Using Active Directory for authorisation in Unix queue managers

Mark E Taylor |Sep 15 2016 Updated

Permissions for accessing MQ functions have traditionally relied on using operating system definitions for users and groups. That could mean you having a requirement to define those users and groups on each system individually, which is challenging enough in a static topology, but becomes even worse in a dynamic environment such as a cloud where systems may be being defined and deleted regularly. And so some central definition of the identities becomes essential.

For Windows systems, the standard way of sharing identities is Active Directory (AD).

For Unix systems, one way of sharing user and group information is through the configuration of services in the PAM and [nsswitch](http://man7.org/linux/man-pages/man5/nsswitch.conf.5.html%22%20%5Ct%20%22_blank) interfaces. Perhaps the most common mechanism for sharing on those systems is to configure nsswitch to use of the NIS (or NIS+) services; implementations also exist for storing the information in LDAP among other stores. Using those nsswitch and PAM services means that MQ does not know how the users or groups are stored; it can treat them as if they were locally defined in /etc/passwd or /etc/group. All of the operating system services such as *getpwnam*() behave transparently, hiding the underlying source of the data.

An alternative now exists in MQ, which can directly access LDAP servers instead of relying on OS services. A common store can be used not just within an operating system family, but across all the MQ Distributed platforms.  MQ V8 added the ability to authenticate users against an LDAP directory. Fixpack 8.0.0.2 on Unix and System i extended that to allow **authorisations**to be managed using LDAP-defined users and groups; the equivalent Windows support for LDAP authorisation arrived in V9. Using LDAP explicitly means that users do not actually need to be defined or available on the operating system where MQ is running. The OS userid that is running an application program has no direct relationship to the identity used for authorisation checks.

This article shows how MQ can be configured to have LDAP refer to users and groups defined in a Windows Active Directory system, even though the queue manager is not running on Windows. In this exercise, the whole environment was running on AWS, using an Amazon-provided AD server, removing the need to configure such a server myself.

MQ supports the use of any LDAP server, including AD and IBM Directory Server, but one thing I was particularly interested in testing here was the use of "standard" Windows accounts in AD, instead of simply using it as a generic LDAP server, where users and groups may be defined in a separate part of the directory tree.

**The architecture**

1. AWS Directory Service is used to run an Active Directory instance.
2. A Windows 2012 server image which is used to administer the directory, including definitions of users and groups.
3. A Linux image where MQ is installed and runs the queue manager. This was also where MQ applications were run for these tests. I used packer JSON files and procedures as the starting point for this image.

**Configuring the directory service**

All that was needed was to provide a domain name ('**mq.hursley.ibm.com**') and a password for the administrator. You may want to set a VPC and some subnet information to control networking, but I left this to default. Once created, two IP addresses are available which are needed for access to the directory.

**Configuring WIndows**

When I created the Windows 2012 instance, I did not initially join the domain. I left that until later. Once the instance is running, and you have logged on as the administrator, then you need to enable the feature that allows you to administer the directory.



Next, the system must be configured to point at the DNS addresses created for the directory service:



Finally, we can join the domain:



**Administering AD from Windows**

The "Administrative Tools" panel should now contain an item for "Active Directory Users and Computers". That gives a UI for inspecting, defining and modifying users and groups. Much of MQ's LDAP configuration relies on knowing details of the directory schema and where objects are located in the directory. This is most easily done by selecting the "Advanced Features" of the program as it turns on some additional items when you drill into directory entries.



In particular you can now use the Attribute Editor to see all of the attributes of an entry, including the full Distinguished Name and the field names associated with other data. As well as the DN, you need to identify a field in the entry that can be used as a unique short identifier for the user. Looking at the properties of this entry, it seems that **employeeID**might be a good candidate here. The short name is used in MQ to fill in the 12 character MQMD UserIdentifier field, and will also appear in the output of commands such as DISPLAY CONN, showing who is using a queue manager.



**Users and groups**

I created several users and groups to which I could then grant different levels of access.

The first new user, **mqmldap**, is intended to be used by the queue manager for its connection to the directory. This user does no MQ work itself, and is purely there to be able to search the directory for identities. Once created, and a password has been set (you will need to know the password later, I used "**MQpassw0rd**'"), the user needs to be granted read authority on the directory. No further authorities are needed here.



I also created an MQUser group and an MQAdmin group, along with a few test ids that were made members of one or both of these groups.

**Configuring MQ**

Once the directory is configured, with suitable identities, we can turn to the queue manager configuration. In this MQSC command, we are giving several sets of information:

* How to connect to the directory (CONNAME, LDAPUSER, LDAPPWD)
* How to find users and extract the shortname (BASEDNU, CLASSUSR, USRFIELD, SHORTUSR)
* How to find groups (BASEDNG, CLASSGRP, GRPFIELD)
* How to discover which groups a user belongs to (AUTHORMD, FINDGRP)

It is not a mistake that the BASEDNG and BASEDNU in this definition have the same value - in AD, both users and groups can be found in the "Users" part of the tree. You may configure AD with more layers of folders between the root and the actual user or group record; that will work as long as MQ can still find unique values lower in the tree; only the highest level container needs to be supplied.

DEFINE AUTHINFO('AWSLDAP')

 AUTHTYPE(IDPWLDAP) +

 ADOPTCTX(YES) +

 CONNAME('172.31.aaa.bbb,172.31.ccc.ddd') +

 CHCKCLNT(OPTIONAL) +

 CHCKLOCL(OPTIONAL) +

 CLASSGRP('GROUP') +

 CLASSUSR('USER') +

 BASEDNG('CN=Users,DC=mq,DC=hursley,DC=ibm,DC=com') +

 BASEDNU('CN=Users,DC=mq,DC=hursley,DC=ibm,DC=com') +

 FINDGRP('member') +

 LDAPUSER('CN=mqmldap,CN=users,DC=mq,DC=hursley,DC=ibm,DC=com') +

 LDAPPWD('MQpassw0rd') +

 SHORTUSR('employeeID') +

 GRPFIELD('sAMAccountName') +

 USRFIELD('sAMAccountName') +

 AUTHORMD(SEARCHGRP) +

 REPLACE

ALTER QMGR CONNAUTH(AWSLDAP)

REFRESH SECURITY

Using the attribute editor for the entities on Windows made it easy to find the elements that make up this MQ command. The**DISPLAY QMSTATUS ALL** command should show that the queue manager has successfully connected to the LDAP server.

**Setting and checking authority**

We can now set authority for users and groups using the **setmqaut**or **SET AUTHREC** commands. The full distinguished name for the group or user does not need to be used; instead the queue manager uses the information in the AUTHINFO object to search for and derive the DN. For example, I can use

setmqaut -t qmgr -m QMLDAP -p met +connect

setmqaut -t q -n Q1 -m QMLDAP -g MQUser +put

to grant myself connect authority to this queue manager, and all members of a group put authority on a queue. If I then run dmpmqaut, I will see records referring to the complete Distinguished Name instead of the shortened 'met'. More examples of the different syntax options for specifying users and groups can be found in the [Knowledge Center](http://www.ibm.com/support/knowledgecenter/SSFKSJ_9.0.0/com.ibm.mq.sec.doc/q123460_.htm). If you get an "invalid principal" error, then there will be more information in the queue manager error log.

**MQ Administrators**

When MQ is using LDAP for its authorisation model, OS users do not get administrative authority on the queue manager simply by virtue of being in the local **mqm**group. Only the user who starts the queue manager has that authority automatically. The mqm group membership is used only to control starting, stopping and deleting the queue manager. Instead, every operation such as defining a queue or altering a channel is processed by the OAM to see if someone has suitable authority. To simplify the task of having a group of MQ administrators, without needing to share an account, a script is provided in the product that will grant full permissions to a specific group. Running

/opt/mqm/samp/bin/amqauthg.sh QMLDAP MQAdmin

executes all of the setmqaut commands needed to give members of the MQAdmin AD group full administrative control over MQ objects. As it is a script, you can easily change it, for example to create "read-only" administrators. The script does not grant "message" authorities to this group, so they cannot put or get messages. Again, that's something you may choose to add.



**Running MQ programs**

The userid on the operating system can now be irrelevant. If my applications use the userid/password feature during the MQCONN, it is that provided userid - resolved to a DN - that will be used for all authorisation, and for filling in the MQMD context. (If the application does not provide a userid/password, and it is not enforced by the AUTHINFO rule, then the OS userid is mapped to the DN as if it were the SHORTUSR, so it is not required that applications be updated, but it is recommended.) The common sample programs allow userids to be given to demonstrate the process. For example:

export MQSAMP\_USER\_ID=met

/opt/mqm/samp/bin/amqsput Q1 QMLDAP

Sample AMQSPUT0 start

Enter password: passw0rd

...

This also of course works with client connections, such as with **amqsputc**.

**Limitations**

There are two main limitations with this configuration

1. Amazon does not support SSL/TLS communication to the directory server. That means that you need to ensure the network configuration is suitably secure, and cannot be snooped on. From an MQ configuration perspective, this means that the AUTHINFO object's SECCOMM attribute must be set to NO. Of course, if you are running the AD service yourself, then TLS would be recommended.
2. Active Directory does not support complex authentication configurations when using the LDAP bind APIs. MQ is not calling Windows APIs, just the standard LDAP APIs. So that rules out configurations such as authenticating through cross-forest trust relationships.

And remember that all explicit use of LDAP for MQ authentication and authorisation, regardless of which directory implementation is being used, must have a way to return a unique short name for the user. Depending on your directory schema there may or may not be a convenient field already in existence.

**Summary**

In this article I've shown how you can use a Windows AD configuration to provide user and group directly to a non-Windows queue manager. This can simplify provisioning of operating systems and queue managers, with no need to create new OS identities.