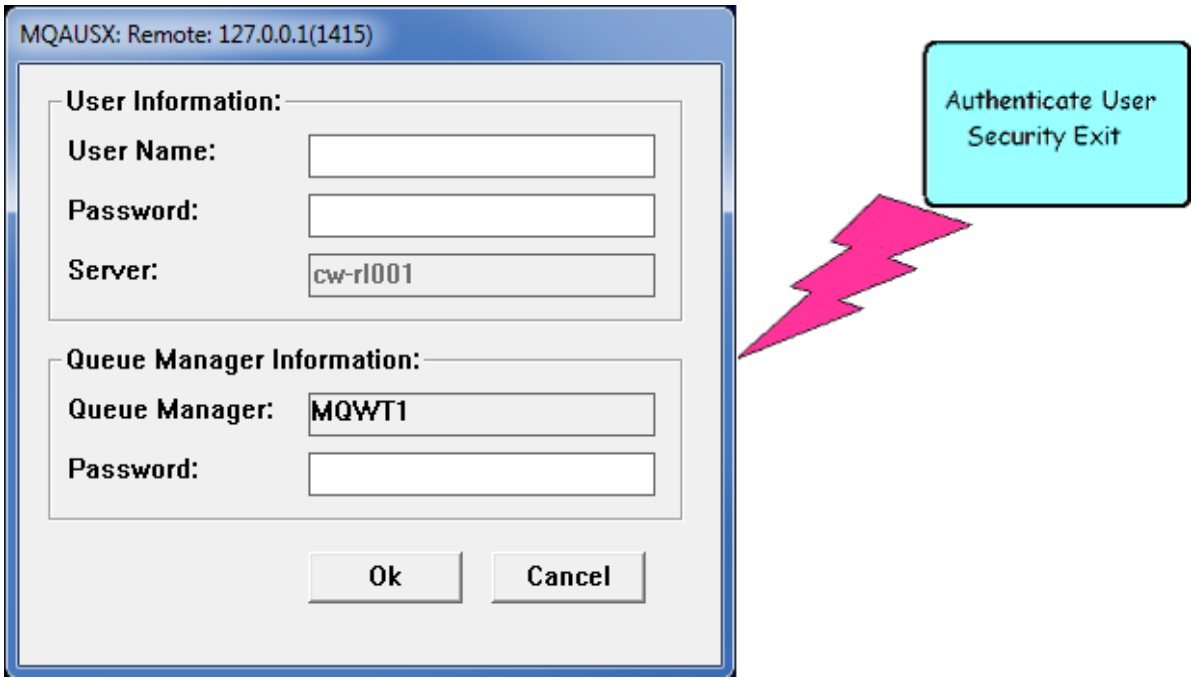
MQAUSX Server-side Installation and Operation Manual



Capitalware Inc.
Unit 11, 1673 Richmond Street, PMB524
London, Ontario N6G2N3
Canada
sales@capitalware.com
https://www.capitalware.com

Last Updated: January 2022.

Table of Contents

1 Introduction

1.1 Overview

MQ Authenticate User Security Exit (MQAUSX) is solution that allows a company to fully authenticate a user who is accessing a IBM MQ resource. It authenticates the user's UserId and Password (and possibly Domain Name) against the server's native OS system, LDAP server, Microsoft's Active Directory, Quest Authentication Services, Centrify's DirectControl, Unix/Linux PAM (Pluggable Authentication Module) or an encrypted MQAUSX FBA file.

The security exit will operate with IBM MQ v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2 in Windows, Unix and Linux environments. It works with Server Connection, Client Connection, Sender, Receiver, Server and Requestor channels of IBM MQ queue manager.

The MQ Authenticate User Security Exit solution is comprised of 2 components: client-side security exit and server-side security exit.

1.1.1 Client-Side Security Exit

The ***client-side security exit*** first checks if the server-side exit is defined for the particular channel. The client-side exit will receive a security token to be used in the encryption process of the user's password. It will prompt the user for his / her UserId and Password (and domain name for Windows), encrypt the data and send it to the server-side security exit.

For each connection attempt, the server-side security exit will verify that it is an acceptable client exit attempting the connection. If so, then the server-side will send a unique security token. When the server-side security exit receives the encrypted data, it will decrypt the incoming data and then perform UserId and Password (and domain) authentication against the native OS system, LDAP server, Microsoft's Active Directory, Quest Authentication Services, Centrify's DirectControl, Unix/Linux PAM (Pluggable Authentication Module) or an encrypted MQAUSX FBA file. If successful, the connection will be allowed.

1.1.2 Server-Side Security Exit

The ***server-side security exit*** supports the concept of 'Proxy IDs'. After a user has been successfully authenticated against the native OS system, LDAP server, Microsoft's Active Directory, Quest Authentication Services, Centrify's DirectControl, Unix/Linux PAM (Pluggable Authentication Module) or an encrypted MQAUSX FBA file and the 'Proxy Mode' flag is set, then the server-side security exit will look up the user's UserID in the Proxy file for their Proxy ID. The Proxy ID will be used for all MQ interactions.

An MQAdmin can define a password for a queue manager. Hence, when enabled, a back-end application and/or end-user would need to not only know their UserID and Password but also the queue manager's Password to successfully log in. Defining and requiring a queue manager password in MQAUSX is equivalent to adding perimeter security to your system.

The server-side security exit has the ability to allow or restrict users from logging in with the 'mqm' or 'MUSR_MQADMIN' or 'QMQM' UserIDs. This is controlled by the server-side security exit's property keyword 'Allowmqm'.

The server-side security exit has the capability to allow or limit the incoming channel connections according to the name of the associated Server Connection channel (SVRCONN). Each Server Connection channel can be allocated a maximum number of connections and the server-side security exit will ensure that this maximum is not exceeded.

Client connections to a queue manager are limited by either channel name or the 'DefaultMCC' property keyword in the initialization file. In today's use of J2EE applications, it is a possibility that one J2EE application could overwhelm the queue manager with client connections, thus preventing any connections being made from other applications.

The MQAdmin can enable Excessive Client Connections alerting system that counts the number of connections over a period of time (i.e. Day / Hour / Minute) and writes a message to the log when the count exceeds a particular value. If the keyword WriteToEventQueue is set to 'Y' then an event message is also written to an event queue. ECC feature is designed to catch applications that are poorly written, for example, applications that continuously connect and disconnect from the queue manager for every message sent or received.

The server-side security exit has the ability to allow or restrict the incoming IP address, hostname and/or SSL DN. The server-side security exit uses a regular expression parser to parse the incoming client IP address, hostname, and/or SSL DN against a predefined regular expression pattern.

The server-side security exit has the ability to allow or restrict the incoming UserID against a group. A list of groups can be queried for the incoming UserID. The groups can be in the local OS or a group file. If MQAUSX is authenticating against an LDAP server then the group querying can be against the LDAP server.

For those channels where authentication is not required, the server-side security exit can be set to not perform this function. This is controlled by the server-side security exit's property keyword 'NoAuth'.

The server-side security exit, when in non-authentication mode, has the ability to allow or restrict users from connecting with a blank UserID value. This is controlled by the server-side security exit's property keyword 'AllowBlankUserID'.

The server-side security exit, when in non-authentication mode, has the ability to allow or restrict the incoming UserID. The server-side security exit uses a regular expression parser to parse the incoming client UserID against a predefined regular expression pattern.

On AIX, HP-UX, Linux, Solaris and Windows, MQAUSX can be configured and used with a non-default installation of MQ in a multi-install MQ environment.

Note: Raspberry Pi is a Linux ARM 32-bit OS (Operating System). Hence, simply follow the Linux 32-bit instructions for installing and using the solution on a Raspberry Pi.

MQAUSX is 4 products in 1

1. If the client application is configured with the client-side security exit then the user credentials are encrypted and sent to the remote queue manager. This is the best level of security.

2. If the client application is not configured with the client-side security exit and the client-side **AND** server-side are at MQ V8 then MQ V8 will encrypt the user credentials as they flow from the client application to the queue manager.

3. If the client application is not configured with the client-side security exit then the user credentials are sent in plain text to the remote queue manager. This feature is available for Java/JMS, Java and C# DotNet client applications. For native applications (i.e. C/C++), then the application must use and populate the MQCSP structure with the UserID and Password.
    - Using MQAUSX with No Client-side Security Exit - Part 1 (coding examples) http://www.capitalware.com/rl_blog/?p=638
    - Using MQAUSX with No Client-side Security Exit - Part 2 (configuring tools like MQ Explorer, SupportPac MO71, MQ Visual Edit, etc..) http://www.capitalware.com/rl_blog/?p=659

4. If the MQAdmin sets the MQAUSX IniFile parameter NoAuth to Y then it functions just like MQ Standard Security Exit (MQSSX). MQSSX does not authenticate. It filters the incoming connection based on UserID, IP address, hostname and/or SSL DN.

1.2 Executive Summary

The ***MQ Authenticate User Security Exit*** solution is comprised of 2 components: client-side security exit and server-side security exit.

1.2.1 Server-Side Security Exit

The server-side security exit is available in 3 forms:

- Windows DLL
- Shared library for AIX, HP-UX, Linux, and Solaris.
- IBM i exit module

The major features of the server-side security exit are as follows:

- Authenticate a user against the server's native OS system, LDAP server, Microsoft's Active Directory, Quest Authentication Services*, Centrify's DirectControl*, PAM* or FBA file.
- Allows or restricts the incoming UserID against a Group
- Provides support for Proxy UserIDs
- Ability to assign a Password to a queue manager for client authentication
- Allows or restricts the incoming IP address against a regular expression pattern
- Allows or restricts the incoming SSL DN against a regular expression pattern
- Allows or restricts the incoming UserID against a regular expression pattern
- Allows or restricts the incoming AD server name against a regular expression pattern**
- Allows or restricts the use of 'mqm', 'MUSER_MQADMIN' or 'QMQM' UserIDs
- Ability to turn off server-side authentication
- Ability to set the maximum number of allowable connections per a given channel (MCC)
- Ability to monitor for excessive client connections (ECC) and then generate an alert
- Provides monitoring tool tie-in by using custom MQ event messages
- Provides logging capability for all connecting client applications regardless if they are successful or not.

* Unix/Linux only
** Windows Only

1.2.2 Client-Side Security Exit

The client-side security exit is available in 5 forms:

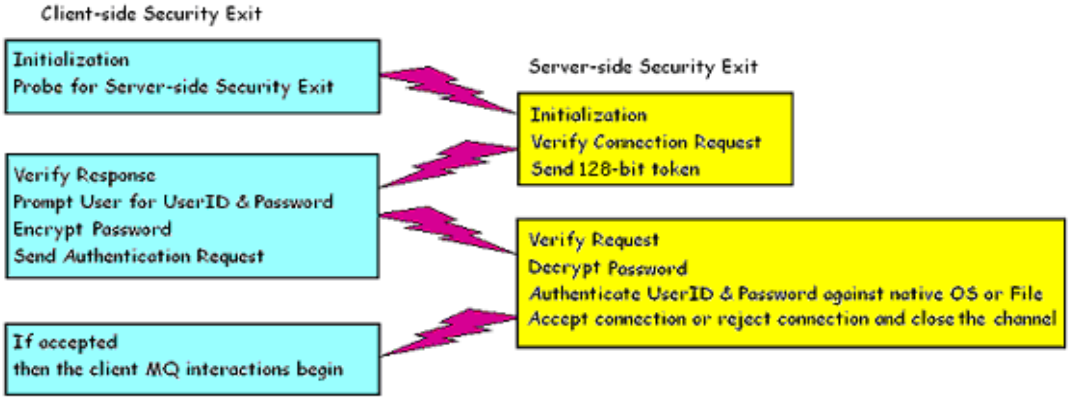
- Windows DLL (32-bit & 64-bit)
- Windows DLL for managed .NET (32-bit & 64-bit)
- Java JAR
- Non-GUI shared library for AIX, HP-UX, Linux, and Solaris.
- IBM i exit module

The client-side security exit has been tested against the following MQ client programs:

- IBM's MQ Explorer v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2
- SupportPac MO71 (MQMon)
- IBM's WBIMB Eclipse Tool Kit
- WebSphere Message Broker Explorer V8.0 or higher
- IBM DataPower
- BMC Middleware Management - Administration (BMM Admin)
- BMC's Administration for IBM MQ (AppWatch)
- webMethods MQ Adapter
- Mercury's SiteScope
- Capitalware's MQ Visual Edit
- Capitalware's MQ Visual Browse
- Capitalware's MQ Batch Toolkit
- Capitalware's Universal File Mover
- J2EE application servers i.e. WAS, WebLogic, Jboss, etc...
- Any program that uses Client Channel Tables (i.e. SupportPac MS03, WatchQ, etc.)

1.3 Context Diagram (Logical View)

1.4 Security Message Flow (Logical View)



1.5 Prerequisites

This section provides the minimum supported software levels. These prerequisites apply to both client-side and server-side installations of MQ Authenticate User Security Exit.

1.5.1 Operating System

MQ Authenticate User Security Exit can be installed on any of the following supported servers:

1.5.1.1 IBM AIX

- IBM AIX 6L version 6.1 or higher

1.5.1.2 HP-UX IA64

- HP-UX v11.23 or higher

1.5.1.3 IBM i (OS/400)

- IBM i V6R1 or higher

1.5.1.4 Linux x86

- Red Hat Enterprise Linux v5, v6, v7, v8
- SUSE Linux Enterprise Server v11, v12, v15

1.5.1.5 Linux x86_64 (64-bit)

- Red Hat Enterprise Linux v5, v6, v7, v8
- SUSE Linux Enterprise Server v11, v12, v15

1.5.1.6 Linux on POWER

- Red Hat Enterprise Linux v5, v6, v7, v8
- SUSE Linux Enterprise Server v11, v12, v15

1.5.1.7 Linux on zSeries (64-bit)

- Red Hat Enterprise Linux v5, v6, v7, v8
- SUSE Linux Enterprise Server v11, v12, v15

1.5.1.8 Raspberry Pi (Linux ARM 32-bit)

- Raspberry Pi OS v9 or higher

1.5.1.9 Sun Solaris

- Solaris SPARC v10 & v11
- Solaris x86_64 v10 & v11

1.5.1.10 Windows

- Windows 2008, 2012 or 2016 Server (32-bit & 64-bit)
- Windows 7, 8, 8.1 & 10 (32-bit & 64-bit)

1.5.2 IBM MQ

- IBM MQ v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2 (both 32-bit and 64-bit)

Operating System	MQ v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2
AIX v6.1 or higher	64-bit
HP-UX IA64 v11.23 or higher	64-bit
IBM i (OS/400)	64-bit
Linux x86	32-bit
Linux x86_64	64-bit
Linux on POWER	64-bit
Linux on zSeries	32-bit & 64-bit
Raspberry Pi ARM	32-bit
Solaris SPARC v8 or higher	64-bit
Solaris x86_64 v10	64-bit
Windows 2008, 2012, 2016, 7, 8, 8.1 & 10	32-bit & 64-bit

1.5.3 LDAP Server

- Microsoft's Active Directory for Windows 2000 Server or higher
- Novell's eDirectory v8 or higher
- Oracle 9i Internet Directory or higher
- OpenLDAP v2.1 or higher
- IBM Tivoli Directory Server
- z/OS Integrated Security Services LDAP Server v1.6 or higher

1.5.3.1 Dependencies

- An LDAP client needs to be installed on the same server as the MQ queue manager.
- To use SSL, the LDAP server and the LDAP client must be configured for SSL.

1.5.4 Windows 64-bit

The following is the software prerequisite for Windows 64-bit:

- Microsoft Visual C++ 2010 Redistributable Package (x64)
  https://download.microsoft.com/download/1/6/5/165255E7-1014-4D0A-B094-B6A430A6BFFC/vcredist_x64.exe

2 Installing MQ Authenticate User Security Exit

This section describes how to install Capitalware's MQ Authenticate User Security Exit. For Windows, it is available as a Windows installer package called: **mqausx-server-setup.exe**. When the user runs this package, it will install server-side security exit.

2.1 Server-side Security Exit Installation

2.1.1 Windows 32-bit Installation

To install the security exit on Windows, first unzip the **mqausx.zip** and then run the ***mqausx-server-setup-32bit.exe*** file from the ***Windows-Server*** directory. Follow the on-screen instructions and the security exit will be installed in the **C:\Capitalware\MQAUSX** directory (default installation).

The user may copy or ftp the mqausx.dll, mqausxldap.exe, mqausxldapssl.exe, mqausx.ini, AddRegistryEntries.bat and mqausx.reg files from one Windows server to another Windows server.

To use the MQAUSX LDAP component on Windows, the MQAUSX install directory needs to be added to the System's PATH environment variable.

Steps to configure the System PATH environment variable on Windows NT / 2000 / 2003 / 2008 / 2012 / XP / Vista / 7 / 8 / 8.1:

1. Open the Control Panel
2. Click on the **System** icon
3. Select the **Advanced** tab
4. Click the **Environment Variables** button
5. Select **PATH** variable from the **System variables** (not User Variables) and then click the **Edit** button
6. Update the **Variable value** with the MQAUSX installation path i.e. C:\Capitalware\MQAUSX
7. Click the **OK** button on the **Edit System Variable** window
8. Click the **OK** button on the **Environment Variables** window
9. Click the **OK** button on the **System Properties** window

2.1.2 Windows 64-bit Installation (MQ v8.0 & higher)

To install the security exit on Windows, first unzip the **mqausx.zip** and then run the ***mqausx-server-setup-64bit.exe*** file from the ***Windows-Server*** directory. Follow the on-screen instructions and the security exit will be installed in the **C:\Capitalware\MQAUSX** directory (default installation).

The user may copy or ftp the mqausx.dll, mqausxldap.exe, mqausxldapssl.exe, mqausx.ini, AddRegistryEntries.bat and mqausx.reg files from one Windows server to another Windows server.

To use the MQAUSX LDAP component on Windows, the MQAUSX install directory needs to be added to the System's PATH environment variable.

Steps to configure the System PATH environment variable on Windows 2003 / 2008 / 2012 / Vista / 7 / 8 / 8.1:

1. Open the Control Panel
2. Click on the **System** icon
3. Select the **Advanced** tab
4. Click the **Environment Variables** button
5. Select **PATH** variable from the **System variables** (not User Variables) and then click the **Edit** button
6. Update the **Variable value** with the MQAUSX installation path i.e. C:\Capitalware\MQAUSX
7. Click the **OK** button on the **Edit System Variable** window
8. Click the **OK** button on the **Environment Variables** window
9. Click the **OK** button on the **System Properties** window

2.1.3 Linux 32-bit Installation

To install the 32-bit version of MQAUSX on Linux, first unzip the **mqausx.zip** and then select the appropriate TAR file for the target platform. You will find the following 2 TAR files in the original ZIP file:

- **Linux_x86/mqausx_linux.tar**
- **RaspberryPi_ARM/mqausx_raspberrypi_arm.tar**

You will need to ftp or copy the selected TAR file to the target platform to the /var/mqm/exits directory. Next telnet to the Unix or Linux server and 'cd' (change directory) to the /var/mqm/exits directory and untar the TAR file.

i.e. For AIX, do the following command:

```
cd /var/mqm/exits/
tar -xvf  mqausx_linux.tar
```

2.1.4 Unix and Linux 64-bit Installation

To install the 64-bit version of MQAUSX on Unix or Linux, first unzip the **mqausx.zip** and then select the appropriate TAR file for the target platform. You will find the following 7 TAR files in the original ZIP file:

- **AIX/mqausx_aix71_64.tar** for AIX v7.1 or higher
- **HPUX_IA64/mqausx_hpux64.tar**
- **Linux_x86_64/mqausx_linux_x86_64.tar**
- **Linux_POWER/mqausx_linux_power64.tar**
- **Linux_zSeries/mqausx_linux_zseries64.tar**
- **Solaris_SPARC/mqausx_solaris10_64.tar** for Solaris SPARC v10 or higher
- **Solaris_x86_64/mqausx_solaris_x86_64.tar**

You will need to ftp or copy the selected TAR file to the target platform to the /var/mqm/exits64 directory. Next telnet to the Unix or Linux server and 'cd' (change directory) to the /var/mqm/exits64 directory and untar the TAR file.

i.e. For Linux AIX 7.1, do the following command:

```
cd /var/mqm/exits64/
tar -xvf  mqausx_aix71_64.tar
```

2.1.5 IBM i Installation

To install the MQAUSX on IBM i , first unzip the **mqausx.zip** and then select the files in the **IBM i** (IBM i) directory.

- **mqausx.savf** is the IBM i 'Save File' that contains the library with the security exit.
- **mqausx_iseries.tar** is the IBM i IFS TAR file that contains a sample initialization file for the server-side security exit and sample MQSC script to define MQ channels with the security exits.

Steps to install the server-side security exit:

1. Log onto the target IBM i server and do the following command:

   ```
   CRTSAVF FILE(QGPL/MQAUSX)
   ```

2. ftp the IBM i files to the IBM i server as follows:

   ```
   ftp -s:mqausx_iseries.ftp  iseries_hostname
   ```

   ```
   your-IBM i-userid
   your-IBM i-password

   binary
   cd QGPL
   put mqausx.savf

   quote SITE NAMEFMT 1

   cd /QIBM/UserData/mqm/
   put mqausx_iseries.tar
   quit
   ```

3. Log onto the target IBM i server and do the following commands:

   ```
   RSTLIB SAVLIB(MQAUSX) DEV(*SAVF) SAVF(QGPL/MQAUSX)
   CLRSAVF FILE(QGPL/MQAUSX)
   CHGOBJOWN  OBJ(MQAUSX) OBJTYPE(*LIB)  NEWOWN(QMQM)
   qsh
   cd /QIBM/UserData/mqm/
   tar -xvf mqausx_iseries.tar
   chown -R QMQM mqausx
   rm mqausx_iseries.tar
   ```

2.1.6 MQAUSX-GUI Installation

This section will describe how to install the MQAUSX-GUI. The user will find 2 files in the software package listed as follows:

- **MQAUSX-GUI/mqausxgui-wthjre.exe** (for Windows)
- **MQAUSX-GUI/mqausxgui.zip** (for Unix, Linux or macOS)

2.1.6.1 MQAUSX-GUI Installation on Windows

To install MQAUSX-GUI on Windows, run the **mqausxgui-withjre.exe** file located in the MQAUSX-GUI directory. Follow the on-screen instructions and the program will be installed in the **C:\Capitalware\MQAUSX-GUI** directory (default installation).

2.1.6.2 MQAUSX-GUI Installation on Unix, Linux or macOS

To install MQAUSX-GUI on Unix or Linux, you will need to ftp or copy the selected TAR file to the target platform to the /home/mqm/ directory. Next, one must telnet to the Unix, Linux or macOS server and 'cd' (change directory) to the /home/mqm/ directory and unzip the archive file.

i.e. Do the following command:

```
unzip mqausxgui.zip
```

2.2 Client-side Security Exit

For more information, please read the ***MQAUSX Client-side Configuration*** Manual.

3 Security Configuration

This section describes the necessary steps to enable the server-side security exit to perform the UserId and Password (and Domain Name) verification. The server-side security exit will work with IBM MQ v7.1, v7.5, v8.0, v9.0, v9.1 and v9.2 or higher.

3.1 Windows Security Configuration

This section describes how to configure Windows to allow the server-side security exit to perform the UserId and Password (and Domain Name) verification.

Note: This section is only applicable to older releases of Windows NT and Windows 2000. Newer releases of Windows 2003 / 2008 / 2012 Server do not require this configuration.

Before doing the following steps, first attempt to use the server-side security exit without the modification. If the server-side server exit outputs the error message 'ERROR_PRIVILEGE_NOT_HELD (1314) security error' then proceed with the modification.

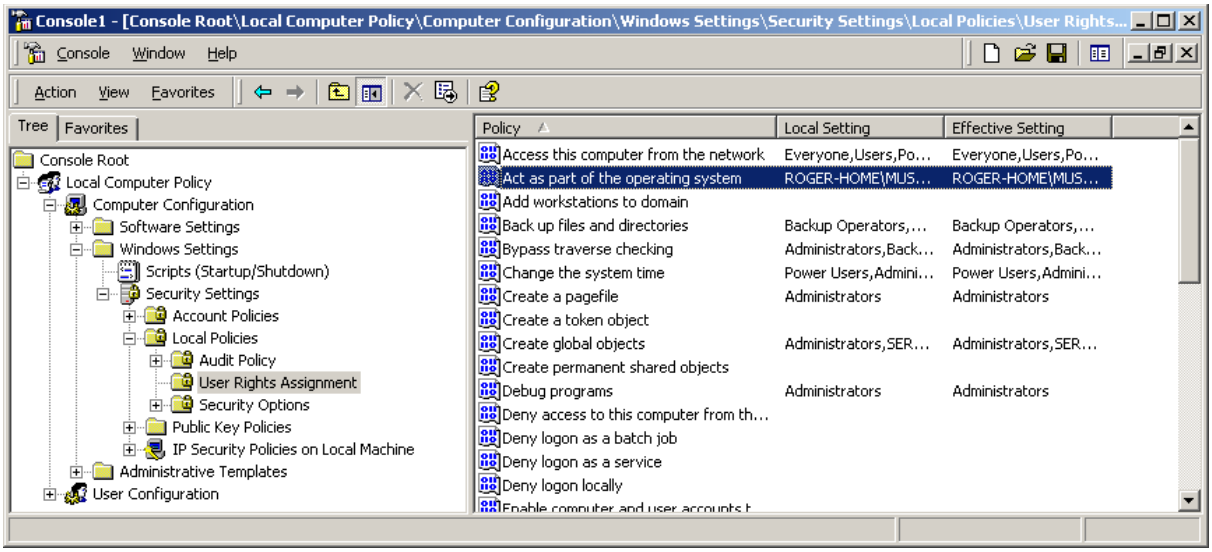
Steps to configure Windows 2000 Server:

1. Click the **Start** button, then select the **Control Panel**
2. Double click the **Administrative Tools** icon
3. Double click the **Group Policy** icon
   Then go to step 8 below (see next page)

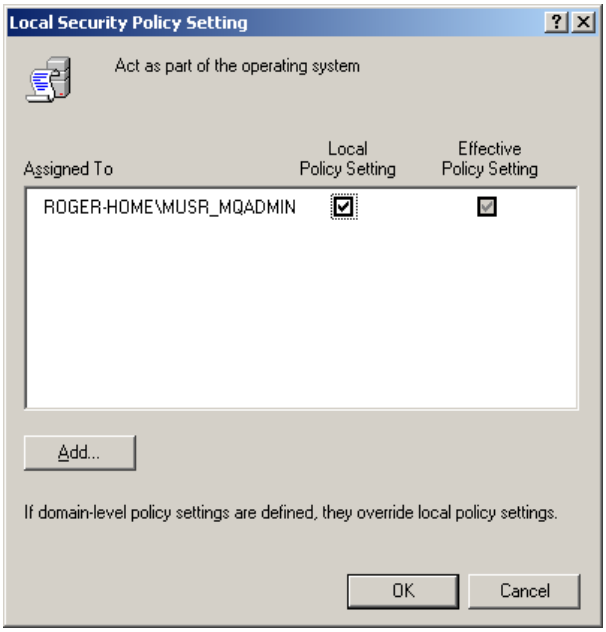
Steps to configure Windows 2000 Pro or Windows NT or Windows XP Pro:

1. Click the **Start** button, then select the **Run**
2. Type **mmc** and click the **OK** button
3. Select File and then select **Add/Remove Snap-in** menu item
4. In the **Add/Remove Snap-in** window, on the **Standalone** tab, then click the **Add** button
5. Scroll down, select **Group Policy**, click the **Add** button and then the **Finish** button
6. Click the **Close** button on the **Add Standalone Snap-in** window
7. Click the **OK** button on the **Add/Remove Snap-in** window
   Then go to step 8 below (see next page)

8. Expand the Console Tree pane as follows:
   - Local Computer Policy
   - Windows Settings
   - Security Settings
   - Local Policies
   - User Rights Assignment
   - Act as part of the operating system



9. Double click on 'Act as part of the operating system'
10. Add the **MUSR_MQADMIN** UserID to this Window



11. Click **OK**.
12. Because **MUSR_MQADMIN** is a system account, you need to **reboot** the box.

3.2 AIX, HP-UX, Solaris and Linux Security Configuration

This section describes how to configure AIX, HP-UX, Solaris and Linux to allow the server-side security exit to perform the UserId and Password verification.

3.2.1 Normal Security

Steps to configure server-side security exit:

IBM MQ 32-bit

1. ftp ***mqausx*** and ***mqausxvfy*** to ***/var/mqm/exits/*** sub-directory or un-tar the mqausx_xxx.tar file into the ***/var/mqm/exits/*** sub-directory (xxx is either aix, hpux, solaris or linux, see section 2.1.2 more information)
2. Change directory to ***/var/mqm/exits/***

IBM MQ 64-bit

1. ftp ***mqausx*** and ***mqausxvfy*** to ***/var/mqm/exits64/*** sub-directory or un-tar the mqausx_xxx.tar file into the ***/var/mqm/exits64/*** sub-directory (xxx is either aix, hpux, solaris or linux, see section 2.1.3 more information)

2. Change directory to ***/var/mqm/exits64/***

3. Next, do the following commands:

```
chown mqm:mqm mqausx cwchad mqausx*.so testldap* enc_*
chmod 550 mqausx cwchad mqausx*.so testldap* enc_*

chown mqm:mqm mqausxclnt cwdspver
chmod 755 mqausxclnt cwdspver

chown mqm:mqm *.jar
chmod 644 *.jar
```

4. Finally, do the following commands, ***you will need to be logged in as root:***

```
chown root:mqm mqausxvfy mqausxqas
chmod 550 mqausxvfy mqausxqas
chmod u+s mqausxvfy mqausxqas
```

The above commands are included in a Unix shell scripts called: ***setausx.sh*** . To execute setausx.sh shell script, first the user will need to enable execution:

```
chmod +x setausx.sh
```

Then ***login as root*** and run the script as follows:

```
./setausx.sh
```

3.2.2 High-level of Security

Steps to configure server-side security exit:

1. Create a Unix group called: ***mqausx***
2. Put mqm in the mqausx group - ***do not put any other UserIDs in this new Unix group.***

IBM MQ 32-bit

3. ftp ***mqausx*** and ***mqausxvfy*** to ***/var/mqm/exits/*** sub-directory or un-tar the mqausx_xxx.tar file into the ***/var/mqm/exits/*** sub-directory (xxx is either aix, hpux, solaris or linux, see section 2.1.2 more information)
4. Change directory to ***/var/mqm/exits/***

IBM MQ 64-bit

3. ftp ***mqausx*** and ***mqausxvfy*** to ***/var/mqm/exits64/*** sub-directory or un-tar the mqausx_xxx.tar file into the ***/var/mqm/exits64/*** sub-directory (xxx is either aix, hpux, solaris or linux, see section 2.1.3 more information)

4. Change directory to ***/var/mqm/exits64/***

5. Next, do the following commands:

```
chown mqm:mqausx mqausx cwchad mqausx*.so testldap* enc_*
chmod 550 mqausx cwchad mqausx*.so testldap* enc_*

chown mqm:mqausx mqausxclnt cwdspver
chmod 755 mqausx mqausxclnt cwdspver

chown mqm:mqm *.jar
chmod 644 *.jar
```

6. Next, do the following commands against ***mqausxvfy***, ***you will need to be logged in as root***:

```
chown root:mqausx mqausxvfy mqausxqas
chmod 550 mqausxvfy mqausxqas
chmod u+s mqausxvfy mqausxqas
```

3.3 IBM i Security Configuration

There are no specific security settings for MQAUSX on IBM i.

4 Configuring Server-side Security Exit

This section describes how to configure the server-side security exit.

4.1 MQAUSX Authentication

Starting with IBM v8.0, IBM has included a basic authentication feature in the base product. Exiting queue managers that are migrated to MQ v8.0 or higher will have this feature disabled but when the MQAdmin creates a new queue manager, this feature will be enabled. Hence, to use MQAUSX's authentication, issue the following MQSC command to disable the builtin authentication mechanism:

```
ALTER QMGR CONNAUTH(' ')
```

4.2 MQAUSX Filtering

Starting with IBM v7.1, IBM has included a feature called channel authentication record. Channel authentication record feature allows for the filtering of incoming client connections. Exiting queue managers that are migrated to MQ v7.1 or higher will have this feature disabled but when the MQAdmin creates a new queue manager, this feature will be enabled. Hence, to use MQAUSX's filtering feature, issue the following MQSC command to disable channel authentication record mechanism:

```
ALTER QMGR CHLAUTH(DISABLED)
```

4.3 Security User Data (SCYDATA)

Security User Data (SCYDATA) field must NOT exceed 32 characters. In order to work with this limitation, MQAUSX supports 3 ways to specify an IniFile path: absolute path, relative path and environment variable.

Note: The IniFile path that is determined by MQAUSX server-side security exit will also be used for the following IniFile keywords (if no pathing is specified for these keywords): FBAFile, LicenseFile, LogFile, ProxyFile and SSLCertFileName.

4.3.1 Absolute Path

Absolute pathing (specifying the complete path) for the SCYDATA works on Linux, Unix and Windows platforms.

E.g. Windows

```
DEFINE CHANNEL ('SYSTEM.ADMIN.SVRCONN') CHLTYPE(SVRCONN) +
       TRPTYPE(TCP) +
       SCYEXIT('C:\Capitalware\MQAUSX\mqausx(SecExit)') +
       SCYDATA('C:\Capitalware\MQAUSX\mqausx.ini') +
       REPLACE
```

Hence, MQAUSX will use the following path as the IniFile path:
C:\Capitalware\MQAUSX

4.3.2 Relative Path

Relative pathing for the SCYDATA is supported on Linux, IBM i, Unix and Windows platforms. MQAUSX will extract the path from SCYEXIT field and prefix it to the IniFile specified in the SCYDATA field in order to locate the IniFile.

E.g. Unix

```
DEFINE CHANNEL ('SYSTEM.ADMIN.SVRCONN') CHLTYPE(SVRCONN) +
       TRPTYPE(TCP) +
       SCYEXIT('/var/mqm/exits/mqausx(SecExit)') +
       SCYDATA('mqausx.ini') +
       REPLACE
```

Hence, MQAUSX will use the following path as the IniFile path:
/var/mqm/exits/

4.3.3 Environment Variables

4.3.3.1 Global Environment Variable

MQAUSX supports the use of the MQAUSX_HOME environment variable which holds the directory path information. MQAUSX_HOME environment variable is supported on Linux, IBM i, Unix and Windows platforms.

E.g. Unix

```
export MQAUSX_HOME=/really/long/path/MQHA/QMgrName/data/
```

```
DEFINE CHANNEL ('SYSTEM.ADMIN.SVRCONN') CHLTYPE(SVRCONN) +
       TRPTYPE(TCP) +
       SCYEXIT('/var/mqm/exits64/mqausx(SecExit)') +
       SCYDATA('mqausx.ini') +
       REPLACE
```

Hence, MQAUSX will use the following path as the IniFile path:
/really/long/path/MQHA/QMgrName/data/

4.3.3.2 Queue Manager Specific Environment Variable

MQAUSX supports the use of the MQAUSX_HOME environment variable post-fixed with the queue manager name which holds the directory path information. MQAUSX_HOME environment variable is supported on Linux, IBM i, Unix and Windows platforms.

e.g. Unix with a queue manager name of MQA1

```
export MQAUSX_HOME_MQA1=/really/long/path/MQHA/QMgrName/data2/
```

```
DEFINE CHANNEL ('SYSTEM.ADMIN.SVRCONN') CHLTYPE(SVRCONN) +
       TRPTYPE(TCP) +
       SCYEXIT('/var/mqm/exits64/mqausx(SecExit)') +
       SCYDATA('mqausx.ini') +
       REPLACE
```

Hence, MQAUSX will use the following path as the IniFile path:
/really/long/path/MQHA/QMgrName/data2/

Note: If both environment variables are specified then the queue manager specific environment variable will be used.

4.4 SVRCONN Channel

This section describes the necessary entries to enable the server-side security exit. The MQ Administrator will need to update 2 fields of the SVRCONN Channel that the server-side security exit will be applied to.

Platform		Directory	Exit Module Name
Windows	32-bit	C:\Capitalware\MQAUSX\	mqausx.dll
Linux/Unix	32-bit	/var/mqm/exits/	mqausx
Linux/Unix	64-bit	/var/mqm/exits64/	mqausx

MQAUSX now supports MQ's multi-install in a non-default directory.

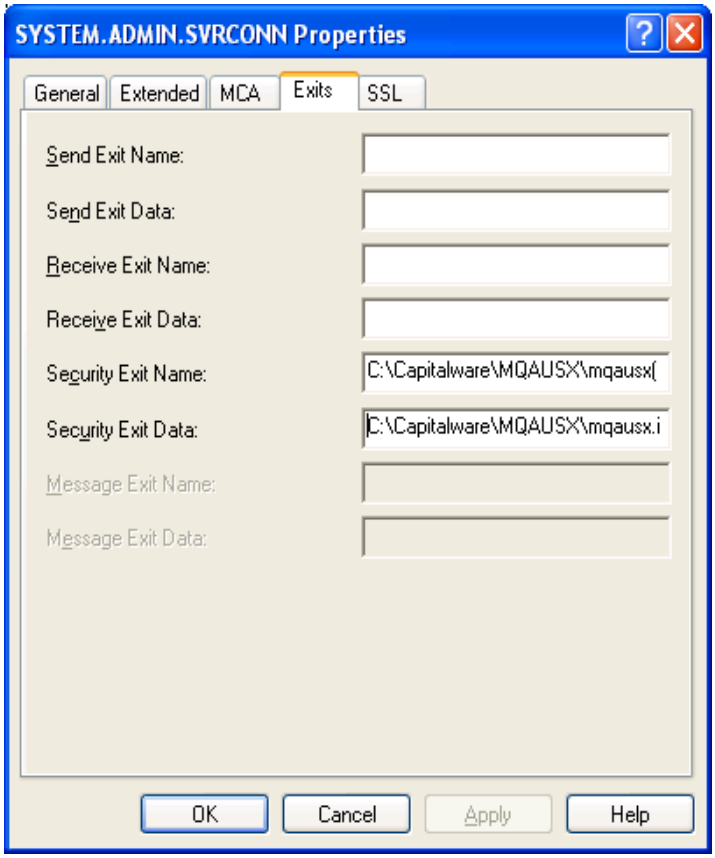
Note: The Security Exit Data (SCYDATA) field must NOT exceed 32 characters.

4.4.1 Windows

On Windows, SCYEXIT and SCYDATA will contain the following values assuming a default install:

- SCYEXIT
  `C:\Capitalware\MQAUSX\mqausx(SecExit)`
- SCYDATA
  `C:\Capitalware\MQAUSX\mqausx.ini`

```
DEFINE CHANNEL ('SYSTEM.ADMIN.SVRCONN') CHLTYPE(SVRCONN) +
       TRPTYPE(TCP) +
       SCYEXIT('C:\Capitalware\MQAUSX\mqausx(SecExit)') +
       SCYDATA('C:\Capitalware\MQAUSX\mqausx.ini') +
       REPLACE
```



4.4.2 Linux 32-bit

On Linux, SCYEXIT and SCYDATA will contain the following values assuming a default install:

- SCYEXIT
  /var/mqm/exits/mqausx(SecExit)
- SCYDATA
  /var/mqm/exits/mqausx.ini

```
DEFINE CHANNEL ('SYSTEM.ADMIN.SVRCONN') CHLTYPE(SVRCONN) +
       TRPTYPE(TCP) +
       SCYEXIT('/var/mqm/exits/mqausx(SecExit)') +
       SCYDATA('/var/mqm/exits/mqausx.ini') +
       REPLACE
```

4.4.3 Unix and Linux 64-bit

On Unix and Linux (excluding Linux x86), SCYEXIT and SCYDATA will contain the following values assuming a default install:

- SCYEXIT
  /var/mqm/exits64/mqausx(SecExit)
- SCYDATA
  /var/mqm/exits64/mqausx.ini

```
DEFINE CHANNEL ('SYSTEM.ADMIN.SVRCONN') CHLTYPE(SVRCONN) +
       TRPTYPE(TCP) +
       SCYEXIT('/var/mqm/exits64/mqausx(SecExit)') +
       SCYDATA('/var/mqm/exits64/mqausx.ini') +
       REPLACE
```

4.4.4 IBM i

On IBM i, SCYEXIT and SCYDATA will contain the following values assuming a default install:

- SCYEXIT is made up of 10 characters for program name (padded with blanks) followed by 10 characters for the LIBRARY name (padded with blanks).

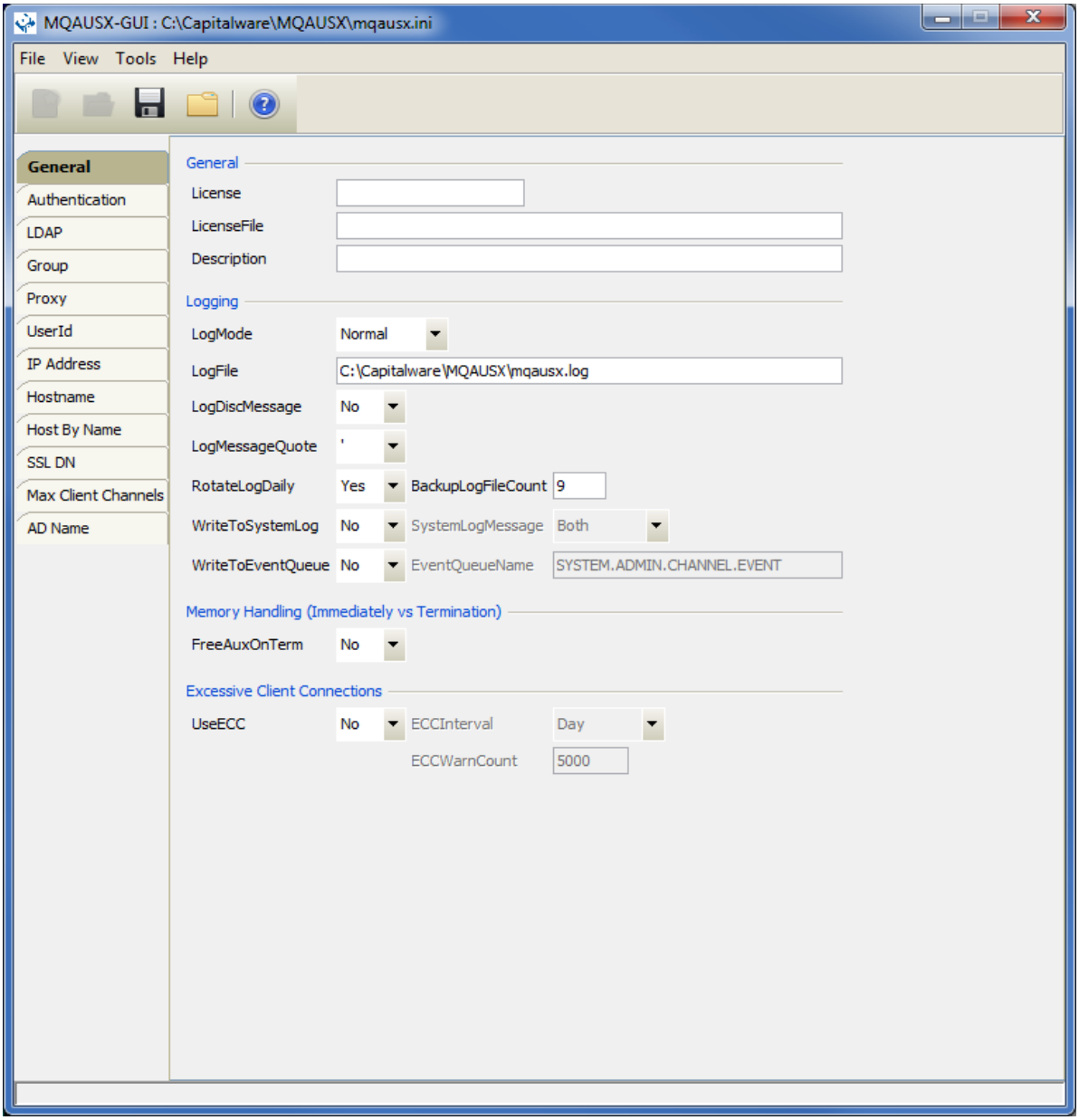
  MQAUSX MQAUSX

- SCYDATA
  mqausx.ini

```
DEFINE CHANNEL ('SYSTEM.ADMIN.SVRCONN') CHLTYPE(SVRCONN) +
       TRPTYPE(TCP) +
       SCYEXIT('MQAUSX    MQAUSX    ') +
       SCYDATA('mqausx.ini') +
       REPLACE
```

4.5 MQAUSX-GUI

This section briefly describes the new graphical program called MQAUSX-GUI. This program assists the user in creating and managing their MQAUSX IniFiles. For more information, please see the ***MQAUSX-GUI User Guide*** manual.



5 IniFile Keywords (Server-side)

This section describes IniFile keywords.

5.1 Logging

This section describes the necessary entries to enable MQAUSX to record log information. To enable and control logging, there are 10 keywords in the IniFile:

1. **LogMode** specifies what type of logging the user wishes to have. LogMode supports 4 values [Q / N / V / D] where Q is Quiet, N is Normal, V is Verbose and D is Debug. The default value is N.

2. **LogFile** specifies the location of the log file. The default is as follows:

   For Windows:
   ```
   LogFile=C:\Capitalware\MQAUSX\mqausx.log
   ```

   For IBM MQ 32-bit on Unix and Linux:
   ```
   LogFile=/var/mqm/exits/mqausx.log
   ```

   For IBM MQ 64-bit on Unix and Linux:
   ```
   LogFile=/var/mqm/exits64/mqausx.log
   ```

   For IBM MQ on IBM i:
   ```
   LogFile=/QIBM/UserData/mqm/mqausx/mqausx.log
   ```

   Token Replacement for LogFile keyword:
   - **%QM%** - Substitutes the name of the queue manager
   - **%CHL%** - Substitutes the name of the channel
   - **%UID%** - Substitutes the UserID
   - **%PID%** - Substitutes the Process ID
   - **%TID%** - Substitutes the Thread ID

3. **LogDiscMessage** specifies whether or not MQAUSX write a disconnect message when the client application closes the channel. The default value is No.

4. **LogMessageQuote** specifies the type of quote (single or double) to be used on the log message. The default value is ' (single quote).

5. **RotateLogDaily** specifies whether or not the log files will be rotated on a daily basis. Setting ' RotateLogDaily' to 'Y' (Yes) will activate this feature; otherwise, the log files will left as is. The default value is Y.

   The RotateLogDaily feature will keep up to 9 backup log files (see BackupLogFileCount keyword for more information). The first connection request after midnight (and not at midnight) will cause it to roll / rotate the log files. If there are already 9 backup log files then the ninth backup log file will be deleted and 8 becomes 9, 7 becomes 8, etc...

6. **BackupLogFileCount** specifies the number of backup log files that should be kept by the security exit. The default value is 9. This keyword is only used if RotateLogDaily is set to 'Y'.

7. **WriteToSystemLog** specifies whether or not MQAUSX will write a log entry to the server's 'logging system'. On Windows, the server's 'logging system' is the Event Log and on Unix / Linux, it is the syslog. Setting ' WriteToSystemLog' to 'Y' (Yes) will activate this feature; otherwise, there will be no logging to the server's logging system. The default value is N.

   The location of the Unix / Linux syslog output can be found for each operating system as follows:
   - AIX: /var/log/messages
   - HP-UX: /var/adm/syslog/syslog.log
   - Linux: /var/log/messages
   - Solaris: /var/adm/messages

   The Windows Event log can be viewed using the Windows Event Viewer.

   Note: If the user did not install MQAUSX server-side security exit using the windows installer then the user will need to run the AddRegistryEntries.bat file to add the registry entries in order to view the MQAUSX log messages in the Event Viewer.

8. **SystemLogMessage** specifies what messages will be written to the system log.. SystemLogMessage supports 3 values [B / A / R] where B is Both, A is Accepted Only, and R is Rejected Only messages. The default value is B.

9. **WriteToEventQueue** specifies whether or not MQAUSX will write an event message containing the log entry information to the event queue. The default value is N.

   WriteToEventQueue provides the ability to write custom MQ Events to System Channel Event Queue to allow MQAUSX to be tied into an MQ Monitoring tool.
   - 9101 for Connection rejected (Authentication failed) event message
   - 9201 for MCC Warning event message
   - 9202 for MCC Exceeded event message
   - 9301 for ECC Warning event message

10. **EventQueueName** specifies the event queue name. The default value is 'SYSTEM.ADMIN.CHANNEL.EVENT'.

```
LogMode=N
LogFile=C:\Capitalware\MQAUSX\mqausx.log
RotateLogDaily=Y
WriteToSystemLog=Y
WriteToEventQueue=Y
EventQueueName=SYSTEM.ADMIN.CHANNEL.EVENT
```

5.2 Order of Authentication

This section describes the necessary steps to enable UserId and Password against multiple authentication sources and the order in which these sources will be tested. Currently, MQAUSX supports 6 authentication sources: ldap, files, cdc, qas, pam and mqausx.

- **UseAuthOrder** allows the user to specify the authentication methods and order of these methods.
- **AuthOrder** specifies which authentication method to be executed and the order of execution. AuthOrder supports the following 5 values:
    - **ldap** means the authentication will be against a LDAP server
    - **files** means the authentication will be against the local OS
    - **cdc** means the authentication will be against Centrify's DirectControl*
    - **qas** means the authentication will be against a Quest Authentication Services*
    - **pam** means the authentication will be against PAM*
    - **mqausx** means the authentication will be against MQAUSX formatted file (i.e. FBA).

Note: If more than authentication method is specified for AuthOrder parameter then the authentication order will be from left to right.

```
UseAuthOrder=Y
AuthOrder=ldap files mqausx
```

* Only supported on AIX, HP-UX, Linux Solaris.

5.3 File Based Authentication

This section describes the necessary steps to enable 'File Based Authentication'. By default, the server-side security exit will do UserId and Password (and Domain Name) authentication against the native OS (Operating System). The company or MQ Administrator can choose to have authentication against a file-based look-up system.

5.3.1 Encrypted FBA

5.3.1.1 Encrypted File Based Authentication Configuration

It is strongly recommended that native OS authentication is used. To enable the encrypted server-side file-based authentication, 2 keywords are needed in the IniFile:

- **UseFBA** allows the UserId and Password to be verified against a file rather than the OS
- **FBAFile** specifies the file name and location of the encrypted FBA file to do the UserId and Password verification

```
UseFBA=Y
FBAFile=c:\fba.enc
```

5.3.1.2 Encrypted file Management

Follow the instructions in Appendix C for creating / managing an encrypted server-side FBA file.

5.3.2 Plain Text FBA

5.3.2.1 File Based Authentication Configuration

It is strongly recommends that you use native OS authentication. To enable the file-based authentication, you need 2 keywords in the IniFile:

- **UseFBA** allows the UserId and Password to be verified against a file rather than the OS
- **FBAFile** specifies the file name and location of the file to do the UserId and Password verification

```
UseFBA=Y
FBAFile=c:\userlist.auth
```

5.3.2.2 File Based Authentication File Layout

The following table specifies the format of the File Based Authentication File:

File Based Verification – File Layout

Field Name	Description	Req'd Y/N	Min/ Max	Data Type	Format	Start Pos.	End Pos.	Justified	Pad Character
UserID	UserID of the user logging in	Y	1/32	Char	Alpha/Numeric	1	32	Left	Space
Password	The password for the specified UserID	Y	1/32	Char	Alpha/Numeric	33	64	Left	Space

5.4 Centrify's DirectControl Based Authentication

This section describes the necessary steps to enable Centrify's DirectControl authentication. By default, the server-side security exit will do UserId and Password authentication against the native OS (Operating System). The company or MQ Administrator can choose to have authentication against Centrify's DirectControl. MQAUSX supports Centrify's DirectControl on AIX, HP-UX, Linux and Solaris.

- **UseCDC** allows the UserId and Password to be authenticated against Centrify's DirectControl rather than the OS

```
UseCDC=Y
```

5.5 Quest Authentication Services Based Authentication

This section describes the necessary steps to enable Quest Authentication Services authentication. By default, the server-side security exit will do UserId and Password authentication against the native OS (Operating System). The company or MQ Administrator can choose to have authentication against Quest Authentication Services. MQAUSX supports Quest Authentication Services on AIX, HP-UX, Linux and Solaris.

- **UseQAS** allows the UserId and Password to be authenticated against Quest Authentication Services rather than the OS

```
UseQAS=Y
```

5.6 Pluggable Authentication Module (PAM)

This section describes the necessary steps to enable Pluggable Authentication Module (PAM). By default, the server-side security exit will do UserId and Password authentication against the native OS (Operating System). The company or MQ Administrator can choose to have authentication against PAM. MQAUSX supports PAM on AIX, HP-UX, Linux and Solaris.

- **UsePAM** allows the UserId and Password to be authenticated against PAM rather than the OS
- **PAMService** is the name of the PAM service to be invoked.

```
UsePAM=Y
PAMService=common-auth
```

5.7 LDAP Based Authentication

This section describes the necessary steps to enable 'LDAP Based Authentication'. By default, the server-side security exit will authenticate UserId and Password against the native OS (Operating System). The company or MQ Administrator can choose to have authentication against a centralized LDAP server.

MQAUSX's LDAP client functions have been tested against the following LDAP servers:

- Microsoft's Active Directory for Windows 2000 Server or higher
- Novell's eDirectory v8 or higher
- Oracle 9i Internet Directory or higher
- OpenLDAP v2.1 or higher
- IBM Tivoli Directory Server
- z/OS Integrated Security Services LDAP Server v1.6 or higher

5.7.1 LDAP Based Authentication Configuration

MQAUSX supports the use of LDAP server to authenticate a user's UserId and Password. To enable the LDAP-based authentication, you need 12 keywords in the IniFile:

1. **UseLDAP** allows the UserId and Password to be verified against a centralized LDAP server
2. **LDAPHost** specifies the hostname or IP address of the LDAP servers
3. **LDAPPort** specifies the port number of the LDAP server (default value is 389)
4. **LDAPTimeOut** specifies, in seconds, how long to wait before timing out
5. **UseLDAPLoadBalance** sgnals that the security exit rotates through the list of LDAP servers to perform load balancing when authenticating the UserID and Password..
6. **LDAPBaseDN** specifies the base DN / LDAP schema to be used when accessing the LDAP server
7. **UseLoginDNPrefix** specifies that a LoginDN Prefix be used.
8. **LoginDNPrefix** specifies the LoginDN Prefix to be used.
9. **UseLDAPAuthCompare** specifies whether to use ldap_compare_s vs ldap_simple_bind_s LDAP API call
10. **UseLDAPBindDN** allows for a particular LDAP UserId and Password to be specified to connect to the remote LDAP server which the authentication will take place with.
11. **LDAPBindDN** specifies the bind DN UserID.
12. **LDAPBindPwd** specifies the bind DN Password - see Appendix D for more information

LDAPHost and LDAPPort keywords can support up to 10 LDAP servers. Separate each LDAPHost and/or LDAPPort pattern is separated by a ';' semi-colon. By default, MQAUSX will authenticate the incoming UserId and Password against the first active LDAP server from the list of LDAP servers in the LDAPHost keyword. To have MQAUSX server-side component cycle through the list of LDAP servers. Tthe UseLDAPLoadBalance keyword is used in order to have MQASUSX to perform load balancing of the LDAP Servers when authenticating the UserID and Password..

The LDAPBaseDN keyword supports multiple “BaseDN” values. Each BaseDN in the LDAPBaseDN keyword needs to be separated by a “|” (pipe character).

For example, here are 3 sample BaseDNs

- CN=Users,DC=zzzz,DC=com
- CN=Admins,DC=zzzz,DC=com
- CN=Other,DC=zzzz,DC=com

Using the above 3 BaseDNs, the IniFile LDAPBaseDN keyword would be:

LDAPBaseDN=”CN=Users,DC=zzz,DC=com|CN=Admins,DC=zzz,DC=com|CN=Other,DC=zzz,DC=com”

The '***Login DN***' will be built by concatenating the following information together:

"CN=**%userid%,%LDAPBaseDN%**"
or
"**%LoginDNPrefix%**=**%userid%,%LDAPBaseDN%**"

If the **LDAPBaseDN** keyword contains more than one BaseDN value then MQAUSX will extract the first BaseDN and construct a Login DN to try against the LDAP server. If it fails, then the next BaseDN value from the LDAPBaseDN keyword will be used in the Login DN and tested against the LDAP server. This process will continue until a successful match is found or all BaseDN values have been rejected.

Examples:

```
UseLDAP=Y
LDAPHost=10.1.10.1;10.1.10.2
LDAPPort=389
LDAPBaseDN=”CN=Users,DC=xxx,DC=yyyy,DC=zzzz,DC=com”
```

```
UseLDAP=Y
LDAPHost=10.1.10.1;10.1.10.2;10.1.10.3;10.1.10.4
LDAPPort=389
UseLDAPLoadBalance=Y
LDAPBaseDN=”CN=Users,DC=xxx,DC=yyyy,DC=zzzz,DC=com”
```

```
UseLDAP=Y
LDAPHost=192.168.10.100
LDAPPort=389
LDAPBaseDN=”CN=Users,DC=zzz,DC=com|CN=Admins,DC=zzz,DC=com|CN=Other,DC=zzz,DC=com”
```

```
UseLDAP=Y
LDAPHost=192.168.10.100;192.168.10.200;192.168.10.300
LDAPPort=389
LDAPBaseDN=”O=acme”
```

```
UseLDAP = Y
LDAPHost = 10.1.10.1;10.1.10.2
LDAPPort = 389
LDAPBaseDN = DC=xxx,DC=yyyy,DC=zzzz,DC=com
UseLDAPBindDN = Y
LDAPBindDN = CN=cntladmin,DC=yyyy,DC=zzzz,DC=com
LDAPBindPwd = @jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA
```

5.7.2 LDAP SSL Based Authentication Configuration

MQAUSX supports the use of a SSL connection to a LDAP server to authenticate a user's UserId and Password. MQAUSX supports making an SSL connection to an LDAP server with or without any server certificate verification.

LDAP SSL is only supported on the following platforms:

Operating System	MQ v7.1, v7.5, v8.0, v9.0 & v9.1
AIX v5.1 or higher	Yes
HP-UX IA64 v11.23 or higher	Yes
HP-UX PA-RISC v11.00 or higher	Yes
IBM i (OS/400)	Yes
Linux x86	Yes
Linux x86_64	Yes
Linux on POWER	Yes
Linux on zSeries	Yes
Solaris SPARC v8 or higher	Yes
Solaris x86_64 v10 or higher	Yes
Windows 2008, 2012, 2016, Vista, 7, 8, 8.1 & 10	Yes

The following keywords relate to LDAP SSL connectivity:

1. **UseLDAPSSL** specifies that the connection to the LDAP server will be done via SSL.
2. **UseLDAPSSLCert** specifies that a SSL certificate will verify the SSL connection.
3. **SSLCertFileName** specifies the path and file name of the SSL certificate.
4. **SSLCertPwd** specifies the Password for the SSL certificate (Netscape/Mozilla style) – see Appendix D for more information.
5. **SSLCertFileType** specifies the type of SSL certificate: DER or B64

5.7.2.1 LDAP SSL connection without any server certificate verification.

```
UseLDAPSSL=Y
```

5.7.2.2 LDAP SSL connection with server certificate verification.

```
UseLDAPSSL=Y
UseLDAPSSLCert =Y
SSLCertFileName=certfile.der
SSLCertFileType=DER
```

5.7.3 LDAP UserID Search

Some companies configure their Microsoft Active Directory (AD) servers such that the attribute sAMAccountName contains the user's UserID (i.e. jdoe) and the user's username attribute to be the user's full name (i.e. John Doe). This makes authentications very difficult for MQAUSX via an LDAP (or LDAP SSL) session as AD will not authenticate a Login DN that contains sAMAccountName attribute but not the username attribute.

Hence, the following logic was added for MQAUSX LDAP (and LDAP SSL) authentication:

- Connect and bind to LDAP server using service account credentials (LDAPBindDN and LDAPBindPwd).
- With the incoming UserID, run an LDAP search to discover the DN of the user object associated with UserID.
- The search should return the current DN (i.e. “cn=John Doe, cn=Users, dc=corp, dc=acme, dc=com”) of a user object (if not, the connection is rejected).
- Attempt a LDAP bind operation using the returned DN along with the provided incoming Password
- If the bind succeeds, the user is authenticated, otherwise the connection is rejected.

LDAP UserID Search keywords:

1. **UseLDAPUserIDSearch** specifies that LDAP UserID Search be used
2. **LDAPUserIDSearchBase** specifies a search base for the LDAP UserID Search
3. **LDAPUserIDSearchFilter** specifies a search filter for the LDAP UserID Search
4. **LDAPUserIDSearchScope** specifies a search scope for the LDAP UserID Search

Note: LDAPBaseDN is not used for LDAP UserID Search.

```
UseLDAPUserIDSearch=Y
LDAPUserIDSearchBase="dc=capitalware,dc=net"
LDAPUserIDSearchFilter="(cn=%USERID%)
(objectclass=organizationalPerson)(dc=capitalware,dc=net)"
LDAPUserIDSearchScope=1
```

Note: The token “***%USERID%*** in either LDAPUserIDSearchBase or LDAPUserIDSearchFilter will be replaced by MQAUSX with the actual user's UserID.

LDAPUserIDSearchScope values and their meaning:

- -1 - LDAP_SCOPE_DEFAULT
- 0 - LDAP_SCOPE_BASE
- 1 - LDAP_SCOPE_ONELEVEL
- 2 - LDAP_SCOPE_SUBTREE
- 3 - LDAP_SCOPE_SUBORDINATE_SUBTREE

5.7.4 ANR (Ambiguous Name Resolution) for LDAP authentication

MQAUSX supports the use of ANR (Ambiguous Name Resolution) for LDAP authentication for a user's UserId and Password. To enable the ANR for LDAP-based authentication, you may use the following 8 keywords in the IniFile:

1. **UseANRforLDAP** specifies that ANR for LDAP authentication be used.
2. **UseANRPrefix** specifies that MQAUSX will prefix a value to the user's UserId.
3. **ANRPrefix** specifies the value to be used as the prefix data.
4. **UseANRPostfix** specifies that MQAUSX will append a value to the user's UserId.
5. **ANRPostfix** specifies the value to be appended.
6. **ExtractUserIDFromANR** specifies that MQAUSX should strip a UserId from the inputted UserId. The default delimiter is the 'at sign' ('@').
7. **UseANRDelimiter** specifies a delimiter to be used when stripping a UserId from the inputted UserId.
8. **ANRDelimiter** specifies a single character to be used as a delimiter.

Note: LDAPBaseDN is not used for ANR for LDAP.

```
UseANRforLDAP=Y
UseANRPostfix=Y
ANRPostfix=@acme.com
ExtractUserIDFromANR=N
UseANRDelimiter=N
```

The '***Login DN***' for ANR for LDAP-based authentication will be built by concatenating the following information together (the ANR base value is the user's UserId):

"**%userid%**"
or
"**%ANRPrefix%%userid%**"
or
"**%userid%%ANRPostfix%**"
or
"**%ANRPrefix%%userid%%ANRPostfix%**"

5.7.5 LDAP Group Search

MQAUSX supports the use of LDAP Group Search to verify that a UserID exists in a group. If there is a successful match then the connection is allowed; otherwise the connection is closed. To enable the LDAP Search, you may use the following 4 keywords in the IniFile:

1. **UseLDAPGroupSearch** specifies that LDAP Group Search be used for group verification
2. **LDAPGroupSearchBase** specifies a search base for the LDAP Group Search
3. **LDAPGroupSearchFilter** specifies a search filter for the LDAP Group Search
4. **LDAPGroupSearchScope** specifies a search scope for the LDAP Group Search

```
UseLDAPGroupSearch=Y
LDAPGroupSearchBase="dc=capitalware,dc=net"
LDAPGroupSearchFilter="(cn=%USERID%)
(objectclass=organizationalPerson)(dc=capitalware,dc=net)"
LDAPGroupSearchScope=1
```

Note: The token "***%USERID%*** in either LDAPGroupSearchBase or LDAPGroupSearchFilter will be replaced by MQAUSX with the actual user's UserID.

LDAPGroupSearchScope values and their meaning:

- -1 - LDAP_SCOPE_DEFAULT
- 0 - LDAP_SCOPE_BASE
- 1 - LDAP_SCOPE_ONELEVEL
- 2 - LDAP_SCOPE_SUBTREE
- 3 - LDAP_SCOPE_SUBORDINATE_SUBTREE

5.8 Credential Cache

This section describes the necessary entries to enable credential cache within MQAUSX. MQAUSX will cache the user credentials (in an encrypted format) for 'x' minutes (default is 5 minutes) in shared memory. When there is a new connection, MQAUSX will first check the cache for the incoming UserID and if found then the entry's timestamp will be checked. If the cache entry has expired then the entry is removed from the cache. If the entry is valid then the cached password is compared to the incoming password. If the passwords match then the connection is allowed. If the passwords do not match then the entry is removed from the cache and MQAUSX will perform an authentication against the target (i.e. LDAP).

To enable credential cache feature, you need 3 keywords in the IniFile:

- **UseCredentialCache** allows the MQAdmin enable credential caching in MQAUSX UseCredentialCache supports 2 values [Y / N]. The default value is Y.

- **CacheLife** specifies the "time to live" for the credentials in the cache. The default value is 5 minutes.

- **CacheSize** specifies the size of the cache. The default value is 100 entries.

```
UseCredentialCache=Y
CacheLife = 7
CacheSize = 150
```

5.9 Queue Manager Password

This section describes the necessary entries to enable an extra layer of security to the standard UserID and Password authentication. An MQAdmin can define a password for a queue manager via the MQAUSX configuration file. Hence, when enabled, a back-end application and/or end-user would need to not only know their UserID and Password but also the queue manager's Password to successfully log in.

Defining and requiring a queue manager Password in MQAUSX is like adding perimeter security to your system or putting your valuables in a safe and putting that safe in another safe.

To enable queue manager password feature, you need 2 keywords in the IniFile:

- **UseQMgrPwd** allows the MQAdmin to assign a password to a queue manager. UseQMgrPwd supports 2 values [Y / N]. The default value is N.

- **QMgrPwd** specifies the encrypted password the MQAdmin is assigning to a queue manager. See Appendix D for details on creating the encrypted password.

```
UseQMgrPwd=Y
QMgrPwd = @jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA
```

5.10 Allow or Restrict the Incoming UserID against a Group

This section describes the necessary entries to enable the feature that allows or restricts the incoming UserID against an OS group or a group file. This feature uses the following four keywords:

- **UseGroups** keyword controls the use of Groups. Set to 'Y' to allow authorization by either OS or a group file.
- **Groups** keyword specifies the authorized groups that can connect to the queue manager. Each group is separated from the next by a semi-colon (';').
- **UseGroupFile** keyword controls the use of GroupFile. Set to Y to activate feature.
- **GroupFile** keyword specifies the location of the group file (i.e. groups.ini)

5.10.1 Authorization against Local OS

When UseGroups is set to 'Y' and UseGroupFile is set to 'N' then MQAUSX will query the local OS for the groups listed in the Groups keyword.

Example:

```
UseGroups=Y
Groups=grpX;grpZ
UseGroupFile=N
```

5.10.2 Authorization against a Group File

When UseGroups is set to 'Y' and UseGroupFile is set to 'Y' then MQAUSX will query the specified group file given in GroupFile keyword. This section describes how to implement groups files. The group files are implemented in a similar manner to the way they are implemented in Unix and Linux (i.e. **/etc/group** file).

Below is an example group file:

```
unique_group_name = UserID1;UserID2;UserID3
```

Example:

```
grp1=fred;wilma;pebbles
grp2=barney;betty;bammbamm
grpA=arnold;rockhead;slate;gazoo
grpB=dino;puss;doozy;hoppy
```

The following are the default values for GroupFile:

For Windows:
GroupFile=C:\Capitalware\MQAUSX\groups.ini

For IBM MQ 32-bit on Unix and Linux:
GroupFile=/var/mqm/exits/groups.ini

For IBM MQ 64-bit on Unix and Linux:
GroupFile=/var/mqm/exits64/groups.ini

For IBM MQ on IBM i:
GroupFile=/QIBM/UserData/mqm/mqme/groups.ini

MQAUSX will check, in order, each group listed in the Groups keyword for a particular UserID. The UserID must exist in one of the groups or else MQAUSX will not decrypt the data.

Example:

```
UseGroups=Y
Groups=grp1;grp2;grpB
UseGroupFile=Y
GroupFile=C:\Capitalware\MQAUSX\groups.ini
```

5.11 Allow or Restrict the Incoming IP Address

This section describes the necessary entries to enable the feature that allows or restricts the incoming IP addresses through the use of regular expression patterns. This feature uses the following two keywords:

- **UseAllowIP** controls the use of AllowIP. Set to Y to activate feature.
- **AllowIP** specifies the regular expression patterns that limit the allowable incoming IP addresses

The server-side security exit will look up the regular expression patterns from the **AllowIP** keyword in order to determine if the entire incoming IP address matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: AllowIP must NOT exceed 2048 characters.

Separate each IP address pattern with a ';' semi-colon.

```
UseAllowIP=Y
AllowIP=192.168.*.*;10.15[0-9].2[0-5][0-9];127.0.0.#
```

5.12 Allow or Restrict the Incoming Hostname

This section describes the necessary entries to enable the feature that allows or restricts the incoming Hostnames through the use of regular expression patterns. This feature uses the following two keywords:

- **UseAllowHostname** controls the use of AllowHostname. Set to Y to activate feature.
- **AllowHostname** specifies the regular expression patterns that limit the allowable incoming Hostnames

The server-side security exit will look up the regular expression patterns from the **AllowHostname** keyword in order to determine if the entire incoming Hostname matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: AllowHostname must NOT exceed 2048 characters.

Separate each Hostname pattern with a ';' semi-colon.

```
UseAllowHostname=Y
AllowHostname=abc01.acme.com;abc02.acme.com
```

5.13 Allow or Restrict the Incoming IP against IP of Hostname

This section describes the necessary entries to enable the feature that allows or restricts the incoming IP against IP of Hostnames that MQAUSX will perform a gethostbyaddr() call against to compare the returned IP address against the incoming IP address.. This feature uses the following two keywords:

- **UseAllowHostByName** controls the use of AllowHostByName. Set to Y to activate feature.

- **AllowHostByName** specifies the Hostnames that MQAUSX will perform a gethostbyaddr() call against to compare the returned IP address against the incoming IP address to allow the incoming connection.

The server-side security exit will perform a gethostbyaddr() call against hostnames from the **AllowHostByName** keyword and used the returned IP address and compare the returned IP address.

Note: AllowHostByName must NOT exceed 2048 characters.

Separate each Hostname pattern with a ';' semi-colon.

```
UseAllowHostByName=Y
AllowHostByName=abc01.acme.com;abc02.acme.com
```

5.14 Allow or Restrict the Incoming SSL DN

This section describes the necessary entries to enable the feature that allows or restricts the incoming SSL DN through the use of regular expression patterns. This feature uses the following two keywords:

- **UseAllowSSLDN** controls the use of AllowSSLDN. Set to Y to activate feature.
- **AllowSSLDN** specifies the regular expression patterns that limit the allowable incoming SSL DN

The server-side security exit will look up the regular expression patterns from the **AllowSSLDN** keyword in order to determine if the entire incoming SSL DN matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: AllowSSLDN must NOT exceed 2048 characters.

```
UseAllowSSLDN=Y
AllowSSLDN=O=Capitalware,DC=net;CN=roger;O=acme
```

5.15 Allow or Restrict the Incoming UserID

This section describes the necessary entries to enable the feature that allows or restricts the incoming UserIDs through the use of regular expression patterns. This feature uses the following two keywords:

- **UseAllowUserID** controls the use of AllowUserID. Set to Y to activate feature.
- **AllowUserID** specifies the regular expression patterns that limit the allowable incoming UserIds

The server-side security exit will look up the regular expression patterns from the **AllowUserID** keyword in order to determine if the entire incoming UserID matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: AllowUserID must NOT exceed 2048 characters.

```
UseAllowUserID=Y
AllowUserID=mq*;hr[0-9][a-f];abc??01
```

5.16 Allow or Restrict the Incoming Active Directory Server Name

This section describes the necessary entries to enable the feature that allows or restricts the incoming Microsoft Microsoft Active Directory Server Name (AD Server Name) through the use of regular expression patterns (***Windows only***). This feature uses the following two keywords:

- **UseAllowADName** controls the use of AllowADName. Set to Y to activate feature.
- **AllowADName** specifies the regular expression patterns that limit the allowable incoming user specified AD Server Name

The server-side security exit will look up the regular expression patterns from the **AllowADName** keyword in order to determine if the entire incoming user specified AD Server Name matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: AllowADName must NOT exceed 2048 characters.

Separate each AD server name pattern with a ';' semi-colon.

```
UseAllowADName=Y
AllowADName=ad001;ads0*;adc0??;ad*.acme.com
```

5.17 Reject the Incoming IP Address

This section describes the necessary entries to enable the feature that rejects the incoming IP addresses through the use of regular expression patterns. This feature uses the following two keywords:

- **UseRejectIP** controls the use of RejectIP. Set to Y to activate feature.
- **RejectIP** specifies the regular expression patterns that explicitly reject the incoming IP Address

The server-side security exit will look up the regular expression patterns from the **RejectIP** keyword in order to determine if the entire incoming IP address matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: RejectIP must NOT exceed 2048 characters.

Separate each IP address pattern with a ';' semi-colon.

```
UseReject=Y
RejectIP=192.161.*.*;10.13[0-9].2[0-5][0-9];10.10.1.15
```

5.18 Reject by Hostname

This section describes the necessary entries to enable the feature that rejects by the Hostnames through the use of regular expression patterns. This feature uses the following two keywords:

- **UseRejectHostname** controls the use of RejectHostname. Set to Y to activate feature.

- **RejectHostname** specifies the regular expression patterns that explicitly reject by hostname

The server-side security exit will look up the regular expression patterns from the **RejectHostname** keyword in order to determine if the entire incoming Hostname matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: RejectHostname must NOT exceed 2048 characters.

Separate each Hostname pattern with a ';' semi-colon.

```
UseReject=Y
RejectHostname=xyz01.acme.com;xyz02.acme.com
```

5.19 Reject by Incoming IP against IP of Hostname

This section describes the necessary entries to enable the feature that rejects the incoming IP against IP of Hostnames that MQAUSX will perform a gethostbyaddr() call against to compare the returned IP address against the incoming IP address.. This feature uses the following two keywords:

- **UseRejectHostByName** controls the use of RejectHostByName. Set to Y to activate feature.

- **RejectHostByName** specifies the Hostnames that MQAUSX will perform a gethostbyaddr() call against to compare the returned IP address against the incoming IP address.

The server-side security exit will perform a gethostbyaddr() call against hostnames from the **RejectHostByName** keyword and used the returned IP address and compare the returned IP address to reject the incoming connection.

Note: RejectHostByName must NOT exceed 2048 characters.

Separate each Hostname pattern with a ';' semi-colon.

```
UseReject=Y
RejectHostByName=xyz01.acme.com;xyz02.acme.com
```

5.20 Reject the Incoming SSL DN

This section describes the necessary entries to enable the feature that rejects the incoming SSL DN through the use of regular expression patterns. This feature uses the following two keywords:

- **UseRejectSSLDN** controls the use of RejectSSLDN. Set to Y to activate feature.
- **RejectSSLDN** specifies the regular expression patterns that reject the incoming SSL DN

The server-side security exit will look up the regular expression patterns from the **RejectSSLDN** keyword in order to determine if the entire incoming SSL DN matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: RejectSSLDN must NOT exceed 2048 characters.

```
UseRejectSSLDN=Y
RejectSSLDN=O=xyz*;O=abc*
```

5.21 Reject the Incoming UserID

This section describes the necessary entries to enable the feature that rejects the incoming UserIDs through the use of regular expression patterns. This feature uses the following two keywords:

- **UseRejectUserID** controls the use of RejectUserID. Set to Y to activate feature.
- **RejectUserID** specifies the regular expression patterns that reject the incoming UserId

The server-side security exit will look up the regular expression patterns from the **RejectUserID** keyword in order to determine if the entire incoming UserID matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: RejectUserID must NOT exceed 2048 characters.

```
UseRejectUserID=Y
RejectUserID=abc*;x[0-9][a-f]
```

5.22 Reject the Incoming Active Directory Server Name

This section describes the necessary entries to enable the feature that rejects the incoming Microsoft Microsoft Active Directory Server Name (AD Server Name) through the use of regular expression patterns (***Windows only***). This feature uses the following two keywords:

- **UseRejectADName** controls the use of RejectADName. Set to Y to activate feature.

- **RejectADName** specifies the regular expression patterns that rejects user specified AD Server Name

The server-side security exit will look up the regular expression patterns from the **RejectADName** keyword in order to determine if the entire incoming user specified AD Server Name matches any of the specified expression patterns. Each regular expression pattern is separated from the next pattern by a semi-colon (';').

In the regular expression pattern:

- '*' matches any sequence of characters (zero or more)
- '?' matches any single character
- '#' matches any single numeric digit (0-9)
- '@' matches any single alphabetic character (A-Z, a-z)
- [SET] matches any of the characters in the specified set
- [!SET] or [^SET] matches any character except those specified in the set (negation).

A SET can be composed of characters or ranges. A range is in the form: 'character – character' (i.e. 0-9 or A-Z). Although this is the simplest range allowed in the [ ] pattern, more complex inclusive ranges such as [0-9a-zA-Z] are allowed. [0-9a-zA-Z] specifies that the character can be 0 through 9 **or** a through z **or** A through Z. Other characters are allowed (ie. 8 bit characters) if your system supports them.

In order to suppress the special syntactic significance of any of these characters '[] * ? # @ ! ^ - \', a backslash ('\') must precede the special character.

Note: RejectADName must NOT exceed 2048 characters.

Separate each AD server name pattern with a ';' semi-colon.

```
UseRejectADName=Y
RejectADName=ad001;ads0*;adc0??;ad*.acme.com
```

5.23 Excessive Client Connections

This section describes the necessary entries to configure Excessive Client Connections (ECC) alert system in MQAUSX. This is controlled by the IniFile's property keyword 'UseECC'.

ECC is an alert system that counts the number of connections over a period of time (i.e. Day / Hour / Minute) and writes a message to the log when the count exceeds a particular value. If the keyword WriteToEventQueue is set to 'Y', an event message is also written to an event queue. ECC feature is designed to catch applications that are poorly written, such as, applications that continuously connect and disconnect from the queue manager for every message sent or received.

To enable the alerting of excessive client connections, you need 3 keywords in the IniFile:

1. **UseECC** enables excessive client connections feature
2. **ECCWarnCount** specifies a count which, when exceeded, will cause an alert to be generated. The default value is 5000.
3. **ECCInterval** specifies a time interval to monitor the incoming number of connections. Valid values are D/H/M (Day, Hour and Minute) The default value is 'D'.

```
UseECC=Y
ECCWarnCount=200
ECCInterval=H
```

5.24 Set Maximum Number of Incoming Connections per Channel

This section describes the necessary entries to set a maximum number of allowable connections per a given channel. This is controlled by the IniFile's property keyword 'UseMCC'. Setting 'UseMCC' to 'Y' (Yes) will cause the server-side security exit to look up channel's name as a property keyword in the IniFile.

To enable the restricting of allowable connections per a given channel, you need 7 keywords in the IniFile:

1. **UseMCC** enables restricting of allowable connections per a given channel
2. **DefaultMCC** specifies the default maximum allowable connections for a particular channel.
3. **MCCEventWarnLevel** specifies the percentage of incoming channels to the maximum allowable number of channels that will cause MQAUSX to write a warning message to the event queue. The default value is 80.
4. **UseMCCRedo** keyword specifies whether or not the PCF 'display channel status' command should be issued. The default value is 'Y'.
5. **MCCRedoMinutes** specifies a time internal to issue the 'display channel status' command. The default value is 720 minutes.
6. **MCCRedoCount** specifies how often the 'display channel status' command should be issued. The default value is 5000.
7. **MCCGetTimeOut** specifies how long the security exit will wait for the reply from the queue manager's command server. The default value is 3 seconds.

For example, if 'UseMCC' is set to 'Y' and the incoming connection is on 'SYSTEM.ADMIN.SVRCONN', the server-side security exit will look up in the IniFile the keyword of 'SYSTEM.ADMIN.SVRCONN'. If the 'SYSTEM.ADMIN.SVRCONN' keyword is not found, then the server-side security exit will look up 'DefaultMCC' keyword in the IniFile.

If the 'DefaultMCC' keyword is not found, the 'UseMCC' keyword is then switched to 'N' (No).

The server-side security exit uses shared memory to keep track of the channel connection and disconnection. MQAUSX was designed to periodically refresh the shared memory counter by issuing a PCF command to get the current channel status. This information is written to the shared memory.

There are 2 IniFile keywords to control how often the PCF Inquire channel status command is issued: 'MCCRedoMinutes' and 'MCCRedoCount'. 'MCCRedoMinutes' keyword states that the server-side security exit should issue PCF command if more than 'x' minutes have passed since the last PCF command was issued. The default value for 'MCCRedoMinutes' is 720 minutes. 'MCCRedoCount' keyword states that the server-side security exit should issue PCF command if more than 'x' connection attempts passed since the last PCF command was issued. The default value for 'MCCRedoCount' is 5000.

MCCEventWarnLevel keyword states that the server-side security exit should write a warning message to the event queue when the number of connections exceeds the percentage level. The default value for 'MCCEventWarnLevel' is 80. Note: Only used if both UseMCC and WriteToEventQueue are each set to 'Y'.

MCCGetTimeOut keyword specifies how long the security exit should wait for a reply from the queue manager's command server. The default value is 3 seconds.

```
UseMCC=Y
SYSTEM.ADMIN.SVRCONN=5
ABC.CH01=50
DEF. CH01=40
SYSTEM.DEF.SVRCONN=5
#
DefaultMCC=25
#
UseMCCRedo=Y
MCCRedoMinutes=900
MCCRedoCount=2000
MCCGetTimeOut=5
```

Note: For queue managers with thousands for active connections, the user may wish to increase the values for 'MCCRedoMinutes' and 'MCCRedoCount' to a higher value. This will keep the overhead to a minimum.

```
UseMCCRedo=Y
MCCRedoMinutes=1440
MCCRedoCount=8000
```

5.25 Proxy ID

This section describes the necessary steps to enable the use of 'Proxy IDs'. Proxy ID allows an authorized User to use a different UserID for MQ interactions.

- **UseProxy** allows an authorized User to use a different UserID for MQ interactions
- **ProxyFile** specifies the location of the file to do alternate UserID look-up

```
UseProxy=Y
ProxyFile = proxy.lst
```

The format of the Proxy file is similar to an IniFile or properties file where each keyword has an associated value. Each keyword and its value is on a separate line. The format is as follows:

```
Validated_UserID = ProxyID
```

Example:

```
Roger=app1
Fred=app2
Barney=app1
```

If the UserID is not found in the Proxy file then the incoming connection is rejected. To have a default Proxy UserID in the Proxy file use the "DefaultProxyID" value.

Example:

```
DefaultProxyID=readonly
```

5.26 ServerName

This section describes the necessary steps to enable 'Windows Active Directory' authentication. By default, the server-side security exit will do UserId and Password against the local Windows local server. The company or MQ Administrator can choose to have authentication against a 'Windows Active Directory'.

- **ServerName** allows the authentication to be performed against Active Directory.
- **AllowUserAlterServerName** allows the user to specified a different ServerName

```
UseServerName=Y
ServerName=company_AD_Server_Name
AllowUserAlterServerName = Y
```

5.27 AllowPlainTextCredentials

This section describes the necessary entries to enable the MQAUSX server-side component to accept UserId and Password in plain text (i.e. no client-side security exit). The default value is Y.

```
AllowPlainTextCredentials=Y
```

5.28 UserIDFormatting

This section describes the necessary entries on how to handle the incoming UserID. 'UserIDFormatting' supports 3 values [A / U / L]. ('As Is, Uppercase and Lowercase). The default value is A.

```
UserIDFormatting=U
```

5.29 Allow Users to login as mqm

This section describes the necessary entries to enable users to login with the mqm or MUSR_MQADMIN or QMQM system account. This is controlled by the IniFile's property keyword 'Allowmqm'. Setting 'Allowmqm' to 'Y' (Yes) will activate this feature; otherwise, it will be blocked.

```
Allowmqm=Y
```

5.30 Turning off Authentication

This section describes the necessary entries to disable authentication in the server-side security exit. ***Be very careful when disabling authentication because the connecting user will not need a client-side security exit to make a valid connection to the channel.*** This is controlled by the IniFile's property keyword 'NoAuth'. Setting 'NoAuth' to 'Y' (Yes) will disable server-side authentication.

```
NoAuth=Y
```

5.31 Allow Connection to have a Blank UserID

This section describes the necessary entries to enable connection to have a blank UserID. ***This parameter is only valid when 'NoAuth' is set to 'Y'.*** This is controlled by the IniFile's property keyword 'AllowBlankUserID'. Setting 'AllowBlankUserID' to 'Y' (Yes) will allow connections to have a blank UserID.

```
AllowBlankUserID=Y
```

5.32 MCAUSER Field

This section describes the necessary steps to enable the use of the channel's MCAUSER field. If this IniFile parameter is set to 'Y' (Yes) then after the authentication process is complete, the connection will use the UserID value specified in the MCAUSER field.

- **UseMCAUser** enables the connection to use the UserID value specified in the channel's MCAUSER field

```
UseMCAUser=Y
```

5.33 CheckFinalUserID

This section describes the necessary entries to enable z/MQSSX to process the final UserID against UseAllowUserID, AllowUserID, UseRejectUserID, RejectUserID and Allowmqm keywords.

```
CheckFinalUserID=Y
```

5.34 SSL Self-Signed Certificate

This section describes the necessary steps to allow or reject SSL Self-Signed Certificates. If the AllowSSLSSCert IniFile parameter is set to 'Y' (Yes) then the SSL Self-Signed Certificate are allowed. If AllowSSLSSCert is set to 'N' (No), the SSL Self-Signed Certificate is disallowed (i.e. the incoming connection is closed).

- **AllowSSLSSCert** specifies whether or not to allow the Self-Signed Certificate on the channel.

```
AllowSSLSSCert=Y
```

5.35 Set UserID from SSL DN

MQAUSX supports the retrieval of the UserID from the channel's SSL DN field. To enable the retrieval of the UserId from the channel's SSL DN field, you may use the following 4 keywords in the IniFile:

1. **UseSSLUserIDFromDN** specifies that the UserID is to be retrieved from a SSL DN entry.
2. **SSLDNAttrName** specifies the SSL DN attribute field name.
3. **SSLDNAttrStartPos** specifies the start position of the retrieval.
4. **SSLDNAttrLength** specifies the length of the field to be extracted (* means all).

```
UseSSLUserIDFromDN = Y
SSLDNAttrName = CN
SSLDNAttrStartPos = 1
SSLDNAttrLength = *
```

5.36 FreeAuxOnTerm

This section describes when MQAUSX should free the auxiliary memory. The default is immediately. By setting FreeAuxOnTerm to Y, MQAUSX will free the memory when the connection is terminicated (i.e. client application disconnects).

```
FreeAuxOnTerm=Y
```

5.37 LicenseFile

This section will describe how to have a file that contains all of the user's MQAUSX license keys.

The format of the LicenseFile is similar to an IniFile or properties file where each keyword has an associated value. Each keyword and its value are on a separate line. The format is as follows:

```
QMgrName = License_Key
```

Example:

```
MQA1 = 10A0-AAAA-BBBBBBBB
MQB1 = 10A0-XXXX-CCCCCCCC
```

If the queue manager name is not found in the LicenseFile then the License keyword will be used to retrieve the license key value.

The following are the default values for LicenseFile:

For Windows:

```
LicenseFile=C:\Capitalware\MQAUSX\mqausx_licenses.ini
```

For IBM MQ 32-bit on Unix and Linux:

```
LicenseFile=/var/mqm/exits/mqausx_licenses.ini
```

For IBM MQ 64-bit on Unix and Linux:

```
LicenseFile=/var/mqm/exits64/mqausx_licenses.ini
```

For IBM MQ on IBM i:

```
LicenseFile=/QIBM/UserData/mqm/mqausx/mqausx_licenses.ini
```

5.38 License Key

This section will describe how to license MQ Authenticate User Security Exit to a particular queue manager.

Note: The License keyword is not required if the user has implemented the LicenseFile keyword or the License file actually exists in the default location.

Your license will look something like: 10A0-AAAA-BBBBBBBB (Note: This is a sample license only and will NOT work).

```
License=10A0-AAAA-BBBBBBBB
```

6 Miscellaneous

This section describes the extra files that were included to help the user get MQAUSX up and running in a very quick manner.

6.1 Windows

Sample IniFile

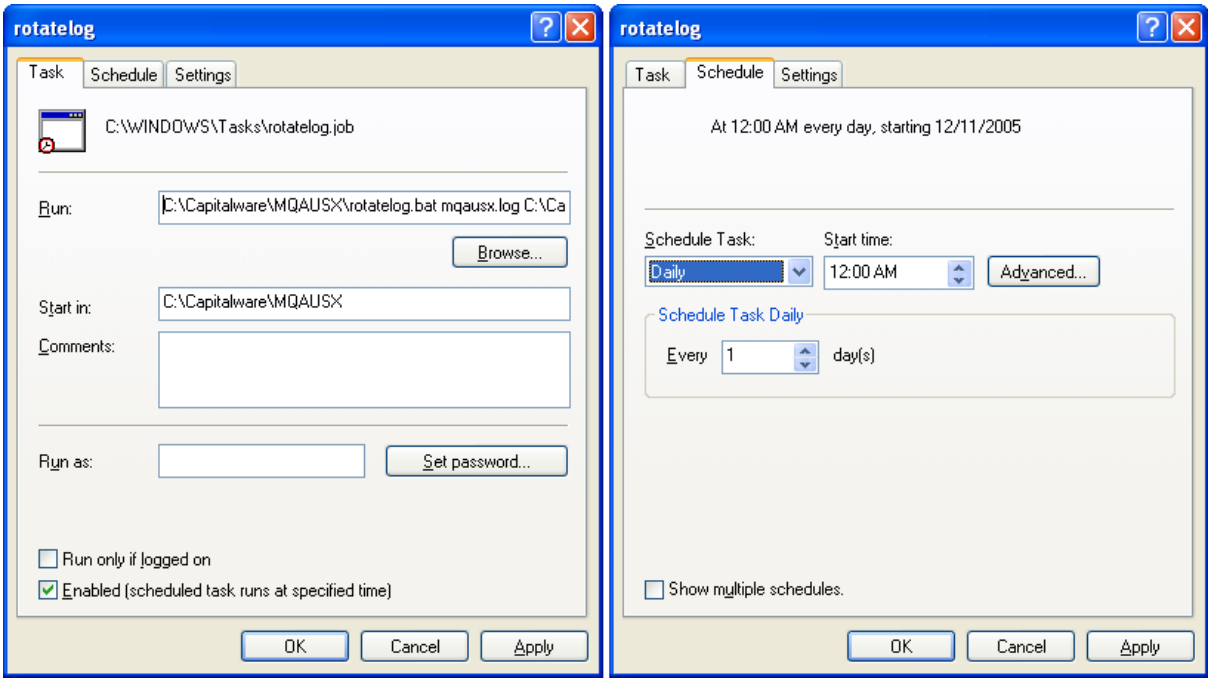
The '***mqausx.ini***' file is a basic MQAUSX IniFile. It has the standard IniFile parameters that the user may need to use or update. The '***mqausx.ini.readme***' file is a plain text help file with a description of the parameters.

Sample MQSC scripts

The '***mqausx.sample.mqsc***' file is a sample MQSC script to update the 2 system defined channels with the MQAUSX security exit information.

Rotate log script

The '***rotatelog.bat***' file is a Windows batch script to rotate (backup) the mqausx.log file. Actually, it is generic in implementation; hence, it can be used to rotate any log file that the user wishes to be rotated. The batch script requires 2 parameters: log file name and the directory of log file.



6.2 Unix and Linux

Sample IniFile
The '***mqausx.ini***' file is a basic MQAUSX IniFile. It has the standard IniFile parameters that the user may need to use or update. The '***mqausx.ini.readme***' file is a plain text help file with a description of the parameters.

Sample MQSC scripts
The '***mqausx.sample.mqsc***' file is a sample MQSC script to update the 2 system defined channels with the MQAUSX security exit information.

Rotate log script
The '***rotatelog.sh***' file is a Unix / Linux shell script to rotate (backup) the mqausx.log file. Actually, it is generic in implementation; hence, it can be used to rotate any log file that the user wishes to be rotated. The shell script requires 2 parameters: log file name and the directory of log file.

Sample daily CRON entry for IBM MQ 32-bit on Unix and Linux:

0 0 * * * /var/mqm/exits/rotatelog.sh mqaux.log /var/mqm/exits/ > /tmp/mqausx.log.run 2 > &1

Sample daily CRON entry for IBM MQ 64-bit on Unix and Linux:

0 0 * * * /var/mqm/exits64/rotatelog.sh mqaux.log /var/mqm/exits64/ > /tmp/mqausx.log.run 2 > &1

6.3 IBM i

Sample IniFile
The '***mqausx.ini***' file is a basic MQAUSX IniFile. It has the standard IniFile parameters that the user may need to use or update. The '***mqausx.ini.readme***' file is a plain text help file with a description of the parameters.

Sample MQSC scripts
The '***mqausx.sample.mqsc***' file is a sample MQSC script to update the 2 system defined channels with the MQAUSX security exit information.

6.4 Server-side Log File

To verify that the process flow was successful, you can view the log file for the events that are generated.

Authentication Log Sample

```
2009/01/15 13:30:58 MQAUSX #02292 I: Connection accepted for UserID='tester' UserSpecifiedServer='cw-rl001' QMgr='MQWT1' C
2009/01/15 13:47:01 MQAUSX #02292 I: Connection accepted for UserID='tester' UserSpecifiedServer='cw-rl001' QMgr='MQWT1' C
```

Non-Authentication Log Sample

```
2006/03/23 19:47:11 MQAUSX #01767 I: Connection accepted for QMgr='MQS1' ChlName='MY.TEST.EXIT' ConName='192.168.10.101' R
2006/03/23 19:47:31 MQAUSX #01767 I: Connection accepted for QMgr='MQS1' ChlName='MY.TEST.EXIT' ConName='192.168.10.101' R
```

7 Appendix A – Summary of IniFile (Server-side)

The sample IniFile below is the mqausx.ini file supplied for Windows. The IniFile supports the following keywords and their values:

```
LogMode=N
LogFile=C:\Capitalware\MQAUSX\mqausx.log
UseFBA = N
Allowmqm=N
UseMCC=N
UseAllowIP=N
UseProxy=N
```

Note: Keywords are case sensitive.

Keyword	Description of Server-side keywords
AllowADName	**AllowADName** specifies a set of regular expression patterns that the incoming user-specified Microsoft Active Directory Server Name (AD Server Name) would be compared against. The default is '*'. You must separate the AD Server Name regular expression patterns with a ';' semi-colon. e.g. AllowADName=ad001;adc0*;ads0*.acme.com Note: Only used if UseAllowADName is set to 'Y'.
AllowBlankUserID	**AllowBlankUserID** specifies whether or not to allow the incoming connection to have a blank UserID value. ***This feature is only valid when the 'NoAuth' keyword is set to 'Y'.*** AllowBlankUserID supports 2 values [Y / N]. The default value is N. e.g. AllowBlankUserID=Y
AllowHostByName	**AllowHostByName** specifies the Hostnames that MQAUSX will perform a gethostbyaddr() call against to compare the returned IP address against the incoming IP address to allow the incoming connection. The default is '*'. You must separate the hostname regular expression patterns with a ';' semi-colon. e.g. AllowHostByName=abc01.acme.com;abc02.acme.com Note: Only used if UseAllowHostByName is set to 'Y'.

Keyword	Description of Server-side keywords
AllowHostname	**AllowHostname** specifies a set of regular expression patterns that the hostname will be compared against. The default is '*'. You must separate the hostname regular expression patterns with a ';' semi-colon. e.g. AllowHostname=abc01.acme.com;abc02.acme.com Note: Only used if UseAllowHostname is set to 'Y'.
AllowIP	**AllowIP** specifies a set of regular expression patterns that the incoming channel's IP address will be compared against. The default is '*'. You must separate the IP regular expression patterns with a ';' semi-colon. e.g. AllowIP=192.168.*.1[0-5][0-9];127.0.0.?;10.*.*.[0-9] Note: Only used if UseAllowIP is set to 'Y'.
Allowmqm	**Allowmqm** specifies whether or not to allow a user to be able to login using 'mqm' (Unix), 'MUSR_MQADMIN' (Windows) or 'QMQM' (OS/400) system account. Allowmqm supports 2 values [Y / N]. The default value is N. e.g. Allowmqm=Y
AllowPlainTextCredentials	**AllowPlainTextCredentials** allows the MQAUSX server-side component to accept UserId and Password in plain text (i.e. no client-side security exit). AllowPlainTextCredentials supports 2 values [Y / N]. The default value is Y. e.g. AllowPlainTextCredentials=Y
AllowSSLDN	**AllowSSLDN** specifies a set of regular expression patterns that the incoming channel's SSL DN will be compared against. You must separate the SSL DN regular expression patterns with a ';' semi-colon. e.g. AllowSSLDN=O=Capitalware,C=CA;O=IBM,DC=com Note: Only used if UseAllowSSLDN is set to 'Y'.

Keyword	Description of Server-side keywords
AllowSSLSSCert	**AllowSSLSSCert** specifies whether or not to allow Self-Signed Certificate on the channel. AllowSSLSSCert supports 2 values [Y / N]. The default value is Y. e.g. AllowSSLSSCert=Y
AllowUserAlterServerName	**AllowUserAlterServerName** specifies whether or not to allow a user to be able to changed the ServerName when authenticating against a Windows server. AllowUserAlterServerName supports 2 values [Y / N]. The default value is N. e.g. AllowUserAlterServerName=N
AllowUserID	**AllowUserID** specifies a set of regular expression patterns that the incoming connection's UserID will be compared against. The default is '*'. You must separate each IP regular expression pattern with a ';' semi-colon. e.g. AllowUserID=mq*;abc??;xyz[0-9][a-f];hr[0-9][0-9] Note: Only used if UseAllowUserID is set to 'Y'.
ANRDelimiter	**ANRDelimiter** specifies a single character to be used as a delimiter. e.g. ANRDelimiter=$
ANRPostfix	**ANRPostfix** specifies the value to be appended to the ANR value (inputted UserId). e.g. ANRPostfix=@acme.com
ANRPrefix	**ANRPrefix** specifies the value to be prefixed to the ANR value (inputted UserId). e.g. ANRPrefix=XXX

Keyword	Description of Server-side keywords
AuthOrder	**AuthOrder** specifies which authentication sources that the UserId and Password will be tested against. The authentication order is from left to right. There are 3 supported values: ldap, files and mqausx. e.g. AuthOrder=ldap files mqausx Note: Only used if UseAuthOrder is set to 'Y'.
BackupLogFileCount	**BackupLogFileCount** specifies the number of backup logfiles that the security exit will be keeping. The default value is 9. e.g. BackupLogFileCount=9
CacheLife	**CacheLife** specifies the "time to live" for the credentials in the cache. The default value is 5 minutes. e.g. CacheLife =12 Note: Only used if UseCredentialCache is each set to 'Y'.
CacheSize	**CacheSize** specifies the size of the cache. The default value is 100 entries. e.g. CacheSize =300 Note: Only used if UseCredentialCache is each set to 'Y'.
CheckFinalUserID	**CheckFinalUserID** specifies whether or not the final UserID will be checked against the UseAllowUserID, AllowUserID, UseRejectUserID, RejectUserID and Allowmqm keywords. CheckFinalUserID supports 2 values [Y / N]. The default value is N. e.g. CheckFinalUserID=Y
DefaultMCC	**DefaultMCC** specifies a default maximum number of incoming connections that a particular channel will allow. There is no default value. e.g. DefaultMCC=25

Keyword	Description of Server-side keywords
ECCInterval	**ECCInterval** specifies a time interval to monitor the incoming number of connections. Valid values are D/H/M (Day, Hour and Minute) The default value is 'D'. e.g. ECCInterval =H Note: Only used if UseECC is each set to 'Y'.
ECCWarnCount	**ECCWarnCount** specifies a count which, when exceeded, will cause an alert to be generated. The default value is 5000. e.g. ECCWarnCount =4000 Note: Only used if UseECC is each set to 'Y'.
EventQueueName	**EventQueueName** specifies the name of the event queue. The default value is 'SYSTEM.ADMIN.CHANNEL.EVENT'. e.g. EventQueueName= SYSTEM.ADMIN.CHANNEL.EVENT Note: Only used if WriteToEventQueue is set to 'Y'.
ExtractUserIDFromANR	**ExtractUserIDFromANR** specifies that MQAUSX should extract a username from the incoming UserId. The username will be used in the MCAUSER field. The default delimiter is the 'at sign' ('@'). value. ExtractUserIDFromANR supports 2 values [Y / N]. The default value is N. e.g. ExtractUserIDFromANR=Y
FBAFile	**FBAFile** specifies the location of the file that will be used for UserId and Password authentication. The default value is userlist.auth. e.g. FBAFile=c:\userlist.auth Note: Only used if UseFBA is set to 'Y'.
FreeAuxOnTerm	**FreeAuxOnTerm** specifies whether or not the auxiliary memory is to be freed immediately or on termination. FreeAuxOnTerm supports 2 values [Y / N]. The default value is N. e.g. FreeAuxOnTerm=Y

Keyword	Description of Server-side keywords
Groups	**Groups** specifies the list of groups that the authorizations will be performed against. e.g. Groups=grpA;grpF;grpM Note: Only used if UseGroups is set to 'Y'.
GroupFile	**GroupFile** specifies the location of the group file. The default value is 'groups.ini'. e.g. GroupFile=C:\Capitalware\MQAUSX\groups.ini Note: Only used if UseGroupFile is set to 'Y'.
LDAPBaseDN	**LDAPBaseDN** specifies the base DN / LDAP schema to be used when accessing the remote LDAP server. e.g. LDAPBaseDN="CN=Users,DC=yyyy,DC=zzzz,DC=com" Note: Only used if UseLDAP is set to 'Y'.
LDAPBindDN	**LDAPBindDN** specifies the UserId when connecting to the remote LDAP server. e.g. LDAPBindDN="CN=barney,DC=yyyy,DC=zzzz,DC=com" Note: Only used if UseLDAPBindDN is set to 'Y'.
LDAPBindPwd	**LDAPBindPwd** specifies the Password for the LDAPBNindDN UserId when connecting to the remote LDAP server. See Appendix D for details on creating the encrypted Password. e.g. LDAPBindPwd=@jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA Note: Only used if UseLDAPBindDN is set to 'Y'.
LDAPHost	**LDAPHost** specifies the hostname or IP address of the remote LDAP server. e.g. LDAPHost=10.1.10.1 Note: Only used if UseLDAP is set to 'Y'.

Keyword	Description of Server-side keywords
LDAPPort	**LDAPPort** specifies the port number of the remote LDAP server. The default value is 389. e.g. LDAPPort=389 Note: Only used if UseLDAP is set to 'Y'.
LDAPGroupSearchBase	**LDAPGroupSearchBase** specifies the LDAP search base for a LDAP group search. e.g. LDAPGroupSearchBase="dc=capitalware,dc=net" Note: Only used if UseLDAPGroupSearch is set to 'Y'.
LDAPGroupSearchFilter	**LDAPGroupSearchFilter** specifies the LDAP search filter for a LDAP group search. e.g. LDAPGroupSearchFilter="(cn=%USERID%) (objectclass=organizationalPerson)(dc=capitalware,dc=net)" Note: Only used if UseLDAPGroupSearch is set to 'Y'.
LDAPGroupSearchScope	**LDAPGroupSearchScope** specifies the LDAP search scope for a LDAP group search. The default value is 1. LDAPGroupSearchScope values and their meaning: • -1 - LDAP_SCOPE_DEFAULT • 0 - LDAP_SCOPE_BASE • 1 - LDAP_SCOPE_ONELEVEL • 2 - LDAP_SCOPE_SUBTREE • 3 - LDAP_SCOPE_SUBORDINATE_SUBTREE e.g. LDAPGroupSearchScope=1 Note: Only used if UseLDAPGroupSearch is set to 'Y'.
LDAPUserIDSearchBase	**LDAPUserIDSearchBase** specifies the LDAP UserID search base for a LDAP UserID search. e.g. LDAPUserIDSearchBase="dc=capitalware,dc=net" Note: Only used if UseLDAPUserIDSearch is set to 'Y'.

Keyword	Description of Server-side keywords
LDAPUserIDSearchFilter	**LDAPUserIDSearchFilter** specifies the LDAP UserID search filter for a LDAP UserID search. e.g. LDAPUserIDSearchFilter="(cn=%USERID%)(objectclass=organizationalPerson)(dc=capitalware,dc=net)" Note: Only used if UseLDAPUserIDSearch is set to 'Y'.
LDAPUserIDSearchScope	**LDAPUserIDSearchScope** specifies the LDAP UserID search scope for a LDAP UserID search. The default value is 1. LDAPUserIDSearchScope values and their meaning: • -1 - LDAP_SCOPE_DEFAULT • 0 - LDAP_SCOPE_BASE • 1 - LDAP_SCOPE_ONELEVEL • 2 - LDAP_SCOPE_SUBTREE • 3 - LDAP_SCOPE_SUBORDINATE_SUBTREE e.g. LDAPUserIDSearchScope=1 Note: Only used if UseLDAPUserIDSearch is set to 'Y'.
License	**License** specifies the queue manager's license key. Your license will look something like: 10A0-AAAA-BBBBBBBB (Note: This is a sample license only and will NOT work). e.g. License=10A0-AAAA-BBBBBBBB

Keyword	Description of Server-side keywords
LicenseFile	**LicenseFile** specifies the location of License file that contains all of the customer's license keys. The following are the default values for LicenseFile: For Windows: LicenseFile=C:\Capitalware\MQAUSX\mqausx_licenses.ini For IBM MQ 32-bit on Unix and Linux: LicenseFile=/var/mqm/exits/mqausx_licenses.ini For IBM MQ 64-bit on Unix and Linux: LicenseFile=/var/mqm/exits64/mqausx_licenses.ini For IBM MQ on IBM i: LicenseFile=/QIBM/UserData/mqm/mqausx/ mqausx_licenses.ini e.g. LicenseFile=/var/mqm/exits64/mqausx_licenses.ini
LogDiscMessage	**LogDiscMessage** specifies whether or not a disconnect message is outputted to the logfile. LogDiscMessage supports 2 values [Y / N]. The default value is N. e.g. LogDiscMessage=Y
LogFile	**LogFile** specifies the location of the log file. The default is as follows: For Windows: LogFile=C:\Capitalware\MQAUSX\mqausx.log For IBM MQ 32-bit on Unix and Linux: LogFile=/var/mqm/exits/mqausx.log For IBM MQ 64-bit on Unix and Linux: LogFile=/var/mqm/exits64/mqausx.log For IBM MQ on IBM i: LogFile=/QIBM/UserData/mqm/mqausx/mqausx.log

Keyword	Description of Server-side keywords
LogMessageQuote	**LogMessageQuote** specifies what type of quote (single or double) is to be used with the log message. LogMessageQuote supports 2 values [' / "] (single or double quote). The default value is ' (single quote). e.g. LogMessageQuote="
LogMode	**LogMode** specifies what type of logging the user wishes to have. LogMode supports 4 values [Q / N / V / D] where Q is Quiet, N is Normal, V is Verbose and D is Debug. The default value is N. e.g. LogMode=N
LoginDNPrefix	**LoginDNPrefix** specifies the LoginDN Prefix to be used. e.g. LoginDNPrefix=UID
MCCEventWarnLevel	**MCCEventWarnLevel** specifies the percentage of incoming channels to the maximum allowable number of channels that will cause MQAUSX to write a warning message to the event queue. The default value is 80. e.g. MCCEventWarnLevel =80 Note: Only used if both UseMCC and WriteToEventQueue are each set to 'Y'.
MCCGetTimeOut	**MCCGetTimeOut** specifies the number of seconds that the security exit will wait for a reply form the queue manager's command server. The default value is 3. e.g. MCCGetTimeOut=3 Note: Only used if UseMCC is set to 'Y'.
MCCRedoCount	**MCCRedoCount** specifies the number of connections attempts to occur before the next PCF 'display current channel status' is issued. The default value is 5000. e.g. MCCRedoCount=5000 Note: Only used if UseMCC is set to 'Y'.

Keyword	Description of Server-side keywords
MCCRedoMinutes	**MCCRedoMinutes** specifies the period of time in minutes to wait before the next PCF 'display current channel status' is issued. The default value is 720 minutes. e.g. MCCRedoMinutes=720 Note: Only used if UseMCC is set to 'Y'.
NoAuth	**NoAuth** specifies whether or not to turn off authentication for the server-side security exit. ***Be VERY careful with this option because if set to 'Y' then the user will not be prompted for their UserID & password on the client-side even if the client-side security exit is enabled.*** NoAuth supports 2 values [Y / N]. The default value is N. e.g. NoAuth=Y
PAMService	**PAMService** specifies the name of the PAM service to be used by MQAUSX. e.g. PAMService=common-auth Note: Only used if UsePAM is set to 'Y'.
ProxyFile	**ProxyFile** specifies the location of the file to do alternate UserID look-up. The default value is proxy.lst e.g. ProxyFile=C:\proxy.lst Note: Only used if UseProxy is set to 'Y'.
QMgrPwd	**QMgrPwd** specifies the encrypted password the MQAdmin has assign to a queue manager. See Appendix D for details on creating the encrypted password. e.g. QMgrPwd=@jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA Note: Only used if UseQMgrPwd is set to 'Y'.

Keyword	Description of Server-side keywords
RejectADName	**RejectADName** specifies a set of regular expression patterns that the incoming user-specified Microsoft Active Directory Server Name (AD Server Name) would be compared against. You must separate the AD Server Name regular expression patterns with a ';' semi-colon. e.g. RejectADName=ad001;adc0*;ads0*.acme.com Note: Only used if UseRejectADName is set to 'Y'.
RejectHostByName	**RejectHostByName** specifies a list of hostnames that MQAUSX will perform a gethostbyaddr() call against the hostname to compare the returned IP address against the incoming IP address to reject the incoming connection. You must separate the hostnames with a ';' semi-colon. e.g. RejectHostByName=xyz01.acme.com;xyz02.acme.com Note: Only used if UseRejectHostByName is set to 'Y'.
RejectHostname	**RejectHostname** specifies a set of regular expression patterns that the hostname will be compared against. You must separate the hostname regular expression patterns with a ';' semi-colon. e.g. RejectHostname=xyz01.acme.com;xyz02.acme.com Note: Only used if UseAllowHostname is set to 'Y'.
RejectIP	**RejectIP** specifies a set of regular expression patterns that the incoming channel's IP address will be compared against. You must separate the IP regular expression patterns with a ';' semi-colon. e.g. RejectIP=192.168.*.1[0-5][0-9];127.0.0.?;10.*.*.[0-9] Note: Only used if UseAllowIP is set to 'Y'.

Keyword	Description of Server-side keywords
RejectSSLDN	**RejectSSLDN** specifies a set of regular expression patterns that the incoming channel's SSL DN will be compared against. You must separate the SSL DN expression patterns with a ';' semi-colon. e.g. RejectSSLDN=O=xyz*,C=CA;O=abc*,DC=net Note: Only used if UseRejectSSLDN is set to 'Y'.
RejectUserID	**RejectUserID** specifies a set of regular expression patterns that the incoming connection's UserID will be compared against. You must separate each IP regular expression pattern with a ';' semi-colon. e.g. RejectUserID=mq*;abc??;xyz[0-9][a-f];hr[0-9][0-9] Note: Only used if UseRejectUserID is set to 'Y'.
RotateLogDaily	**RotateLogDaily** specifies whether or not daily log file rotation should take place. RotateLogDaily supports 2 values [Y / N]. The default value is Y. e.g. RotateLogDaily=Y
ServerName	**ServerName** specifies a default server name for this entity. This value will be transmitted to the end-user. For a Windows Server, you may specify a domain name. The default is the hostname. e.g. ServerName=ABC123 Note: Only used if UseServerName is set to 'Y'.
SSLCertFileName	**SSLCertFileName** specifies the path and file name of the SSL certificate to be used when connecting to a LDAP server. e.g. SSLCertFileName=certfile.der Note: Only used if UseLDAPSSLCert is set to 'Y'.

Keyword	Description of Server-side keywords
SSLCertPwd	**SSLCertPwd** specifies the Password for the SSLCertFileName. This is only used when the client LDAP is Netscape/Mozilla version. See Appendix D for details on creating the encrypted Password. e.g. SSLCertPwd=$jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA Note: Only used if UseLDAPSSLCert is set to 'Y'.
SSLCertFileType	**SSLCertFileType** specifies the type of SSL certificate that will be used when connecting to a LDAP server. SSLCertFileType supports 2 values [DER / B64]. The default value is 'DER'. e.g. SSLCertFileType=DER Note: Only used if UseLDAPSSLCert is set to 'Y'.
SSLDNAttrLength	**SSLDNAttrLength** specifies the length of the extraction of the UserId from the SSL DN attribute. The default value is '*' (* means all). e.g. SSLDNAttrLength=* Note: Only used if UseSSLUserIDFromDN is set to 'Y'.
SSLDNAttrName	**SSLDNAttrName** specifies the name of SSL DN attribute to be used to extract UserId from. The default value is 'CN'. e.g. SSLDNAttrName=CN Note: Only used if UseSSLUserIDFromDN is set to 'Y'.
SSLDNAttrStartPos	**SSLDNAttrStartPos** specifies the start position for the extraction of the UserId from the SSL DN attribute. The default value is '1'. e.g. SSLDNAttrStartPos=1 Note: Only used if UseSSLUserIDFromDN is set to 'Y'.

Keyword	Description of Server-side keywords
SystemLogMessage	**SystemLogMessage** specifies what messages will be written to the system log.. SystemLogMessage supports 3 values [B / A / R] where B is Both, A is Accepted Only, and R is Rejected Only messages. The default value is B. e.g. SystemLogMessage=B Note: Only used if WriteToSystemLog is set to 'Y'.
UseAllowADName	**UseAllowADName** allows MQ Admin to allow or restrict incoming user-specified AD Server Name by comparing it against a regular expression pattern. UseAllowADName supports 2 values [Y / N]. The default value is N. e.g. UseAllowADName=Y
UseAllowHostByName	**UseAllowHostByName** allows MQ Admin to allow or restrict by performing a gethostbyaddr() call against the hostname to compare the returned IP address against the incoming IP address to allow the incoming connection.. UseAllowHostByName supports 2 values [Y / N]. The default value is N. e.g. UseAllowHostByName=Y
UseAllowHostname	**UseAllowHostname** allows MQ Admin to allow or restrict by hostname by comparing it against a regular expression pattern. UseAllowHostname supports 2 values [Y / N]. The default value is N. e.g. UseAllowHostname=Y
UseAllowIP	**UseAllowIP** allows MQ Admin to allow or restrict incoming channel IP address by comparing it against a regular expression pattern. UseAllowIP supports 2 values [Y / N]. The default value is N. e.g. UseAllowIP=Y
UseAllowSSLDN	**UseAllowSSLDN** allows MQ Admin to allow or restrict incoming channel's SSL DN by comparing it against a regular expression pattern. UseAllowSSLDN supports 2 values [Y / N]. The default value is N. e.g. UseAllowSSLDN=Y

Keyword	Description of Server-side keywords
UseAllowUserID	**UseAllowUserID** allows MQ Admin to allow or restrict incoming UserID by comparing it against a regular expression pattern. UseAllowUserID supports 2 values [Y / N]. The default value is N. e.g. UseAllowUserID=Y
UseANRDelimiter	**UseANRDelimiter** specifies a delimiter to be used when extracting the username from the incoming UserId. UseANRDelimiter supports 2 values [Y / N]. The default value is N. e.g. UseANRDelimiter=Y
UseANRforLDAP	**UseANRforLDAP** specifies that ANR will be used for LDAP authentication. UseANRforLDAP supports 2 values [Y / N]. The default value is N. e.g. UseANRforLDAP=Y
UseANRPostfix	**UseANRPostfix** specifies that MQAUSX will append a value to the user's UserId. UseANRPostfix supports 2 values [Y / N]. The default value is N. e.g. UseANRPostfix=Y
UseANRPrefix	**UseANRPrefix** specifies that MQAUSX will prefix a value to the user's UserId. UseANRPrefix supports 2 values [Y / N]. The default value is N. e.g. UseANRPrefix=Y
UseAuthOrder	**UseAuthOrder** allows the connection to be tested against more than one authentication sources. UseAuthOrder supports 2 values [Y / N]. The default value is N. e.g. UseAuthOrder=Y
UseCDC	**UseCDC** allows the UserId and Password to be authenticated against Centrify's DirectControl rather than the OS. UseCDC supports 2 values [Y / N]. The default value is N. e.g. UseCDC=Y

Keyword	Description of Server-side keywords
UseCredentialCache	**UseCredentialCache** allows the MQAdmin enable credential caching in MQAUSX UseCredentialCache supports 2 values [Y / N]. The default value is Y. e.g. UseCredentialCache=Y
UseECC	**UseECC** allows MQ Admin to have MQAUSX generate an alert when the ECCWarnCount is exceeded. UseECC supports 2 values [Y / N]. The default value is N. e.g. UseECC=Y
UseFBA	**UseFBA** allows the UserId and Password to be authenticated against a file rather than the OS. UseFBA supports 2 values [Y / N]. The default value is N. e.g. UseFBA=Y
UseGroups	**UseGroups** allows or restricts the incoming UserID against an OS group or a group file. UseGroups supports 2 values [Y / N]. The default value is N. e.g. UseGroups=Y
UseGroupFile	**UseGroupFile** specifies whether or not a group file. UseGroupFile supports 2 values [Y / N]. The default value is N. e.g. UseGroupFile=Y
UseLDAP	**UseLDAP** allows the UserId and Password to be authenticated against a remote LDAP server rather than the OS. UseLDAP supports 2 values [Y / N]. The default value is N. e.g. UseLDAP=Y
UseLDAPAuthCompare	**UseLDAPAuthCompare** specifies whether to use ldap_compare_s vs ldap_simple_bind_s LDAP API call. UseLDAPAuthCompare supports 2 values [Y / N]. The default value is N. e.g. UseLDAPAuthCompare=Y

Keyword	Description of Server-side keywords
UseLDAPLoadBalance	**UseLDAPLoadBalance** allows the security exit to rotate through the list of LDAP servers to authenticate the incoming UserId and Password against. UseLDAPLoadBalance supports 2 values [Y / N]. The default value is N. e.g. UseLDAPLoadBalance=Y
UseLDAPGroupSearch	**UseLDAPGroupSearch** allows a LDAP group based search of the UserId to be in a particular group. UseLDAPGroupSearch supports 2 values [Y / N]. The default value is N. e.g. UseLDAPGroupSearch=Y
UseLDAPSSL	**UseLDAPSSL** allows the UserId and Password to be authenticated against a remote LDAP server using SSL. UseLDAPSSL supports 2 values [Y / N]. The default value is N. e.g. UseLDAPSSL=Y
UseLDAPSSLCert	**UseLDAPSSLCert** allows LDAP SSL connection to be used with a certificate. UseLDAPSSLCert supports 2 values [Y / N]. The default value is N. e.g. UseLDAPSSLCert=Y
UseLDAPUserIDSearch	**UseLDAPUserIDSearch** allows a LDAP UserID based search be used. UseLDAPUserIDSearch supports 2 values [Y / N]. The default value is N. e.g. UseLDAPUserIDSearch=Y
UseLoginDNPrefix	**UseLoginDNPrefix** specifies that the wishes to use a LoginDN Prefix. UseLoginDNPrefix supports 2 values [Y / N]. The default value is N. e.g. UseLoginDNPrefix s=Y
UseMCAUser	**UseMCAUser** allows the connection to use the UserID value specified in the channel's MCAUSER field. UseMCAUser supports 2 values [Y / N]. The default value is N. e.g. UseMCAUser=Y

Keyword	Description of Server-side keywords
UseMCC	**UseMCC** allows MQ Admin to set a limit on the maximum number of connections to a given channel. UseMCC supports 2 values [Y / N]. The default value is N. e.g. UseMCC=Y
UseMCCRedo	**UseMCCRedo** keyword specifies whether or not the server-side security exit should issue PCF command. UseMCCRedo supports 2 values [Y / N]. The default value is Y. e.g. UseMCCRedo=Y
UsePAM	**UsePAM** allows the UserId and Password to be authenticated against Pluggable Authentication Modules (PAM) rather than the OS. UsePAM supports 2 values [Y / N]. The default value is N. e.g. UsePAM=Y
UseProxy	**UseProxy** allows an authorized User to use a different UserID for MQ interactions. UseProxy supports 2 values [Y / N]. The default value is N. e.g. UseProxy=N
UseQAS	**UseQAS** allows the UserId and Password to be authenticated against Quest Authentication Services rather than the OS. UseQAS supports 2 values [Y / N]. The default value is N. e.g. UseQAS=Y
UseQMgrPwd	**UseQMgrPwd** allows the MQAdmin to assign a password to a queue manager. UseQMgrPwd supports 2 values [Y / N]. The default value is N. e.g. UseQMgrPwd=Y
UseRejectADName	**UseRejectADName** allows MQ Admin to reject incoming user-specified AD Server Name by comparing it against a regular expression pattern. UseRejectADName supports 2 values [Y / N]. The default value is N. e.g. UseRejectADName=Y

Keyword	Description of Server-side keywords
UseRejectHostByName	**UseRejectIHostByName** allows MQ Admin to perform a gethostbyaddr() call against the hostname to compare the returned IP address against the incoming IP address to reject the incoming connection. UseRejectHostByName supports 2 values [Y / N]. The default value is N. e.g. UseRejectHostByName=Y
UseRejectHostname	**UseRejectIHostname** allows MQ Admin to reject a hostname by comparing it against a regular expression pattern. UseRejectHostname supports 2 values [Y / N]. The default value is N. e.g. UseRejectHostname=Y
UseRejectIP	**UseRejectIP** allows MQ Admin to reject incoming channel IP address by comparing it against a regular expression pattern. UseRejectIP supports 2 values [Y / N]. The default value is N. e.g. UseRejectIP=Y
UseRejectSSLDN	**UseRejectSSLDN** allows MQ Admin to reject incoming channel's SSL DN by comparing it against a regular expression pattern. UseRejectSSLDN supports 2 values [Y / N]. The default value is N. e.g. UseRejectSSLDN=Y
UseRejectUserID	**UseRejectUserID** allows MQ Admin to reject incoming UserID by comparing it against a regular expression pattern. UseRejectUserID supports 2 values [Y / N]. The default value is N. e.g. UseRejectUserID=Y
UserIDFormatting	**UserIDFormatting** specifies how MQAUSX will handle the incoming UserID. UserIDFormatting supports 3 values [A / U / L]. ('As Is, Uppercase and Lowercase). The default value is A. UserIDFormatting=U
UseServerName	**UseServerName** allows MQ Admin to designate a server name. UseServerName supports 2 values [Y / N]. The default value is N. UseServerName=Y

Keyword	Description of Server-side keywords
UseSSLUserIDFromDN	**UseSSLUserIDFromDN** specifies to set the channel's UserId to be the value extracted from the SSL DN. UseSSLUserIDFromDN supports 2 values [Y / N]. The default value is N. e.g. UseSSLUserIDFromDN=Y
WriteToEventQueue	**WriteToEventQueue** specifies if MQAUSX will write an event message containing the log entry information to an event queue. WriteToEventQueue supports 2 values [Y / N]. The default value is N. e.g. WriteToEventQueue=Y
WriteToSystemLog	**WriteToSystemLog** specifies if MQAUSX will write a log entry to the server's 'logging system'. On Windows, the server's 'logging system' is the Event Log and on Unix/Linux, it is the syslog. WriteToSystemLog supports 2 values [Y / N]. The default value is N. The Unix/Linux syslog output can be found for each operating system as follows: ➢ AIX: /var/log/messages ➢ HP-UX: /var/adm/syslog/syslog.log ➢ Linux: /var/log/messages ➢ Solaris: /var/adm/messages e.g. WriteToSystemLog =Y

8 Appendix B – MQAUSX Upgrade Procedures

To upgrade an existing installation of MQAUSX from an older version to a newer version, do please do the following in the appropriate section below.

8.1.1 Windows Upgrade

- Stop all of the channels using the MQAUSX server-side security exit or completely stop the queue manager.
- Backup all MQAUSX IniFiles in the MQAUSX install directory
- If MQAUSX was installed using the Windows Installer then
  - Click the ***Start*** -> ***All Programs*** -> ***Control Panel*** -> ***Add or Remove Programs***, select MQAUSX from the list and click the ***Remove*** button then follow the prompts to remove it
  - Run the ***mqausx-server-setup.exe*** file from the ***Windows-Server*** directory to install the new version
- Otherwise copy the following files (latest version) to the MQAUSX install directory:
  - mqausx.dll, mqausxldap.dll, mqausxldapssl.dll, cwchad.dll, AddRegistryEntries.bat, mqausx.reg, LDAPSDK.DLL, LDAPSSL.DLL, LDAPX.DLL, rotatelog.bat, testldap.exe and testldapssl.exe
- Run AddRegistryEntries.bat batch file
- Restore the MQAUSX IniFiles if they were altered / deleted.
- Start all of the channels using the MQAUSX server-side security exit or restart the queue manager if it was previously stopped.

8.1.2 Linux 32-bit Upgrade

- Login under the mqm account
- Stop all of the channels using the MQAUSX server-side security exit or completely stop the queue manager.
- Backup all MQAUSX IniFiles in the MQAUSX install directory
- Copy the appropriate tar file to the ***/var/mqm/exits/*** directory
- ***Login as root*** (Very Important step)
- Un-tar the contents of the tar file.
  i.e. For AIX, do the following command:
  ```
  tar -xvf  mqausx_aix.tar
  ```
- Run the script as follows:
  ```
  ./setausx.sh
  ```
- ***Logout from root***
- Restore the MQAUSX IniFiles if they were altered / deleted.
- Delete the MQAUSX tar file
- Start all of the channels using the MQAUSX server-side security exit or restart the queue manager if it was previously stopped.

8.1.3 Unix and Linux 64-bit Upgrade

- Stop all of the channels using the MQAUSX server-side security exit or completely stop the queue manager.
- Backup all MQAUSX IniFiles in the MQAUSX install directory
- Copy the appropriate tar file to the ***/var/mqm/exits64/*** directory
- ***Login as root*** (Very Important step)
- Un-tar the contents of the tar file.
  i.e. For AIX, do the following command:
  ```
  tar -xvf  mqausx_aix53_64.tar
  ```
- Run the script as follows:
  ```
  ./setausx.sh
  ```
- ***Logout from root***
- Restore the MQAUSX IniFiles if they were altered / deleted.
- Delete the MQAUSX tar file
- Start all of the channels using the MQAUSX server-side security exit or restart the queue manager if it was previously stopped.

8.1.4 IBM i Upgrade

- Stop all of the channels using the MQAUSX server-side security exit or completely stop the queue manager.
- Backup all MQAUSX IniFiles in the MQAUSX install directory
- ftp the IBM i files to the IBM i server as follows:
  ```
  ftp -s:mqausx_iseries.ftp  iseries_hostname
  ```

```
your-IBM i-userid
your-IBM i-password

binary
cd QGPL
put mqausx.savf
quote SITE NAMEFMT 1
cd /QIBM/UserData/mqm/
put mqausx_iseries.tar
quit
```

- Log onto the target IBM i server and do the following commands:

```
CLRLIB LIB(MQAUSX)
RSTLIB SAVLIB(MQAUSX) DEV(*SAVF) SAVF(QGPL/MQAUSX)
CLRSAVF FILE(QGPL/MQAUSX)
CHGOBJOWN  OBJ(MQAUSX) OBJTYPE(*LIB)  NEWOWN(QMQM)
qsh
cd /QIBM/UserData/mqm/
tar -xvf mqausx_iseries.tar
chown -R QMQM mqausx
rm mqausx_iseries.tar
```

- Restore the MQAUSX IniFiles if they were altered / deleted.
- Start all of the channels using the MQAUSX server-side security exit or restart the queue manager if it was previously stopped.

9 Appendix C - FBA Encrypted File

The user can create a file that will contain the UserID and encrypted Password. The ***enc_server*** program is used to create and manage a file that will contain the server-side UserID and encrypted Password. Enc_server functions a lot like the Unix programs: adduser, rmuser and passwd but all combined together. Enc_server uses the same Unix crypt method as the Unix ***passwd*** program to encrypt the password. The enc_server's file format is very similar to the Unix ***/etc/shadow*** password file.

Syntax:

```
enc_server {-a | -d | -r} -u UserId -p Password [-f outfilename]
```

Where :

- -a specifies that a UserID and Password are to be added to the file
- -d specifies that a UserID and Password are to be deleted from the file
- -r specifies that a UserID and Password are to be replaced in the file
- UserId is the user's remote UserID (remote Logon ID)
- Password is the user's Password to be encrypted
- outfilename is the output file name (optional)

9.1 Examples

9.1.1 Windows

To use the ***enc_server*** program on Windows, open a Command prompt and change the directory to `C:\Capitalware\MQAUSX\`

```
Add a UserId & Password:
enc_server.exe -a -u barney -p bedrock
enc_server.exe -a -u barney -p bedrock -f C:\temp\fba.enc
Delete a UserId & Password:
enc_server.exe -d -u barney
enc_server.exe -d -u barney -f C:\temp\fba.enc
Replace a UserId & Password:
enc_server.exe -r -u barney -p bedrock
enc_server.exe -r -u barney -p bedrock -f C:\temp\fba.enc
```

9.1.2 Linux 32-bit

To use the enc_server program on Linux for MQ 32-bit, open a shell prompt and change directory to `/var/mqm/exits/`

```
Add a UserId & Password:
enc_server -a -u barney -p bedrock
enc_server -a -u barney -p bedrock -f /tmp/fba.enc
Delete a UserId & Password:
enc_server -d -u barney
enc_server -d -u barney -f /tmp/fba.enc
Replace a UserId & Password:
enc_server -r -u barney -p bedrock
enc_server -r -u barney -p bedrock -f /tmp/fba.enc
```

9.1.3 Unix and Linux 64-bit

To use the enc_server program on Unix/Linux for MQ 64-bit, open a shell prompt and change directory to `/var/mqm/exits64/`

```
Add a UserId & Password:
enc_server -a -u barney -p bedrock
enc_server -a -u barney -p bedrock -f /tmp/fba.enc
Delete a UserId & Password:
enc_server -d -u barney
enc_server -d -u barney -f /tmp/fba.enc
Replace a UserId & Password:
enc_server -r -u barney -p bedrock
enc_server -r -u barney -p bedrock -f /tmp/fba.enc
```

9.1.4 IBM i

To use the enc_server program on IBM i for MQ, open a shell prompt (**QSH**) and change directory to **/QIBM/UserData/mqm/**

```
Add a UserId & Password:

CALL MQAUSX/ENC_SERVER PARM('-a' '-u' 'barney' '-p' 'bedrock')

CALL MQAUSX/ENC_SERVER PARM('-a' '-u' 'barney' '-p' 'bedrock' '-f'
'/QIBM/UserData/mqm/mqausx/fba.enc')

Delete a UserId & Password:

CALL MQAUSX/ENC_SERVER PARM('-d' '-u' 'barney'

CALL MQAUSX/ENC_SERVER PARM('-d' '-u' 'barney' '-f'
'/QIBM/UserData/mqm/mqausx/fba.enc')

Replace a UserId & Password:

CALL MQAUSX/ENC_SERVER PARM('-r' '-u' 'barney' '-p' 'bedrock')

CALL MQAUSX/ENC_SERVER PARM('-r' '-u' 'barney' '-p' 'bedrock' '-f'
'/QIBM/UserData/mqm/mqausx/fba.enc')
```

9.2 Password Restrictions

The following are the restrictions the MQAdmin must follow when creating FBA passwords with enc_server program.

- It must be at least 8 characters in length and less than or equal to 32.
- It must contain a lowercase character (a-z).
- It must contain an uppercase character (A-Z).
- It must contain a numeric digit (0-9).
- It must contain a punctuation character. i.e. !\"#$%&'()*+,-./:;><=?@[\\]^_`{|}\
- It cannot contain the UserId.
- It cannot contain any spaces.

10 Appendix D - Encrypt Password

The Encrypt Password ('enc_pwd') program is used to encrypt the password for LDAPBindPwd, SSLCertPwd and QMgrPwd keywords.

Syntax:

```
enc_pwd  mytestpwd [-R 1|2]
```

Where :

- Password is the user's Password to be encrypted
- R (release) and the default is 2 (AES encryption). Use 1 for older TEA encryption

enc_pwd outputs the encrypted password to the user's screen as follows:

```
Encrpyted Password: @jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA
```

Copy the 47 characters beginning with the "@" and place them in the IniFile for either LDAPBindPwd, SSLCertPwd or QMgrPwd keywords.

e.g.

```
LDAPBindPwd=@jXzFNIKKwZ52wsQ3CUwqWUBpDaoVRDnLMDkNqhVEOcswMA
```

10.1 Examples

10.1.1 Windows

To use the ***enc_pwd*** program on Windows, open a Command prompt and change the directory to `C:\CapitalWare\MQAUSX\`

```
enc_pwd.exe bedrock
```

10.1.2 Linux 32-bit

To use the ***enc_pwd*** program on Unix/Linux for MQ 32-bit, open a shell prompt and change directory to `/var/mqm/exits/`

```
enc_pwd bedrock
```

10.1.3 Unix and Linux 64-bit

To use the ***enc_pwd*** program on Unix/Linux for MQ 64-bit, open a shell prompt and change directory to `/var/mqm/exits64/`

```
enc_pwd bedrock
```

10.1.4 IBM i

To use the ***enc_pwd*** program on IBM i for MQ, open a shell prompt (`QSH`) and change directory to `/QIBM/UserData/mqm/`

```
CALL MQAUSX/ENC_PWD PARM('bedrock')
```

11 Appendix E – Test LDAP IniFile Values

MQAUSX includes 2 testing tools called: testldap and testldapssl. The purpose of the tools is to facilitate easy testing of UserID and Password against an LDAP server. Note: testldapssl is available on limited platforms.

Syntax:

```
testldap -u UserId -p Password [-f mqausx_inifile]

testldapssl -u UserId -p Password [-f mqausx_inifile]
```

Where :

- UserId is the user's LDAP UserID
- Password is the user's LDAP Password
- mqausx_inifile is the MQAUSX IniFile that is being tested (optional)

11.1 Examples

11.1.1 Windows

To use the testldap or testldapssl program on Windows, open a Command prompt and change the directory to `C:\Capitalware\MQAUSX\`

```
testldap.exe -u barney -p bedrock

testldap.exe -u barney -p bedrock -f mqausx_test.ini

testldapssl.exe -u barney -p bedrock

testldapssl.exe -u barney -p bedrock -f mqausx_test.ini
```

11.1.2 Linux 32-bit

To use the testldap or testldapssl program on Unix/Linux for MQ 32-bit, open a shell prompt and change directory to `/var/mqm/exits/`

```
testldap -u barney -p bedrock

testldap -u barney -p bedrock -f /tmp/mqausx_test.ini

testldapssl -u barney -p bedrock

testldapssl -u barney -p bedrock -f /tmp/mqausx_test.ini
```

11.1.3 Unix and Linux 64-bit

To use the testldap or testldapssl program on Unix/Linux for MQ 64-bit, open a shell prompt and change directory to **/var/mqm/exits64/**

```
testldap -u barney -p bedrock

testldap -u barney -p bedrock -f /tmp/mqausx_test.ini

testldapssl -u barney -p bedrock

testldapssl -u barney -p bedrock -f /tmp/mqausx_test.ini
```

11.1.4 IBM i

To use the testldap program on IBM i for MQ, open a shell prompt (**QSH**) and change directory to **/QIBM/UserData/mqm/**

```
CALL MQAUSX/TESTLDAP PARM('-u' 'barney' '-p' 'bedrock')

CALL MQAUSX/TESTLDAP PARM('-u' 'barney' '-p' 'bedrock' '-f'
'/QIBM/UserData/mqm/mqausx/mqausx_test.ini')
```

12 Appendix F – Capitalware Product Display Version

MQAUSX includes a program to display the product version number. The command to display the product version number is:

`cwdspver`

12.1 Examples

12.1.1 Windows

To use the cwdspver program on Windows, open a Command prompt and change the directory to `C:\Capitalware\MQAUSX\` and type the following:

```
cwdspver.exe
```

12.1.2 Linux 32-bit

To use the cwdspver program on Unix/Linux for MQ 32-bit, open a shell prompt and change directory to `/var/mqm/exits/` and type the following:

```
./cwdspver
```

12.1.3 Unix and Linux 64-bit

To use the cwdspver program on Unix/Linux for MQ 64-bit, open a shell prompt and change directory to `/var/mqm/exits64/` and type the following:

```
./cwdspver
```

12.1.4 IBM i

To use the cwdspver program on IBM i, issue the following command on the Command Prompt:

```
CALL MQAUSX/CWDSPVER
```

13 Appendix G – Encryption

MQ Authenticate User Security Exit Solution uses the Advanced Encryption Standard (AES) for encryption and decryption of the user's password between the client-side security exit and the server-side security exit.

Wikipedia

> *the Advanced Encryption Standard (AES) is an encryption standard adopted by the U.S. government. The standard comprises three block ciphers, AES-128, AES-192 and AES-256, adopted from a larger collection originally published as Rijndael. Each AES cipher has a 128-bit block size, with key sizes of 128, 192 and 256 bits, respectively. The AES ciphers have been analyzed extensively and are now used worldwide, as was the case with its predecessor,[3] the Data Encryption Standard (DES).*
>
> *AES was announced by National Institute of Standards and Technology (NIST) as U.S. FIPS PUB 197 (FIPS 197) on November 26, 2001 after a 5-year standardization process in which fifteen competing designs were presented and evaluated before Rijndael was selected as the most suitable (see Advanced Encryption Standard process for more details). It became effective as a Federal government standard on May 26, 2002 after approval by the Secretary of Commerce. It is available in many different encryption packages. AES is the first publicly accessible and open cipher approved by the NSA for top secret information*

14 Appendix H – Support

The support for MQ Authenticate User Security Exit can be found at the following location:

By email at:
support@capitalware.com

By regular mail at:

Capitalware Inc.
Attn: MQAUSX Support
Unit 11, 1673 Richmond Street, PMB524
London, Ontario N6G2N3
Canada

15 Appendix I – Summary of Changes

- MQ Authenticate User Security Exit v3.5.0
  Server-side:
  - Fixed a bug in the enc_server program when deleting a UserId (-p option is not required).
  - Enhanced the code for dumping the pointers passed into exit.
  - Fixed an issue in the subroutine that removes trailing blanks
  - Fixed issue when an invalid or expired license key is used
  - Fixed an issue with default exit path

- MQ Authenticate User Security Exit v3.4.0
  Server-side:
  - Added code to check the length of the incoming UserId & Password in the MQCSP structure
  - Tuned the code that is called on entry
  - Tuned the logging code

- MQ Authenticate User Security Exit v3.3.0
  Server-side:
  - Added code to check all cache entries if they have expired.
  - For Linux & Windows switched to Novell's latest release of LDAP Libraries for C
  - Fixed an issue in the logging framework where a constant was being modified.

- MQ Authenticate User Security Exit v3.2.0
  Server-side:
  - Added support for log disconnect message (new keyword: LogDiscMessage)
  - Added support for single or double quotes for log message (new keyword: LogMessageQuote)
  - Added support when the auxiliary memory is to be freed - immediately vs on termination (new keyword: FreeAuxOnTerm)
  - Enhanced logging - the LogFile keyword now supports the following tokens: %QM%, %CHL%, %UID%, %PID% & %TID%
  - Fixed an issue with Credential Cache shared memory
  - Fixed an issue with ECC shared memory
  - Fixed an issue with MCC shared memory

- MQ Authenticate User Security Exit v3.1.0
  Server-side:
  - Added Credential Cache - MQAUSX will cache (when enabled) the user credentials (in an encrypted format) for 'x' minutes (default is 5 minutes) in shared memory.
  - Fixed an issue on Windows with freeing environment variable memory (error with FreeEnvironmentStrings Windows API call)

- Fixed an issue with auxiliary memory (if used) not being freed on a connection rejection
- Fixed an issue with using "size_t" variable type when it should have been "int"

Client-side:

- Fixed an issue with the Login window (mqausxclnt) not being displayed from .NET.

- MQ Authenticate User Security Exit v3.0.0

Server-side:

- Added support for Queue Manager Password Authentication (new keywords: UseQMgrPwd & QMgrPwd)
- Added support for Pluggable Authentication Modules (PAM) for Unix and Linux (new keywords: UsePAM & PAMService)
- Ability to monitor for excessive client connections (ECC) and then generate an alert (new keywords: UseECC, ECCWarnCount & ECCInterval)
- Fixed an issue with shared memory on Windows
- Fixed an issue with the installer script on Windows missing the mqausx_MI.dll file

Client-side:

- Fixed an issue with the .NET MQAUSX component using the MQAUSX_CLNT environment variable

- MQ Authenticate User Security Exit v2.1.0

Server-side:

- Added new CheckFinalUserID keyword to MQAUSX. It willl take the final UserID and reprocess it against UseAllowUserID, AllowUserID, UseRejectUserID, RejectUserID and Allowmqm keywords.
- Added code to display 'Remote' UserID that the client application is actually running under
- Added code to display MQAUSX client-side version, type of application (Native/DotNet/Java) and platform it is running on
- Improved the IniFile processing speed.
- Fixed an issue with Enterprise License key not being loaded from a License file.
- Fixed an issue with MQAUSX not recognizing the filename specified for FBAFile keyword.
- Tested with MQ v8.0
- Tested with Windows 8/8.1

Client-side:

- Added code to send the UserID (labeling it as Remote UserID) that the application is running under
- Added code to send the MQAUSX version, what type of application and platform that the application is running on
- Fixed a bug on Windows 7, GetDlgItemText buffer was not being cleared of trailing garbage

- Fixed an issue with MQAUSXJ, MQ v8.0 and SecExitData field being null vs blank
- Fixed an issue with MQAUSXJ being re-instantiates and a flag was set incorrectly for reuse (after a failure).

- MQ Authenticate User Security Exit v2.0.1

  Server-side:

  - Added UseMCCRedo flag to control MCCRedoCount, MCCRedoMinutes and MCCGetTimeOut
  - Added UserIDFormatting flag to force lowercase/uppercase/as_is UserID formatting on all platforms
  - Renamed AllowMQCSPAuth flag to AllowPlainTextCredentials

- MQ Authenticate User Security Exit v2.0.0

  Server-side:

  - MQAUSX server-side security exit defaults to use AES 256-bit encryption for user credentials
  - Added support for authentication against Quest Authentication Services (QAS) aka Vintela Authentication Services on Unix/Linux
  - Added support for authentication against Centrify's DirectControl (CDC) on Unix/Linux
  - Added keyword UseLDAPGroupSearchBindDN so that the bindDN and BindPswd will be used for an LDAP Search if UseLDAPBindDN is set to No
  - Added keyword UseLDAPAuthCompare so that the ldap bind will be used for authentication rather than ldap compare
  - Added keyword UseAllowHostname and AllowHostname to only allow hosts by name (reverse lookup of incoming IP address)
  - Added keyword UseRejectHostname and RejectHostname to explicitly reject a hostname (reverse lookup of incoming IP address)
  - Added keyword UseAllowHostByName and AllowHostByName to only allow hosts by name
  - Added keyword UseRejectHostByName and RejectHostByName to explicitly reject a hostname
  - Added keyword SystemLogMessage to control what type of messages ('accepted' and/or 'rejected') are written to system log
  - Added keywords UseGroups, Groups, UseGroupFile & GroupFile
  - Added program cwdspver to display the product version number
  - Added code in the Ini parser to distinguish between 'ABC' and 'ABCDEF' keywords
  - enc_pwd program defaults to use AES 256-bit encryption
  - Increased the accepted IniFile parameter length from 1024 to 2048 characters
  - Added support non-default install for MQ v7.1 & higher multi-install feature on IBM i, Linux, Unix and Windows
  - Added 64-bit client-side security exit for Windows
  - Updated LDAPGroupSearchBase and LDAPGroupSearchFilter processing to replace all occurrences of %USERID% - not just the 1st occurrence

- Updated the "Connection accepted" log record to include the UserID set for the connection.
- Updated MCC logic so that a command server failure does not affect the exit.
- Changed MCCRedoCount default value from 1000 to 5000
- Fixed a bug with LDAP ANR processing
- Fixed a bug with ConnectionName when both IPv4 and IPv6 stacks are used
- Fixed a bug with UseAuthOrder and AuthOrder
- Fixed a bug in the in-memory Ini parser
- Fixed a bug with Proxy file processing
- Fixed a bug in the AllowSSLDN processing
- Fixed a bug in CWCHAD when NoAuth is used
- Fixed a bug in AllowHostname on Linux
- Fixed an issue with BackupLogFileCount
- Fixed a bug with SSLPeerNamePtr field.
- Fixed a memory leak with LogonUser API call on Windows
- Fixed a bug with LDAP SSL looping again when a failed authentication happens
- Fixed weird error with dlsym on Solaris
- Tested with MQ v7.5

Client-side:

- MQAUSX client-side security exit defaults to use AES 256-bit encryption for user credentials
- enc_clnt program defaults to use AES 256-bit encryption
- Fixed the font for CCDTE and Encrypted Client GUI
- Added support to explicitly reject an IP address and/or hostname (RejectConName / MQAUSX_REJECT_CONNAME)
- Added support to explicitly reject a queue manager (RejectQMgrName / MQAUSX_REJECT_QMGR_NAME)
- Fixed a bug in MQAUSXJ in processing SCYDATA using inline parameters (u=...;p=...)
- Added code to get around APAR IZ69820

- MQ Authenticate User Security Exit v1.5.0

Server-side:

- Added AllowSSLSSCert IniFile keyword to enable the check for Self-signed Certificate
- Added UseSSLUserIDFromDN, SSLDNAttrName, SSLDNAttrStartPos and SSLDNAttrLength IniFile keywords to extract the UserID from the channel's SSL DN field
- Added Netscape/Mozilla style LDAP SSL support for AIX, HP-UX and Solaris
- Added SSLCertPwd IniFile keyword for Netscape/Mozilla style LDAP SSL
- Added UseLDAPBindDN, LDAPBindDN and LDAPBindPwd IniFile keywords to enable the use of a Bind UserID and Password for a LDAP connection
- Created program enc_pwd to encrypt the password used for LDAPBindPwd and SSLCertPwd
- Added support for "OR" of LDAPBaseDN values.
- Added LicenseFile to support multiple license keys in a single file

- Added support for MQAUSX_HOME environment variable to explicitly define an IniFile location
- Fixed a bug on HP-UX IA64 using LDAP
- Fixed a bug with Proxy file processing
- Fully tested and supported for Windows 7 Professional

- MQ Authenticate User Security Exit v1.4.0

Server-side:

- New supported platform: IBM i (OS/400)
- New supported platform: AIX 6.1
- Major performance and tuning to many modules - a 7% - 12% improvement in speed depending on features used
- Added ***enc_server*** - it is used to create an encrypted server-side FBA file (i.e. /etc/shadow). enc_server is similar/combination to the Unix programs: useradd, userdel and passwd including Unix crypt.
- Added ***testldap*** and ***testldapssl*** helper programs to allow quick testing of the the LDAP keywords in the MQAUSX IniFile.
- Added support for up to 10 LDAP servers.
- Added UseLDAPLoadBalance keyword which is used for LDAP load balancing when 2 or more LDAP servers are specified
- Added support for LDAP group lookup. (UseLDAPGroupSearch, LDAPGroupSearchBase, LDAPGroupSearchFilter and LDAPGroupSearchScope)Added support for LDAP group lookup. (UseLDAPGroupSearch, LDAPGroupSearchBase, LDAPGroupSearchFilter and LDAPGroupSearchScope)
- Added the ability to explicitly reject an incoming IP address based on a pattern-matching (UseRejectIP and RejectIP).
- Added the ability to explicitly reject an incoming UserId based on a pattern-matching (UseRejectUserID and RejectUserID).
- Added the ability to explicitly reject an incoming Active Directory Name based on a pattern-matching (UseRejectADName and RejectADName) * Windows only *.
- Added the code to disable Event Warning messages when WriteToEventQueue is being used.
- Added code to limit the number of messages written to the event queue when WriteToEventQueue is being used.
- Added MCCGetTimeOut keyword to allow the user to define how long to wait on the "DIS CHL(<ChannelName>)" command when UseMCC is being used.
- Added BackupLogFileCount which is used to control the number of backup log files (Default value is 9)
- Fixed a shared memory issue on Windows when UseMCC is being used.

Client-side:

- Fixed a null pointer problem in the client-side code

- MQ Authenticate User Security Exit v1.3.0

Server-side:

- Created new CHAD (Channel Auto-Definition) exit so that MQAUSX can work with cluster channels.
- Added the ability to filter the incoming connection request by the incoming user inputted AD Server Name (Windows only). Purpose is to allow/restrict Microsoft Active Directory names. New keywords: UseAllowADName and AllowADName
- Added the ability to filter the incoming connection request by authenticated UserId (previously it was only when NoAuth=Y).
- Added the ability to write custom MQ Events to System Channel Event Queue to allow MQAUSX to be tied into an MQ Monitoring tool.
  - 9101 for Connection rejected (Authentication failed) event message
  - 9201 for MCC Warning event message
  - 9202 for MCC Exceeded event message
- Incorporated the rotatelog.bat and rotatelog.sh scripts into server-side security exit.
- New supported platform: HP-UX IA64
- Successfully tested with MQ v7.0
- Fixed a bug related to uppercasing a very long UserId.
- Fixed a bug related to a memory leak when using the MCC feature under extreme load.
- Fixed a bug related to a pointer getting clear between shmdt and shmctl calls.
- Created 3 MQAUSX manuals: MQAUSX Cluster Configuration, MQAUSX Programming Guide and MQAUSX Queue Manager To Queue Manager Configuration.

Client-side:

- Added grouping of servers to support quasi-single sign on feature.
- Created a new Windows GUI program called: Client Channel Table Editor. It is a stand-alone Windows GUI program that interacts with MO72 to create/update/delete CLNTCONN channels in a client channel table. It does NOT require MQ (server or client) to be installed on the PC.
- Created a new Windows GUI program called: Encrypt Client File. It is a stand-alone Windows GUI program that will create an encrypted client file for use by the client-side security exit.
- Added 2 new custom MQ API calls: CMQCONN & CMQCONNX. They replace MQCONN & MQCONNX
- Added 48 complete programming samples for C, C++, C-Sharp, Java and VB to demonstrate client-side programming utilizing the client-side security exit.

- MQ Authenticate User Security Exit v1.2.3
  - Fixed an issue with Max Channel Connections on Solaris 8

- MQ Authenticate User Security Exit v1.2.2
  - Fixed an issue with LDAP header files on Linux
  - Fixed an issue with LDAP authentication on Windows
  - Added more debug information related to memory pointers.
  - Changed MCCRedoSeconds keyword to MCCRedoMinutes.

- Changed default values for MCCRedoMinutes and MCCRedoCount
- For client-side exit, use PartnerName if QMgrName is blank

- MQ Authenticate User Security Exit v1.2.1
  - Fixed a bug with File Based Authentication
  - Updated Java client-side security exit to work better with MQ Explorer v6
  - Created Windows batch scripts to use SupportPac MO72 to create client channel tables for use with MQ Explorer v6
  - Updated IniFile keyword handling

- MQ Authenticate User Security Exit v1.2.0
  - Provided a new graphical program, MQAUSX-GUI, to assist the user in creating and managing their MQAUSX IniFiles.
  - Added support for LDAP SSL authentication (with and without any server certification verification)
  - Added support for ANR (Ambiguous Name Resolution) for LDAP authentication - includes 8 new keywords for managing the ANR usage.
  - Added support for user selectable LDAP 'LoginDN Prefix'.
  - Added the ability to write log message to system log facility. (For Unix and Linux, it writes to the syslog and for Windows it writes to the Event Log)
  - Added support for comments in the FBA file.
  - Added support for relative file access for IniFile, LogFile, ProxyFile and SSLCertFileName. Ideal for cross-platform clustered channels.
  - Created a new memory manager - now uses 50% less memory per connection
  - Improved the speed of the IniFile handle.
  - Fixed a bug: with rejected messages not being written to the log file when using UseAllowIP and normal logging.
  - Fixed a bug: with DefaultProxyID not being found when using UseProxy.
  - Fixed a bug: in the IniFile when the last line does not have a trailing CRLF for Windows or LF for Unix/Linux, it did not make use of key word on the last line.

- MQ Authenticate User Security Exit v1.1.7
  - Added a new client-side Java class especially for MQ Explorer V6.
  - Added support for MQAUSX on the following 3 new platforms:
    - 'Linux on zSeries both 32-bit and 64-bit'
    - 'Linux on x86_64' (64-bit)
    - 'Solaris 10 on x86_64' (64-bit)

- MQ Authenticate User Security Exit v1.1.6
  - Added tighter controls for allowing the mqm UserId.
  - Fixed a bug with LDAP on HP-UX.

- MQ Authenticate User Security Exit v1.1.5
  - Added support for IBM MQ v6.0
  - Added support for IBM MQ v6.0's new MQCSP security structure.
  - Added a new supported platform 'Linux on POWER' for MQAUSX.

- MQ Authenticate User Security Exit v1.1.0
  - Added support for direct connection to a remote LDAP server to verify the incoming UserId and Password.
  - Added remote connection name (conname) to the title bar of the client-side popup UserId and Password window.

- MQ Authenticate User Security Exit v1.0.7
  - Added support for NIS on AIX.
  - Added error message if specified IniFile does not exist.
  - Added support for older Linux servers with glibc 2.2.5

- MQ Authenticate User Security Exit v1.0.6
  - Added support for HP-UX trusted mode.

- MQ Authenticate User Security Exit v1.0.5
  - Added support for using the channel's MCAUSER value.

- MQ Authenticate User Security Exit v1.0.4
  - Enhanced the speed for reading IniFile parameters.

- MQ Authenticate User Security Exit v1.0.3
  - General bug fixes.

- MQ Authenticate User Security Exit v1.0.0
  - Initial release.

16 Appendix J – License Agreement

This is a legal agreement between you (either an individual or an entity) and Capitalware Inc. By opening the sealed software packages (if appropriate) and/or by using the SOFTWARE, you agree to be bound by the terms of this Agreement. If you do not agree to the terms of this Agreement, promptly return the disk package and accompanying items for a full refund.
SOFTWARE LICENSE

1. GRANT OF LICENSE. This License Agreement (License) permits you to use one copy of the software product identified above, which may include user documentation provided in on-line or electronic form (SOFTWARE). The SOFTWARE is licensed as a single product, to an individual queue manager, or group of queue managers for an Enterprise License. This Agreement requires that each queue manager of the SOFTWARE be Licensed, either individually, or as part of a group. Each queue manager's use of this SOFTWARE must be covered either individually, or as part of an Enterprise License. The SOFTWARE is in use on a computer when it is loaded into the temporary memory (i.e. RAM) or installed into the permanent memory (e.g. hard disk) of that computer. This software may be installed on a network provided that appropriate restrictions are in place limiting the use to registered queue managers only. Each licensed queue manager will be provided with a perpetual license key and the licensee may continue to use the SOFTWARE, so long as the licensee is current on the Yearly Maintenance Fee. If the licensee stops paying the Yearly Maintenance Fee, then the SOFTWARE must be removed from all systems at the end of the current maintenance period.

2. COPYRIGHT. The SOFTWARE is owned by Capitalware Inc. and is protected by United States Of America and Canada copyright laws and international treaty provisions. You may not copy the printed materials accompanying the SOFTWARE (if any), nor print copies of any user documentation provided in on-line or electronic form. You must not redistribute the registration codes provided, either on paper, electronically, or as stored in the files mqausx.ini, mqausx_licenses.ini or any other form.

3. OTHER RESTRICTIONS. The registration notification provided, showing your authorization code and this License is your proof of license to exercise the rights granted herein and must be retained by you. You may not rent or lease the SOFTWARE, but you may transfer your rights under this License on a permanent basis, provided you transfer this License, the SOFTWARE and all accompanying printed materials, retain no copies, and the recipient agrees to the terms of this License. You may not reverse engineer, decompile, or disassemble the SOFTWARE, except to the extent the foregoing restriction is expressly prohibited by applicable law.

LIMITED WARRANTY

LIMITED WARRANTY. Capitalware Inc. warrants that the SOFTWARE will perform substantially in accordance with the accompanying printed material (if any) and on-line documentation for a period of 365 days from the date of receipt.

CUSTOMER REMEDIES. Capitalware Inc. entire liability and your exclusive remedy shall be, at Capitalware Inc. option, either (a) return of the price paid or (b) repair or replacement of the SOFTWARE that does not meet this Limited Warranty and that is returned to Capitalware Inc.

with a copy of your receipt. This Limited Warranty is void if failure of the SOFTWARE has resulted from accident, abuse, or misapplication. Any replacement SOFTWARE will be warranted for the remainder of the original warranty period or thirty (30) days, whichever is longer.

NO OTHER WARRANTIES. To the maximum extent permitted by applicable law, Capitalware Inc. disclaims all other warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to the SOFTWARE and any accompanying written materials.

NO LIABILITY FOR CONSEQUENTIAL DAMAGES. To the maximum extent permitted by applicable law, in no event shall Capitalware Inc. be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use or inability to use the SOFTWARE, even if Capitalware Inc. has been advised of the possibility of such damages.

17 Appendix K – Notices

Trademarks:

AIX, IBM, MQSeries, OS/2 Warp, OS/400, IBM i, MVS, OS/390, WebSphere, IBM MQ and z/OS are trademarks of International Business Machines Corporation.

HP-UX is a trademark of Hewlett-Packard Company.

Intel is a registered trademark of Intel Corporation.

Java, J2SE, J2EE, Sun and Solaris are trademarks of Sun Microsystems Inc.

Linux is a trademark of Linus Torvalds.

Mac OS X is a trademark of Apple Computer Inc.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation.

UNIX is a registered trademark of the Open Group.

WebLogic is a trademark of BEA Systems Inc.