WebSphere MQ Security

Morag Hughson – hughson@uk.ibm.com

WebSphere MQ Security - Notes

N O T E S

- 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

- **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.
- **Limit Access to authorised users only**
  - Access Control:- Protects critical resources in a system by limiting access only to authorised users and their applications. It prevents unauthorised use of a resource or the use of a resource in an unauthorised manner.
- **Track who does what to what and when**
  - Auditing:- Tracking who has done what to what and when.

Objectives and Terminology

- **Protect your sensitive data from unauthorised viewing**
  - Confidentiality:- Protects sensitive information from unauthorised disclosure.
- **Check unauthorised changes have not been made to data**
  - Data Integrity:- Detects whether there has been unauthorsied 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

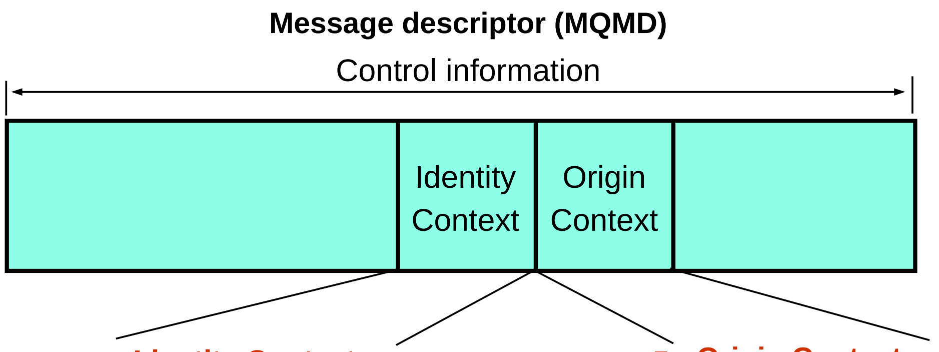
- **Identification**
  - O/S User IDs
  - Context
  - Link-level considerations (later)
- **Authentication**
  - O/S Logon
  - MQCONNX
  - Link-level considerations (later)
- **Access Control**
  - Link-level considerations (later)
- **Auditing**
- **Confidentiality**
  - Application Level Security
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

WebSphere MQ Security - Notes

N O T E S

- 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?



- **Identity Context**
  - **UserIdentifier**
  - **AccountingToken**
  - **ApplIdentityData**
- **Origin Context**
  - **PutApplType**
  - **PutApplName**
  - **PutDate**
  - **PutTime**
  - **ApplOriginData**

Origin Context Information - Notes

N O T E S

- 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...
- PutApplName: 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 HHMMSSTH
  - 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 authorised 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

N

O

T

E

S

- 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 authorised 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.

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;
```

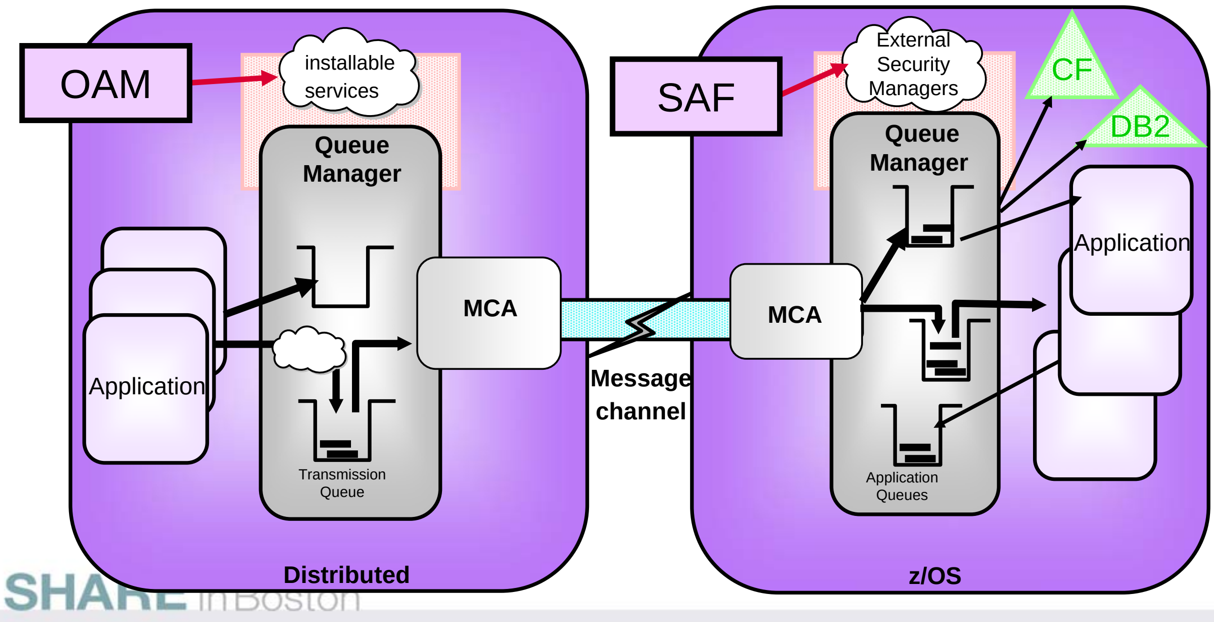
Authentication - MQCONNX - Notes

N O T E S

- On MQCONNX an application can provide a user ID and password (in the Connection Security Parameters (MQCSP) structure in the MQCNO), which are passed to 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.

Access Control Mechanisms

- **Two equivalent mechanisms**
- **To secure the following tasks**
  - Administer WebSphere MQ
  - Work with WebSphere MQ objects



Access Control Mechanisms - Notes

N O T E S

- 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 Authorisation 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 Authorisation Service - to provide access control for MQ resources. MQ supplies an Object Authority Manager (OAM) as an authorisation 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 Authorisation 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. setmqaut (UNIX & Windows), GRTMQMAUTH (i5/OS)
  - e.g. runmqsc (UNIX & Windows), STRMQMMQSC (i5/OS)
  - use mqm group and OS Facilities 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 (i5/OS)
  - 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 (MO05)**

Administering WebSphere MQ - Notes

N O T E S

- 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 qmgrs 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. For the supremely cautious, placing of the WebSphere MQ commands onto diskette will certainly restrict access to diskette holders only !!
- setmqaut (UNIX & Windows) or GRTMQMAUTH (i5/OS) are used to grant authorities to other users to access WebSphere MQ resources. 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

N O T E S

- 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 unauthorised 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.

Access Control - API Security

- **Queue Managers**
- **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**

Access Control - API Security - Notes

N O T E S

- 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

N O T E S

- 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 authorised 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 for 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 objects - Queues**
  - When an application opens a Queue using an MQOPEN or MQPUT1.
  - Alias queues that resolve to a topic
  - Dynamic queues can involve more than one check on creation
    - Model queue
    - Dynamic queue
  - When an application close deletes a permanent dynamic queue using an MQCLOSE
  - When an MQSUB or DEFINE SUB specifies a destination queue for publications
  - Named resource that is checked
  - Fully qualified Remote queues
    - Check on XMITQ associated with non-local queue
    - ToQmgrName
    - SYSTEM.CLUSTER.TRANSMIT.QUEUE

Updated V7

API Security - Queues - Notes

N O T E S

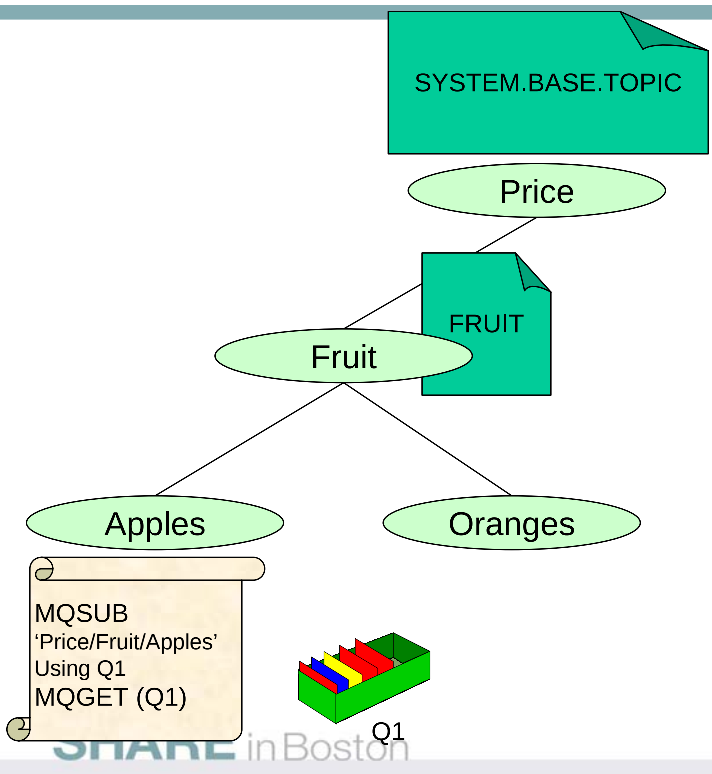
MQSUB or DEFINE SUB command

- 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 qmgr name is also specified), then the application requires access to 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; the SYSTEM.CLUSTER.TRANSMIT.QUEUE if you are using clustering
- This is particularly relevant for the ReplyToQueue/ReplyToQMgr which may be being used for sending responses from a server application.

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

New for V7

API Security - Topics - Notes

N O T E S

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 authorisation 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 = MQOO_OUTPUT
        | MQOO_ALTERNATE_USER_AUTHORITY
        | MQOO_FAIL_IF_QUIESCING;

strncpy(ObjDesc.AlternateUser,
        “FRED”,
        MQ_USER_ID_LENGTH);

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

API Security - Alternate Userid - Notes

N O T E S

- 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);
```

New for V7

API Security - Alternate Userid - Notes

N O T E S

- 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

- **Message Context**
  - When an application wishes to 'do something' with the message context, the check is carried out on the MQOPEN or MQPUT1, or MQSUB

- **Save Context**
  - MQOPEN – MQOO_SAVE_ALL_CONTEXT
  - Save it from MQGET of a message

- **Set Identity Context**
  - MQOPEN – MQOO_SET_IDENTITY_CONTEXT
  - MQPUT – MQPMO_SET_IDENTITY_CONTEXT

- **Pass Identity Context**
  - MQOPEN – MQOO_PASS_IDENTITY_CONTEXT
  - MQPUT – MQPMO_PASS_IDENTITY_CONTEXT

- **Set All Context**
  - MQOPEN – MQOO_SET_ALL_CONTEXT
  - MQPUT – MQPMO_SET_ALL_CONTEXT

- **Pass All Context**
  - MQOPEN – MQOO_PASS_ALL_CONTEXT
  - MQPUT – MQPMO_PASS_ALL_CONTEXT

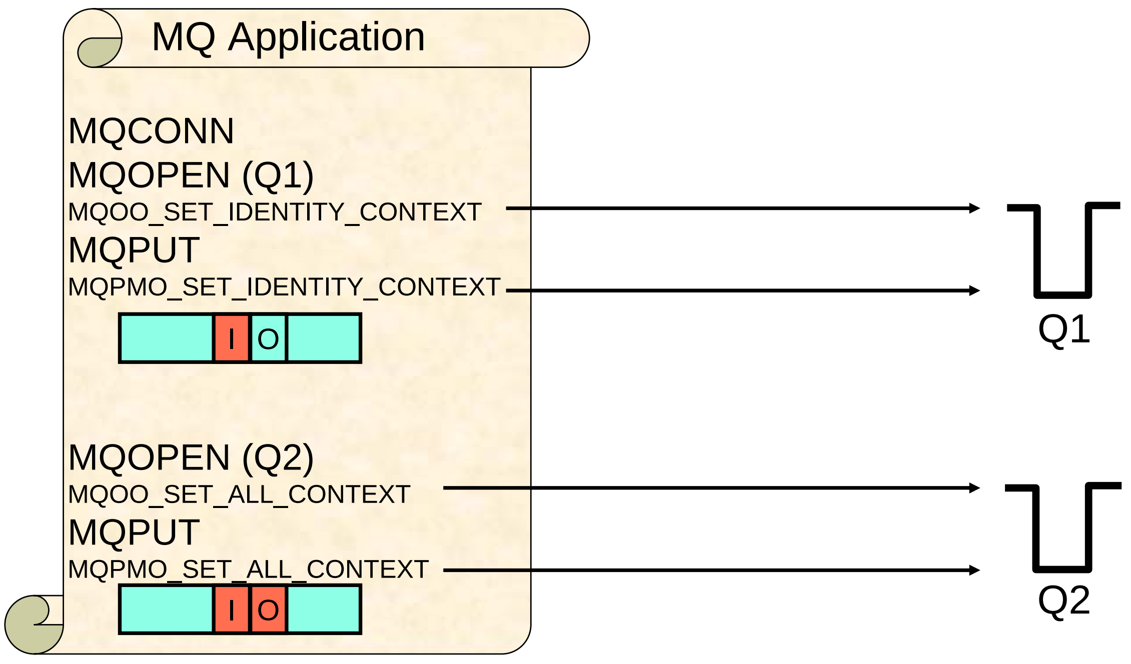
- **Set Publication message Identity Context**
  - MQSUB – MQSO_SET_IDENTITY_CONTEXT

API Security - Context Information - Notes

N O T E S

- 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



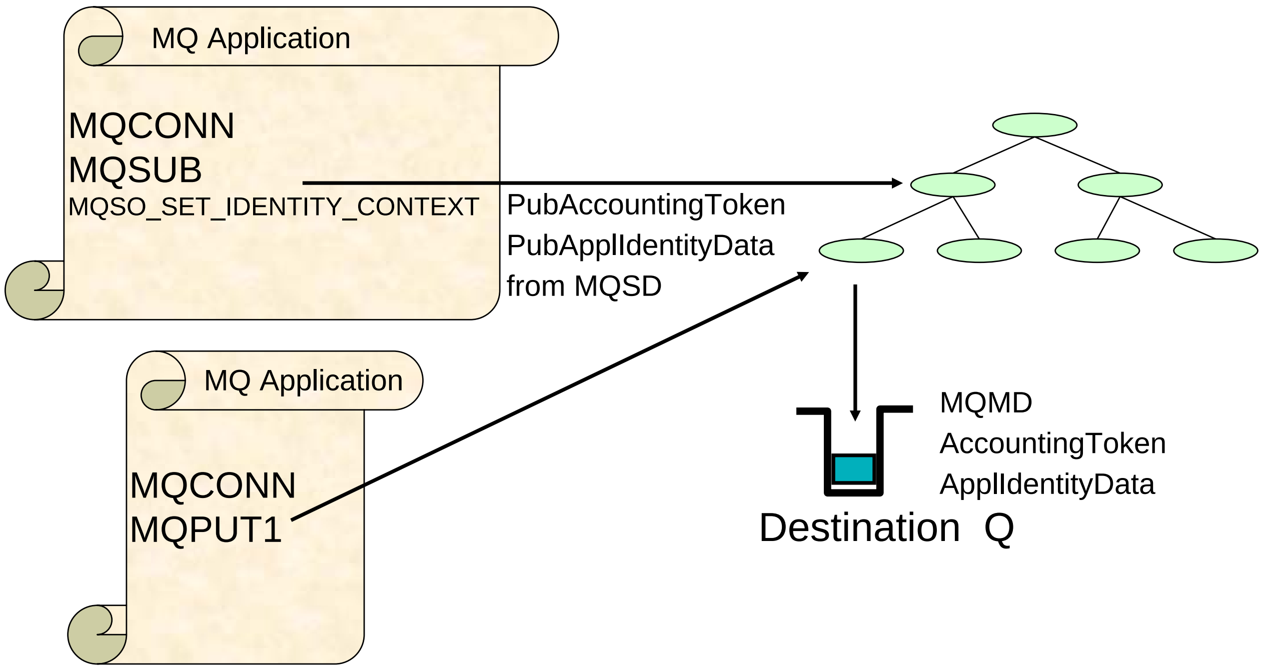
API Security - Setting Context - Notes

N O T E S

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



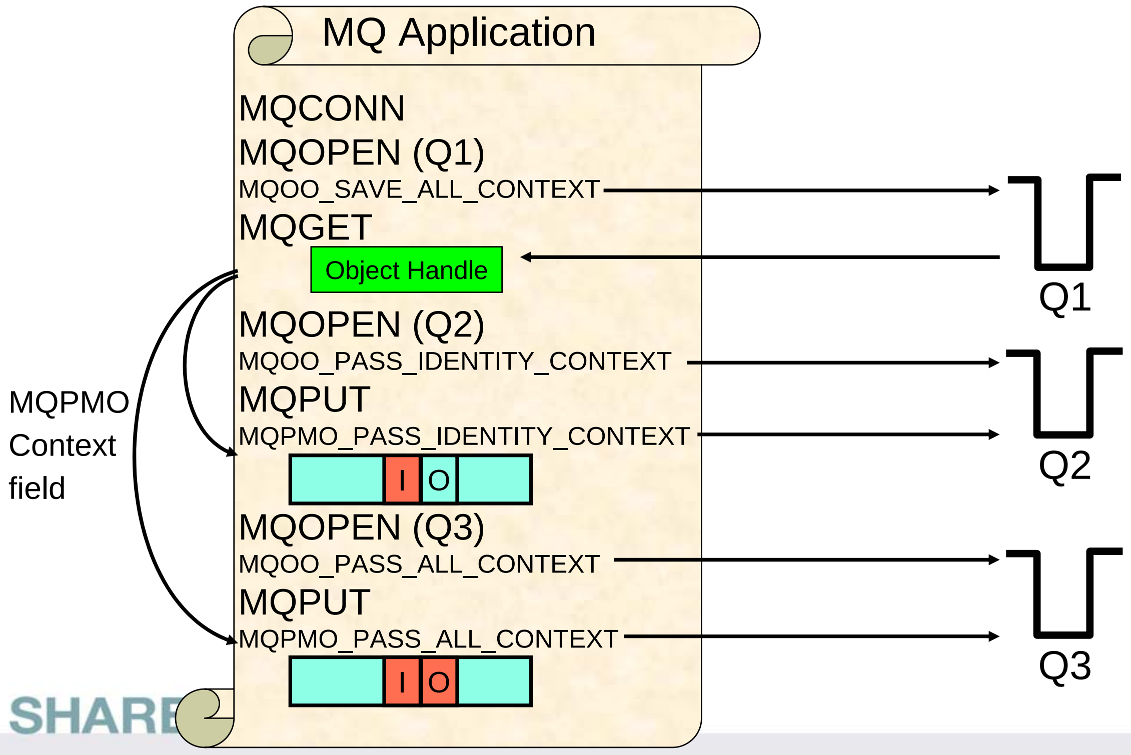
API Security - Setting Context - Notes

N O T E S

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

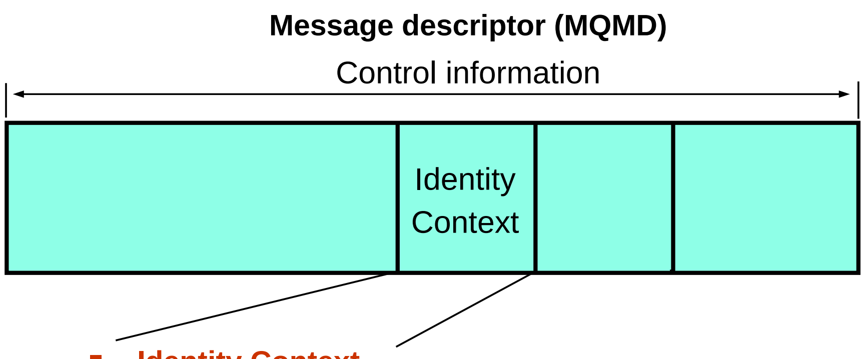
N O T E S

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

N O T E S

- 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.

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

N O T E S

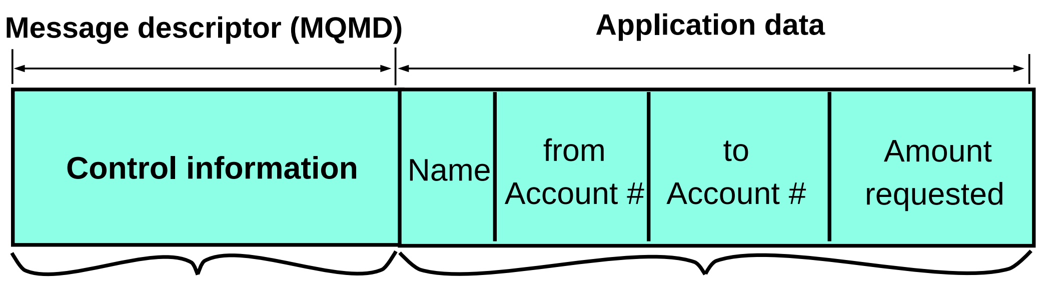
z/OS Platform

- 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 MQRC_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.

What is a message?



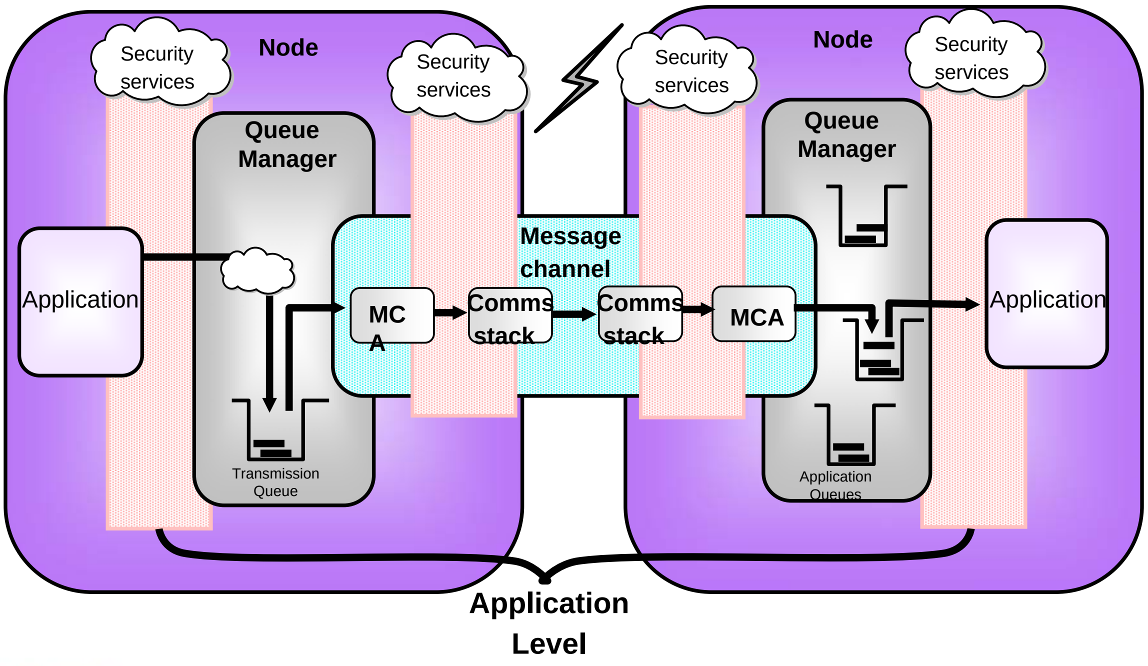
- **Contains things like**
  - Type of message
  - Identifier for message
  - Context information
- **Contains your data**
  - Anything you want to send

What is a message? - Notes

N O T E S

- 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

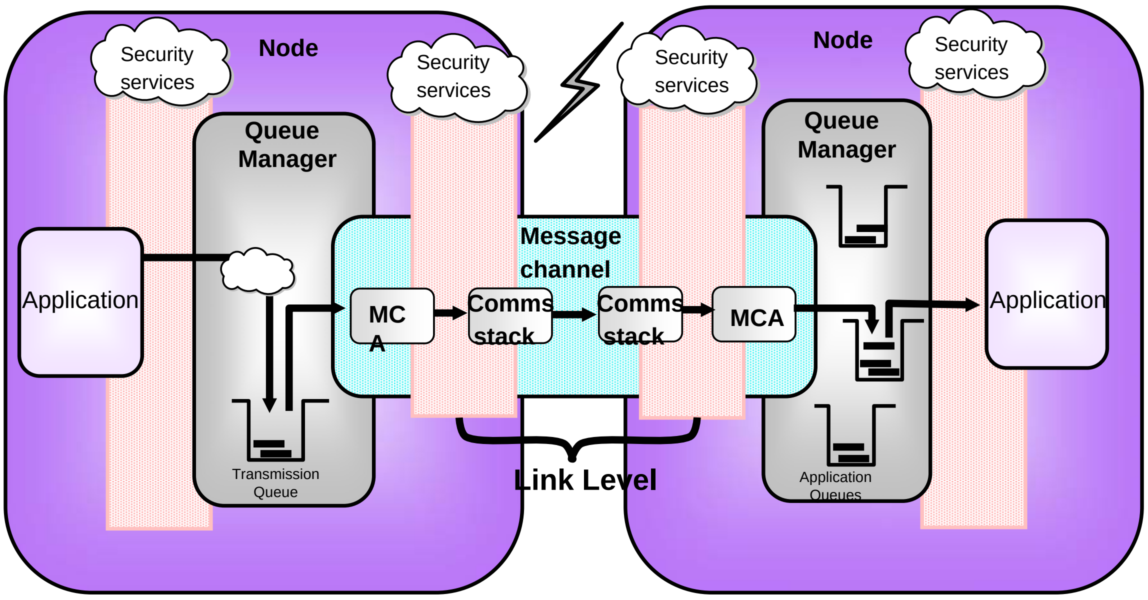


Application Level Security - Notes

N O T E S

- 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 is 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 Extended Security Edition (ESE), 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

N O T E S

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

Link Level Security - Notes

N O T E S

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

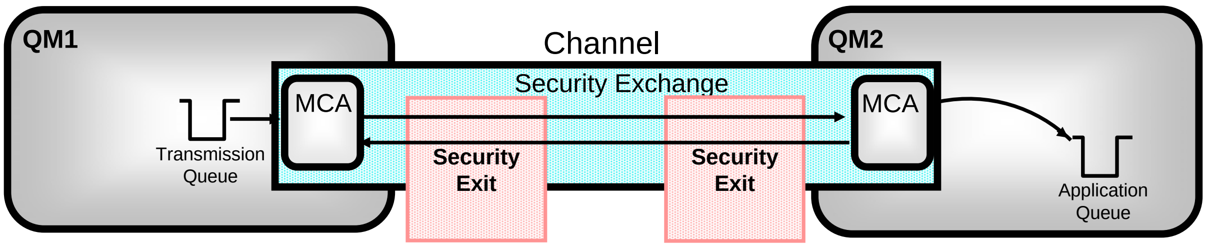
- Authentication is the way in which a channel ensures that the other end of the channel is who they say they are. In WebSphere MQ V5.3, channels can make use of SSL to authenticate a digital certificate sent by the partner. Another method employed by Channels is to use security exits to perform this function.
- Once a remote partner has been authenticated, 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

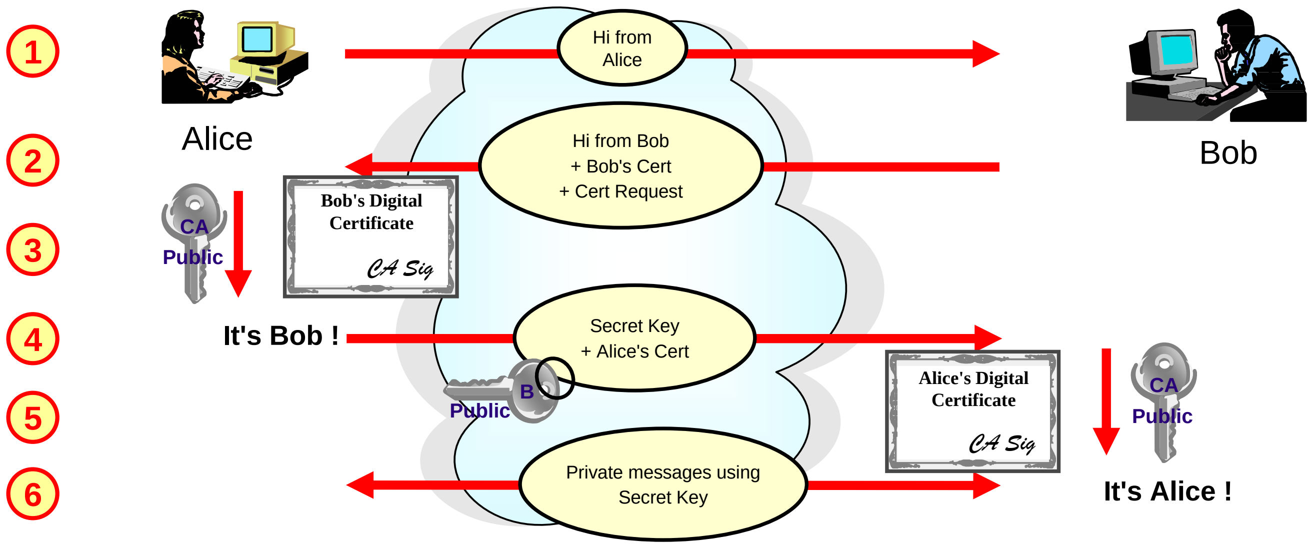


Link Level .. Identification and Authentication - Notes

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. This is built into channels in WebSphere MQ V5.3 and is the subject of a separate presentation.
- 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.

SSL Handshake



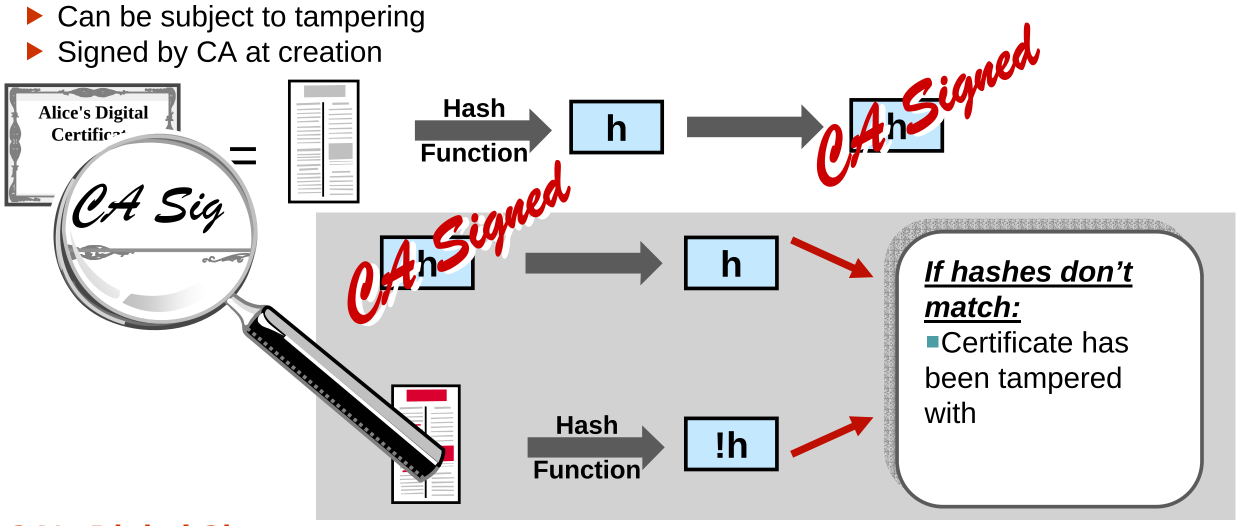
SSL Handshake

N O T E S

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.
2. The 'Server Hello'
   – Bob sends Alice some random text and chooses the CipherSpec be used, from Alice's list. He will also send over his certicate - the Server Certificate and may optionally request that Alice sends him her certificate - the Client Certificate Request.
3. 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).
4. 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).
5. Verify Client Certificate
   – Bob will verify Alice's certificate is valid by checking this digital signature made by the CA
6. 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

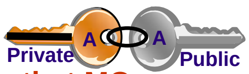
Trusting a Digital Certificate - Notes

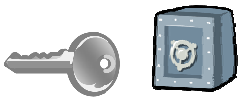
N O T E S

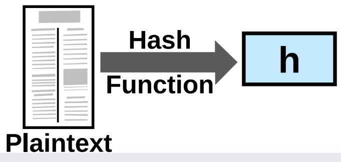
- Digital signatures combine the use of the one-way hash function and public/private key encryption.
- The message is hashed to provide a number, the hash number or message digest. This hash number is encrypted using the sender's private key to create the digital signature. The recipient of the message 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 for Authentication**
  - Digital Certificates
  - Asymmetric Keys
- **Put your certificates in a place that MQ can use**
- **Decide if your business needs revocation status checking**
- **Configure your channels to use SSL for Confidentiality**
  - Symmetric Key Cryptography
- **… and Data Integrity**
  - Hash Function







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

SSLKEYR(QM1KEYRING)

SSLCRLNL(LDAPNL)

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

Link Level .. SSL - Notes

N O T E S

- 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 use of certificate revocation lists (SSLCRLNL on the queue manager).

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**

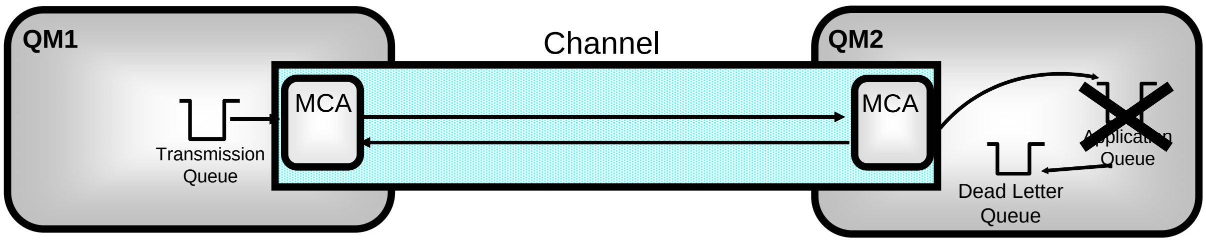
Link Level .. Access Control - Notes

N O T E S

- 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**



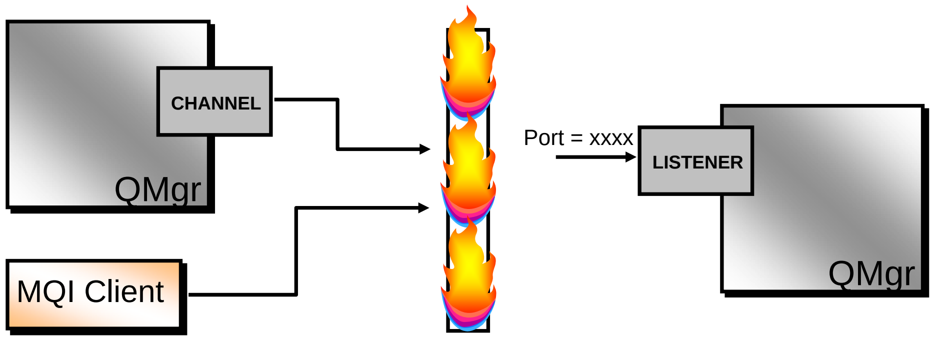
Dead Letter Queue - Notes

N O T E S

- Note that security failures will result in an attempt to put to the Dead Letter Queue. If the user is also not authorised 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))



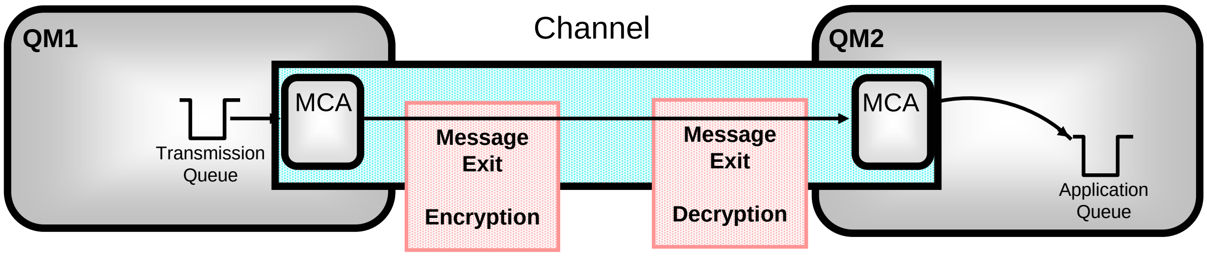
Firewalls - Notes

N O T E S

- 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.

Link Level .. Confidentiality

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



Link Level .. Confidentiality - Notes

N O T E S

- 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/Recieve Exits, or in WebSphere MQ V5.3, it can be done using SSL. SSL is the subject of a separate presentation.
- 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.