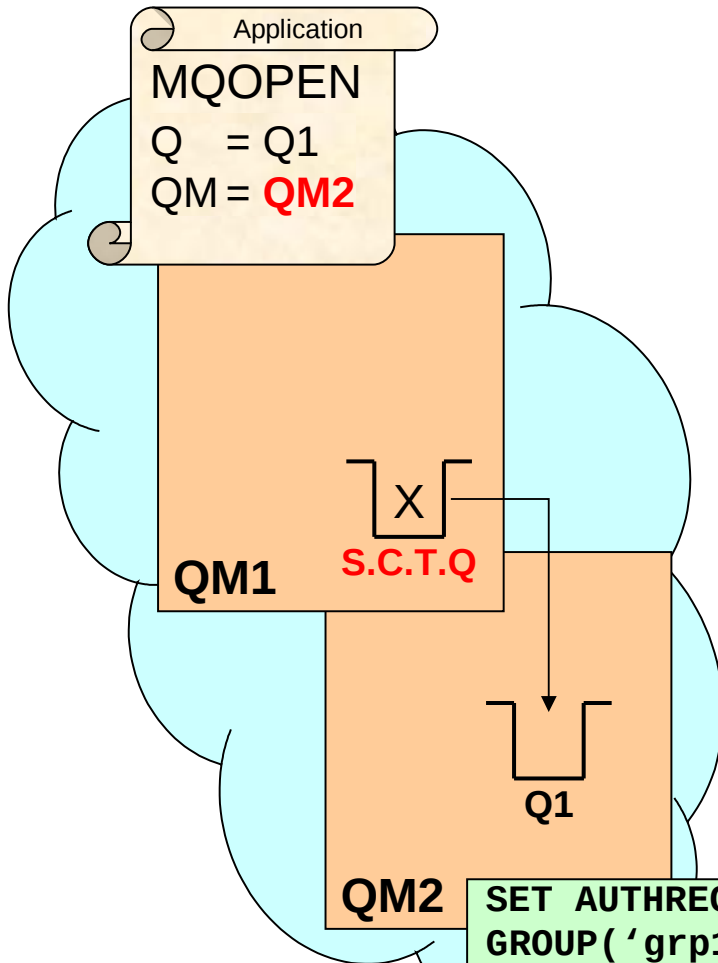
- Cluster queue hosted locally
 - Access check on named queue profile

Cluster Queue Security in WebSphere MQ V7.1



- Cluster queue hosted locally
 - Access check on named queue profile
- Cluster queue hosted remotely
 - Access check on named queue profile
 - **Location of hosted queue now makes no difference to access control**
 - Previously checked the SYSTEM.CLUSTER.TRANSMIT.QUEUE
 - Controlled in qm.ini using `ClusterQueueAccessControl = Xmitq | RQMName`

Cluster Queue Security in WebSphere MQ V7.1



- Cluster queue hosted locally
 - Access check on named queue profile
- Cluster queue hosted remotely
 - Access check on named queue profile
 - **Location of hosted queue now makes no difference to access control**
 - Previously checked the SYSTEM.CLUSTER.TRANSMIT.QUEUE
 - Controlled in qm.ini using `ClusterQueueAccessControl = Xmitq | RQMName`
- Fully qualified cluster queue
 - Access check on named queue manager profile
 - Previously checked the SYSTEM.CLUSTER.TRANSMIT.QUEUE
 - Same qm.ini control (as above)
 - New profile type rqmname

```
SET AUTHREC OBJTYPE(RQMNAME) PROFILE(QM2)
GROUP('grp1') AUTHADD(PUT)
```

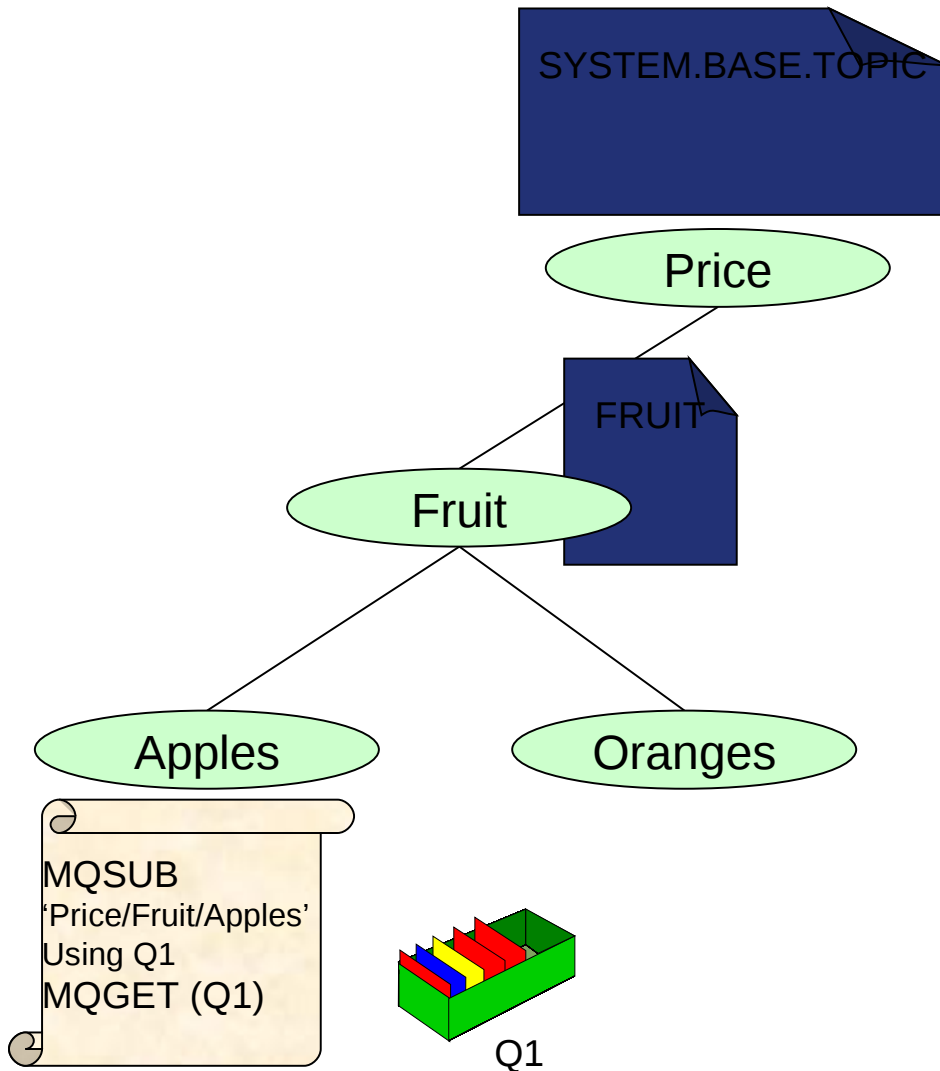
```
setmqaut -m QM1 -t rqmname -n QM2 -g grp1
+put
```

API Security – Remote/Cluster Queues – Notes



- If your application is opening a cluster queue which is hosted remotely then on z/OS and on Distributed platforms from WebSphere MQ 7.1 and onwards, the named object is the profile that is checked. On distributed platforms prior to WebSphere MQ V7.1, the profile for the SYSTEM.CLUSTER.TRANSMIT.QUEUE was checked.
- If your application is opening a cluster queue using the fully-qualified technique, then on z/OS and on Distributed platforms from WebSphere MQ 7.1 and onwards, a profile which represents the 'ToQmgr' is checked. On distributed platforms prior to WebSphere MQ V7.1, the profile for the SYSTEM.CLUSTER.TRANSMIT.QUEUE was checked.
- Control or whether the SYSTEM.CLUSTER.TRANSMIT.QUEUE is used or the new mode, is done by means of setting in the qm.ini file `ClusterQueueAccessControl = Xmitq | RQMName`
- The default in both cases is to continue to do the old way, checking the SYSTEM.CLUSTER.TRANSMIT.QUEUE, but once you are ready and have made the appropriate new profiles, this change can be made and the queue manager restarted to run in the new mode.
- For the fully-qualified case, there is a new profile type called 'rqmname' which is used to set accesses for remote queue managers.

Topic Security



- When an application Subscribes or Publishes to a Topic using
 - MQSUB
 - MQOPEN / MQPUT1
- When an application removes a subscription using
 - MQCLOSE - with option MQCO_REMOVE_SUB
- Authority check on topic objects
 - “Walk up the tree”
 - May be more than one check
- Authority check on destination queue
 - When not using MQSO_MANAGED
 - Check is for PUT to that queue

API Security - Topics - Notes

MQSUB

- A security is performed during MQSUB processing to see whether the application making the request has the required access to that topic.
- An additional authorization check is done for an MQSUB call when the application wishes to use a specific destination queue (i.e. is not using the MQSO_MANAGED option). In this case we also check that this user ID has authority to PUT to that destination queue.

MQOPEN, MQPUT1

- if an application is making a publication to a topic using an MQOPEN or MQPUT1 request a security check is performed to see whether that application has the required access to that topic.
- If an application is making a publication to a topic via an alias queue that resolves to a topic then two checks will take place, one to ensure that the application has access to the alias queue and then one to ensure that the application has the required access to the topic. This is additional processing to that when the alias queue points to another queue and is done to ensure that no matter how the topic tree is accessed, the same security is applied to it.

MQCLOSE

- A check can also be performed when an MQCLOSE is performed for a subscription using the remove sub option.
- In our example we have called MQSUB at the point in the topic tree, “Price/Fruit/Apples”. There is no topic object at this point in the topic tree, so to find the profile we need to check authorities against we walk up the topic tree to find a node which does have a topic object. The next point is “Price/Fruit”. This does have a topic object, FRUIT, so we will check that this user ID has subscribe authority on the profile for the FRUIT topic. If that user ID does have authority, our search stops there. If it does not, we carry on searching up the topic tree and will check the SYSTEM.BASE.TOPIC to see if this user ID has subscribe authority there. This means that the structure of your topic tree and the administration of it requires careful thought.

API Security - Alternate Userid

- Alternate Userid
 - When an application wants to open/put a message to a queue or topic using an Alternate user ID, the check is carried out on the MQOPEN or MQPUT1
 - If Frederick is allowed to use an alternate user ID of “FRED”, then the check on the resource being MQOPENed will be done using “FRED”

- Logged-on UserId(Frederick)

```
OpnOpts = MQ00_OUTPUT
          | MQ00_ALTERNATE_USER_AUTHORITY
          | MQ00_FAIL_IF QUIESCING;

strncpy(ObjDesc.AlternateUser,
        "FRED",
        MQ_USER_ID_LENGTH);

MQOPEN( hConn,
        &ObjDesc,
        OpnOpts,
        &hObj,
        &CompCode,
        &Reason);
```

API Security - Alternate Userid - Notes

- On MQOPEN an application can provide an alternate user ID (in the AlternateUserId field in the MQOD) by using open option MQOO_ALTERNATE_USER_AUTHORITY, for checks to be carried out on instead of the user ID running the application. The user ID requesting use of the AlternateUserId needs authority to be able to do so.
- On z/OS only, if the Queue Manager generated the context information (rather than it being specifically set by the application - as we will discuss next), the given Alternate User ID passed on the MQOPEN is placed in the UserIdentifier field in the MQMD on the MQPUT.

API Security - Alternate Userid

- Alternate Userid
 - When an application wants to subscribe to a topic using an Alternate user ID, the check is carried out on the MQSUB
 - If Alexandra is allowed to use an alternate user ID of “ALEX”, then the check on the topic being subscribed to will be done using “ALEX”
 - DEFINE SUB has a parameter SUBUSER

- Logged-on UserId(Alexandra)

```
SubDesc.Options = MQSO_CREATE
                 | MQSO_ALTERNATE_USER_AUTHORITY
                 | MQSO_FAIL_IF_QUIESCING;

strncpy(SubDesc.AlternateUser,
        "ALEX",
        MQ_USER_ID_LENGTH);

MQSUB( hConn,
        &SubDesc,
        &hObj,
        &hSub,
        &CompCode,
        &Reason);
```


API Security - Alternate Userid - Notes

- On MQSUB an application can provide an alternate user ID (in the AlternateUserId field in the MQSD) by using the sub option MQSO_ALTERNATE_USER_AUTHORITY.
- The user ID requesting use of the AlternateUserId needs authority to be able to do so.
- If you specify MQSO_ALTERNATE_USER_AUTHORITY, the alternate userid is the one used for the authorization check for the subscription and for output to the destination queue (specified in the Hobj parameter of the MQSUB call), in place of the user identifier that the application is currently running under.
- If successful, the user identifier specified in this field is recorded as the subscription owning user identifier in place of the user identifier that the application is currently running under.
- If defining a subscription using a DEFINE SUB command there is a parameter SUBUSER that performs a similar function. It becomes the owning userid in the subscription and is used in the check against the destination queue.

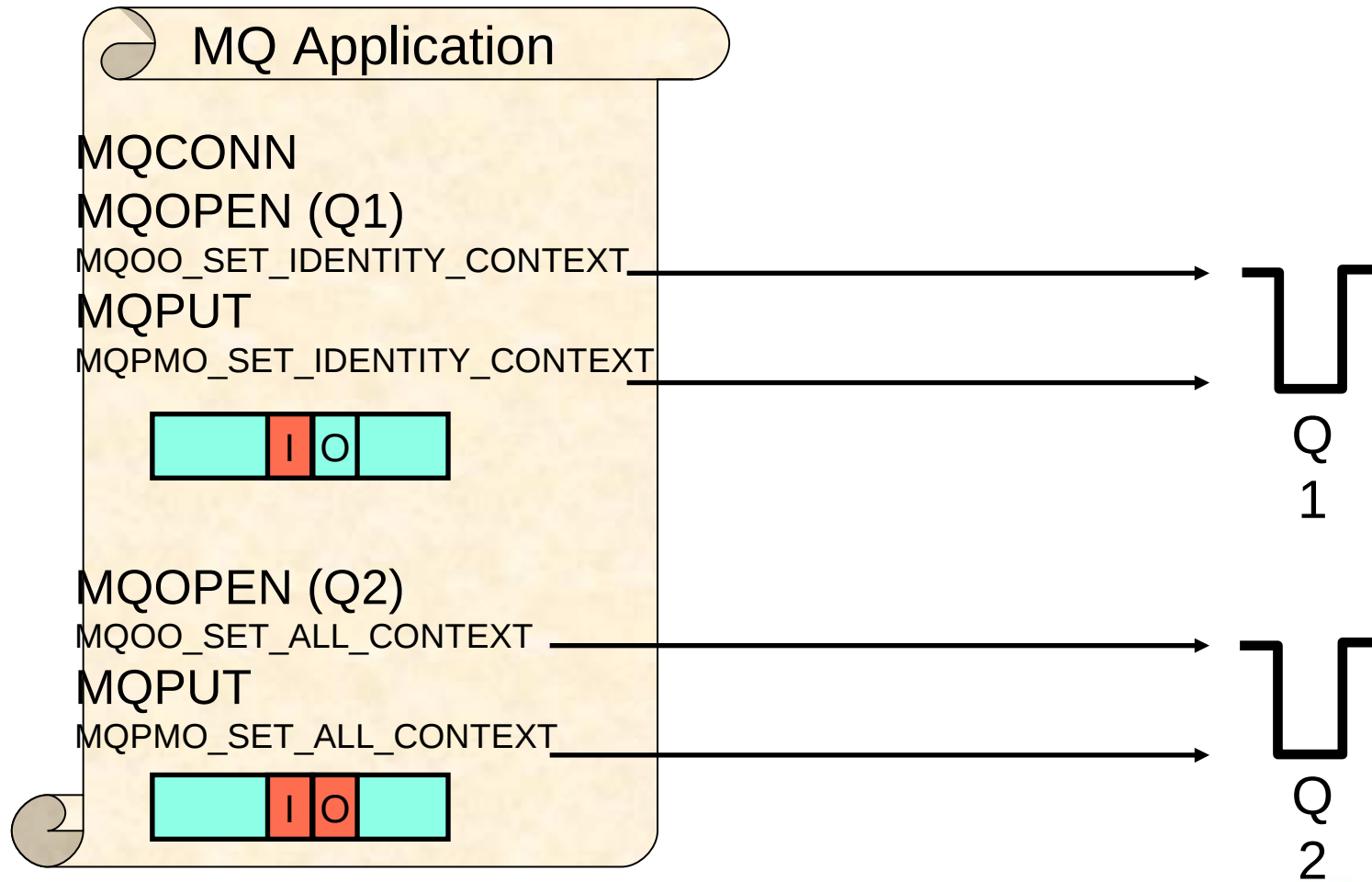
API Security - Context Information

- Set Identity Context
 - MQOPEN –
MQOO_SET_IDENTITY_CONTEXT
 - MQPUT –
MQPMO_SET_IDENTITY_CONTEXT
- Set All Context
 - MQOPEN –
MQOO_SET_ALL_CONTEXT
 - MQPUT –
MQPMO_SET_ALL_CONTEXT
- Set Publication message Identity Context
 - MQSUB –
MQSO_SET_IDENTITY_CONTEXT
- Save Context
 - MQOPEN –
MQOO_SAVE_ALL_CONTEXT
 - Save it from MQGET of a message
- Pass Identity Context
 - MQOPEN –
MQOO_PASS_IDENTITY_CONTEXT
 - MQPUT –
MQPMO_PASS_IDENTITY_CONTEXT
- Pass All Context
 - MQOPEN –
MQOO_PASS_ALL_CONTEXT
 - MQPUT –
MQPMO_PASS_ALL_CONTEXT

API Security - Context Information - Notes

- Context information is controlled by the MQOO and MQPMO options used on MQOPEN, MQPUT1 and MQPUT call. If you have not specified anything in the options field the queue manager may overwrite any existing information held there with context information it has generated for your message. This is the same as specifying MQPMO_DEFAULT_CONTEXT.
- You may want default context when creating a new message or you may want to set the context yourself, but may not necessarily want default context when you are passing on a message. In this case you may wish to pass on the context information
- Whatever you plan to do with context information you need to have opened the queue with the correct authority in order to do so.

API Security – Setting Context

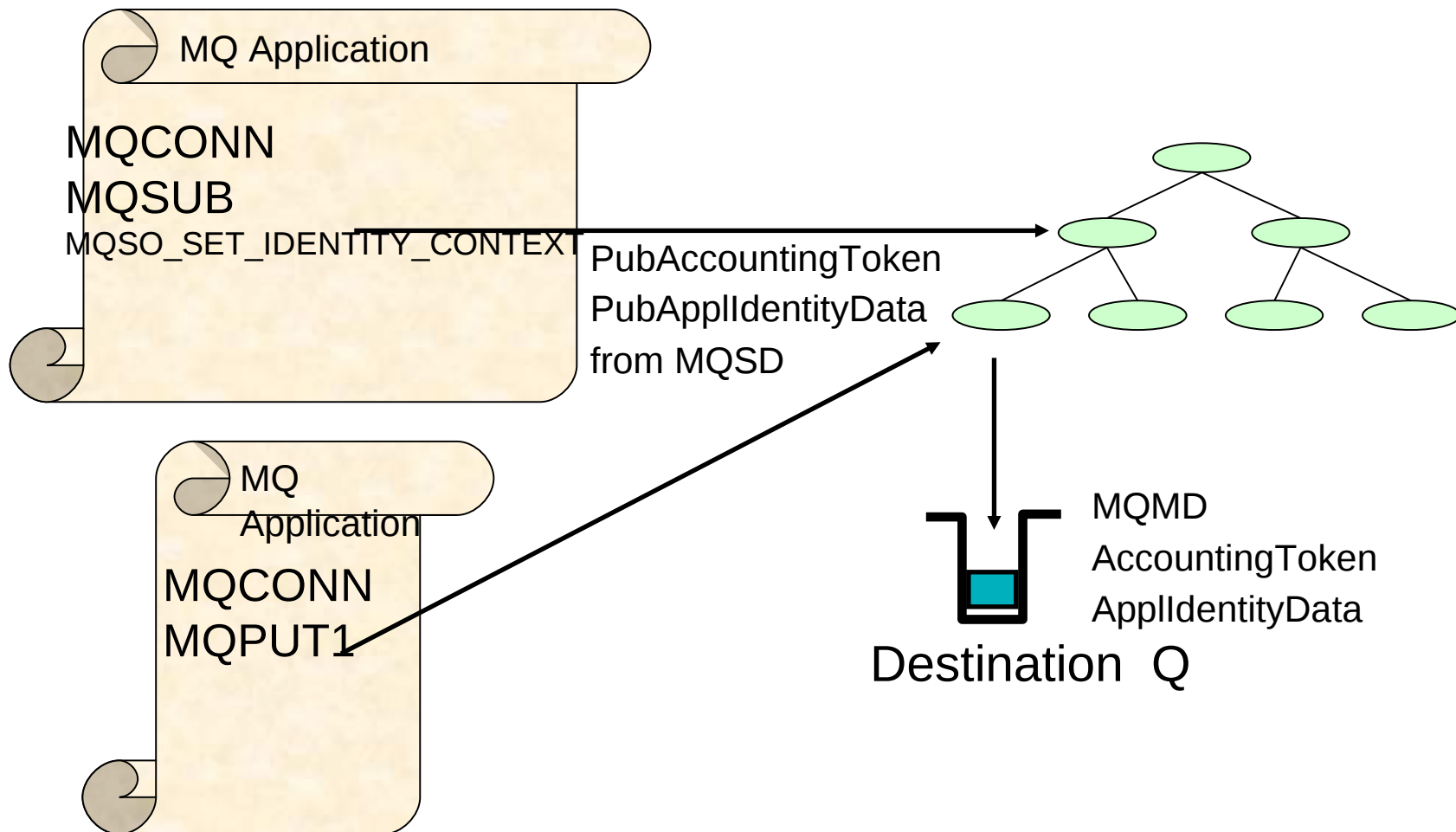


API Security - Setting Context - Notes

MQOPEN and MQPUT options

- If you want to be able to set the identity context of a message and let the queue manager set the origin context of a message then the two options you need are:
 - MQOO_SET_IDENTITY_CONTEXT on the MQOPEN of the queue and
 - MQPMO_SET_IDENTITY_CONTEXT on the MQPUT of the message to the queue
- If you want to be able to set both the identity and the origin context of the message, the options you need are:
 - MQOO_SET_ALL_CONTEXT on the MQOPEN of the queue and
 - MQPMO_SET_ALL_CONTEXT on the MQPUT of the message to the queue
- Appropriate authority is required to be able to do any of these.

API Security - Setting Context - MQSUB

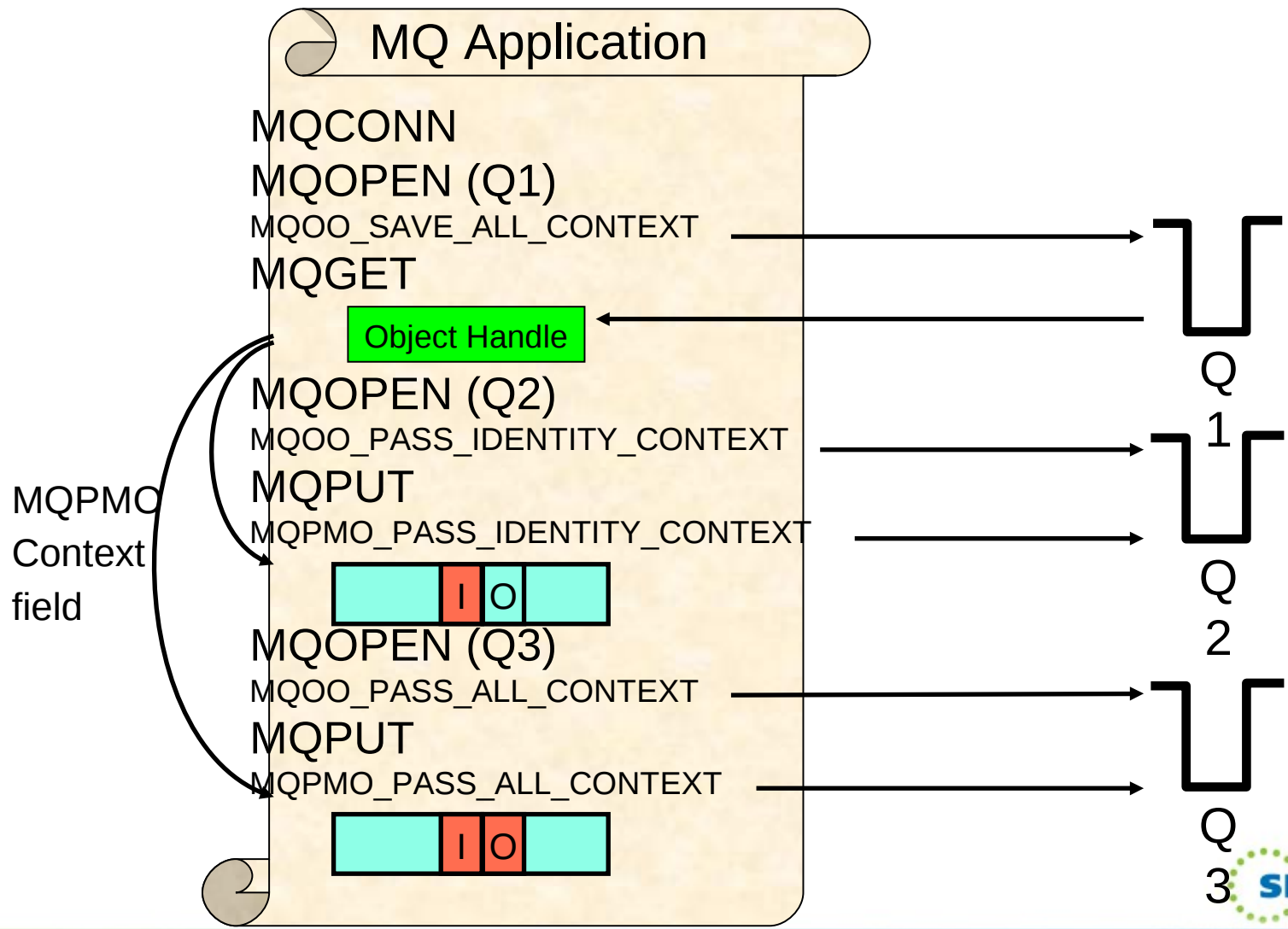


API Security - Setting Context - Notes

MQSUB

- If you want to be able to set the identity context associated with a subscription then the option need is:
 - MQSO_SET_IDENTITY_CONTEXT on the MQSUB of a topic
- Appropriate authority is required in order to do this.
- This will result in a the related context information held in the subscription being put into the message that is published to that subscription and placed on its destination queue
- If an MQSUB has been carried out without this option then the subscription will hold default context information and this is passed onto the publication message for this subscription.

API Security - Passing Context



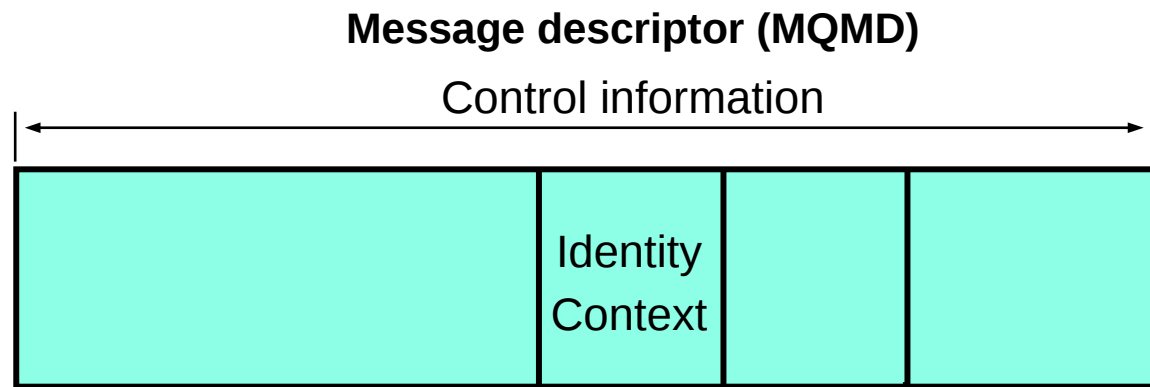
API Security - Passing Context - Notes

MQOPEN and MQPUT options

- If you want to be able to pass the identity context of a message then the options you need are, on the MQOPEN of the queue for which you are going to do the destructive MQGET (pass context cannot be used with browse):
 - MQOO_SAVE_ALL_CONTEXT
- For the queue you are opening to put the message, the options you need are:
 - MQOO_PASS_IDENTITY_CONTEXT or
 - MQOO_PASS_ALL_CONTEXT
- depending upon whether you are passing just the identity context or all the context.
- For the put of the message you need to put the object handle of the message you retrieved using the MQGET into the MQPMO context field for the message you are going to put, and you also need on the options for the MQPUT of the message to the queue, either:
 - MQPMO_PASS_IDENTITY_CONTEXT or
 - MQPMO_PASS_ALL_CONTEXT
- Appropriate authority is required to be able to do any of these.

Context - Security Checking

- User Identifier
 - Originating userid
 - Alternate Userid
- Accounting Token
 - Windows SID
- Report messages
 - Expiry
 - COD
 - Use UserIdentifier



- Identity Context
 - **UserIdentifier**
 - **AccountingToken**
 - **ApplIdentityData**

Context - Security Checking - Notes

- We will now take a brief look at what context information can be used for security checking.
- The User Identifier field in the MQMD either contains the Originating user ID if no context changes (as described in the previous foils) have been made, or it contains an Alternate user ID which was filled in by a set context, for example.
- The Accounting Token field in the MQMD can contain the Windows Security Identifier (SID) to uniquely identify the user Identifier.
- When generating some report messages (expiry and COD) the user ID of the putting application of the original message cannot be used because it is no longer available (it can be used for COA because that report message is still within the applications UOW). In these cases, the user ID in the context information for the original message is used instead.

WebSphere MQ Security

- Identification
 - O/S User IDs
 - Context
 - Link-level considerations (later)
- Authentication
 - O/S Logon
 - MQCONN
 - Link-level considerations (later)
- Access Control
 - Link-level considerations (later)
- Auditing
- Confidentiality
- Data Integrity
- Non-Repudiation

Auditing

z/OS Platform

- Standard External Security Manager (ESM) facilities, to record
 - changes to security profiles and access to them
 - failed accesses to resources controlled by those profiles
 - successful accesses to resources controlled by those profiles
- Reslevel audit records
 - RACROUTE REQUEST=AUDIT
- Controlled via
 - ZPARM: RESAUDIT(YES|NO)
- IMS Bridge audit records
 - RACROUTE REQUEST=AUDIT

Distributed Platforms

- MQRC_NOT_AUTHORIZED events
 - written to
SYSTEM.ADMIN.QMGR.EVENT
queue
- Type 1: MQCONN
- Type 2: MQOPEN/MQPUT1
 - MQPUT1 ==> MQOPEN
- Type 3: MQCLOSE
 - For deletion of dynamic queues
- Type 4: Commands
 - WebSphere MQ PCF commands
- Type 5: MQSUB
 - subscribe check failed
- Type 6: MQSUB
 - destination queue check failed

Auditing - Notes

z/OS Platform

- When using the WebSphere MQ for z/OS queue manager, you can use the standard External Security Manager (ESM) facilities to create an audit trail for any changes made to your security set up.
- This can be set up to do any / all of the list shown depending on the ESM.
- In addition to the standard ESM facilities, there are two other types of audit records written. Due to the different way the enquiry is made to RACF, normal RACF audit records are not written so MQ requests a general audit record is written for these two types.
- Whether these RACF audit records are written for RESLEVEL security checks is controlled by ZPARM RESAUDIT(YES|NO).
- These RACF audit records for the IMS bridge cannot be turned off.

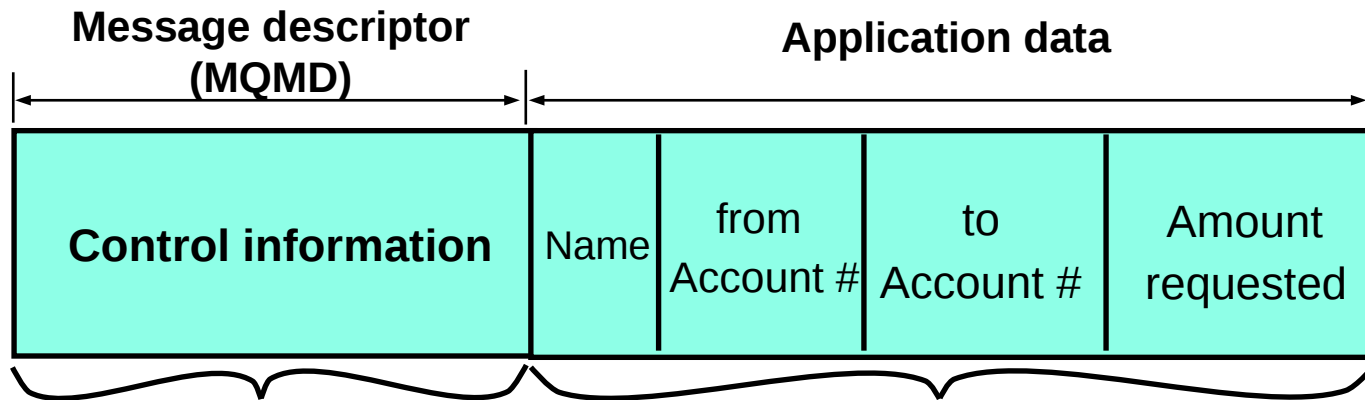
Distributed Platforms

- On the non z/OS platforms, an audit trail of access failures is kept by means of event messages which are written to the SYSTEM.ADMIN.QMGR.EVENT queue. There are several different types of MQRQ_NOT_AUTHORIZED events showing specifically what kind of access was attempted. Each of these types has a different reason qualifier recorded in the event message.
 - MQRQ_CONN_NOT_AUTHORIZED
 - MQRQ_OPEN_NOT_AUTHORIZED
 - MQRQ_CLOSE_NOT_AUTHORIZED
 - MQRQ_CMD_NOT_AUTHORIZED
 - MQRQ_SUB_NOT_AUTHORIZED
 - MQRQ_SUB_DEST_NOT_AUTHORIZED
- and, where applicable, there is information in each event message to show the user ID and application that made the failed access attempt.

WebSphere MQ Security

- Identification
 - O/S User IDs
 - Context
 - Link-level considerations (later)
- Authentication
 - O/S Logon
 - MQCONNX
 - Link-level considerations (later)
- Access Control
 - Link-level considerations (later)
- Auditing
- Confidentiality
 - Application Level Security
 - Link-level considerations (later)
- Data Integrity
 - Application Level Security
 - Link-level considerations (later)
- Non-Repudiation
 - Application Level Security

What is a message?

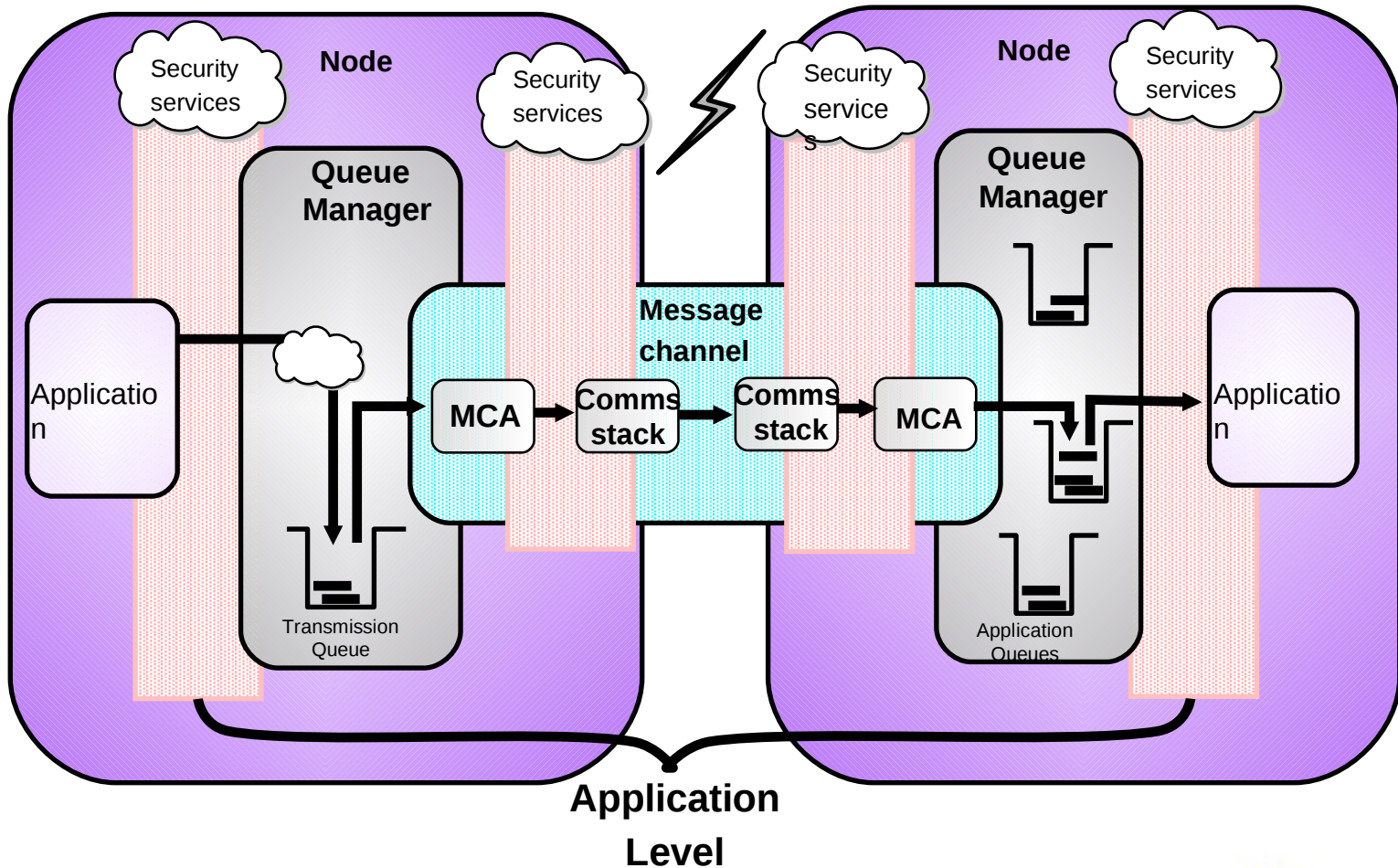


- Contains things like
 - Type of message
 - Identifier for message
 - Context information
- Contains your data
 - Anything you want to send

What is a message? - Notes

- A message in WebSphere MQ is merely a sequence of bytes in a buffer of a given length. The current products support up to 100MB in a single message although the vast majority of messages are in the order of a few thousand bytes.
- Messages have various attributes associated with them, which are contained in the header called the Message Descriptor (MQMD), such as their format and their identifier. This header also contains information detailing where and who the message came from. This information is known as the context information and we will look at it in more detail later.
- The Application data follows the Message Descriptor header. This is your information that you want to pass in a message - this could be information that
 - Is not private or confidential , and that you don't really care if anyone sees it, for example an update to a public notice board for train arrival times, so you might want to control access to it but not necessarily want to do anything else with it.
 - Is very private and confidential, and you do want to protect it, so you might want to control access to it and also want to protect it whilst in transit and when stored.

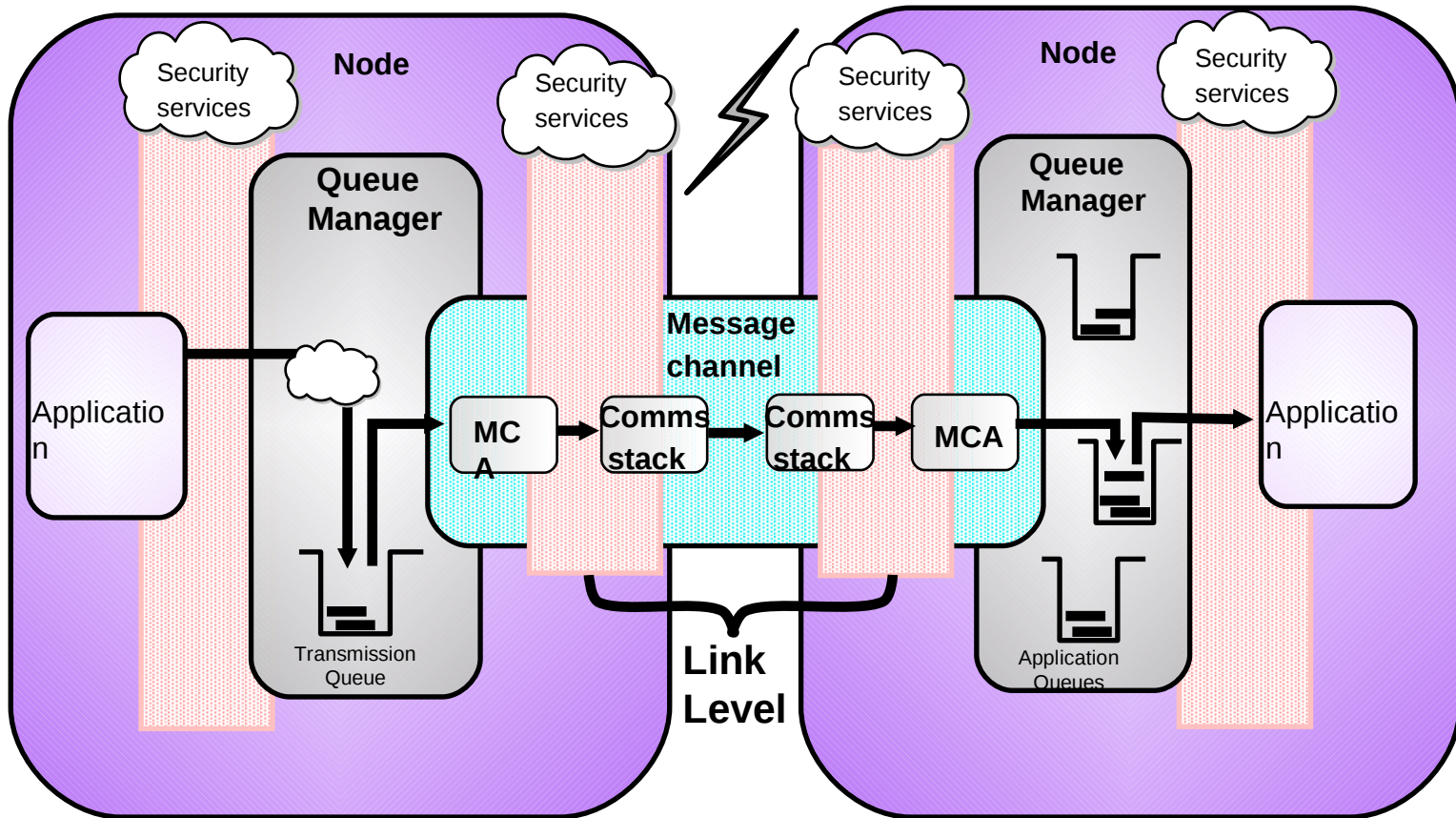
Application Level Security



Application Level Security - Notes

- Application level security (also known as end-to-end security or message level security) fits into the picture at the interface between the application and the queue manager. One example of a service provided by application level security is queue level encryption.
- The application is unaware of the service and so the application programmer need not worry about coding it into his application, however, before the message is even placed on the queue it can be encrypted, thus ensuring that it's contents are never exposed. The message is encrypted while it resides on the queue, while it is transported across the network - the channels are unaware that the content is encrypted since they are content agnostic anyway - and is still encrypted when it is placed on the target queue. At the point where the receiving application gets the message off the queue the application level security service decrypts the data and presents it to the application.
- Application level security facilities such as message level encryption for confidentiality purposes can be achieved with the WebSphere MQ Advanced Message Security (AMS), with API wrappers, or with an API Exit. API Exits allow various vendors to provide different offerings with these facilities.

WebSphere MQ Security - Link Level Security



Link Level Security – Notes

- This diagram illustrates what we mean by Link Level security. We have security facilities which operate on the data which flows over the wire (i.e. the link). The services have no effect on messages when they are at rest on queues in between the links, and if there are multiple hops through various queue managers before the message reaches its final destination, then these security services will be applied multiple times, once on each link.

WebSphere MQ Security - Link Level Security

- Identification
 - Message context
 - Channel Authentication Records
 - Security Exits
- Authentication
- Access Control
- Confidentiality
- Data Integrity

Link Level Security - Notes

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.
- Messages that flow from a remote queue manager already contain identity context inside the message. Later we will see how this identity can be chosen as the identity for access control.

Authentication

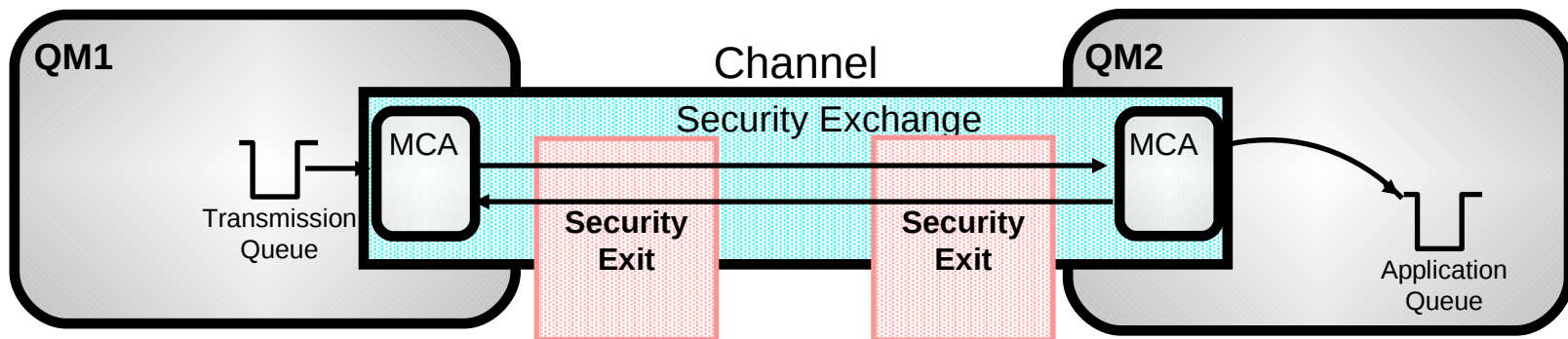
- Authentication is the way in which a channel ensures that the other end of the channel is who they say they are. Channels can make use of SSL to authenticate a digital certificate sent by the partner. In WebSphere MQ V7.1 Channel Authentication Records can be used to do many of the jobs a security exit can do, such as allowing or blocking a channel based on IP address, SSL Peer Name, 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 on the channels.

Link Level .. Identification and Authentication

- Security Exits - Channel 'Gate Keeper'
 - Indefinite exchange of data between exits
 - No defined format
 - No communications knowledge required
- Can end channel
 - Can set MCAUSER



Link Level .. Identification and Authentication - Notes

- One of the problems with authentication is that the industry can 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 and Public/Private key encryption. 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.

Link Level .. Identification and Authentication

- 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 Peer Name matching



Link Level .. Identification and Authentication

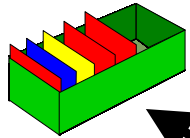
- Rules can use any of the following identifying characteristics of the inbound connection
 - IP Address
 - SSL/TLS Subject's Distinguished Name
 - Client asserted user ID
 - Remote queue manager name



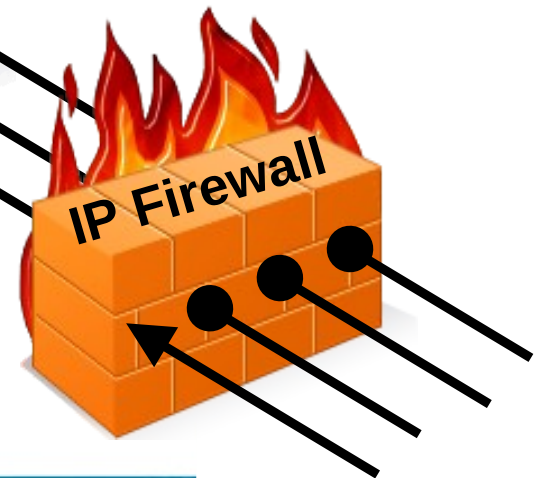
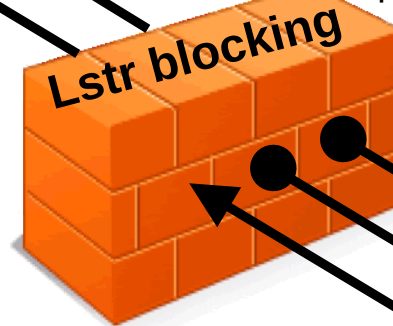
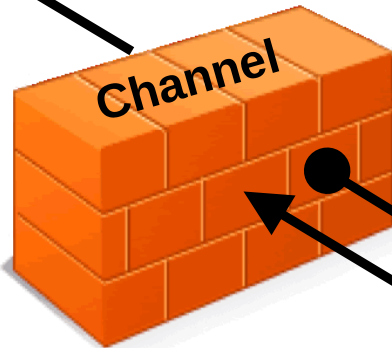
Channel Authentication Records – Notes

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

Channel Access Blocking Points



ACLs



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

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

Channel Access Blocking Points – Notes

- In this picture we illustrate that there are a number of points that an inbound connection must get through in order to actually make use of an MQ queue.
- First, we remind you that your IP firewall is included in this set of blocking points and should not be forgotten, and is not superseded by this feature in MQ.
- One point of note, the inbound connections can be from any version of MQ. There is no requirement that the clients or remote queue managers also be on WebSphere MQ V7.1 to be blocked or mapped by these rules.

Channel Access Blocking Points – Notes

- Second, there is a list of IP addresses that the listener configuration will have access to. If any of these IP addresses attempts to start an inbound channel connection, the listener will bounce the connection prior to starting a channel instance to process any data, for example SSL Handshake, that might need to be read before knowing what the channel name is. If the queue manager is not running it will still have access to the configuration and it will still block the specified IP addresses. **THIS IS NOT A REPLACEMENT FOR AN IP FIREWALL!** However, it does provide a way for an MQ Administrator to implement temporary blocking until the IP firewall updated, or for a short period of time making it not worthwhile to update the IP firewall. The intention is that there shouldn't be many entries in the list.
- A Denial of Service (DoS) attack on the listener; whilst really the place of the firewall to deal with; would mean high CPU in the listener if it had to deal with a repeated connection from a inbound connection. This and the fact that we would like to quietly ignore network pingers if they don't send any data and only raise blocking events on them if they do send data, means that the listener will hold any sockets that come from blocked IP address open for a period of time prior to closing the socket, rather than immediately rejecting it. This will stall the repetitiveness of the attack and protect the listener process allowing it some time to process real requests, and additionally give the network pingers time to close the socket before we do allowing us to detect we don't need to emit an event for that case. By default this time will be 30 seconds.
- Thirdly we come to the rules that work on specific channels. You can set up rules to match against all of the identifying characteristics of an inbound channel (see next notes page). These rules can either indicate that a channel matching the rule should be blocked; should be allowed and assigned a provided user ID to use when it runs; or allowed and the user ID provided by the channel is to be used.

Channel Access Blocking Points – Notes



IP Address

- Rules can be made to be used should a connection arrive from the specified IP address.
- If the client asserted a banned user ID, but its IP address is in this list to map it to another user ID it will be deemed to have asserted the mapped user ID and not the banned one, so the problem of an older Java client sending up blank can be fixed by mapping and it doesn't have to end up banned. In other words mapping happens before the blocked list of user IDs is checked. If you have a port forwarder, DMZ session break, or any other setup which will change the IP address presented to the queue manager, then mapping IP addresses is not necessarily suitable for your use.
- The patterns that can be used to specify IP addresses are described later. Additionally, we will find the most specific rule in order to do the mapping, so an additional pattern is allowed in this configuration – that is a single asterisk which means 'match everything'.

SSL/TLS DN

- Rules can be made to be used should a connection arrive from the specified DN.
- If the client asserted a banned user ID, but its DN is in this list to map it to another user ID it will be deemed to have asserted the mapped user ID and not the banned one in the same way as described above.
- DNs can be provided with pattern matching in the same way as the SSLPEER attribute that already exists. That defines the match everything pattern to be "CN=*". As for IP addresses, we will find the most specific duplet in order to do the mapping; this requires us to define a precedence order of substrings in the DN in order to decide which to use if we have two DN patterns that both match the full DN. An example will follow.

Remote Queue Manager Name

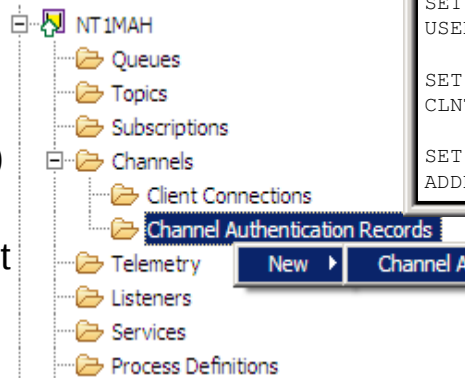
- Rules can be made to be used should a connection arrive from the specified remote queue manager.
- There is also pattern matching on queue manager names, although it is by its very nature, much simpler patterns than IP addresses or SSL Peer Names.

Client asserted User ID

- Rules can be made should a connection arrive asserting the specified user ID. It is often likely that the client asserted user ID is not even defined on the server system, so this can be used to map it to an MCAUSER user ID which of course will be a server side defined user ID.
- If the client asserted a banned user ID, but that user ID is in this to map it to another user ID it will be deemed to have asserted the mapped user ID and not the banned one in the same way as the other mappings.
- There is a list of user IDs which, if any of them are asserted, will cause the channel to end. The checking against this list of blocked user IDs runs after all things that can set the MCAUSER have completed, including security exits and the MCAUSER mapping feature we just spoke of. *MQADMIN is a recognition that we logically want to ban the 'mqm' user if you like. However, this is actually a differing set of user IDs dependant on platform, so we will discover at run-time whether we have a privileged user, and if *MQADMIN is blocked, then privileged users are blocked.

Channel Authentication Records – Configuration

- Create rules using
 - MQSC: SET CHLAUTH
 - PCF
 - MQ Explorer GUI Wizard
- Pattern matching
 - Channel Name
 - Beginning, middle, end
 - IP addresses (IPV4 or IPV6)
 - '*' in any segment
 - '-' ranges in any segment
 - SSL Peer Name (as today)
 - QMgr Name – as channel name
- Restricting rules further by IP Address
 - Rules matching on
 - SSL Peer Name
 - Remote QMgr Name
 - Client User ID
 - Can add IP address



```

C:\ Command Prompt - runmqsc TEST1

Starting MQSC for queue manager TEST1.

SET CHLAUTH('*') TYPE (ADDRESSMAP) ADDRESS('*') USERSRC (NOACCESS)


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

SET CHLAUTH('SYSTEM.ADMIN.*') TYPE (SSLPEERMAP) SSLPEER('O=IBM')
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('CN="Morag Hughson"')
ADDRESS ('9.*') MCAUSER ('hughson')
  
```



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

- Precedence matching
 - Most specific rule is matched
 - Within SSL Peer Name matching
 - Most specific substring is matched

Order	Identity mechanism
0	Channel Name
1	SSL Distinguished Name
2=	Client asserted User ID
2=	Queue Manager Name
4	IP address

Channel Authentication Records - Examples

- SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS)
- SET CHLAUTH('*') TYPE(ADDRESSMAP) ADDRESS('9.20.1-3.*')
USERSRC(CHANNEL)
- SET CHLAUTH('SYSTEM.ADMIN.*') TYPE(SSLPEERMAP) SSLPEER('O=IBM')
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('CN="Morag Hughson"')
ADDRESS('9.*') MCAUSER('hughson')

New Channel Authentication Record

Summary

Channel authentication rule summary and command preview.

Press the finish button to save this rule in a channel authentication record in the queue manager.

Settings to use to create the new channel authentication rule:

Create a rule which applies to channels whose names match the pattern "SYSTEM.*".

Allow inbound connections from SSL/TLS Distinguished Names which match pattern "CN=Morag, OU=Devt,O=IBM".

Assign a user ID of "hughson" for access control checks for these connections.

Command preview:

```
SET CHLAUTH('SYSTEM.*') TYPE(SSLPEERMAP) SSLPEER('CN=Morag,
OU=Devt,O=IBM') USERSRC(MAP) MCAUSER('hughson') ACTION(ADD)
```

? < Back Next > Finish Cancel

Channel Authentication – Configuration – 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. Additionally, the MQ Explorer GUI provides a wizard to walk you through the steps for setting up these rules and at the end of the wizard, the MQSC command that would do the same job as you have done in the wizard, is displayed in a window that you can cut'n'paste from to put the command into a script for future use.
- Some of these examples illustrate the pattern matching that can be applied to channel names, IP addresses, 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.
- 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.
- 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 IP addresses are considered the least specific since clearly more than one queue manager or client can be connecting from the same IP address.

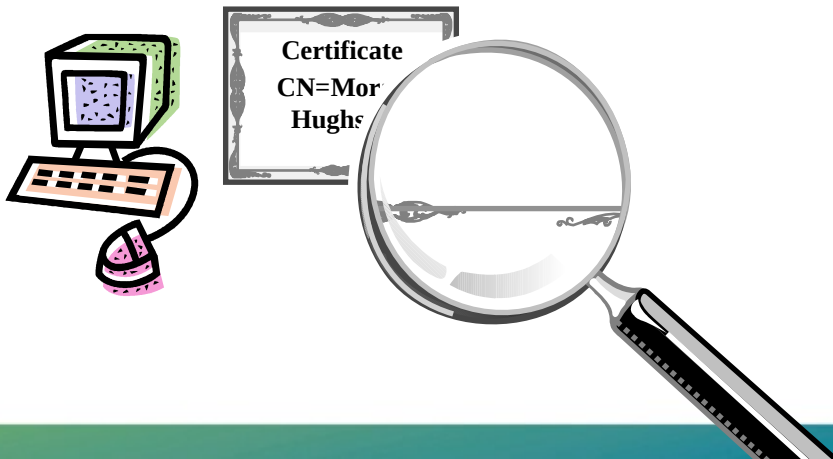
SSL DN Precedence Mapping Example

```
SET CHLAUTH(*) TYPE(SSLPEERMAP) SSLPEER('OU="MQ
Dev"') 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
Dev.
O=IBM UK.L=Hursley.C=UK

SSL DN Precedence Mapping Example – Notes

- 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, search the MQ Information Centre for “Distinguished Names”.

What happens if...?

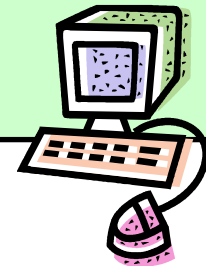
```
DISPLAY CHLAUTH(SYSTEM.ADMIN.SVRCONN) MATCH(RUNCHECK)
        SSLPEER('CN="Morag Hughson", O="IBM UK"')
        CLNTUSER('mhughson')
        ADDRESS('9.180.165.163')
```

returns ==>

```
CHLAUTH(SYSTEM.ADMIN.SVRCONN)
TYPE(ADDRESSMAP)
ADDRESS('9.180.165.163') MCAUSER(MORAG)
```

Order	Identity mechanism
0	Channel Name
1	SSL Distinguished Name
2=	Client asserted User ID
2=	Queue Manager Name
4	IP address

```
Chl: SYSTEM.ADMIN.SVRCONN
DN: CN=Morag Hughson.O=IBM UK
UID: mhughson
IP: 9.180.165.163
```



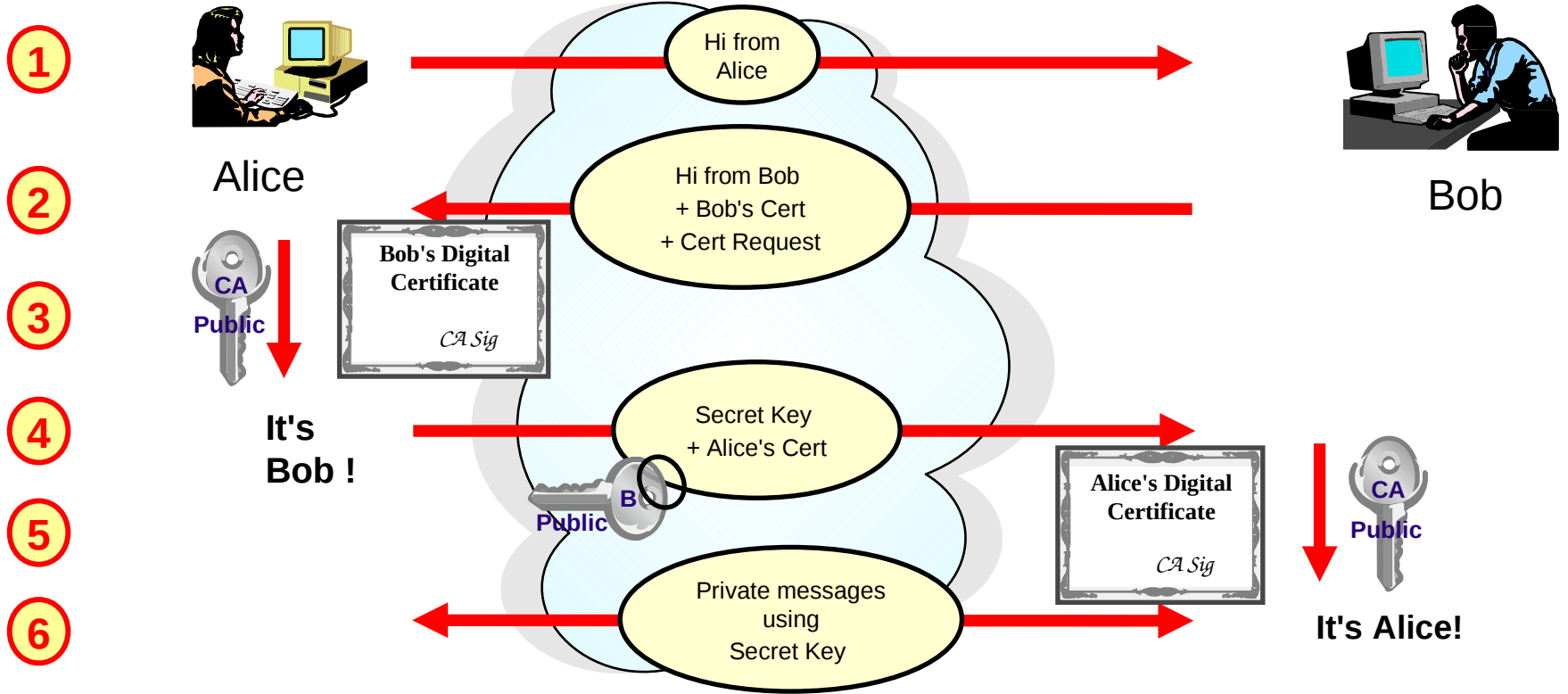
What happens if...? - Notes

- Here is an example of the special matching version of the DISPLAY command to show exactly what would happen should a channel matching these identifying attributes, connect into the system. This should serve as a useful testing tool, service aid, and validation tool, although we would of course recommend not creating such complicated rules that you need it in the first place!

WebSphere MQ Security - Link Level Security

- Identification
 - Message context
 - Channel Authentication Records
 - Security Exits
- Authentication
 - Secure Sockets Layer (SSL)
 - Channel Authentication Records
 - Security Exits
- Access Control
- Confidentiality
- Data Integrity

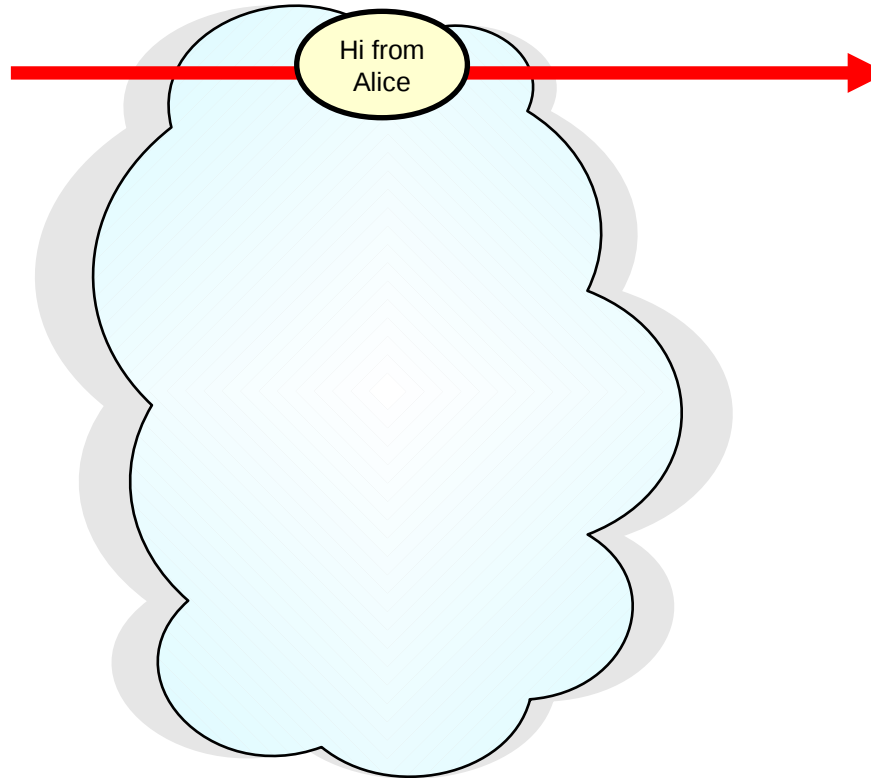
SSL Handshake



SSL Handshake



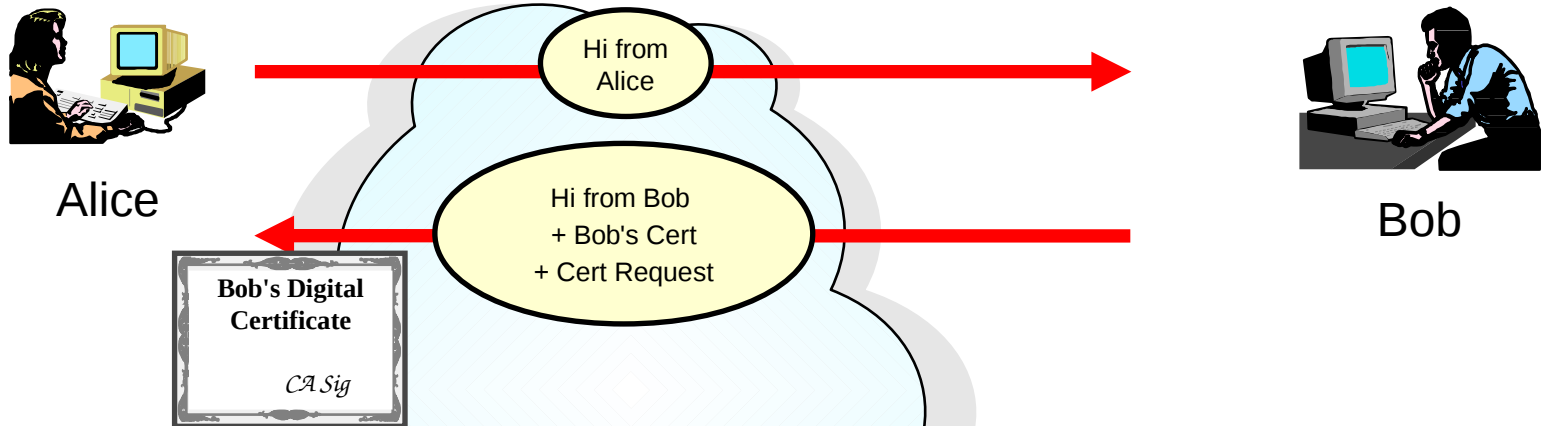
Alice



Bob

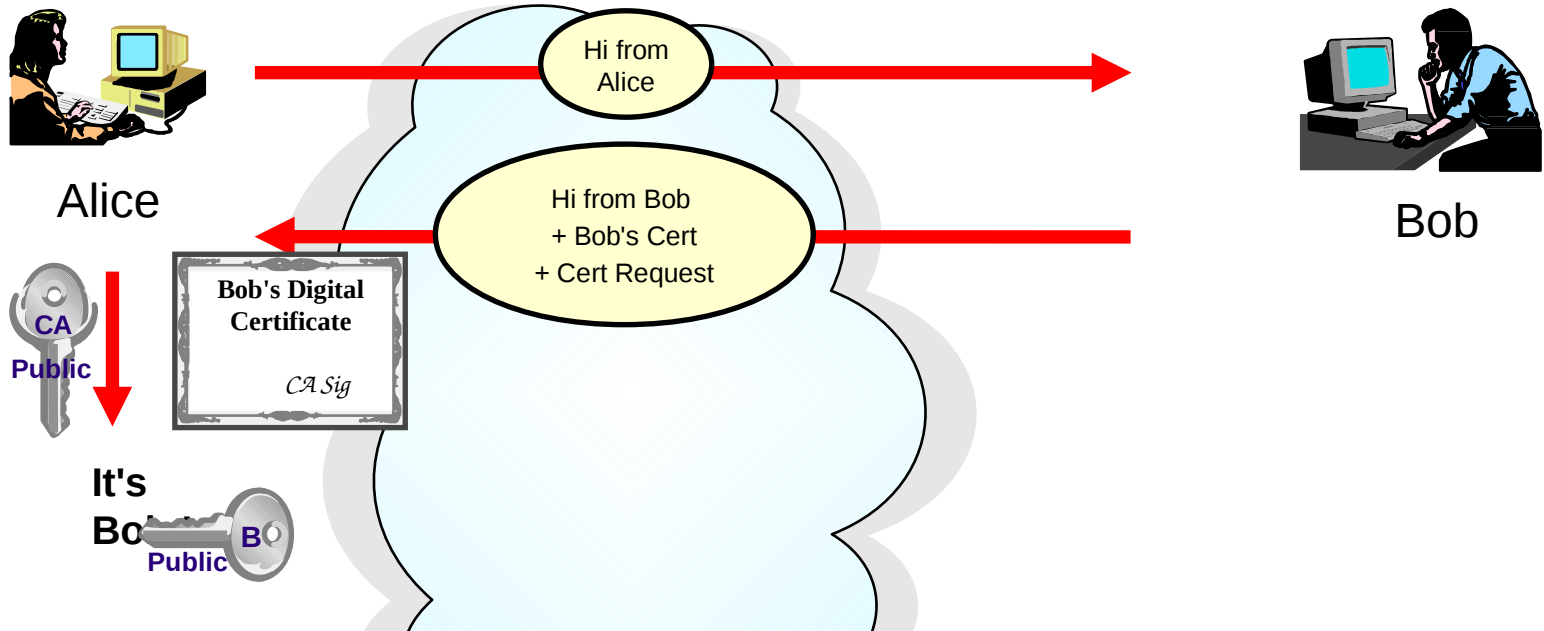
- The 'Client Hello'
 - Alice sends Bob some random text
 - Also sends what CipherSpecs and compression methods she can use
 - Alice is considered the client since she started the handshake

SSL Handshake



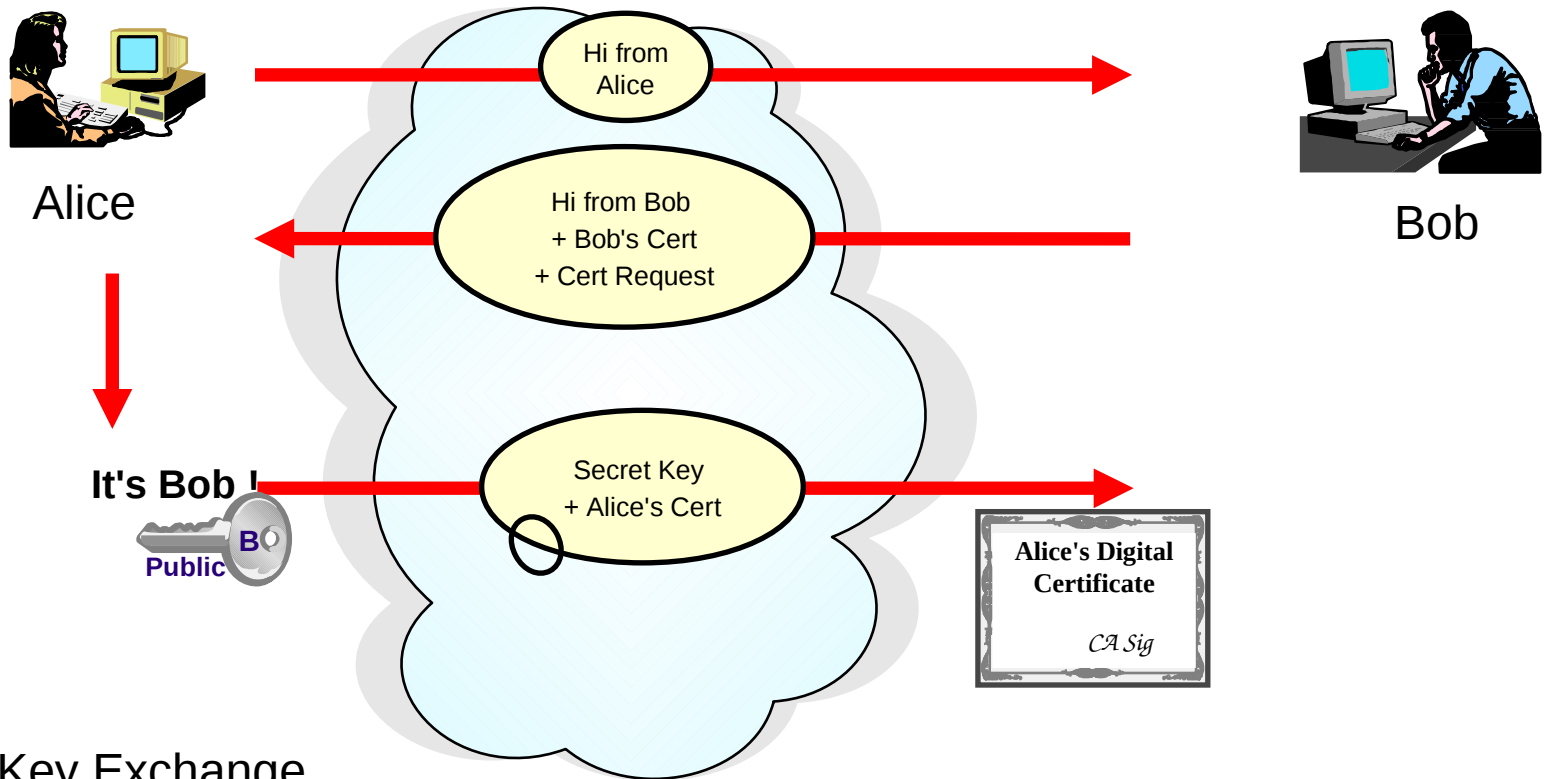
- The 'Server Hello'
 - Bob sends Alice some random text
 - Bob chooses the CipherSpec to be used, from Alice's list
- The Server Certificate
- The Client Certificate Request

SSL Handshake



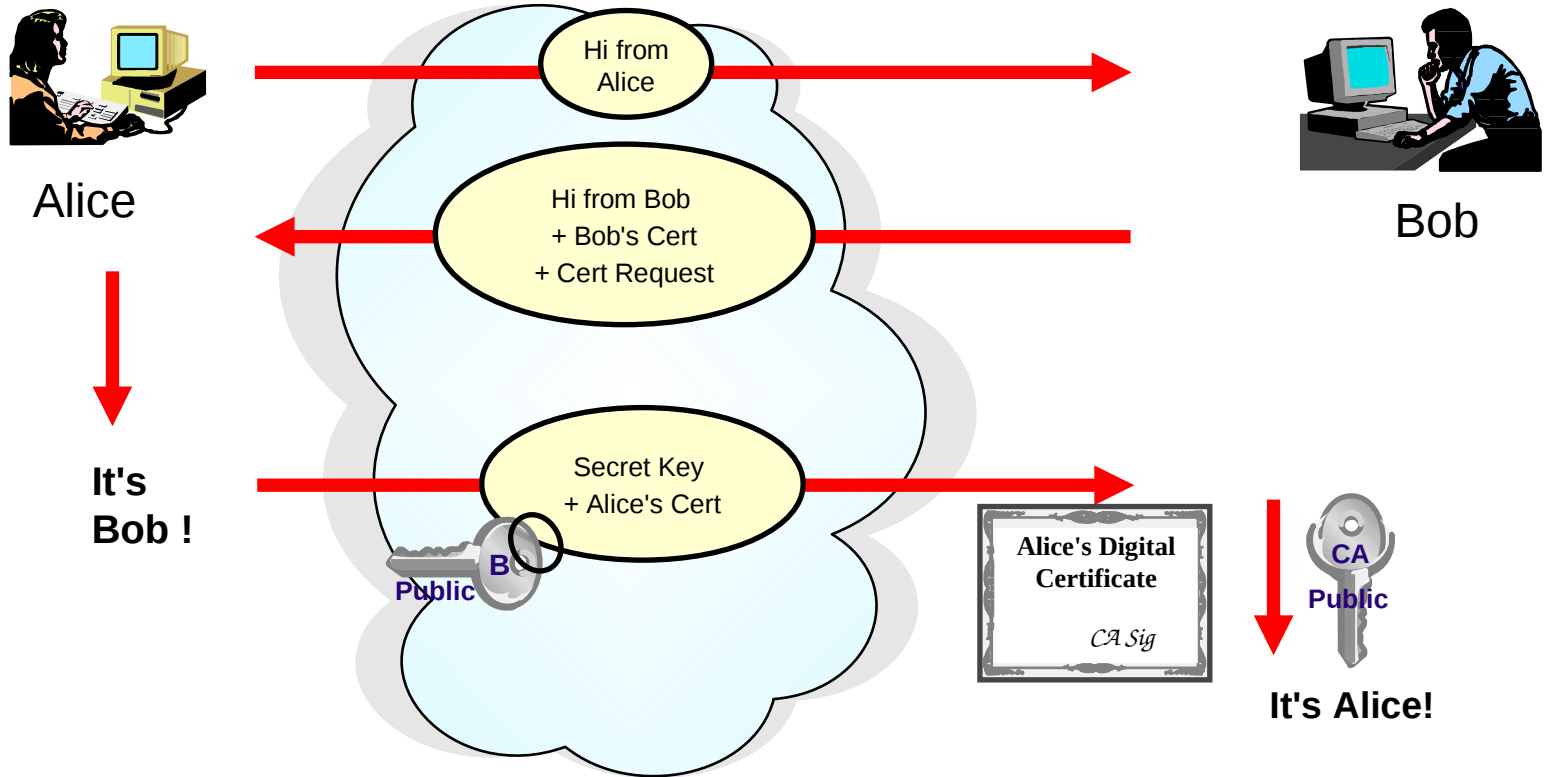
- Verify Server Certificate
 - Check Validity Period
 - Decrypt using CA's Public Key - verifies that certificate is trusted
 - Check Domain Name and/or Distinguished Name
- Also receives Bob's Public Key

SSL Handshake



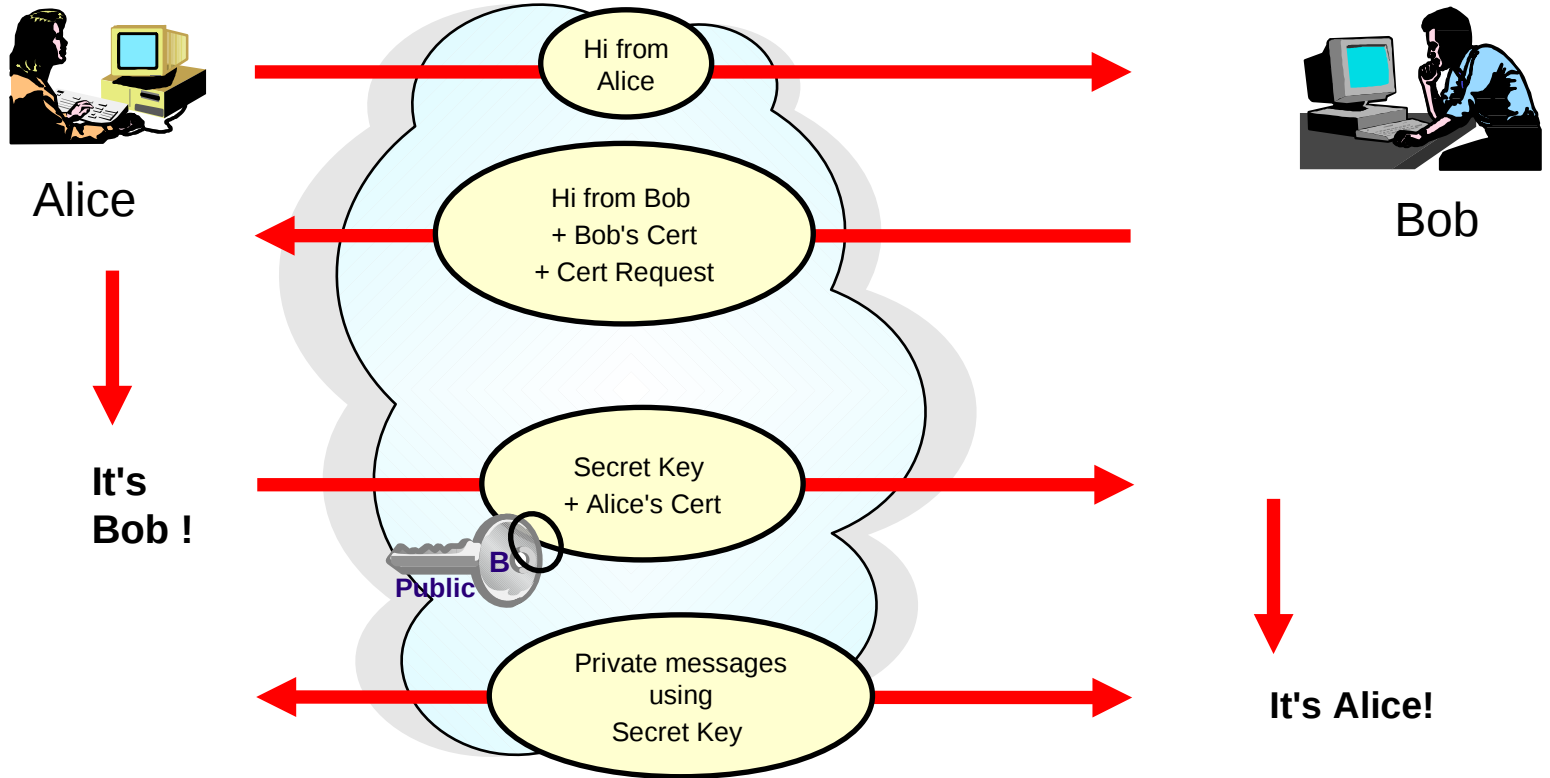
- Client Key Exchange
 - Alice sends Bob the Secret Key to use
 - This is encrypted with Bob's Public Key
 - Also sends her certificate

SSL Handshake



- Verify Client Certificate
 - Decrypt using CA's public Key

SSL Handshake



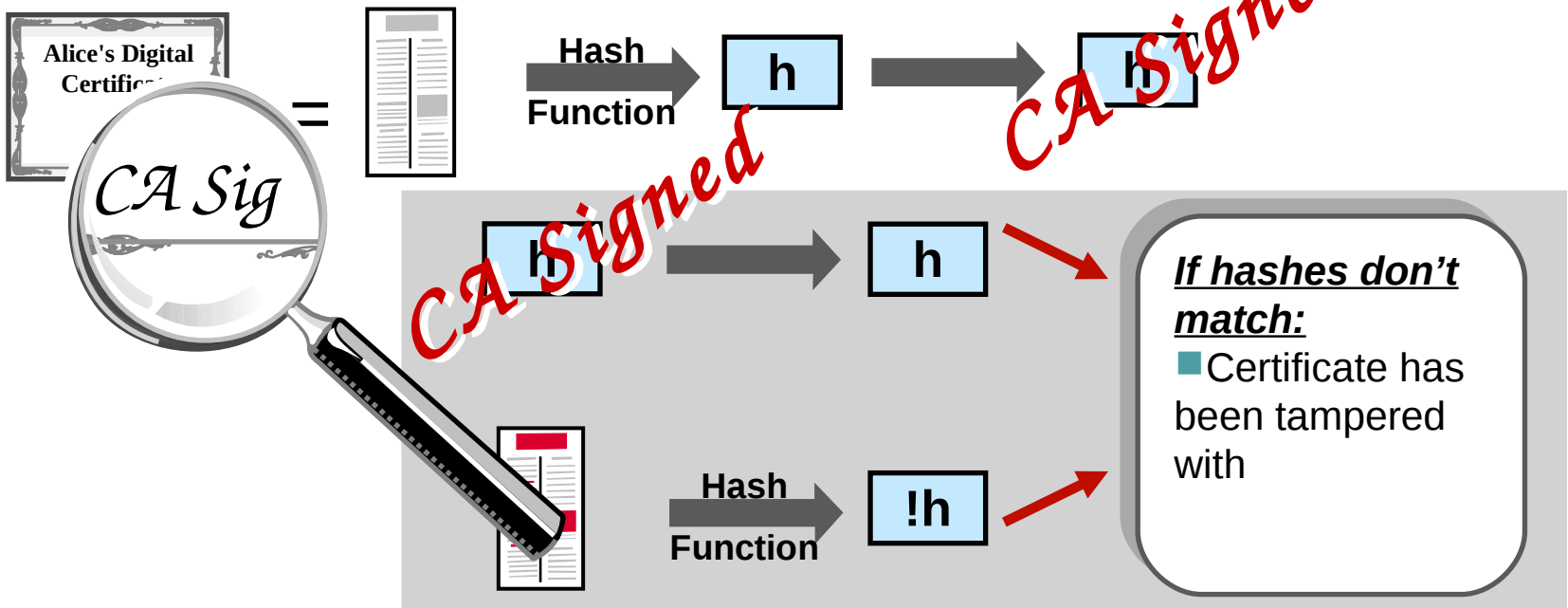
- Send Information using agreed Secret Key
 - Randomly generated 1-time key
- This is now a 'secure line'

SSL Handshake

1. The 'Client Hello'
 - Alice sends Bob some random text, she also sends what CipherSpecs and compression methods she can use. Alice is considered the client since she started the handshake.
1. The 'Server Hello'
 - Bob sends Alice some random text and chooses the CipherSpec to be used, from Alice's list. He will also send over his certificate - the Server Certificate and may optionally request that Alice sends him her certificate - the Client Certificate Request.
1. Verify Server Certificate
 - Alice will verify Bob's certificate is valid by checking this digital signature made by the CA (see next foil for more details).
1. Client Key Exchange
 - Alice builds the Secret Key and sends it to Bob to use in a message encrypted with Bob's Public Key. This means that only Alice who invented the Secret Key and Bob who can decrypt the message containing it, know the Secret Key. Alice also sends her certificate (since Bob requested it).
1. Verify Client Certificate
 - Bob will verify Alice's certificate is valid by checking this digital signature made by the CA
1. This is now a 'secure line'
 - Bob and Alice can now send Information using agreed Secret Key which is a randomly generated 1-time key just used for this session.

Trusting a Digital Certificate

- Digital Certificate = Plaintext
 - Can be subject to tampering
 - Signed by CA at creation



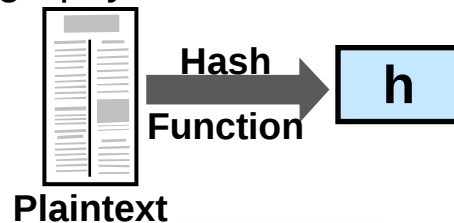
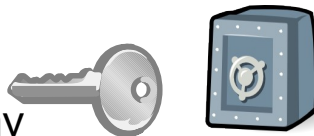
- CA's Digital Signature
 - Allows tampering to be detected

Trusting a Digital Certificate - Notes

- Digital signatures combine the use of the one-way hash function and public/private key encryption.
- The message is hashed to provide a number, the hash number or message digest. This hash number is encrypted using the sender's private key to create the digital signature. The recipient of the message can also hash the message to get a hash number, and can decrypt the digital signature using the sender's public key, to get her hash number. If these numbers match then the message did come from the sender and also we know it hasn't been changed since it was signed.
- A digital certificate can simply be thought of as a piece of plaintext that could be subject to tampering. After all it is just a file on your computer. How can we detect if someone has tampered with the certificate we are going to use. This is where the CA signature comes into effect. The same technique described above is used to determine whether a digital certificate has been tampered with.
- The CA calculate the hash value of the plaintext (our certificate) and then signs that hash value with the CA private key to generate a CA digital signature. To check that the certificate is valid, the CA's digital signature can be decrypted using the CA public key (well known CA public keys are installed in many of the security products that use SSL) to check that the hash values match.

Using SSL with WebSphere MQ

- Get your certificates or Authentication
 - Digital Certificates
 - Asymmetric Keys
- Put your certificates in a place that MQ can use
- 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 for Confidentiality
 - Symmetric Key Cryptography
- ... and Data Integrity
 - Hash Function



- WebSphere MQ SSL Wizard (MO04)

SSLKEYR(QM1KEYRING)

SSLCRLNL(REVOKE.NL)

SSLFIPS(NO)
SUITEB(NONE)

New in
V7.1

SSLCIPH(RC4_MD5_US)
SSLRKEYC(999 999 999)
SSLPEER('O=IBM')
SSLCAUTH(REQUIRED)

Using SSL with WebSphere

- Get your certificates or Authentication

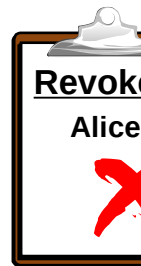
- Digital Certificates



- Asymmetric Keys

- Put your certificates in a place that MQ can use

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



- Decide if you need cipher spec restriction (FIPS or SUITEB)

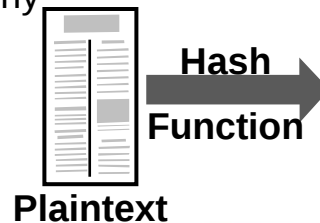
- Configure your channels to use SSL for Confidentiality

- Symmetric Key Cryptography



- ... and Data Integrity

- Hash Function



WebSphere MQ SSL Wizard

File Edit Help

Previous **Generated instructions** Save instructions

Create SSL client keyring on winmvs41

```
RACDCERT ID(MQ23USR) +  
ADDRING(MQ23RING)
```

Create SSL server key database on localhost

```
gsk7capicmd -keydb -create -db "D:\mqscripts\key.kdb" -pw serverpass -type cms  
-expire 365 -stash
```

Save the public CA certificate to the SSL client and SSL server machines

Store the public CA certificate supplied by your CA to a dataset on winmvs41.

Store the public CA certificate supplied by your CA to a file on localhost.

Link Level .. SSL - Notes

- The three main issues that SSL 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. 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).
- Digital Certificates and Public Keys are found in a key repository which can be specified to WebSphere MQ (SSLKEYR on the queue manager). We can also check that we are talking to the partner we expect to be talking to (SSLPEER on a channel) and 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).
- 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

WebSphere MQ Security - Link Level Security

- Identification
 - Message context
 - Channel Authentication Records
 - Security Exits
- Authentication
 - Secure Sockets Layer (SSL)
 - Channel Authentication Records
 - Security Exits
- Access Control
 - Put Authority (see notes for reference)
 - MCA User
 - Message Userid
- Firewalls
 - Port numbers (see notes for reference)
 - Internet passthru (SPac MS81) (see notes for reference)
- Confidentiality
- Data Integrity

Link Level .. Access Control

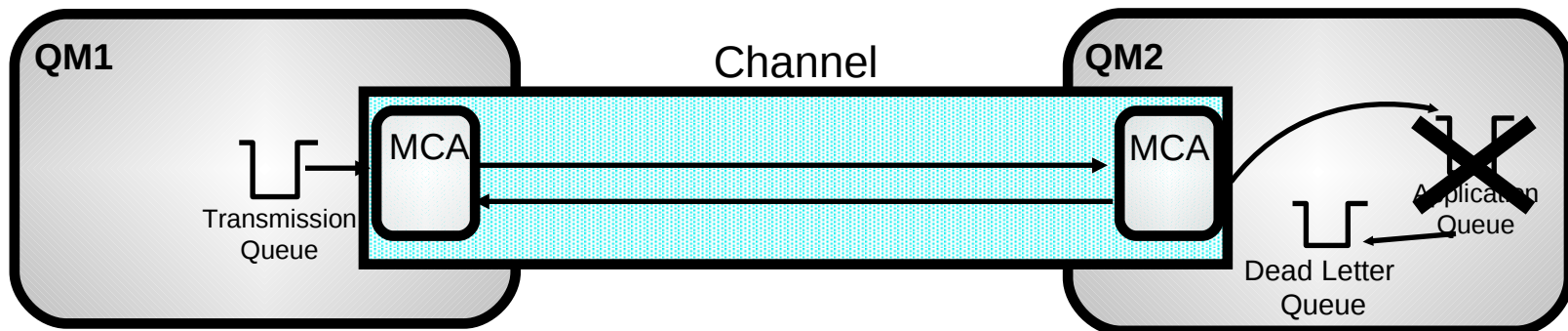
- Put Authority (PUTAUT) receiver channel attribute
 - Default - use the channels MCAUser
 - OnlyMCA - z/OS equivalent
 - Context - use the userid in the message
 - ALTMCA - z/OS equivalent
- MCAUser
 - Effective userid this channel should run under
- Normal MQ access control
- Failed puts put to Dead Letter Queue

Link Level . Access Control - Notes

- Once the security exits have authenticated with the partner, i.e. determined that the partner really is who they say they are, we then have the issue of access control. Essentially the user needs to choose between treating all messages from a remote Queue Manager as the same level of authority or setting access rights on a per message basis.
- This is controlled by the PUTAUT (put authority) attribute of a receiver type channel.
- On Distributed platforms it essentially has two values
 - Context - this tells the channel to take the userid from the Message and use that userid to perform the access control check
 - Default - this tells the channel to perform the access rights for all messages using a single userid for the channel for all messages. This userid is in the MCAUser attribute. This field can either be entered at channel definition time or can be filled in by a security exit at authentication time.
- On z/OS, more than one userid can be checked for access, and the PUTAUT attribute has more values. Default, Context, OnlyMCA and ALTMCA, The z/OS equivalents of the descriptions given above are ALTMCA and ONLYMCA respectively. Context and Default have slightly different meanings on z/OS and all the definitions can be found in the Security chapter of WebSphere MQ for z/OS System Setup Guide.
- When using CONTEXT or ALTMCA the userid of the message is used to determine access control. This means that userids may need to cross platform boundaries. In some cases this is neither practical or desirable. It may be appropriate to change the userid in the message to one suitable for use on the target system. The channel message exit is a suitable place to perform this transformation.

Dead Letter Queue

- Security Failures
 - Messages placed on Dead Letter Queue
- Access Control on DLQ
- Be careful with Dead Letter Handler

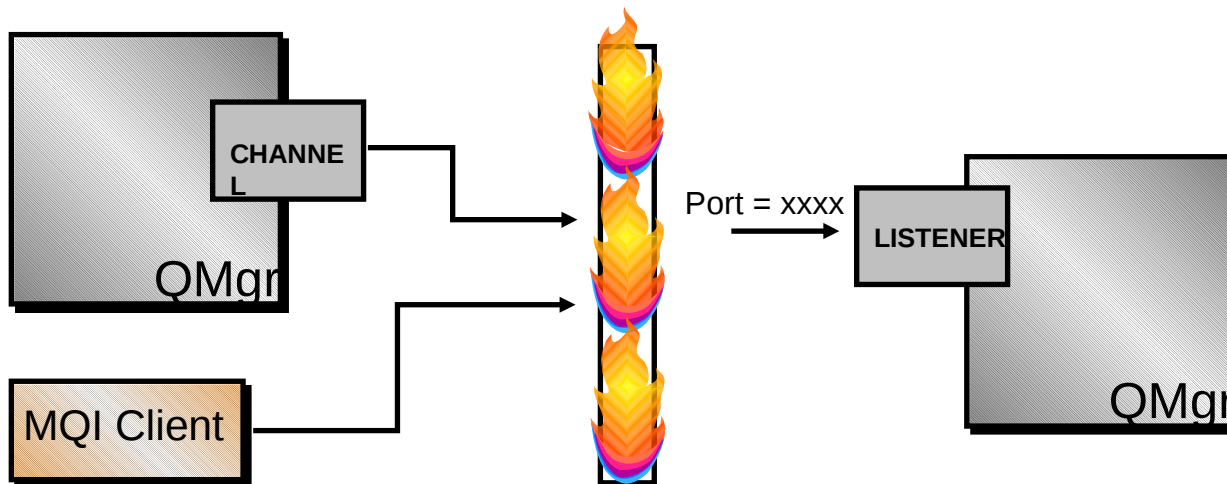


Dead Letter Queue - Notes

- Note that security failures will result in an attempt to put to the Dead Letter Queue. If the user is also not authorized to put to the Dead Letter Queue then for a recoverable message the channel will come down. It is therefore generally a good idea not to make the Dead Letter Queue access control too restrictive.
- Be careful with the process used by the Dead Letter Queue Handler. Make sure it doesn't simply re-PUT messages that failed with a security error back to the queue they were destined for. The user ID running the Dead Letter Queue Handler will probably have higher authority and so you may find that security is being bypassed by messages going via the Dead Letter Queue. Ensure security failures are processes correctly by the Dead Letter Queue Handler or even are placed on a separate queue to be dealt with by another process.

Firewalls

- Requires target port configuration
 - Configurable for the TCP Listener
- Requires source port configuration
 - Configurable for MQI Client
 - Configurable for Channel Definition
 - `DEFINE CHL(TO.NTC1) LOCLADDR((1000,2000))`

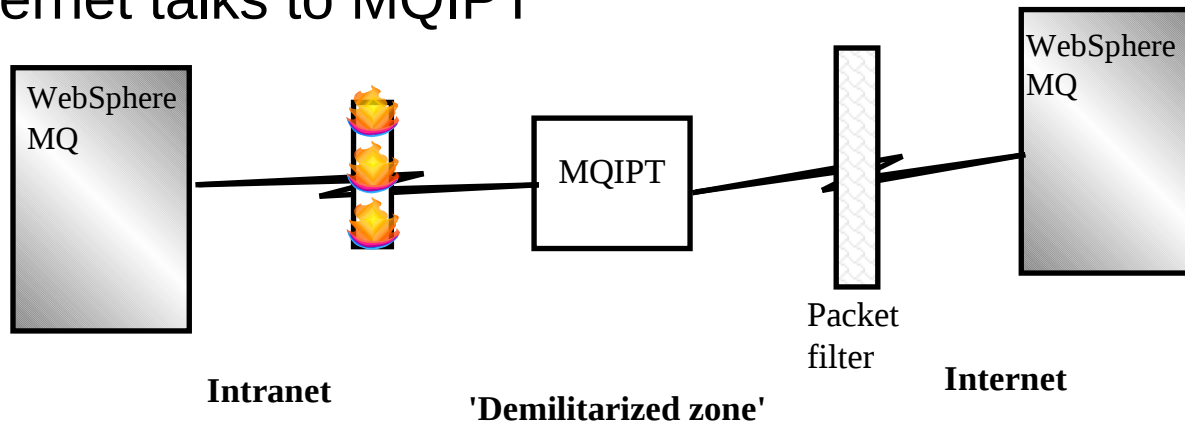


Firewalls - Notes

- To aid with the security of their installation many people install firewalls between machines which have access to machines outside the enterprise. Different firewall products implement slightly different levels of checking. Some tend to be packet based whereas others are just connection/socket based. One common factor though is the ability to open a hole in the firewall given a pair of network addresses and a pair of port numbers.
- Inside the CONNAME field of the sender channel attribute you can specify the port number of the remote partner machine (by default 1414) and in the LOCLADDR field you can specify your own local port number. By default this is chosen by the TCP/IP stack itself. However, if you're using a firewall product you probably want to control the local port number that is chosen.
- The LOCLADDR field is supported in WebSphere MQ V5.3 and later.

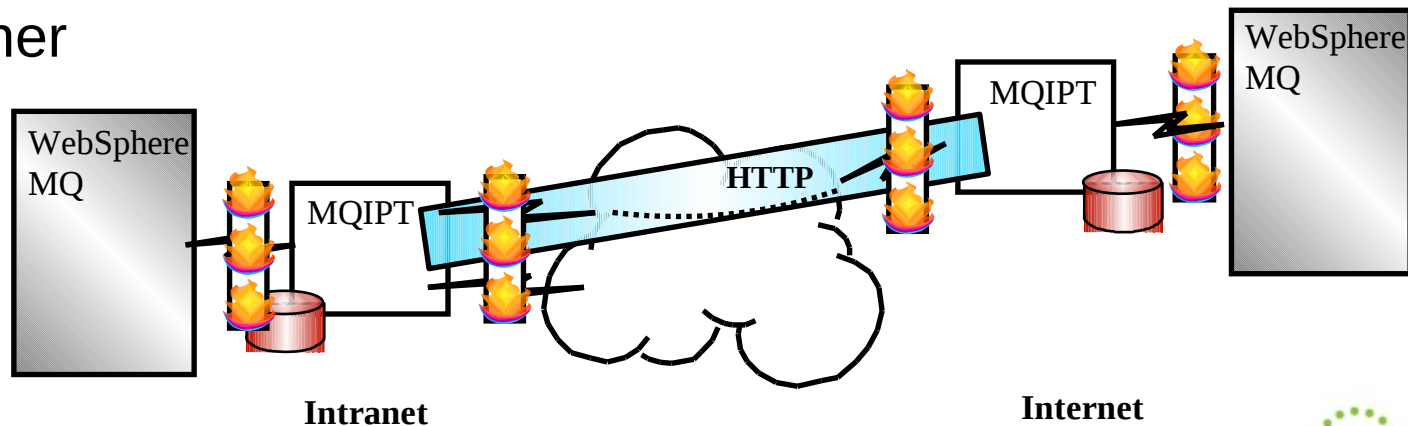
WebSphere MQ Internet Passthru (MQIPT)

- "Proxy" for MQ protocol
 - Suitable for installation in the DMZ
 - Single point of Entry
 - Protects internal IP addresses
 - Internet talks to MQIPT



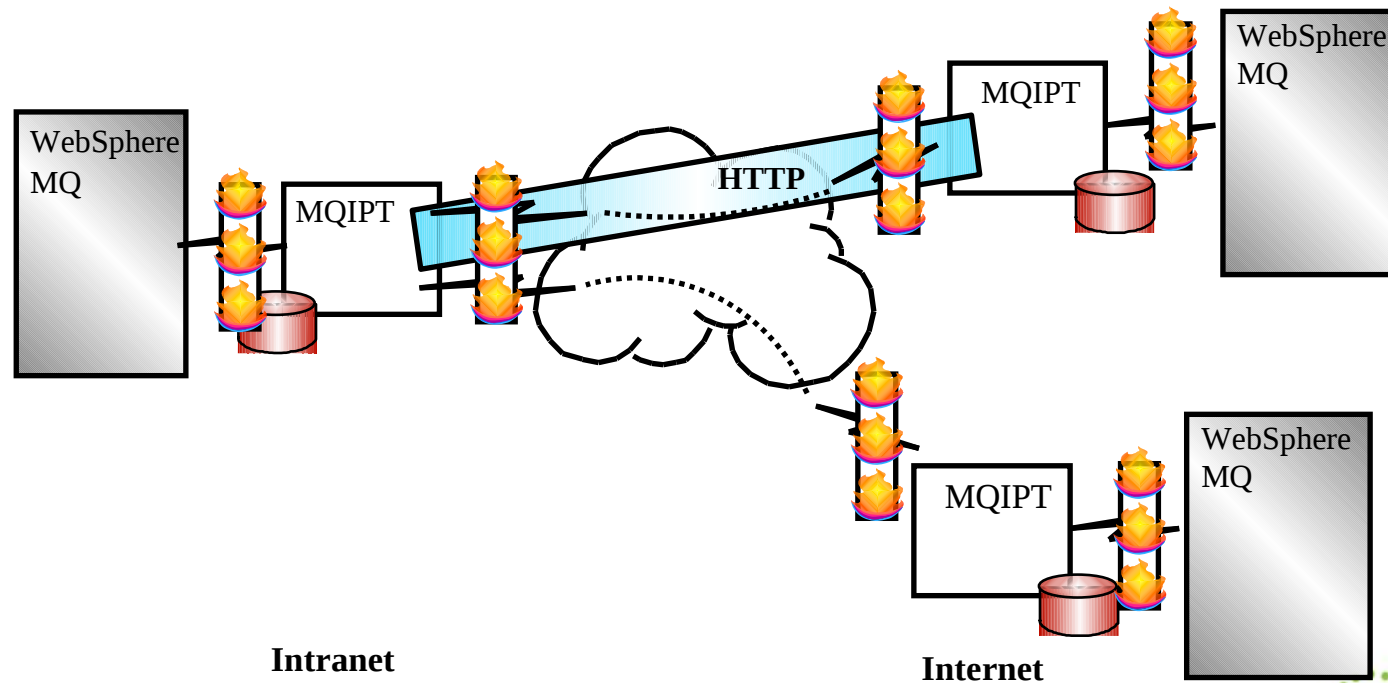
WebSphere MQ Internet Passthru (MQIPT)

- Support for http, https, socks
- Disconnected communications channels
 - Use a pair of MQIPT servers
 - allows use of HTTP wrappers
 - No direct connection between queue managers
 - A servlet version of IPT can be run on a Server end
- In SSL proxy mode , allows SSL flow to go from one end to the other



WebSphere MQ Internet Passthru (MQIPT)

- Concentrator
- Single MQIPT supports multiple targets
- Category 3 SupportPac - MS81



WebSphere MQ Internet Passthru - Notes

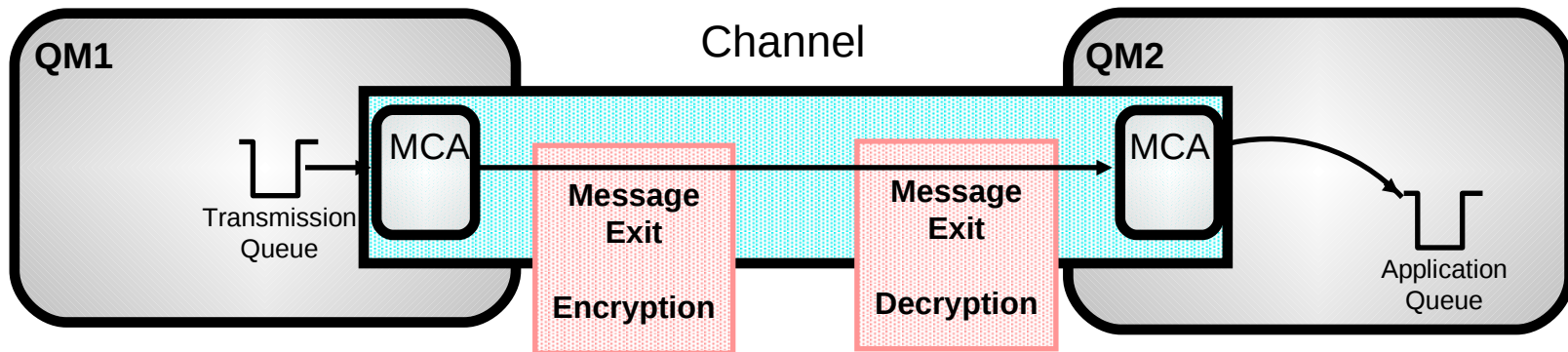
- WebSphere MQ Internet Passthru (MQIPT) is an MQ base product extension that can be used to implement messaging solutions between remote sites across the Internet.
- MQIPT makes the passage of MQ channel protocols into and out of a firewall simpler and more manageable by tunneling the protocols inside HTTP, or by acting as a proxy. The latest version of the SupportPac includes the ability to tunnel inside HTTPS (SSL) and to work via a SOCKS proxy
- Possible uses
- Used as a proxy, MQIPT is placed in the Demilitarised Zone (DMZ) outside an Internet firewall, and relays WebSphere MQ protocol flows from an MQ client or Queue Manager on the external Internet, to a destination Queue Manager inside the firewall. This enables inbound MQ communication through the firewall, from an address which is in the secure DMZ, which is likely to be more acceptable to firewall administrators than an arbitrary external Internet address.
- Placing a pair of MQIPT servers in the path of an WebSphere MQ channel connection enables HTTP wrappers to be added to the protocol flow, which enables the WebSphere MQ connection to pass inbound through an HTTP application firewall.
- MQIPT can also act as a concentrator of MQ connections, which simplifies firewall configuration when multiple WebSphere MQ clients or Queue Managers require access through an Internet firewall.
- These modes of operation of MQIPT give great flexibility to the connection of MQ channels through a variety of firewall and network topologies, and facilitates many application models, particularly in the B2B environment.
- MQIPT does not require any changes to MQ application code, and requires only minor modification to hostname/port setting in MQ channel definitions.

WebSphere MQ Security - Link Level Security

- Identification
 - Message context
 - Security Exits
- Authentication
 - Secure Sockets Layer (SSL)
 - Security Exits
- Access Control
 - Put Authority (see notes for reference)
 - MCA User
 - Message Userid
- Firewalls
 - Port numbers (see notes for reference)
 - Internet passthru (SPac MS81) (see notes for reference)
- Confidentiality
 - Secure Sockets Layer (SSL)
 - Exits
- Data Integrity
 - Secure Sockets Layer (SSL)

Link Level .. Confidentiality

- Encryption at Channel Level
 - SSL
 - Message Exits
 - Messages in the clear on the Queues
 - Applications unaware



Link Level .. Confidentiality - Notes

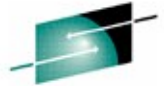
- Encrypting data is an expensive operation and is generally only employed where necessary. Often only certain key elements of a message might be encrypted with most of it still in the clear.

Channel Level Encryption

- Often an enterprises physical security is good enough so that only data on the network is at significant risk of being tampered with. In these cases adding encryption/decryption to the channel operation may be the ideal solution. This can be done in the Channel Message Exits, in the Send/Receive Exits, or it can be done using SSL.
- Encryption algorithms are readily available and easy to incorporate into MQ wrappers and/or Channel Exits. However, if writing a local solution is not to your liking both techniques are well supported by third party security products.

Summary - WebSphere MQ Security

- Identification
 - O/S User IDs
 - Context
 - Channel Authentication Records
 - Channel Security Exits
- Authentication
 - O/S Logon
 - MQCONN
 - Channel Authentication Records
 - Channel Security Exits
 - Secure Sockets Layer (SSL)
- Access Control
 - Command Security
 - API Security
 - Put Authority on Channels
 - Firewalls
- Auditing
- Confidentiality
 - Application Level Security
 - Secure Sockets Layer (SSL)
 - Channel Exits
 - API Exits (also in Clients)
- Data Integrity
 - Application Level Security
 - Secure Sockets Layer (SSL)
- Non-Repudiation
 - Application Level Security



SHARE
Technology • Connections • Results

Questions & Answers



This was session MQ Security - The rest of the week



	Monday	Tuesday	Wednesday	Thursday	Friday
08:00			Free MQ! - MQ Clients and what you can do with them.	MQ Performance and Tuning on distributed	
09:30		The MQ API for dummies - the basics	The Dark Side of Monitoring MQ - SMF 115 and 116 record reading and interpretation	The even darker arts of SMF	CICS Programs Using WMQ V7 Verbs
11:00		Putting the web into WebSphere MQ: A look at Web 2.0 technologies	Message Broker administration	The Do's and Don'ts of z/OS Queue Manager Performance	
		The Doctor is in. Hands-on Lab and Lots of Help with the MQ Family			
12:15		WebSphere MQ: Highly scalable publish subscribe environments		MQ & DB2 – MQ Verbs in DB2 & Q-Replication	
01:30	WebSphere MQ 101: Introduction to the world's leading messaging provider	What's new in WebSphere Message Broker V8.0	The Do's and Don'ts of Message Broker Performance	Diagnosing problems for MQ	
03:00	WebSphere Message Broker 101: The Swiss army knife for application integration	What's new in WebSphere MQ V7.1	WebSphere MQ Security - with V7.1 updates	Diagnosing problems for Message Broker	
04:30	Introduction to the WebSphere MQ Product Family - including what's new in the family products	Under the hood of Message Broker on z/OS - WLM, SMF and more	MQ Java zero to hero	Shared Q including Shared Message Data Sets	
06:00			For your eyes only - WebSphere MQ Advanced Message Security	MQ Q-Box - Open Microphone to ask the experts questions	