IBM MQ for z/OS
Security review

Carl Farkas
IBM Europe zWebSphere consultant
Internet : farkas@fr.ibm.com
(based largely upon slides from Morag Hughson and Marcel Amrein)

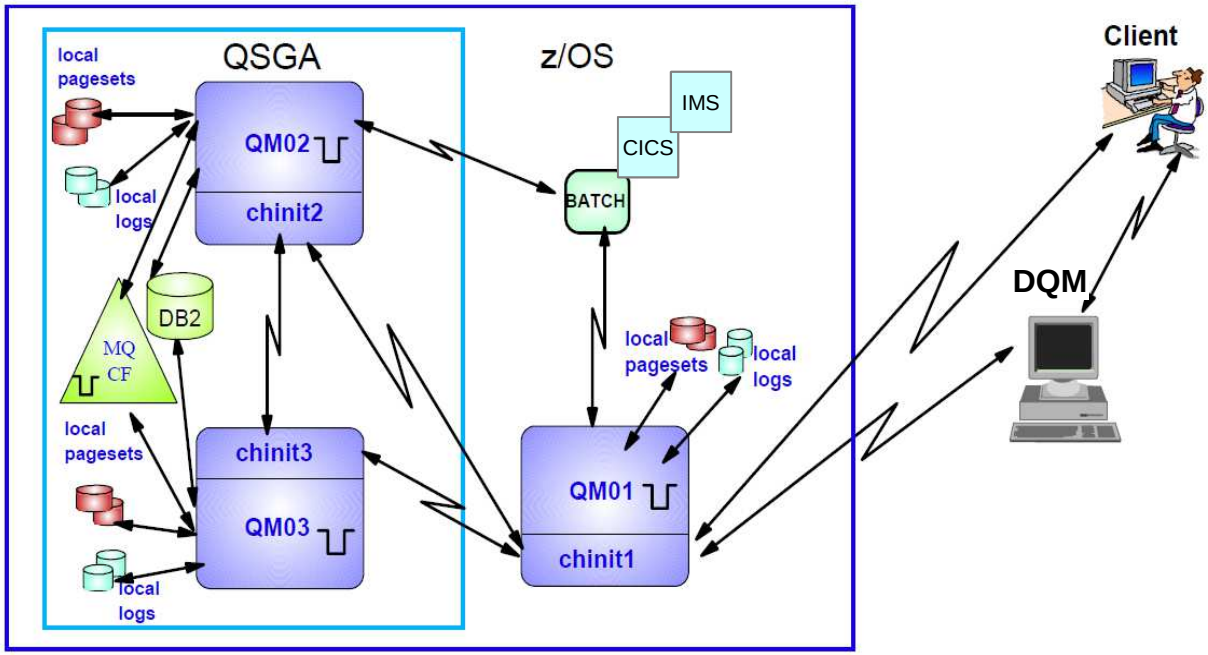
What we'll be covering

- **Identification**
  - Being able to uniquely identify a user of a system or an application
- **Authentication**
  - Prove that a user is who he says he is
- **Access Control (authorisation)**
  - Protects critical resources in a system by limiting access only to authorised users
- **Auditing**
  - Tracking who has done what to what and when
- **Confidentiality**
  - Protect your sensitive data from unauthorised disclosure
- **Data Integrity**
  - Check unauthorised changes have not been made to data

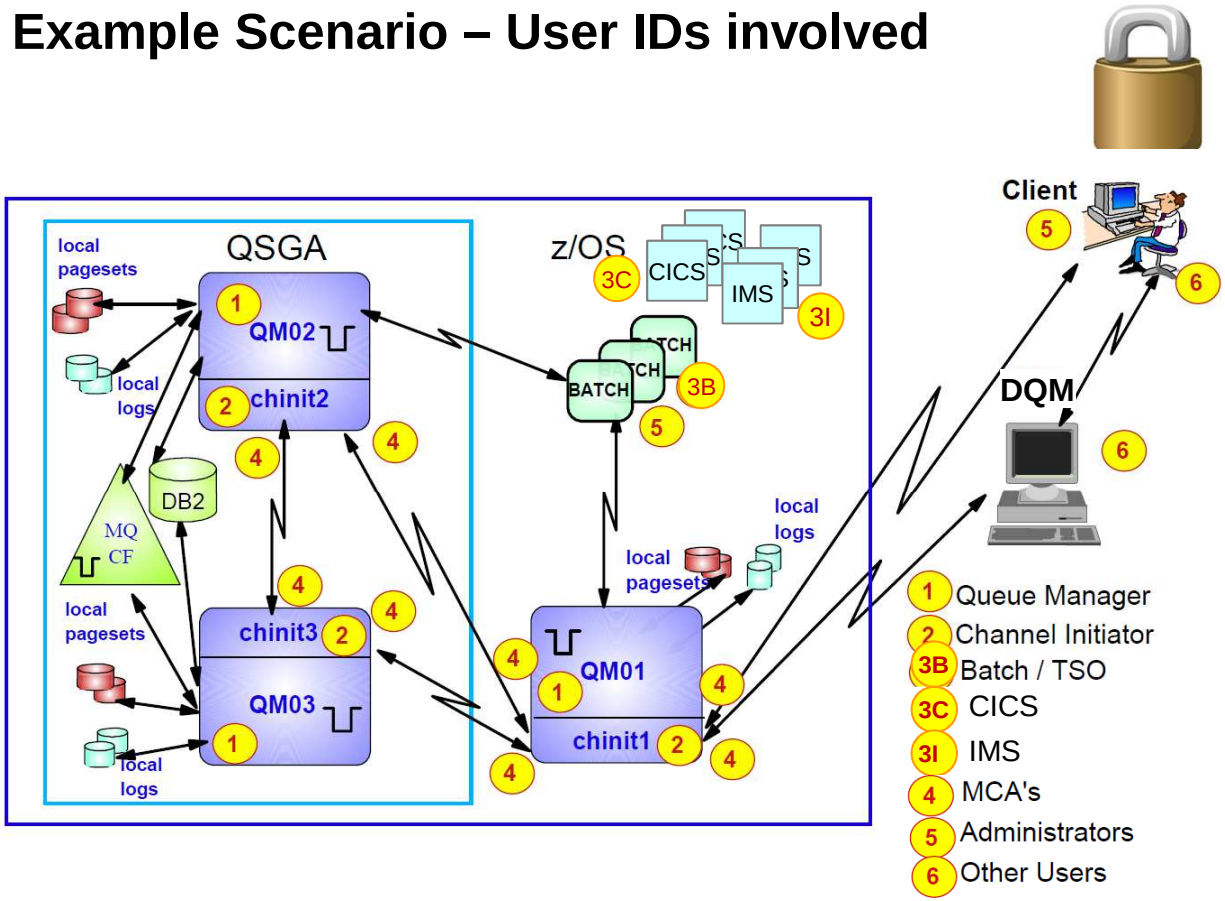
Agenda

- **Look at an example scenario**
  - What protectable resources do we have?
  - Who administers / uses the system?
  - Who is accessing what?
- **MQ for z/OS SAF Concepts**
  - Determine the level of protection – Switch profiles
  - How to protect various types of resources and operations
- **Protecting Network Access to MQ**
  - MQ Clients
  - SSL/TLS
  - Channel Authentication (introduced with MQv7.1)
  - Connection authentication (introduced with MQv8)

Example Scenario



Example Scenario – User IDs involved



Example Scenario – User IDs - Detail

1

Queue Manager id's

QM01MSTR - QM01USR
QM02MSTR - QM02USR
QM03MSTR – QM03USR

QM02 and QM03 in Queue Sharing Group QSGA

2

Channel Initiator id's

QM01CHIN - MVR1USR
QM02CHIN - MVR2USR
QM03CHIN – MVR3USR

3B

Batch / TSO

TSOUSR1
TSOUSR2

3C

CICS

STCCI01,STCCI02
CICUSR1,CICUSR2

3I

IMS

STCIMS1,STCIMS2
IMSUSR1,IMSUSR2

4

Message Channel Agents (MCA's)

QM01- channels
MCAUSR1
MCAUSR2
MCAUSR3
MCAUSR4

QM02 - channels
MCAUSR5
MCAUSR6

QM03 - channels
MCAUSR7
MCAUSR8

5

Administrators

TSOADM1
CLTADM1

6

Other Users

CLTUSR1
DQMUSR1

Agenda

- **Look at an example scenario**
  - What protectable resources do we have?
  - Who administers / uses the system?
  - Who is accessing what?
- **MQ for z/OS SAF* Concepts**
  - Determine the level of protection – Switch profiles
  - How to protect various types of resources and operations
- **Protecting Network Access to MQ**
  - MQ Clients
  - SSL/TLS
  - Channel Authentication (introduced with MQv7.1)
  - Connection authentication (introduced with MQv8)

* We have used the term “System Authorization Facility” (SAF) here, but the examples are all based upon IBM’s RACF product. An equivalent product should work in a similar way.

MQ z/OS dataset security best practices

Data Sets	MQ STC User	MQ Admin	Batch TSO	CICS IMS	App Dev
<hlq>.SCSQAUTH <hlq>.SCSQANLE	READ	READ	READ	READ	
<hlq>.SCSQMVR1	READ				
<hlq>.SCSQLOAD	READ	READ	READ	READ	READ
<hlq>.SCSQCICS	READ			READ	
<hlq>.SCSQMSGE <hlq>.SCSQPMLE <hlq>.SCSQTBLE <hlq>.SCSQSKL <hlq>.SCSQEXEC <hlq>.SCSQPNLS		READ			(READ)
<hlq>.SCSQPROC <hlq>.SCSQINST		(UPDATE)			

Fixed MQ SAF Classes

Class	Group Class	used for ...

MQADMIN
MXADMIN
	GMQADMIN
	GMXADMIN
		• Overall MQ admin security
		  • Component security "switches“
		  • RESLEVEL special profile
		  • Command resources

MQCONN (upper only)	N/A	Application connection security
MQCMDS (upper only)	N/A	Admin Command security
MQQUEUE MXQUEUE	GMQQUEUE GMXQUEUE	Queue Resource profiles
MQPROC MXPROC	GMQPROC GMXPROC	Process Resource profiles
MQNLIST MXNLIST	GMQNLIST GMNLIST	Namelist Resource profiles
MXTOPIC (mixed only)	GMXTOPIC	Topic profiles

SCYCASE(MIXED)

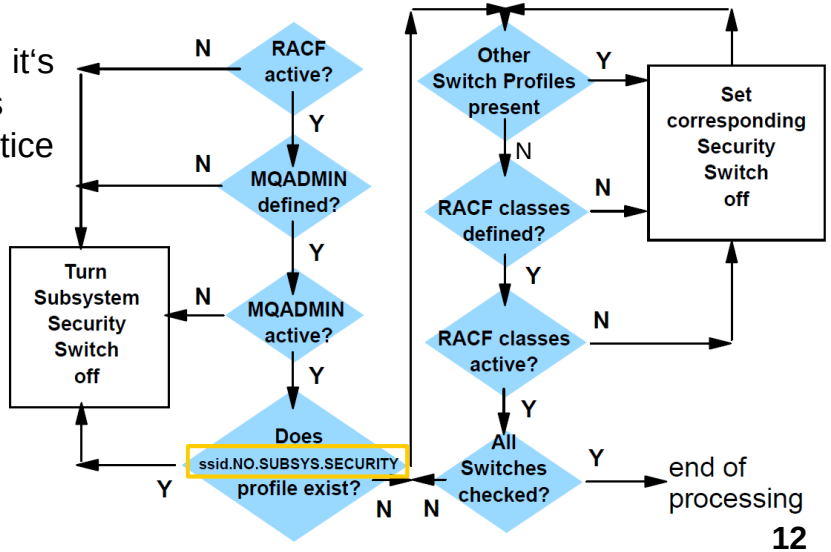
SCYCASE(UPPER)

MQADMIN switch profile functions

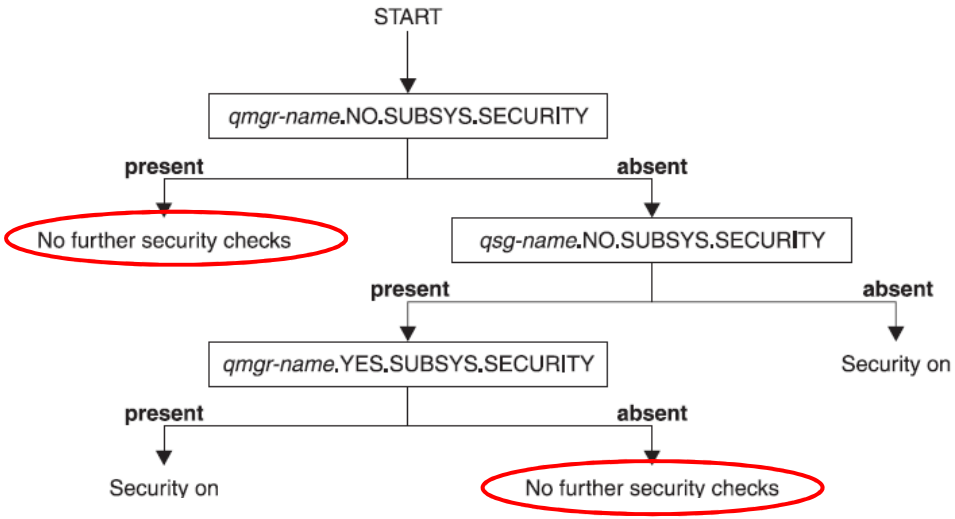
Profile	Turns OFF security for ...
ssid.NO.SUBSYS.SECURITY **qsg.NO.SUBSYS.SECURITY**	… **all** security checking (“main switch“) → qmgr running unsecured
ssid.NO.CONNECT.CHECKS **qsg.NO.CONNECT.CHECKS**	… connections → no checks on MQCONN
ssid.NO.QUEUE.CHECKS **qsg.NO.QUEUE.CHECKS**	… queue resources → no checks on q name on MQOPEN
ssid.NO.CMD.CHECKS **qsg.NO.CMD.CHECKS**	… commands → no checks on command processing
ssid.NO.CMD.RESC.CHECKS **qsg.NO.CMD.RESC.CHECKS**	… command resource checks → no checks resource names for cmds
ssid.NO.CONTEXT.CHECKS **qsg.NO.CONTEXT.CHECKS**	… MQMD context manipulation → no checks approp. MQOPEN option
ssid.NO.ALTERNATE.USER.CHECKS **qsg.NO.ALTERNATE.USER.CHECKS**	… Alternate user usage → no checks approp. MQOPEN option
ssid.NO.PROCESS.CHECKS **qsg.NO.PROCESS.CHECKS**	… process resources → no checks process inquiry
ssid.NO.NAMELIST.CHECKS **qsg.NO.NAMELIST.CHECKS**	… namelist resources → no checks namelist inquire
ssid.NO.TOPIC.CHECKS **qsg.NO.TOPIC.CHECKS**	… Pub/Sub resources → no checks on topic on MQOPEN

Switch profiles: the “main switch”

- Overall security settings and the areas to be protected is controlled by **switch profiles** in SAF class **MQADMIN** (use of uppercase profiles) or **MXADMIN** (mixed case profiles). Use SCYCASE() QMGR attribute for UPPER or MiXeD case.
- The existence or absence of particular profiles determines the security checkings performed by a queue manager or QSG. By default, all switches are ON for security.
- Ideally, to be fully secure, it‘s best to have NO switches enabled, although in practice most sites have some switches in place.



SUBSYS switch profile precedence



- ***Turning off security is clearly not a "best practice", and would generally only be seen in a test environment***

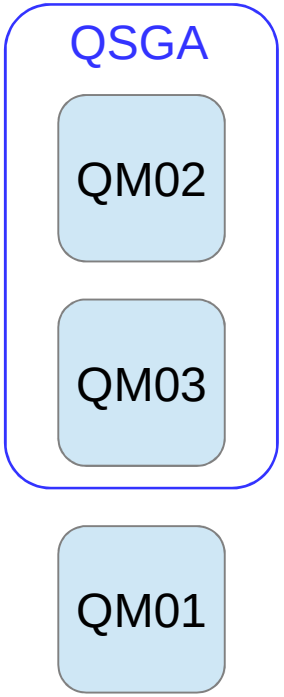
Queue Manager (ssid) or QSG profiles ?

- MQ always uses prefixed SAF profiles
  - ✓ For a stand-alone queue manager, the prefix is the queue manager name, the z/OS subsystem ID
  - ✓ In a queue-sharing group environment, the QSG name may be used as the prefix.
- Additional “switch profiles” in CL(MQADMIN / MXADMIN) are used for finer control
- The use (or non-use) of these additional switch profiles can be controlled:

qmgr-name.NO.QMGR.CHECKS	no qmgr level checks for this queue manager
qsg-name.NO.QMGR.CHECKS	no qmgr level checks for this queue-sharing group
qmgr-name.YES.QMGR.CHECKS	queue manager level checks **override** for this queue manager
qmgr-name.NO.QSG.CHECKS	no QSG level checks for this queue manager
qsg-name.NO.QSG.CHECKS	no QSG level checks for this queue-sharing group
qmgr-name.YES.QSG.CHECKS	QSG level checks **override** for this queue manager

- Note that if you have SUBSYS security on, MQ will prevent you from turning off all checks for BOTH the QSG and the given QMGR.
- Use a simple approach requiring very few and clear switch profiles

“Main Switch” Settings for our scenario



- For the QSGA queue-sharing group (queue managers QM02 and QM03), we want pure QSG-level checking- so we will permanently have this profile defined:

```
RDEF CL(MQADMIN)
    QSGA.NO.QMGR.CHECKS UACC(NONE)
```

  - *Note that any universal access (UACC) level may be defined to the switch profiles- there is no access list!*

- For the single queue manager QM01, we don't have to define anything to let security be active:
- ~~`QM01.NO.SUBSYS.SECURITY`~~

If setting security on for your queue manager, you might want to consider using RACF WARNING mode for the resource profiles to start so you can identify those without access before locking them out!

Start with “sub-area checks” switched OFF

- A practical starting point for a bit-by-bit implementing of MQ security is, to have subsystem security (formally) activated, but inhibit any checks by switching OFF all sub-areas.
- Profiles to be defined to class MQADMIN are:

QM01

```
QM01.NO.CONNECT.CHECKS
QM01.NO.CMD.CHECKS
QM01.NO.CMD.RESC.CHECKS
QM01.NO.QUEUE.CHECKS
QM01.NO.TOPIC.CHECKS
QM01.NO.PROCESS.CHECKS
QM01.NO.NLIST.CHECKS
QM01.NO.CONTEXT.CHECKS
QM01.NO.ALTERNATE.USER.CHECKS
```

QSGA (QM02 & QM03)

```
QSGA.NO.CONNECT.CHECKS
QSGA.NO.CMD.CHECKS
QSGA.NO.CMD.RESC.CHECKS
QSGA.NO.QUEUE.CHECKS
QSGA.NO.TOPIC.CHECKS
QSGA.NO.PROCESS.CHECKS
QSGA.NO.NLIST.CHECKS
QSGA.NO.CONTEXT.CHECKS
QSGA.NO.ALTERNATE.USER.CHECKS
```

Example RACF command to be used:

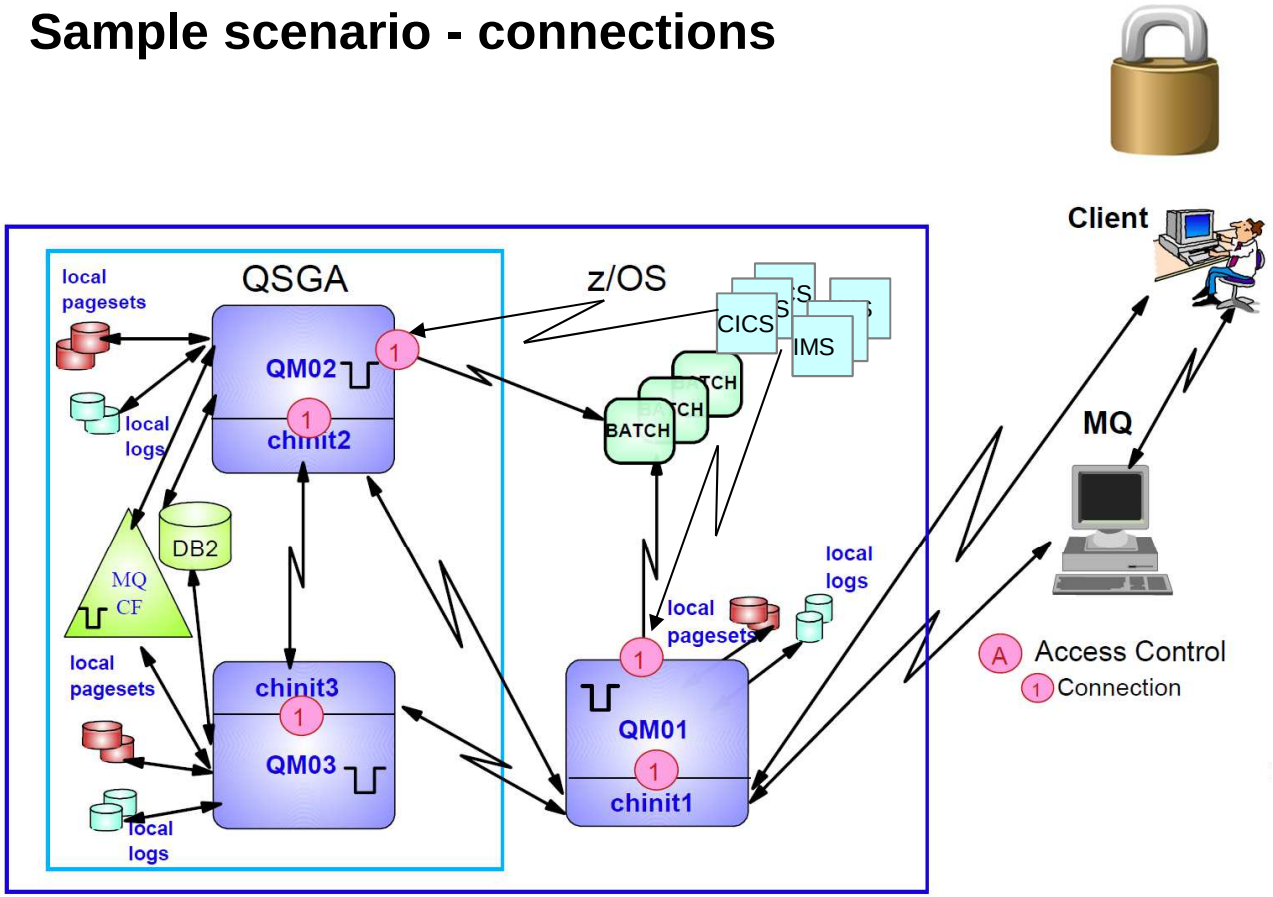
```
RDEFINE MQADMIN QM01.NO.CONNECT.CHECKS UACC(NONE)
```

Setting up connection security

- Connection security is used if ***no*** ssid.NO.CONNECT.CHECKS switch profile is in place
- MQ connection security checks a user's permission to **MQCONN**ect to a system *from/within a particular environment*.
- Checks are performed for **READ** access against profiles in class MQCONN for these different “environments”:

		What user ?
<prefix>.CHIN	channel (MCA) users	The Channel Initiator address space user ID
<prefix>.IMS	Connections from IMS	IMS region address space user ID(s)
<prefix>.CICS	Connections from CICS	CICS region address space user ID(s)
<prefix>.BATCH	All other connections: batch, TSO	Batch, TSO user

Sample scenario - connections



Sample scenario MQCONN profiles

With reference to our sample scenario, we have to define the following profiles in class MQCONN:

QM01.BATCH	connections coming into QM01 from Batch,TSO
QM01.CHIN	connections coming into QM01 from CHIN1
QM01.CICS	connections coming into QM01 from CICS
QM01.IMS	connections coming into QM01 from IMS
QSGA.BATCH	connections coming into QM02,QM03 from Batch,TSO
QSGA.CHIN	connections coming into QM02,QM03 from CHIN2 and CHIN3, respectively
QSGA.CICS	connections coming into QM02,QM03 from CICS
QSGA.IMS	connections coming into QM02,QM03 from IMS

… using RACF commands as follows:

```
RDEFINE MQCONN QM01.BATCH UACC(NONE)
RDEFINE MQCONN QSGA.BATCH UACC(NONE)
...
```

If setting security on for your queue manager, you might want to consider using RACF WARNING mode for the resource profiles to start so you can identify those without access before locking them out!

Authorizing users for connections

2 Channel Initiator id's

QM01CHIN - MVR1USR
QM02CHIN - MVR2USR
QM03CHIN – MVR3USR

… need access to their respective QMs

```
PERMIT QM01.CHIN CLASS(MQCONN)
       ACC(READ) ID(MVR1USR)
PERMIT QSGA.CHIN CLASS(MQCONN)
       ACC(READ)ID(MVR2USR,MVR3USR)
```

3B Batch / TSO

TSOUSR1
TSOUSR2

… need access to all queue managers

```
PERMIT QM01.BATCH CLASS(MQCONN) ACC(READ)
ID(TSOUSR1,TSOUSR2)
PERMIT QSGA.BATCH CLASS(MQCONN) ACC(READ)
ID(TSOUSR1,TSOUSR2)
```

3C CICS

STCCI01,STCCI02
CICUSR1, CICUSR2

… the CICS address space (STC) user IDs need access to all queue managers

```
PERMIT QM01.CICS CLASS(MQCONN) ACC(READ)
       ID(STCCI01,STCCI02)
PERMIT QSGA.CICS CLASS(MQCONN) ACC(READ)
       ID(STCCI01,STCCI02)
```

3I IMS

STCIMS1,STCIMS2
IMSUSR1, IMSUSR2

… so do the IMS address space (STC) user IDs

```
PERMIT QM01.IMS CLASS(MQCONN) ACC(READ)
       ID(STCCI01,STCCI02)
PERMIT QSGA.IMS CLASS(MQCONN) ACC(READ)
       ID(STCIMS1,STCIMS2)
```

Activating MQ connection security

When the MQCONN profiles access lists have been created, connection security can be switched ON for all queue managers

1) Remove the profiles that switch OFF connection security

```
RDELETE MQADMIN QM01.NO.CONNECT.CHECKS
RDELETE MQADMIN QSGA.NO.CONNECT.CHECKS
```

2) Issue the following RACF commands to pick up changes in RACF

```
SETROPTS RACLIST(MQADMIN) REFRESH
SETROPTS GENERIC(MQADMIN) REFRESH
```

3) Issue the following MQ commands to refresh security settings on queue managers

```
+QM01 REFRESH SECURITY(*)
+QM02 REFRESH SECURITY(*)
+QM03 REFRESH SECURITY(*)
```

Note: the **`REFRESH SECURITY`** MQ command has to be issued against every single queue manager –whether stand-alone or member of a QSG.

The RESLEVEL profile for z/OS

- There is just one RESLEVEL profile for each system – **`ssid/qsg.RESLEVEL`**
- The RESLEVEL profile is *used to determine how many userids are to be checked* for resource (i.e. mainly **queue**) checks carried out on a given connection for the life of that connection.
- Note that setting RESLEVEL to CONTROL/ALTER also prevents AUDIT records for that id by default!
- At MQCONN time, the connecting user's authority is checked, and resource security is performed thereafter, as per the following rules:

Userids checked?	NONE	READ	UPDATE	CONTROL/ALTER
TSO/job userid (or alternate)	Yes	Yes	Yes	No checking
CICS address space	Yes	Yes	Yes	No checking
CICS task userid (or alternate)	Yes	No	as RESSEC	No checking
IMS address space	Yes	Yes	Yes	No checking
IMS second userid (or alternate)	Yes	No	No	No checking
CHIN ID	2	1	1	None

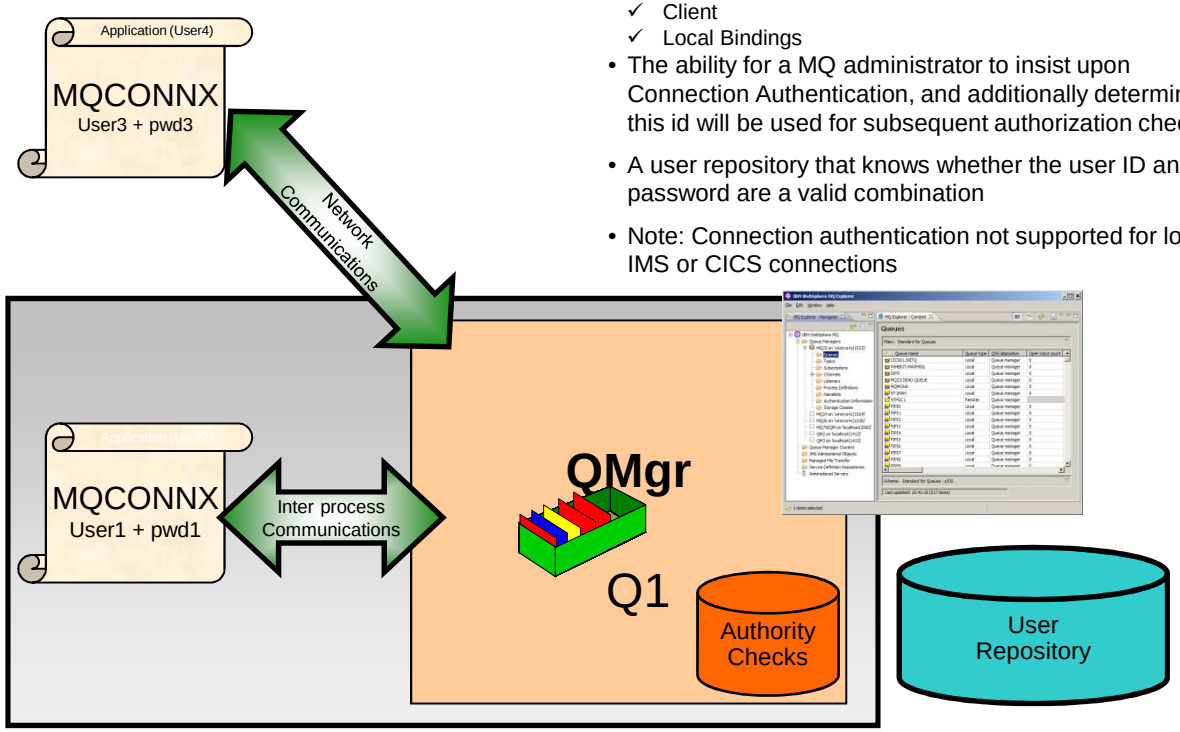
- A common (but not necessarily best!) practice is setting UACC(READ) with no further access list:

```
RDEFINE MQADMIN Q%%%.RESLEVEL UACC(READ)
```

MQ v8 Connection Authentication

MQv8 introduced a new optional Connection Authentication feature:

- The ability for an application to provide a user ID and password
  - ✓ Client
  - ✓ Local Bindings
- The ability for a MQ administrator to insist upon Connection Authentication, and additionally determine if this id will be used for subsequent authorization checking
- A user repository that knows whether the user ID and password are a valid combination
- Note: Connection authentication not supported for local IMS or CICS connections



Setting up Command Security

- Commands against MQ for z/OS may be issued in different ways, by different tools and methods
- Depending on the way you enter an MQ command, securiy checking differs:

Origin of command	User ID checked
Qmgr / Chin startup-JCL CSQINP*,CSQINPX data sets	No security check
Commands put to **`SYSTEM.COMMAND.INPUT.QUEUE`** (eg. Rexx menus, perhaps MQ Explorer if no MCAUSER on channel)	MQMD.UserID
z/OS console	Signed-on user od CSQOPR (ZPARM)
SDSF / TSO	TSO or job user ID
CSQUTIL command option	Job user ID

Command Security Profiles

- Note that MQ command security just protects the command itself and, in case of addressing objects, the **type** of the object affected by the command (the “primary keyword” pkw), which maybe “QUEUE”, “QREMOTE”, “CHANNEL”, etc., *but not the object’s particular name*.
- You can restrict to particular queues, channels, etc… via Command Resource profiles (we’ll see later).
- As follows, some examples of MQCMDS profiles – refer to the MQ Knowledge Center for the full list and associated access levels

Command	MQCMDS profile	Access need
ALTER <pkw>	**<prefix>.ALTER.pkw**	ALTER
DEFINE <pkw>	**<prefix>.DEFINE.pkw**	ALTER
DELETE <pkw>	**<prefix>.DEFINE.pkw**	ALTER
DISPLAY <pkw>	**<prefix>.DISPLAY.pkw**	READ
MOVE QLOCAL	**<prefix>.DEFINE.QLOCAL**	ALTER
ARCHIVE LOG	**<prefix>.ARCHIVE.LOG**	CONTROL
START CHANNEL	**<prefix>.START.CHANNEL**	CONTROL

Authorizing Users for Commands

- What controls are required?
  - Probably lots of profiles to provide suitable protection to MQ commands
  - Use generic profiles where appropriate; implement granular control with specific profiles
- An extract of profiles and permissions to be defined
  - Permit the systems administrator to define channels and queues for queue-sharing group QSGA

```
RDEFINE MQCMDS QSGA.DEFINE.CHANNEL UACC(NONE)
PERMIT  MQCMDS QSGA.DEFINE.CHANNEL ACC(ALTER)ID(TSOADM1)
RDEFINE MQCMDS QSGA.ALTER.QLOCAL UACC(NONE)
PERMIT  MQCMDS QSGA.ALTER.QLOCAL ACC(ALTER)ID(TSOADM1)
```

  - For QM01, let anyone perform DISPLAY commands, but grant full command control only to administrators

```
RDEFINE MQCMDS QM01.** UACC(READ)
PERMIT  MQCMDS QM01.** ACC(ALTER)ID(TSOADM1,CLTAMD1)
```

Command Resource Security Profiles

- If you have a need to differentiate command permissions based on the ***names of resources addressed by the command***, you have to activate Command resource security and provide appropriate profiles in class **MQADMIN.**

Command	MQADMIN cmd resource profile	Access needed
ALTER-DEFINE-DELETE <any type of queue>	**<prefix>.QUEUE.<qname>**	ALTER
ALTER-DEFINE-DELETE PROCESS	**<prefix>.PROCESS.<procname>**	ALTER
ALTER-DEFINE-DELETE NAMELIST	**<prefix>.NAMELIST.<nlistname>**	ALTER
DISPLAY <any type of resource>	**NO CHECK !**	N/A

- Applies to other resources, such as CHANNELs, AUTHINFOs, etc.

Authorizing Users for Command Resources

- **What controls are required?**
  - A practical use case for resource command security could be, to allow developers to maintain (DEFINE, ALTER, etc.) the objects of their own application/project, but to that of other ones.
  - In that case particularly, you will appreciate the value of a good naming convention
- **Example profiles and permissions for command resource protection**

```
RDEFINE MQADMIN QM01.QUEUE.ABC*.* UACC(NONE)
 PERMIT MQADMIN QM01.QUEUE.ABC*.* ACC(ALTER) ID(GRADMIN,GRABC)
RDEFINE MQADMIN QM01.QUEUE.XYZ*.* UACC(NONE)
 PERMIT MQADMIN QM01.QUEUE.XYZ*.* ACC(ALTER) ID(GRADMIN,GRXYZ)
```

  - **However, if command resource security is active, it applies also to processes and namelists – so you may consider to define ...**

```
RDEFINE MQADMIN QM01.*.ABC*.* UACC(READ)
 PERMIT MQADMIN QM01.*.ABC*.* ACC(ALTER) ID(GRADMIN,GRABC)
```

Securing Access to Queues

- You are very likely to have a need to protect your queues
- Queue security checks are always performed at **MQOPEN** time based on the OPEN options (MQOO) specified by the application
- For dynamic queues, another check is performed at MQCLOSE
- **The profile is always `<prefix>.<qname>` in class MQQUEUE**

MQOPEN option	Access needed
MQOO_INQUIRE MQOO_BROWSE	READ
MQOO_INPUT_*	UPDATE
MQOO_OUTPUT	UPDATE
MQOO_SET	ALTER

MQCLOSE option	Access needed
MQCO_DELETE MQOO_DELETE_PURGE	ALTER

- The same rules apply to namelist and process objects for which protection is controlled via a separate switch profile, with profiles
  - `<prefix>.<nlistname>` in class **MQNLIST**
  - `<prefix>.<processname>` in class **MQPROC**

Queue Profile Samples

- **What controls are required?**
  - You probably start with protecting the SYSTEM queues
  - Transmission queues and dead-letter queues tend to require special permissions
  - A good naming convention for application queues allows you to use (few) generic profiles
- **Example profiles and permissions for queue protection for QSGA (roughly designed profiles)**

```
RDEFINE MQQUEUE QSGA.SYSTEM.* UACC(NONE)
RDEFINE MQQUEUE QSGA.ALIAS.*.DEADQ UACC(NONE)
RDEFINE MQQUEUE QSGA.ABC*.*  UACC(NONE)

PERMIT  MQQUEUE QSGA.SYSTEM.* ACC(ALTER)  ID(GRADMIN,GRSTC)
PERMIT  MQQUEUE QSGA.ALIAS.*.DEADQ ACC(ALTER)  ID(GRADMIN,CLTADM1)
PERMIT  MQQUEUE QSGA.ABC.*.* ACC(UPDATE)  ID(GRABC,GRADMIN)
```

Queue Access Requirements for CHIN User ID

- **Just to mention that the CHIN started task ID has very special security needs for SYSTEM.* queues**
  - In case you have to implement a granular queue protection

SYSTEM.ADMIN.CHANNEL.EVENT	UPDATE access required
SYSTEM.CHANNEL.INITQ	UPDATE access required
SYSTEM.CHANNEL.SYNCQ	UPDATE access required
SYSTEM.CLUSTER.COMMAND.QUEUE	ALTER access required
SYSTEM.CLUSTER.REPOSITORY.QUEUE	UPDATE access required
SYSTEM.CLUSTER.TRANSMIT.QUEUE	ALTER access required
SYSTEM.COMMAND.QUEUE	UPDATE access required
SYSTEM.COMMAND.REPLY.MODEL	UPDATE access required
SYSTEM.CSQXCMD.*	UPDATE access required
SYSTEM.QSG.CHANNEL.SYNCQ	UPDATE access required
SYSTEM.QSG.TRANSMIT.QUEUE	UPDATE access required

ALIAS Queue Considerations

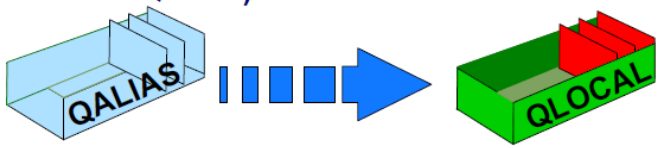
OPTIONS = MQOO_OUTPUT
MQOD_OBJECTNAME = 'REQ1A'
MQOPEN

DEFINE QALIAS(REQ1A)

TARGQ(APPL1.REQUEST)

OPTIONS = MQOO_OUTPUT
MQOD_OBJECTNAME = 'APPL1.REQUEST'
MQOPEN

DEFINE QLOCAL(APPL1.REQUEST)



RACF profile checked
ssid.REQ1A in class MQQUEUE

No check against
ssid.APPL1.REQUEST

RACF profile checked
ssid.APPL1.REQUEST in class MQQUEUE

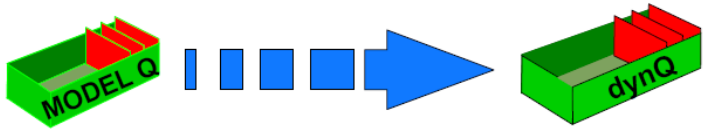
No check against
ssid.REQ1A

Basic rule: a base queue and all its aliases all need specific protection

Model Queue and Dynamic Queues

Requester side

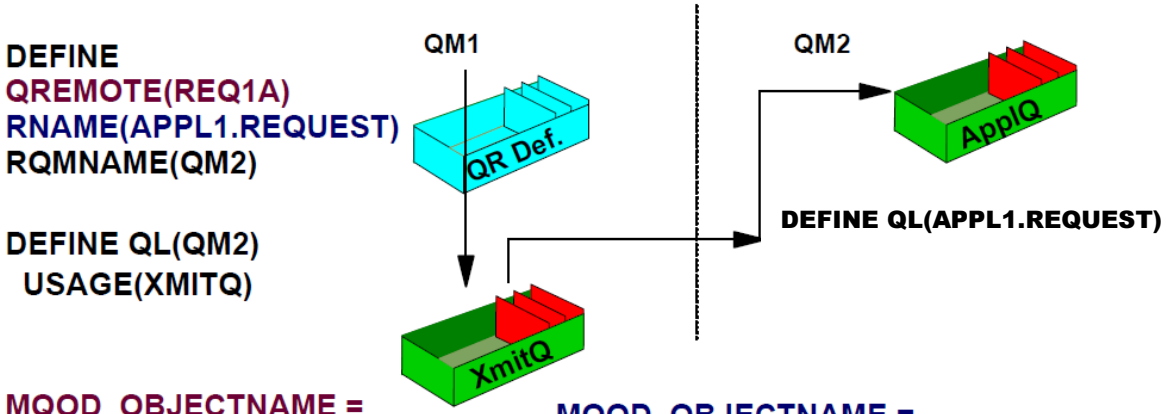
Replier side




```
DEFINE QMODEL(REPLMODA) +
  DEFTYPE(PERMDYN)

MQOD_OBJECTNAME = 'REPLMODA'
MQOD_DYNAMICQNAME =
                    'APPL1.REPLY*'
MQOPEN

Profiles checked

ssid.REPLMODA
ssid.APPL1.REPLY*
```

```
Dynamic 'APPL1.REPLY123456. . . .'


MQOD_OBJECTNAME =
                    'APPL1.REPLY123456. . .
'

MQOPEN, MQCLOSE with MQCO

Profile checked

ssid.APPL1.REPLY*
```

Controling Access to Remote Queues

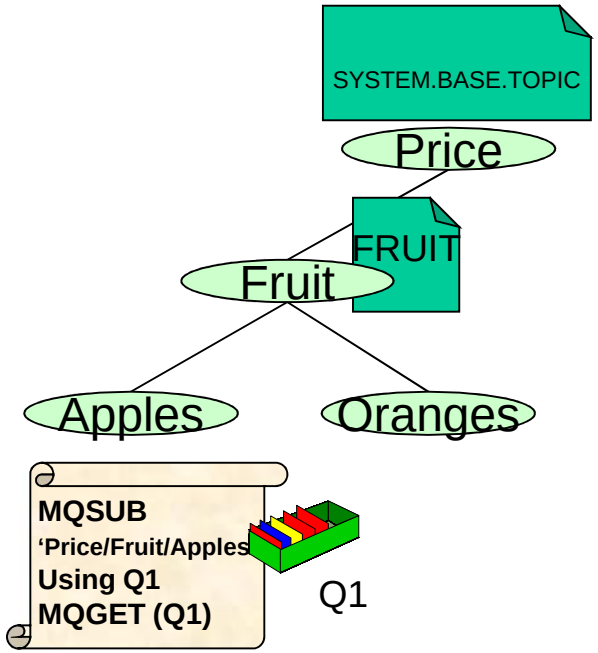


Using a QREMOTE

Using explicit remote QMgr & Q

Topic (Pub/Sub) Security

Security Checks occur:



- ALTER needed when an application Subscribes or Publishes to a Topic using
  - MQSUB
  - MQOPEN / MQPUT1
- ALTER needed 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, eg. Q1
  - When not using MQSO_MANAGED, check is for PUT to that queue

How the TOPIC Profiles Look Like

- SAF class **MXTOPIC**
- **Publish** protected by profiles **`ssid.PUBLISH.<topicname>`**
- **Subscribe** protected by profiles **`ssid.SUBSCRIBE.<topicname>`**
- A practical approach:

```
RDEFINE MXTOPIC WMQ7.SUBSCRIBE.PRICE ACC(NONE)
```

so by default, no one can access PRICE or its children

- Authorize the user group that needs access

```
PERMIT WMQ7.SUBSCRIBE.PRICE CLASS(MXTOPIC)
ID(GROUP) ACC(ALTER)
```

- Of course, generic profiles are possible- but handle with care:

```
PERMIT WMQ7.SUBSCRIBE.PRICE.** CLASS(MXTOPIC)
ID(GROUP) ACC(ALTER)
```

Queue Manager start-up console messages

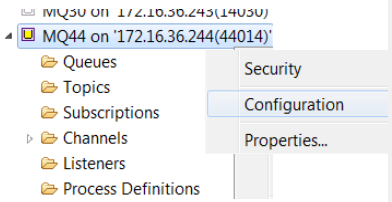
```
CSQH001I +MQ44 CSQHINIT Security using uppercase classes
CSQH024I +MQ44 CSQHINIT SUBSYSTEM security switch set
ON, profile 'MQ44.NO.SUBSYS.SECURITY' not found
CSQH024I +MQ44 CSQHINIT CONNECTION security switch
set ON, profile 'MQ44.NO.CONNECT.CHECKS' not found
CSQH024I +MQ44 CSQHINIT COMMAND security switch set
ON, profile 'MQ44.NO.CMD.CHECKS' not found
CSQH021I +MQ44 CSQHINIT CONTEXT security switch set
OFF, profile 'MQ44.NO.CONTEXT.CHECKS' found
CSQH021I +MQ44 CSQHINIT ALTERNATE USER security
switch set OFF, profile 'MQ44.NO.ALTERNATE.USER.CHECKS' found
CSQH021I +MQ44 CSQHINIT COMMAND RESOURCES security
switch set OFF, profile 'MQ44.NO.CMD.RESC.CHECKS' found
CSQH021I +MQ44 CSQHINIT PROCESS security switch set
OFF, profile 'MQ44.NO.PROCESS.CHECKS' found
CSQH021I +MQ44 CSQHINIT NAMELIST security switch set
OFF, profile 'MQ44.NO.NLIST.CHECKS' found
CSQH024I +MQ44 CSQHINIT QUEUE security switch set ON,
profile 'MQ44.NO.QUEUE.CHECKS' not found
CSQH024I +MQ44 CSQHINIT TOPIC security switch set ON,
profile 'MQ44.NO.TOPIC.CHECKS' not found
```

DISPLAY SECURITY Command Result

- May be issued at any time for the active Queue Manager
- Provides the same result- a bit more clearly arranged

```
+MQ44 DISPLAY SECURITY
CSQH015I +MQ44 Security timeout = 54 minutes
CSQH016I +MQ44 Security interval = 12 minutes
CSQH037I +MQ44 Security using uppercase classes
CSQH030I +MQ44 Security switches ...
CSQH034I +MQ44 SUBSYSTEM: ON, 'MQ44.NO.SUBSYS.SECURITY' not found
CSQH034I +MQ44 CONNECTION: ON, 'MQ44.NO.CONNECT.CHECKS' not found
CSQH034I +MQ44 COMMAND: ON, 'MQ44.NO.CMD.CHECKS' not found
CSQH031I +MQ44 CONTEXT: OFF, 'MQ44.NO.CONTEXT.CHECKS' found
CSQH031I +MQ44 ALTERNATE USER: OFF, 'MQ44.NO.ALTERNATE.USER.CHECKS'
found
CSQH031I +MQ44 PROCESS: OFF, 'MQ44.NO.PROCESS.CHECKS' found
CSQH031I +MQ44 NAMELIST: OFF, 'MQ44.NO.NLIST.CHECKS' found
CSQH034I +MQ44 QUEUE: ON, 'MQ44.NO.QUEUE.CHECKS' not found
CSQH034I +MQ44 TOPIC: ON, 'MQ44.NO.TOPIC.CHECKS' not found
CSQH031I +MQ44 COMMAND RESOURCES: OFF, 'MQ44.NO.CMD.RESC.CHECKS' found
CSQ9022I +MQ44 CSQHPDTC ' DISPLAY SECURITY' NORMAL COMPLETION
```

MQ Explorer Display



Security Switches

Security switch	Security switch setting	Security switch profile
Topic	Switch on; profile not found	MQ44.NO.TOPIC.CHECKS
SubSystem	Switch on; profile not found	MQ44.NO.SUBSYS.SECURITY
Queue	Switch on; profile not found	MQ44.NO.QUEUE.CHECKS
Connection	Switch on; profile not found	MQ44.NO.CONNECT.CHECKS
Command	Switch on; profile not found	MQ44.NO.CMD.CHECKS
Resources	Switch off; profile found	MQ44.NO.CMD.RESC.CHECKS

Activating Changes to MQ SAF Profiles

- After adding/deleting profiles or changing the access list to existing profiles ...

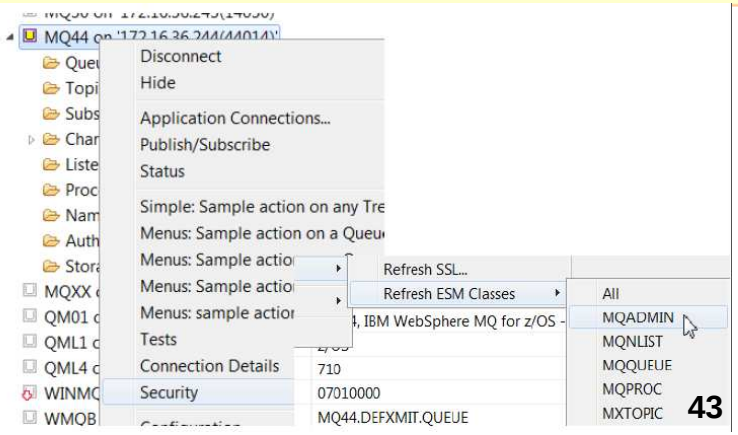
1) Issue the following RACF commands to pick up changes in RACF

```
SETROPTS RACLIST(MQ...) REFRESH
SETROPTS GENERIC(MQ...) REFRESH
```

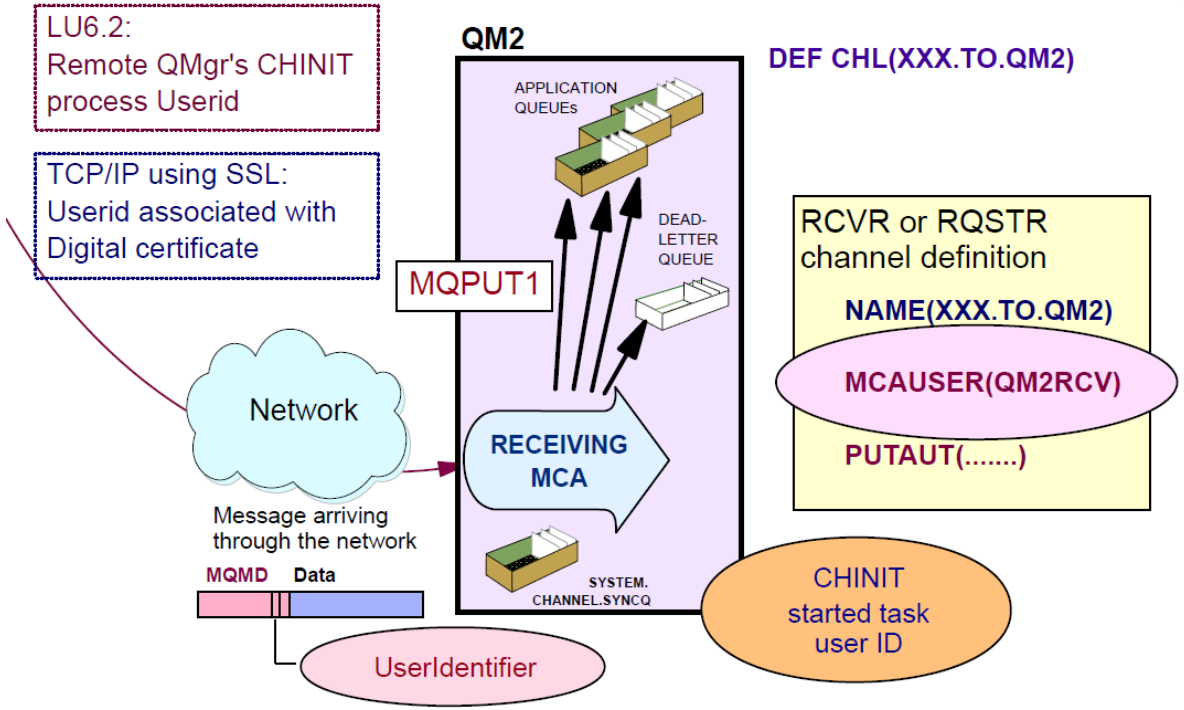
2) Issue the following MQ commands to refresh security settings on queue managers

```
+MQ44 REFRESH SECURITY(MQ... | *)
```

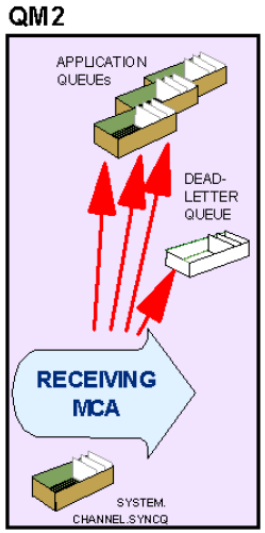
- The MQ commands can also be issued from the MQ Explorer



Securing Remote Queue Access



Resulting Security checks



1 check

PUTAUT specification	Channel User	MCA User	**MQMD Context User**
DEFAULT	X		
CTX	X		
ONLYMCA		X	
ALTMCA		X	

2 checks

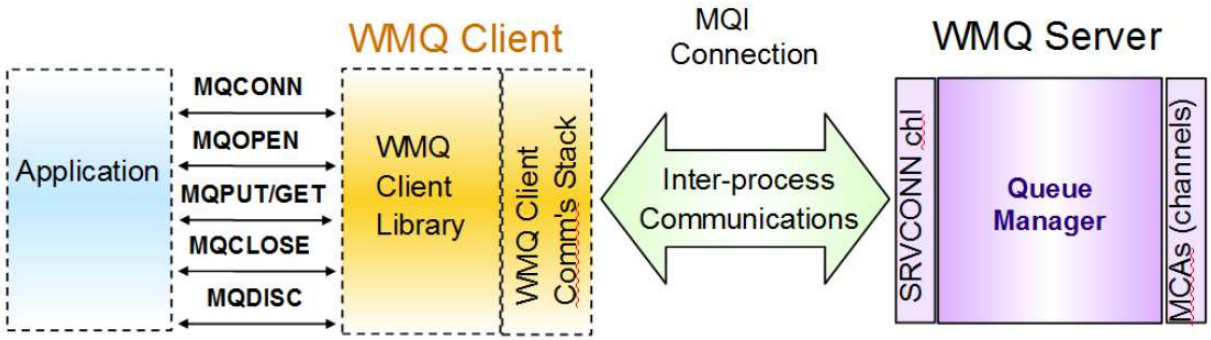
PUTAUT specification	Channel User	MCA User	**MQMD Context User**
DEFAULT	X	X	
CTX	X		X
ONLYMCA		X	
ALTMCA		X	X

- **„1 check“ or „2 checks“ depends by the Chinit STC user's access permission to the RESLEVEL profile**

Agenda

- **Look at an example scenario**
  - What protectable resources do we have?
  - Who administers / uses the system?
  - Who is accessing what?
- **MQ for z/OS SAF Concepts**
  - Determine the level of protection – Switch profiles
  - How to protect various types of resources and operations
- **Protecting Network Access to MQ**
  - MQ Clients
  - SSL/TLS
  - Channel Authentication (introduced with MQv7.1)
  - Connection authentication (introduced with MQv8)

Connecting to MQ on z/OS as a Client



What user ID takes effect in the MQ on z/OS environment ?

- determined by SVRCONN channel configuration
- DEF CHL(<svrconn-chlname>) MCAUSER(' ')
  → *the client app can specify the user ID that runs the channel*
  - By default, that user ID is applied without authentication!
- Client (SVRCONN) channels **MUST** be protected in some way
  - Channel Security Exit (need removed with MQ V8)
  - SSL →, Channel Authentication →
- „Close down“ unused SVRCONN channels by coding something like MCAUSER('NODODY')

SSL/TLS Support for MQ Channels

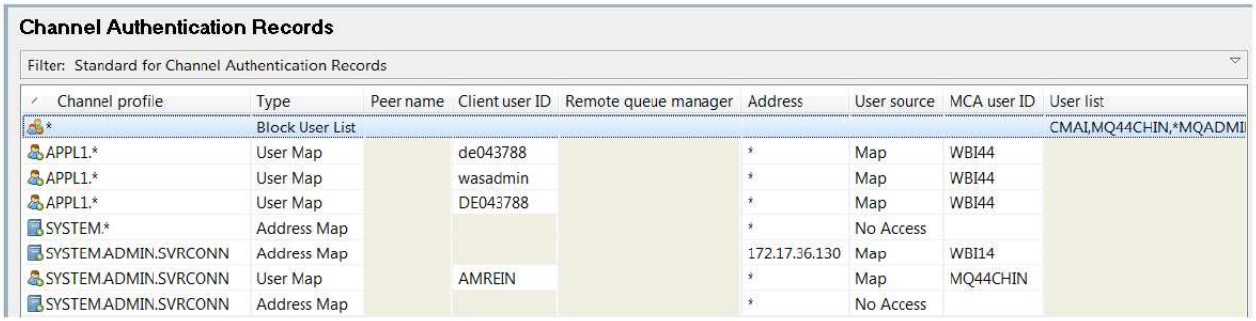
- Queue Manager can enforce the use of SSL/TLS with any type of channel, e.g.
  - `DEF SVRCONN(Client.to.MQ44) … SSLCIPH(RC4_SHA_US)`
  - Of course, SSL/TLS must be activated on the Queue Manager level
    - ALTER QMGR SSLKEYR(<keyring>) SSLTASKS(<integer>)
  - Only clients with a valid certificate are able to connect
- Using RACF certificate name filtering, the Distinguished Name from the certificate can be mapped to a user ID, e.g.:

```
SETROPTS CLASSACT(DIGTNMAP) RACLIST(DIGTNMAP)
```

```
RACDCERT ID(USER1) MAP WITHLABEL('filter1') TRUST
SDNFILTER('O=IBM.C=UK') IDNFILTER('O=ExampleCA.L=Internet')
```

- With MQ V8, each channel may have its own certificate –
  per default, all channels use the Queue Manager cerificate labeled „ibmWebSphereMQssid“

Channel Authentication – introduced by V7.1



Channel Authentication Records

Filter: Standard for Channel Authentication Records

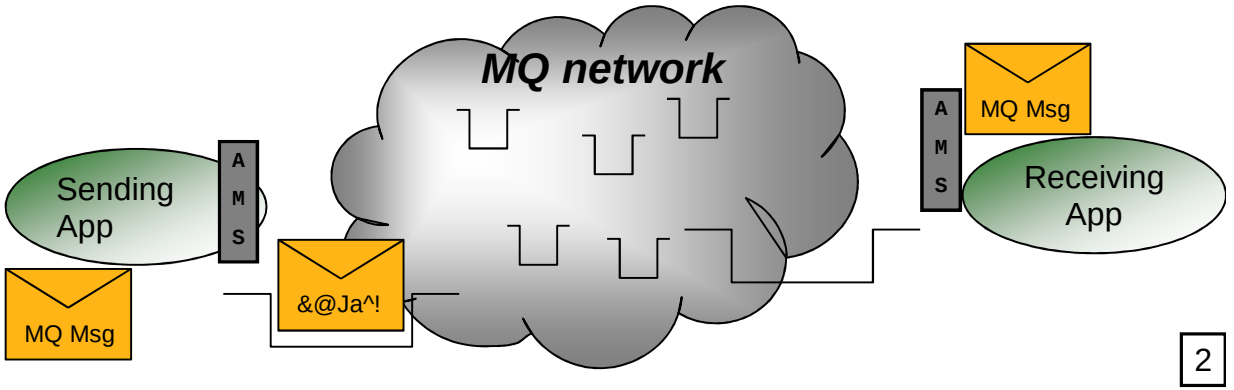
Channel profile	Type	Peer name	Client user ID	Remote queue manager	Address	User source	MCA user ID	User list
*	Block User List							CMAI,MQ44CHIN,*MQADMI
APPL1.*	User Map		de043788		*	Map	WBI44	
APPL1.*	User Map		wasadmin		*	Map	WBI44	
APPL1.*	User Map		DE043788		*	Map	WBI44	
SYSTEM.*	Address Map				*	No Access		
SYSTEM.ADMIN.SVRCONN	Address Map				172.17.36.130	Map	WBI14	
SYSTEM.ADMIN.SVRCONN	User Map		AMREIN		*	Map	MQ44CHIN	
SYSTEM.ADMIN.SVRCONN	Address Map				*	No Access		

- Authentication rules kept inside MQ (as messages on a system queue) to control channel access
  - Rules are related to (generic) channel names and can be „blocking“ or „allowing“, based on the partner's
    - User ID
    - IP address
    - DN pattern from certificate (if SSL/TLS)
    - ...
  - For „allowing“ rules, an MCA user ID can be specified
    - This may remove the need for a security exit or RACF name filtering

What is MQ AMS?

WebSphere MQ Advanced Message Security

- Announced and available since 2010
- Provides security for MQ messages, **end-to-end** with no application changes
- It is a simple “add-on” product that enhances IBM MQ v7, v8
- Security policies are used to define the security level **per queue** which leverage X.509 certificates



Bibliography

- MQ Knowledge Center, for the Security topic, search “q009710_”
- WebSphere MQ for z/OS System Setup Guide (SC34-6927)
- WebSphere MQ Security (SC34-6832)
- IBM DeveloperWorks “WebSphere MQ for z/OS security” article at http://www.ibm.com/developerworks/websphere/library/techarticles/0906_schneider/0906_schneider.html
- IBM DeveloperWorks “What you didn’t know about WebSphere MQ security” article at http://www.ibm.com/developerworks/websphere/techjournal/0701_col_wyatt/0701_col_wyatt.html
- WebSphere MQ Security in an Enterprise Environment (SG24-6814)
- IBM MQ V8 Features and Enhancements (SG24-8218)
- Secure Messaging Scenarios with WebSphere MQ (SG24-8069)
- IBM WebSphere MQ V7.1 and V7.5 Features and Enhancements (SG24-8087)