MQ Integration with Directory Services

Mark Taylor, IBM Hursley

marke_taylor@uk.ibm.com

Session Overview

- MQ was always designed not to rely on external software
  - OS and networking should be sufficient for core function
  - If some other centralised tool is a prereq, then what happens if its down
- But it also always wanted to enable exploitation of optional products
  - HA, Directories, Databases, Application Servers, Transaction Managers etc
  - SOE page lists all the supported optional features
- This session will discuss various features MQ has (or has had) for integration with directory services
  - Directories include, but not limited to, LDAP and Active Directory
  - Almost everything explicit is confined to Distributed platforms – some services on z/OS may transparently exploit LDAP

Why and why not use directory services

- A single point of control
  - Define resources once
  - Everyone can see the same thing at the same time

- What happens if it is down
- Who controls the directory format and content
- Security
- Performance

HISTORIC CURIOSITIES

Start with a quiz

- Who knows what this does

DEFINE QLOCAL(NEW.QUEUE)
SCOPE(CELL)

SCOPE(CELL) – a day-zero feature!

- Allows discovery of which remote queue manager hosts the queue
  - Before clustering was invented
  - Like BIND_ON_OPEN, without workload rebalancing or auto-channel features
  - Converts MQOPEN(queue) to MQOPEN(queue@qmgr)

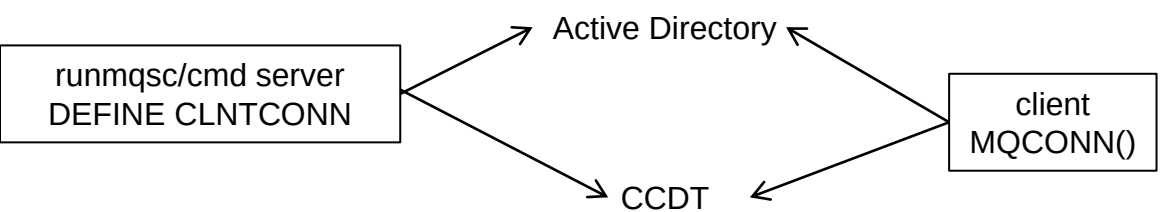
- A pluggable interface gets invoked when defining and opening a queue
  - Documented API (the Naming Installable Service) – similar to OAM for security
  - Sample implementation was provided in source and binary
  - Made calls to DCE Directory Services to insert and query definitions

- Although DCE exits were removed after V5.3, the API still works

Channel definitions

- DEFINE CLNTCONN on Windows populates Active Directory
  - And MQCONN from Windows client searches AD before CCDT
  - Enables a central definition point for client channels
  - setmqscp

- Feature added in V5.1
  - Noone used it: several releases later we found it was broken and noone had complained
  - But it's still there (fixed). And still noone (that I can find) uses it.

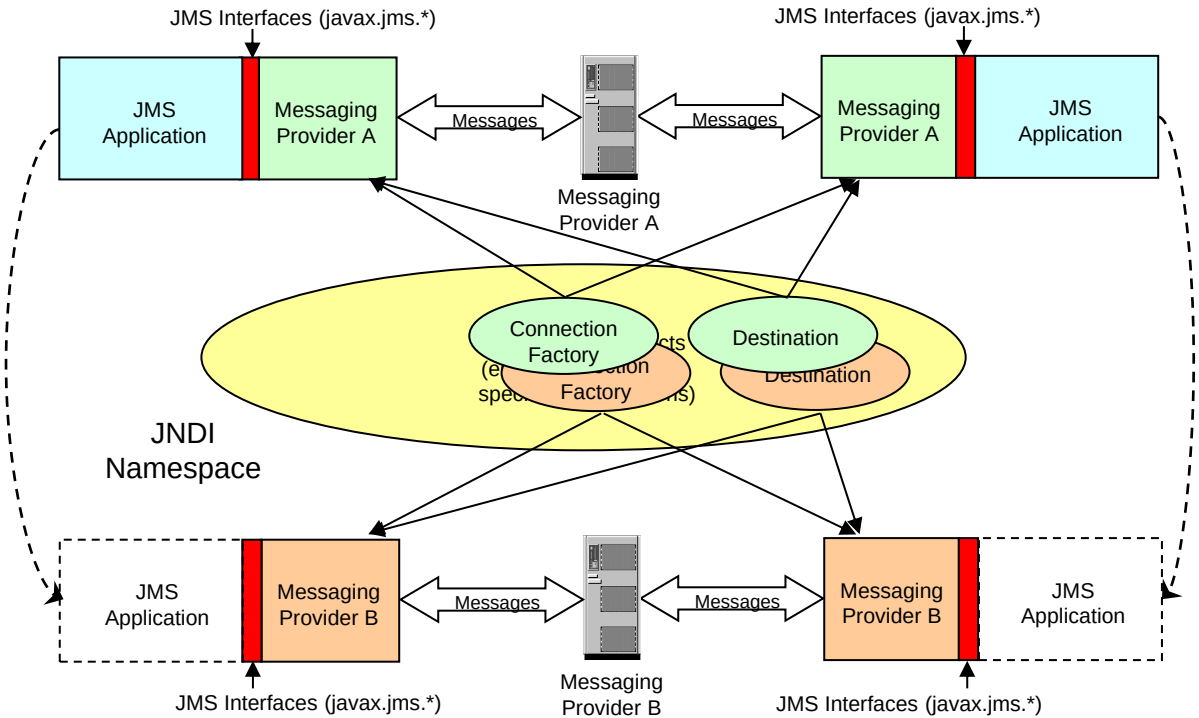


JMS JNDI

Configuration for JMS

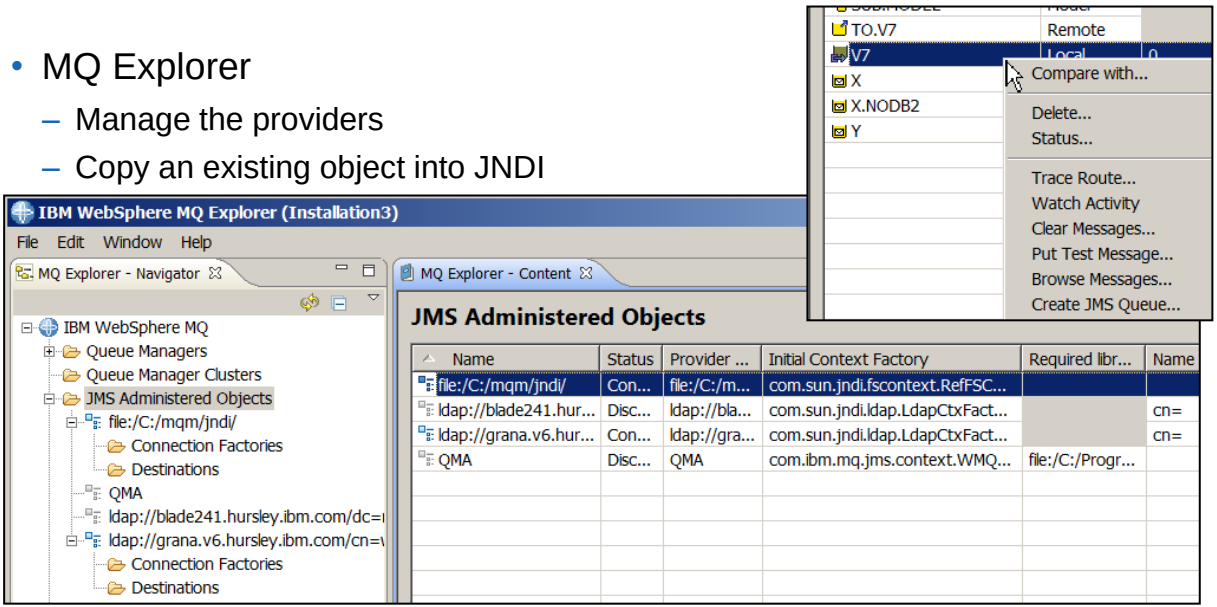
- More successful was a similar concept for JMS programs
- Required by the JMS standard to give an abstract view of configuration
  - Allowing vendor-independent JMS programs to be written
- JNDI gives an abstract interface to directory services
  - Two common implementations, LDAP and local filesystem
  - Old SupportPac ME01 allows a queue manager to be its own JNDI backend
- Putting definitions in LDAP allows central configuration
  - Alternative for connections is a central CCDT accessed via http:// syntax
- Both Connection Factories and Destinations can be accessed this way
  - But there is no scoping of destinations within connections. A destination name must be unique across all connections

JMS using JNDI



Creating JNDI definitions for MQ

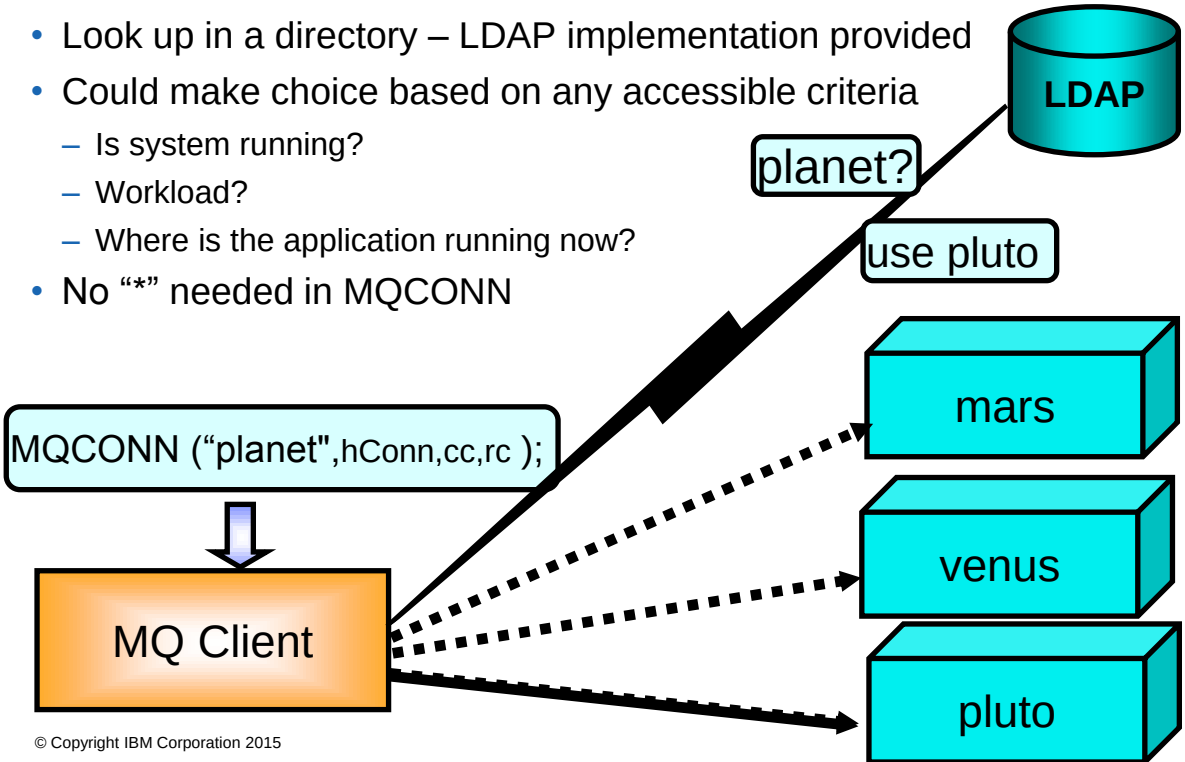
- JMSAdmin command-line program
  - Configuration file lists providers and locations
- MQ Explorer
  - Manage the providers
  - Copy an existing object into JNDI



PRE-CONNECT EXIT

Pre-connect exit overview

- For C clients on all client platforms
- Look up in a directory – LDAP implementation provided
- Could make choice based on any accessible criteria
  - Is system running?
  - Workload?
  - Where is the application running now?
- No “*” needed in MQCONN



LDAP Sample exit details

- Sample source included in product
  - Needs an LDAP client installed to build/run
- Configuration in mqclient.ini requires a new stanza

```
PreConnect:
  Data = ldap://myLDAPServer.com:389/cn=wmq,ou=ibm,ou=com
  Function=LdapPreconnectExit
  Module=amqlcelp
  Sequence=1
```

- An LDAP schema ibm-amq.schema is shipped with exit
  - Schema must be imported to LDAP server
  - Schema attributes are registered with IANA

```
attributetype ( 1.3.18.0.2.4.3566
        NAME 'ibm-amqClientChannelWeight'
        DESC 'Specifies weighting to influence which definition is used.'
        EQUALITY integerMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE
 )
```

LDAP exit – the good and bad

- Another way to centralise definitions
  - Only need to update in one place
  - No "push" to each client system
- Requires mqclient.ini to be configured to drive exit
  - But no client application changes
- No automatic import/export from queue manager into LDAP
- Requires an LDAP client installed to build and run
- Does not apply to Java or .Net programs
  - Update procedures for JMS JNDI could also update pre-connect LDAP tree

SECURITY – BEFORE MQ V8

Authorisation using directories

- Before V8, MQ's security use of directories was mostly transparent
  - Only explicit controls for certificate revocation checks AUTHINFO(CRLLDAP)
  - gskit has an inbuilt LDAP client for this
- Authorisation uses knowledge of the groups to which a user belongs
  - getgrent() etc on Unix
  - NetUserGetLocalGroups, NetUserGetGroups etc on Windows
- The OS may be configured to delegate these calls to a directory
  - Active Directory for Windows
  - nsswitch interface on Unix redirects to a variety of implementations including generic LDAP and explicit Active Directory
- In these setups, MQ does not know how the OS derived the data
  - Use of AD on Windows is limited – does not support all options for nesting etc
  - All users are "local"

Basic Unix configuration – no central config

Local Machine

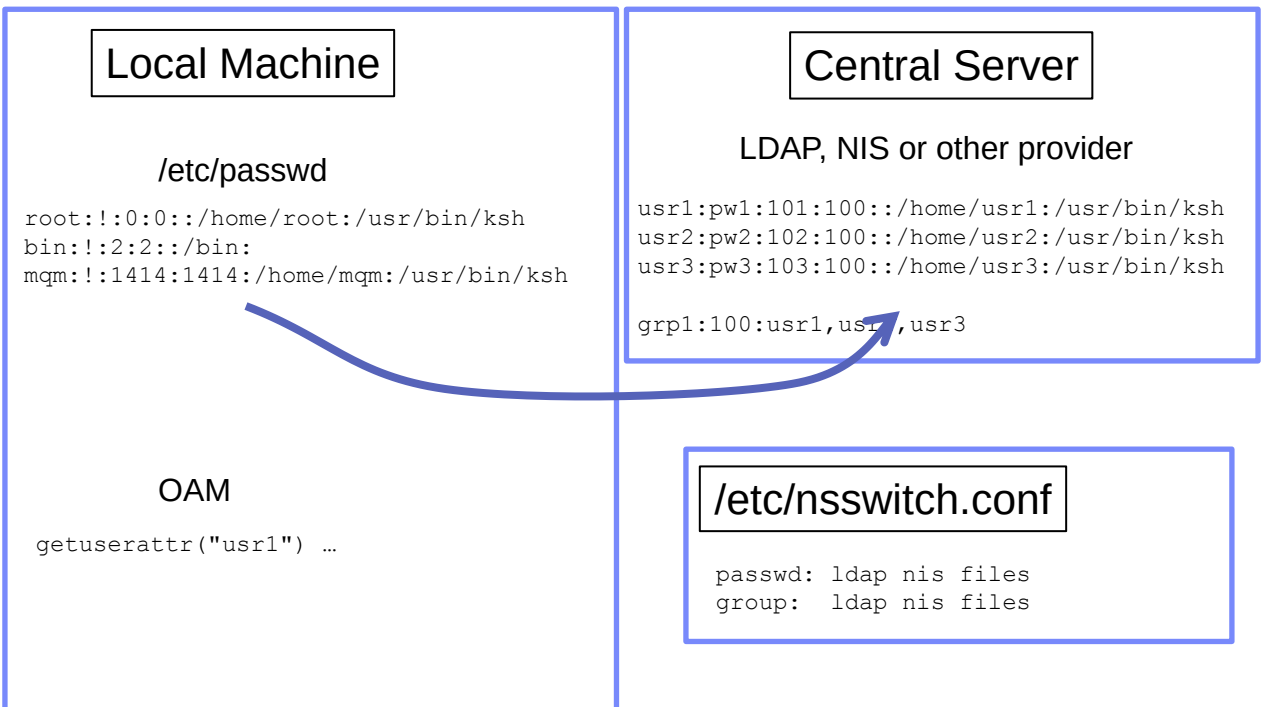
/etc/passwd

```
root:!:0:0::/home/root:/usr/bin/ksh
bin:!:2:2::/bin:
mqm:!:1414:1414:/home/mqm:/usr/bin/ksh
usr1:pw1:101:100::/home/usr1:/usr/bin/ksh
usr2:pw2:102:100::/home/usr2:/usr/bin/ksh
usr3:pw3:103:100::/home/usr3:/usr/bin/ksh
```

OAM

```
getuserattr("usr1") …
```

nsswitch enables centralisation of users/groups



MQ V8

MQ V8 adds authentication and LDAP-based authorisation

- V8 enhanced MQ to verify passwords given during MQCONNX
  - Checks them using local OS methods
  - Or against a common enterprise-wide LDAP repository on Distributed platforms
- FixPack 2 built on that for discovery of group information from LDAP
  - For authorisation
  - On Unix/Linux/400. Not (yet) on Windows
- FixPack 3 added to the "Local OS" authentication with PAM support
  - Which makes it possible to transparently use central repositories as before V8, but also now delegating the authentication

nsswitch and PAM enables centralisation of users/groups

Local Machine

/etc/passwd

```
root:!:0:0::/home/root:/usr/bin/ksh
bin:!:2:2::/bin:
mqm:!:1414:1414:/home/mqm:/usr/bin/ksh
```

OAM

```
pam_authenticate("usr1",password)
getuserattr("usr1") …
```

Central Server

LDAP, NIS or other provider

```
usr1:pw1:101:100::/home/usr1:/usr/bin/ksh
usr2:pw2:102:100::/home/usr2:/usr/bin/ksh
usr3:pw3:103:100::/home/usr3:/usr/bin/ksh

grp1:100:usr1,usr2,usr3
```

/etc/nsswitch.conf

```
passwd: ldap nis files
group:  ldap nis files
```

/etc/pam.d/ibmmq

```
auth     required pam_ldap.so
account  required pam_ldap.so
```

Explicitly using LDAP - Overview

- Reduce/remove the need for locally-defined users and groups
  - The "mqm" group loses a lot of its automatic power
- Almost everything is done based on Distinguished Names
- Authorities can be set for individual users
  - Does not use "primary groups"

```
setmqaut –t qmgr –p "cn=User 1,ou=users,o=ibm,c=uk" +connect
setmqaut –t qmgr –g "cn=Group 1,ou=groups,o=ibm,c=uk" +connect
```

Configuring MQ to use LDAP

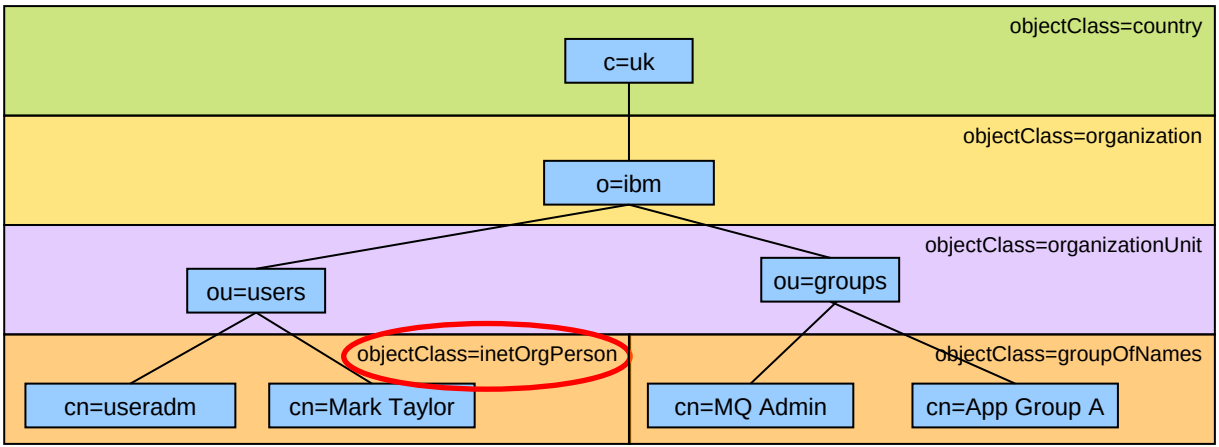
- Several classes of information needed in MQ to use LDAP
  - All set up in a single object definition
  - Which is then referenced from a queue manager attribute
- Define an AUTHINFO(name) TYPE(IDPWLDAP)
  - Authentication requirements (required, optional etc)
  - How to connect to LDAP server
  - How to find userids in the repository
  - How to find groups and group memberships in the repository
- Set queue manager CONNAUTH to point at the AUTHINFO object
  - And **REFRESH SECURITY** to make changes active
  - No separate LDAP client code needed: embedded with MQ
- Queue manager must be able to exploit V8.0.0.2 features

```
strmqm –e CMDLEVEL=801 QMgr
```

LDAP communication configuration

AUTHINFO	
LDAPUSER	If a userid is needed to connect to LDAP. Often anonymous access is permitted.
LDAPPWD	Password for connection
CONNAME	Can accept multiple addresses or even formats such as "ldap://"
SECCOMM	Use TLS for communications to server. Recommended to set ANON or YES Uses other queue manager config such as default qmgr certificate

LDAP Repository layout



Distinguished Name	Type of DN
cn=useradm,ou=users,o=ibm,c=uk	User
cn=Mark Taylor,ou=users,o=ibm,c=uk	User
cn=MQ Admin,ou=groups,o=ibm,c=uk	Group
cn=App Group A,ou=groups,o=ibm,c=uk	Group

Requirement for a short name

- All users must have a field in their LDAP record that holds a short (<=12 char) name
- This name is used in various fields
  - MQMD.UserIdentifier
  - Connection context
  - etc
- Have to be able to map DN to shortnames and back again
  - Inbound messages on a channel with PUTAUT(CTX)
  - Authentication and default connection context
- No need for similar mapping on groups
  - Group information is never transferred outside a queue manager

Attributes for AUTHINFO(IDPWLDAP) at V8 GA

V8 GA	
CLASSUSR	
USRFIELD	
BASEDNU	
SHORTUSR	

- Attributes show how to find where a user's record sits in LDAP
  - So that you do not have to always use a full DN
  - USRFIELD and SHORTUSR do not need to refer to elements of the DN
- Values for these attributes will have to be provided by someone who understands the enterprise LDAP schema

Example of discovering the full DN

```
CLASSUSR('inetOrgPerson')
BASEDNU('ou=users,o=ibm,c=uk')
USRFIELD('cn')
```

Application MQCSP provides	USRFIELD	BASEDNU
cn=useradm,ou=users,o=ibm,c=uk		
cn=useradm		Searches in ou=users,o=ibm,c=uk
useradm	Adds cn=	Searches in ou=users,o=ibm,c=uk

Will then authenticate "cn=useradm,ou=users,o=ibm,c=uk"

Attributes for AUTHINFO(IDPWLDAP) with V8.0.0.2

V8 GA	Added for Authz
CLASSUSR	CLASSGRP
USRFIELD	GRPFIELD
BASEDNU	BASEDNG
SHORTUSR	
	NESTGRP
	FINDGRP
	AUTHORMD

- Three new attributes show how to find groups in exactly the same way as for users
- No need for a "shortgrp" as groups do not need to be mapped to 12-chars
- Values for these attributes will have to be provided by someone who understands the enterprise LDAP schema

Attributes – AUTHORMD (Authorisation Method)

- **OS** means to use operating system calls to find groups as in V8 GA
- **SEARCHGRP:** A group entry in LDAP contains an attribute listing all the users who belong to this group. Membership is indicated by the attribute defined in FINDGRP (typically "member" or “uniqueMember").
- **SEARCHUSR:** A user entry in LDAP contains an attribute listing all the groups to which this user belongs. The attribute to query is defined by the FINDGRP value (typically “memberOf”).
- Many servers use an attribute of the group object to determine group membership
  - This value should therefore be set to SEARCHGRP.
  - But Microsoft AD typically stores group memberships as a user attribute (so use SEARCHUSR)
  - The IBM Tivoli Directory Server supports both methods.

Attributes – NESTGRP, FINDGRP

- **NESTGRP** says to check if groups are members of other groups
  - If YES, each group that a user belongs to is searched recursively to see which group it also belongs to. Clearly "YES" would be slower, but it's only done the first time we need to discover a user's groups.
  - If NO, no such search is done

- **FINDGRP** is the attribute used within an LDAP entry to determine group membership.
  - When AUTHORMD=SEARCHGRP, this attribute is typically set to "member" or "uniqueMember".
  - When AUTHORMD=SEARCHUSR, this attribute is typically set to "memberOf".
  - If left blank, the queue manager uses “member” or “memberOf” depending on the AUTHORMD setting.

Example LDAP records – additional data in records

- In the **inetOrgPerson** class:

```
dn="cn=Mark Taylor, ou=users, o=ibm, c=uk"
    email=marke_taylor@uk.ibm.com [ie >12 chars]
    shortu=metaylor
    Phone=1234567
```

- In the **groupOfNames** class:

```
dn="cn=App Group A, ou=groups, o=ibm, c=uk"
    longname=ApplicationGroupA [ie >12 chars]
    member="cn=useradm, ou=users, o=ibm, c=uk",
           "cn=Mark Taylor, ou=users, o=ibm, c=uk"
```

- Probable AUTHINFO settings:

```
USRFIELD(email) SHORTUSR(shortu)
  BASEDNU(ou=users,o=ibm,c=uk) CLASSUSR(inetOrgPerson)
GRPFIELD(longname)
  BASEDNG(ou=groups,o=ibm,c=uk) CLASSGRP(groupOfNames)
  AUTHORMD(SEARCHGRP)
   + <other attributes to connect to LDAP server>
```

Referring to identities – equivalent examples

Command	Note
`setmqaut -m QM -t qmgr -p metaylor +connect`	This is a flat, unqualified name, resolved via SHORTUSR
`setmqaut -m QM -t qmgr -p marke_taylor@uk.ibm.com +connect`	Also a flat, unqualified name, resolving via USRFIELD to the same entity
`setmqaut -m QM -t qmgr -p email=marke_taylor@uk.ibm.com +connect`	Using a named attribute
`setmqaut -m QM -t qmgr -p "cn=Mark Taylor,ou=users,o=ibm,c=uk" +connect`	Using the full DN
`setmqaut -m QM -t qmgr -p "phone=1234567" +connect`	Using another named attribute which does not have to be any of those configured on the AUTHINFO
`MQSC: SET AUTHREC OBJTYPE(QMGR) PRINCIPAL('marke_taylor@uk.ibm.com') AUTHADD(connect)`	Equivalent MQSC command
`PCF: MQCMD_SET_AUTH_REC with the MQCACF_PRINCIPAL_ENTITY_NAMES element containing the string "cn=Mark Taylor,ou=users,o=ibm,c=uk"`	Same thing again using PCF

Displaying authorities

```
dspmqaut -m QM -t qmgr -p metaylor
Entity metaylor has the following authorisations for object QM:
            connect

dspmqaut -m QM -t qmgr -p email=marke_taylor@uk.ibm.com
Entity email=marke_taylor@uk.ibm.com has the following
authorisations for object QM:
            connect

dmpmqaut -m QM -t qmgr -p metaylor
-----------------------------------
profile: self
object type:qmgr
entity:cn=Mark Taylor,ou=users,o=ibm,c=uk
entity type: principal
authority: connect

dmpmqcfg -m QM -x authrec
-----------------------------------
SET AUTHREC PROFILE(SELF) +
            PRINCIPAL('cn=Mark Taylor,ou=users,o=ibm,c=uk') +
            OBJTYPE(QMGR)
            AUTHADD(CONNECT)
```

Errors

- Errors are shown in the MQ error log
  - For example AMQ5530 or AMQ5531
- Typical problems are that the id cannot be found in LDAP
  - Often because BASEDNU/G or USR/GRPFIELD are incorrectly defined
- Sometimes because the search returns multiple matches
  - Must always have unique responses
- Tracing the agent process (amqzlaa0) shows calls to functions such as ldap_search
  - Can see the actual constructed queries and responses

User-mode authorisation

- V8 on Unix introduced user-mode authorisation option
  - qm.ini setting
- Setting an authority for a user means for that user alone
  - Not for a primary group
  - LDAP does not even have a concept of "primary" group
  - Users may be members of no groups

- When using LDAP authorisation model, user-mode is always in effect
  - Regardless of qm.ini
- No special "nobody" group
  - In OS mode on Unix, can set authorities for every user via "nobody" group
  - Concept not carried into LDAP authorisations

"mqm" group loses a lot of power

- When running in LDAP mode, the ONLY id that has superuser authority is the OS real identity of the person who ran strmqm
  - And, on System i, the QMQM userid
  - May not be the "mqm" userid, though that still exists and setuid still in effect
- Other mqm group members are not treated specially
  - Though they can still run OS commands such as endmqm and dltmqm

- Sample script provided to help define an administrative LDAP group
  - samp/bin/amqauthg.sh issues setmqaut commands for a group
  - Same as the Explorer role-based wizard

Relationship to ADOPTCTX in AUTHINFO object

- There is no requirement for applications to provide authentication information, or for the ADOPTCTX attribute to be set to YES.
  - Subject to CHCKLOCL/CHCKCLNT of course

- If an application does not explicitly authenticate, or if ADOPTCTX is set to NO for the active CONNAUTH object, the identity context associated with the application is initially taken from the operating system userid in the traditional way. Then, when authorisations need to be applied, that 12-char context is mapped to an LDAP identity using the same rules as for the setmqaut commands.

Relationship to ADOPTCTX – example

- Consider running a program by the **su1** userid
  - DN “cn=lu1, ou=users,o=ibm,c=uk” has been granted connect authority
  - The DN has the associated “sn=su1” attribute
  - "amqsauth" is a simple program to do MQCONNX with optional id/pw parms

Comand	ADOPTCTX Config	Comment
amqsauth	Either	**Success**
amqsauth LU1 Pass1	Either	**Success**
amqsauth LU2 Pass2	YES	**Fail** because LU2 is not authorised
amqsauth LU2 Pass2	NO	**Success** because su1/lu1 is authorised, even though we’ve used lu2’s password

MQI

- MQOPEN, MQPUT1 and MQSUB allow an alternate 12-char userid
  - The id is mapped to a DN using the same rules as on the *mqaut commands.
- MQPUT and MQPUT1 may be authorised to set MQMD UserIdentifier
  - This field is not policed during PUT, and is settable to any 12-char value.
  - UserIdentifier may be used for authorisation at later stages such as when PUTAUT(CTX) is defined on a receiving channel. Then the identifier will be checked for authorisation using the config of the receiving queue manager – which may be LDAP or OS-based.
- Wherever an id is provided to a program in an MQI structure, it is the 12-character shortname version associated with the connection
  - For example, the MQAXC.UserId value for API Exits is the shortname returned from the LDAP mapping

Switching between OS and LDAP

- Possible to switch dynamically between OS and LDAP (and back)
  - REFRESH SECURITY TYPE(CONNAUTH)
- Authorisations that have been set for the "other" mode are unused and hidden
  - But are maintained, and will reappear if mode is reverted
- So after switching to LDAP mode, all the default permissions associated with "mqm" group vanish
  - dmpmqaut, dmpmqcfg will only show explicitly-made LDAP authorisations
  - But newly-defined objects still get a (hidden) record for the mqm OS group
    - Which will reappear if authz mode switched back to OS

Summary

- Centralising resource definitions can simplify managing large environments
- MQ has several mechanisms for accessing different types of resource
- Transparent to application programmers
- Directory administrators have to cooperate with MQ administrators

Any questions?