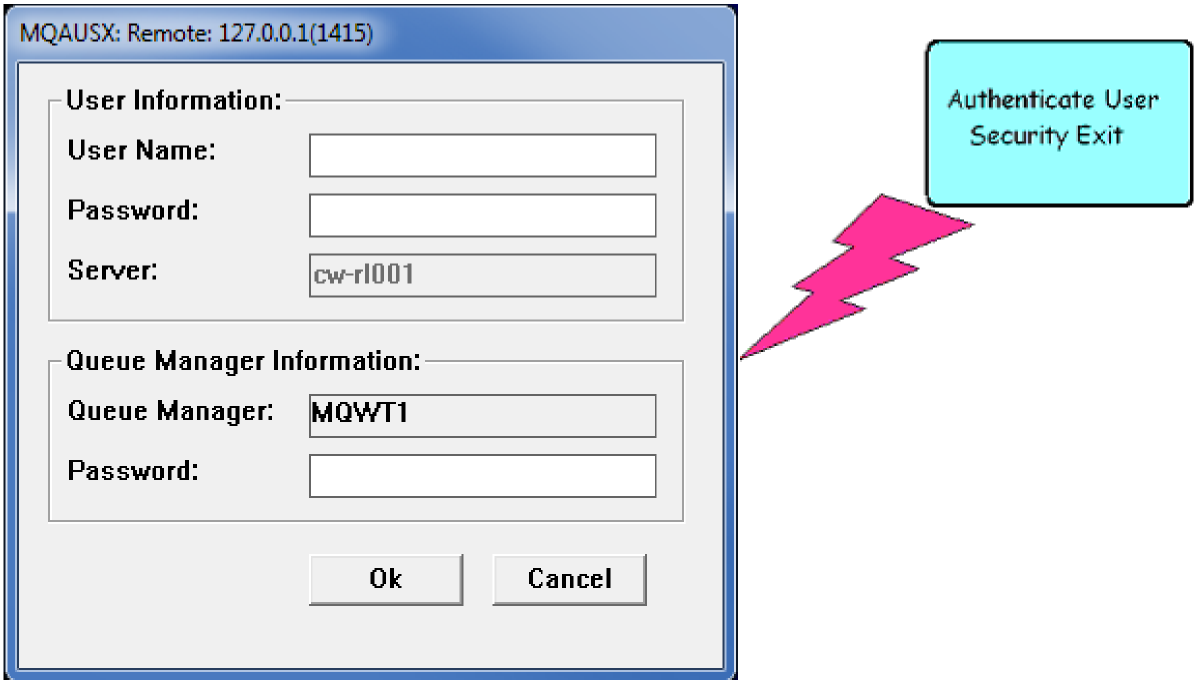
MQAUSX for z/OS Server-side Installation and Operation Manual



MQAUSX: Remote: 127.0.0.1(1415)

User Information:

User Name:

Password:

Server: cw-rl001

Queue Manager Information:

Queue Manager: MQWT1

Password:

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

Last Updated: July 2020.

Table of Contents

1 Introduction

1.1 Overview

MQ Authenticate User Security Exit for z/OS (z/MQAUSX) is a solution that allows a company to fully authenticate a user who is accessing a IBM MQ resource. It verifies the User's UserId and Password against the z/OS server's native OS system.

The security exit will operate with IBM MQ v5.3.1, v6.0, v7.0, v7.1, v8.0, v9.0, v9.1 and v9.2 in z/OS v1.4 or higher environments. It works with Server Connection, Client Connection, Sender, Receiver, Server, Requester, Cluster-Sender and Cluster-Receiver channels of IBM MQ queue manager.

The MQ Authenticate User Security Exit for z/OS 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, 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 authentication against the native OS or an encrypted z/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 z/OS or file based validation data 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 z/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 'CHIN' or the CHIN's Started-task 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.

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.

z/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 z/MQAUSX IniFile parameter NoAuth to Y then it functions just like MQ Standard Security Exit for z/OS (z/MQSSX). z/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 for z/OS*** 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 as:

- z/OS load-module

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

- Authenticate a user against the server's native z/OS or a z/MQAUSX 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 hostname against a regular expression pattern
- Allows or restricts the incoming SSL DN against a regular expression pattern
- Allows or restricts the use of 'CHIN' or the CHIN's Started-task UserIds
- Ability to turn off server-side authentication
- Allows or restricts the incoming UserID against a regular expression pattern when authentication is off
- 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 logging capability for all connecting client applications regardless if they were successful or not.
- Provides logging capability via Write To Operator (WTO) facility.

1.2.2 Client-Side Security Exit

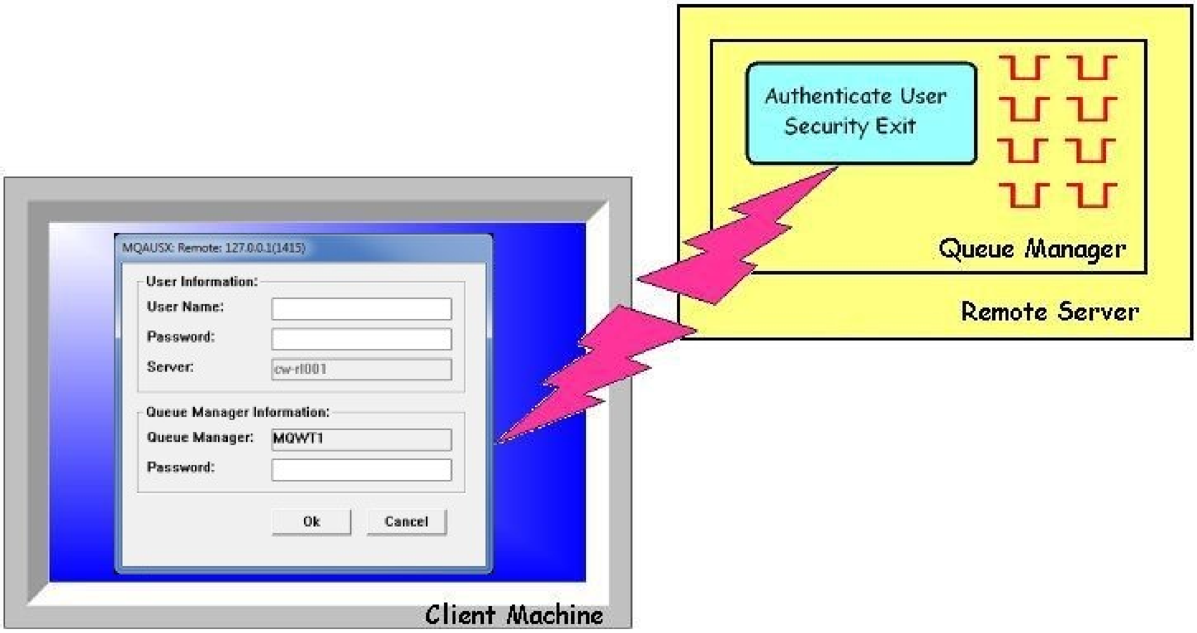
The client-side security exit is available in 4 forms:

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

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

- IBM's MQ Explorer v7.0, 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)



MQAUSX: Remote: 127.0.0.1(1415)

User Information:

User Name:

Password:

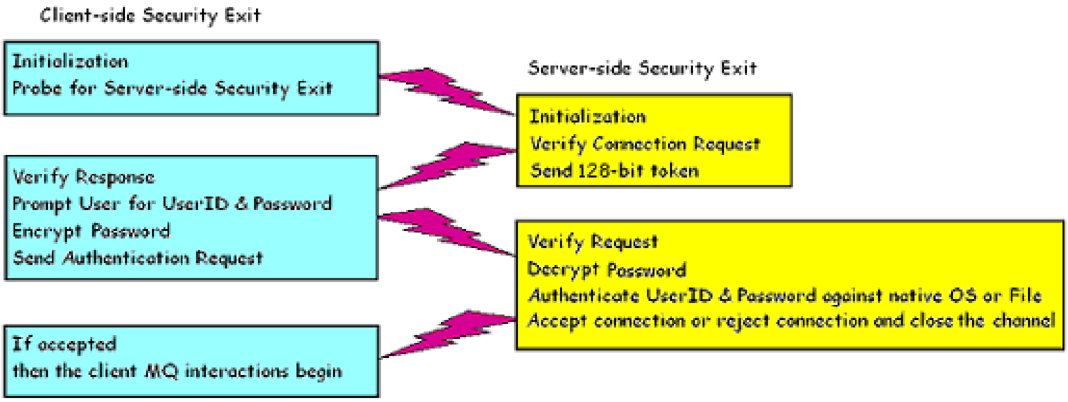
Server: cw-rl001

Queue Manager Information:

Queue Manager: MQWT1

Password:

1.4 Security Message Flow (Logical View)



Client-side Security Exit

Initialization
Probe for Server-side Security Exit

Server-side Security Exit

Initialization
Verify Connection Request
Send 128-bit token

Verify Response
Prompt User for UserID & Password
Encrypt Password
Send Authentication Request

Verify Request
Decrypt Password
Authenticate UserID & Password against native OS or File
Accept connection or reject connection and close the channel

If accepted
then the client MQ interactions begin

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 for z/OS.

1.5.1 Operating System

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

1.5.1.1 IBM z/OS

- IBM z/OS v1.4 or higher

1.5.2 IBM MQ

- IBM MQ for z/OS v5.3.1, v6.0, v7.0, v7.1, v8.0, v9.0, v9.1 and v9.2

2 Installing MQ Authenticate User Security Exit for z/OS

This section describes how to install Capitalware's MQ Authenticate User Security Exit for z/OS.

2.1 Server-side Security Exit

The following files are the platform specific server-side security exits and the required initialization file (IniFile):

2.1.1 z/OS Installation

To install the MQAUSX for z/OS, first unzip the **mqausx_zos-setup.zip**. The zip file contains 2 z/OS XMIT prepared datasets.

- **MQAUSX.LOAD.ZOS** is the XMIT dataset that contains the z/OS load-module.
- **MQAUSX.SYSIN.ZOS** is the XMIT dataset 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. ftp the z/OS XMIT prepared datasets to the z/OS LPAR.

```
ftp -s:mqausx.ftp  z/OS_hostname
```

```
your-z/OS-userid
your-z/OS-password

binary
quote SITE recfm=fb lrecl=80 blksize=3120
put MQAUSX.LOAD.ZOS
put MQAUSX.SYSIN.ZOS
quit
```

If the user receives the following error message then they will need to pre-allocate the z/OS datasets:

```
ftp> put MQAUSX.LOAD.ZOS
200 Port request OK.
550-SVC99 RETURN CODE=4 S99INFO=0 S99ERROR=38656 HEX=9700 S99ERSN code X'000003F3'.
550 Unable to create data set xxxxx.MQAUSX.LOAD.ZOS for STOR command.
ftp> put MQAUSX.SYSIN.ZOS
200 Port request OK.
550-SVC99 RETURN CODE=4 S99INFO=0 S99ERROR=38656 HEX=9700 S99ERSN code X'000003F3'.
550 Unable to create data set xxxxx.MQAUSX.SYSIN.ZOS for STOR command.
```

To pre-allocating the XMIT datasets go to option 3.2 of ISPF and allocate both datasets: MQAUSX.LOAD.ZOS and MQAUSX.SYSIN.ZOS.

Use the following dataset attributes when allocating both datasets:

Space	
Units	BLOCKS
Primary Quantity	40
Secondary Quantity	40
Directory Blocks	0
DCB Parameters	
RECFM	FB
LRECL	80
BLKSIZE	3120
DsnType	Blank

After the user has pre-allocated the datasets, they can redo the ftp commands.

2. Log on to z/OS LPAR and issue the following TSO RECEIVE commands:

```
TSO RECEIVE INDATASET(MQAUSX.LOAD.ZOS)
TSO RECEIVE INDATASET(MQAUSX.SYSIN.ZOS)
```

After issuing the above commands, the following product datasets will appear:

- **+HLQ+.CPTLWARE.MQAUSX.LOAD** is the dataset that contains the z/OS load-module.
- **+HLQ+.CPTLWARE.MQAUSX.SYSIN** is a dataset that contains a sample initialization file for the server-side security exit and sample MQSC script to define MQ channels with the security exits.

2.1.2 z/MQAUSX DataSets

z/MQAUSX solution is comprised of 2 datasets: +HLQ+.CPTLWARE.MQAUSX.LOAD and +HLQ+.CPTLWARE.MQAUSX.SYSIN.

2.1.2.1 +HLQ+.CPTLWARE.MQAUSX.LOAD

- **MQAUSX** is the actual security exit z/OS load-module that will be invoked by the MQ Server component.

2.1.2.2 +HLQ+.CPTLWARE.MQAUSX.SYSIN

- **MQAUSXIN** is a sample initialization file for the server-side security exit.
- **AUSXMQSC** is a sample MQSC script to define MQ channels with the security exits.

2.1.3 z/OS CHIN JCL

This section describes the required JCL for z/MQAUSX.

2.1.3.1 CSQXLIB DDName

The MQAUSX load-module needs to be put in the executable path for the CHINIT started-task. There are 2 options for achieving this:

1. Add the dataset to the CSQXLIB concatenation of the CHINIT's CSQXLIB.

```
//CSQXLIB   DD DISP=SHR,DSN=+MQHLQ+.+QMGRNAME+.USERAUTH
//          DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQAUSX.LOAD
```

2. Copy the MQAUSX load-module to your existing MQ exit / link-edited parameter dataset. Here is a sample JCL to copy the MQAUSX load-module:

```
//COPY1   EXEC PGM=IEBCOPY,REGION=1024K
//SYSPRINT DD  SYSOUT=*
//SYSUT3    DD  DSN=&&SYSUT3,UNIT=SYSDA,DISP=(,DELETE),
//          SPACE=(CYL,(5,1))
//SYSUT4    DD  DSN=&&SYSUT4,UNIT=SYSDA,DISP=(,DELETE),
//          SPACE=(CYL,(5,1))
//*
//IN        DD  DISP=SHR,DSN=+HLQ+.CPTLWARE.MQAUSX.LOAD
//*
//OUT       DD  DISP=SHR,DSN=+MQHLQ+.+QMGRNAME+.USERAUTH
//*
//SYSIN     DD  *
 COPYMOD OUTDD=OUT,INDD=((IN,R))
      S M=MQAUSX
/*
```

2.1.3.2 MQAUSXIN DDName

MQAUSXIN is the DDName that points to a dataset containing the IniFile parameters.

Add the following line to the CHINIT's JCL.

```
//MQAUSXIN DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQAUSX.SYSIN(MQAUSXIN)
```

2.1.3.3 PROXY DDName - Optional

PROXY is the DDName that points to a dataset containing the UserId proxy values.

Add the following line to the CHINIT's JCL.

```
//PROXY    DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQAUSX.PROXY
```

To allocate the PROXY dataset go to option 3.2 of ISPF and allocate a dataset using the following dataset attributes:

Space	
Units	BLOCKS
Primary Quantity	40
Secondary Quantity	40
Directory Blocks	0
DCB Parameters	
RECFM	FB
LRECL	80
BLKSIZE	27920
DsnType	Blank

See section 4.12 for more information on the use of Proxy UserIds.

2.1.3.4 FBA - Optional

2.1.3.4.1.1 FBAENC DDName (Encrypted File) - Optional

FBAENC is the DDName that points to a dataset containing the UserId and Encrypted Password values.

Add the following line to the CHINIT's JCL.

```
//FBAENC DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQAUSX.FBAENC
```

To allocate the FBAENC dataset go to option 3.2 of ISPF and allocate a dataset using the following dataset attributes:

Space	
Units	BLOCKS
Primary Quantity	40
Secondary Quantity	40
Directory Blocks	0
DCB Parameters	
RECFM	FB
LRECL	80
BLKSIZE	27920
DsnType	Blank

See section 5.11. for more information on the use of FBA UserId and Password values.

2.1.3.4.1.2 FBAFile DDName (Plain Text File) - Optional

FBAFILE is the DDName that points to a dataset containing the UserId and Password values.

Add the following line to the CHINIT's JCL.

```
//FBAFILE DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQAUSX.FBAFILE
```

To allocate the FBAFILE dataset go to option 3.2 of ISPF and allocate a dataset using the following dataset attributes:

Space	
Units	BLOCKS
Primary Quantity	40
Secondary Quantity	40
Directory Blocks	0
DCB Parameters	
RECFM	FB

LRECL	80
BLKSIZE	27920
DsnType	Blank

See section 5.11.2 for more information on the use of FBA UserId and Password values.

2.1.4 MQAUSX-ISPF-GUI for z/OS Installation

Read section 2 of the ***MQAUSX-ISPF-GUI for z/OS User Guide*** for information on the installation process.

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 verification. The server-side security exit will work with IBM MQ v5.3.1, v6.0, v7.0 or higher.

3.1 z/OS Security Configuration

This section describes how to configure z/OS to allow the server-side security exit to perform the UserId and Password authentication.

The following RACF commands need to be executed on z/OS to allow z/MQAUSX to authenticate the UserId and Password against z/OS:

```
RALTER PROGRAM * ADDMEM('+MQHLQ+.SCSQAUTH'//NOPADCHECK)
RALTER PROGRAM * ADDMEM('+HLQ+.CPTLWARE.MQAUSX.LOAD'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('SYS1.SCSQANLE'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('SYS1.SCSQAUTH'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('SYS1.SCSQLINK'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('SYS1.SCSQLOAD'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('SYS1.SCSQMVR1'//NOPADCHK) UACC(READ)
RALTER PROGRAM * ADDMEM('SYS1.SCSQSNLE'//NOPADCHK) UACC(READ)
SETROPTS WHEN(PROGRAM) REFRESH
```

4 Configuring Server-side Security Exit

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

4.1 z/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 z/MQAUSX's authentication, issue the following MQSC command to disable the builtin authentication mechanism:

```
ALTER QMGR CONNAUTH(‘ ‘)
```

4.2 z/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 z/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)

MQAUSX supports 2 ways to specify an IniFile via the Security User Data (SCYDATA) field: DD Name and DD Name with a Member Name.

4.3.1 SCYDATA with DD Name

In this case, only the DD Name is used to specify the IniFile. The DD Name provided in the SCYDATA field must match the DD Name in the CHIN's JCL. The DD statement's DSN keyword can contain either a fully qualified Partition DataSet with the Member name or a Sequential DataSet.

4.3.1.1 SCYDATA with DD Name using Partition DataSet

The CHIN's DD Name references the DSN keyword which contains the fully qualified Partition DataSet Name (highlighted in **red**) and member name (highlighted in **blue**). Since the Member Name is included in the CHIN'S DD DSN keyword, do not put the Member Name in the SCYDATA field.

e.g.
SCYDATA('**DDName**')

CHIN JCL using Partition DataSet

```
//MQAUSXIN DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQAUSX.SYSIN(MQAUSXIN)
```

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

4.3.1.2 SCYDATA with DD Name using Sequential DataSet

The CHIN's DD Name specifies a DSN which will contain the Sequential DataSet. As seen below, the DD Name in the SCYDATA field matches the DD Name in the CHIN's JCL.
e.g.
SCYDATA('**DDName**')

CHIN JCL using Sequential DataSet

```
//MQAUSXIN DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQAUSX.SYSIN.SEQ
```

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

4.3.2 SCYDATA with DD Name and Member Name

In this case, both the DD Name (highlighted in **red**) and the Member Name (highlighted in **blue**) are used to specify the IniFile since the DSN keyword of the DD statement only contains the Partition DataSet Name. In other words, the user specifies the Member Name as a parameter to the SCYDATA field. This is a dynamic configuration that allows for different IniFiles for different channels.

e.g.
SCYDATA('**DDName**(**MemberName**)')

CHIN JCL using Partition DataSet

```
//MQAUSXIN DD DISP=SHR,DSN=+HLQ+.CPTLWARE.MQAUSX.SYSIN
```

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

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.

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

4.4.1 z/OS

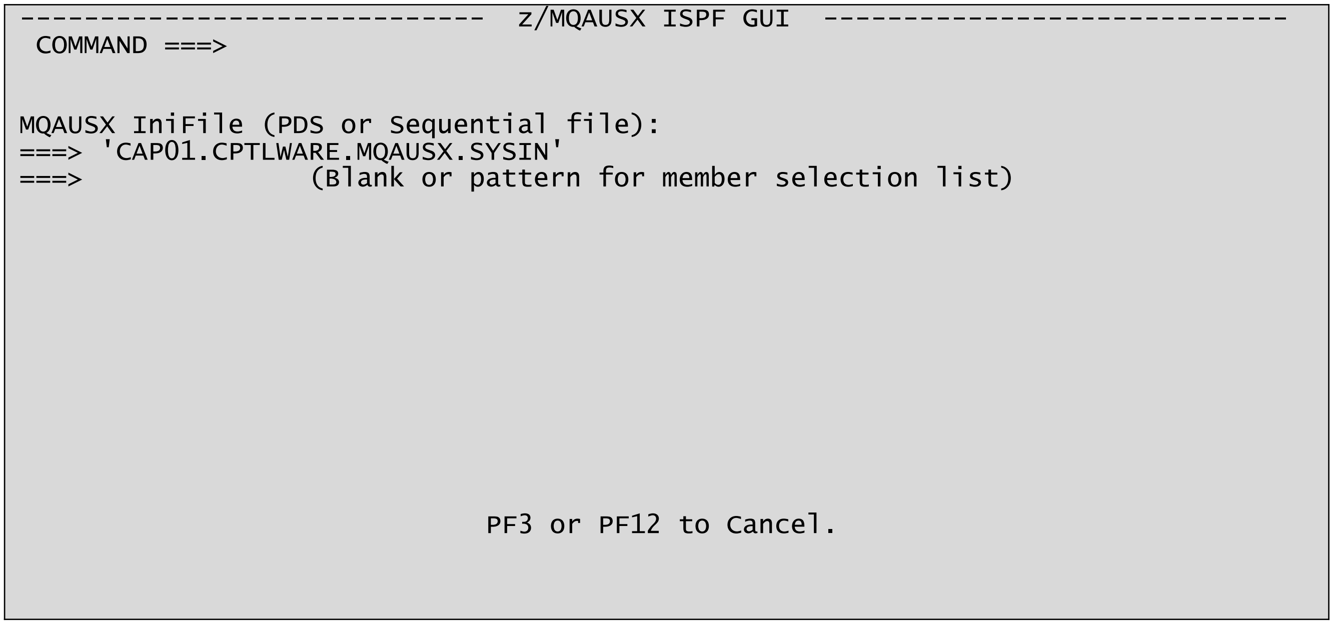
On z/OS, SCYEXIT and SCYDATA will contain the following values assuming a default install:

- SCYEXIT
  MQAUSX
- SCYDATA
  MQAUSXIN

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

4.5 MQAUSX-ISPF-GUI for z/OS

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



```
-----------------------------  z/MQAUSX ISPF GUI  -----------------------------
 COMMAND ===>


MQAUSX IniFile (PDS or Sequential file):
===> 'CAP01.CPTLWARE.MQAUSX.SYSIN'
===>              (Blank or pattern for member selection list)










                             PF3 or PF12 to Cancel.
```

5 IniFile Keywords (Server-side)

This section describes IniFile keywords.

5.1 Logging

This section describes the necessary entries to enable z/MQAUSX to record log information. To enable and control logging, you need 9 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 z/OS:

   ```
   LogFile=SYSPRINT
   ```

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. **WriteToSystemLog** specifies that z/MQAUSX write a log entry to the server's 'logging system'. On z/OS, the server's 'logging system' is JES. The default value is N.

6. **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.

7. **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

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

9. **UseFormFeed** specifies that a FormFeed command be issued once a day at midnight. UseFormFeed supports 2 values [Y / N]. The default value is N.

```
LogMode=N
LogFile=SYSPRINT
WriteToSystemLog=Y
```

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, z/MQAUSX supports 2 authentication sources: files 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 2 values:
    - **files** means the authentication will be against the local z/OS
    - **mqausx** means the authentication will be against z/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= files mqausx
```

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 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:

1. **UseFBA** allows the UserId and Password to be verified against a file rather than the OS
2. **FBAFile** specifies the DD of the encrypted FBA file to do the UserId and Password verification

```
UseFBA=Y
FBAFile=FBAENC
```

Note: For MQAUSX to recognize that it is using an Encrypted FBA file, the DD must end with the letters "ENC". i.e. The DD can be "FBAENC" or "FREDENC", etc..

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

5.3.3 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 DDName of the file to do the UserId and Password verification

```
UseFBA=Y
FBAFile=FBAFILE
```

5.3.3.1 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 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
```

5.5 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
```

5.6 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 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

5.6.1 Authorization against a Group File

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 IniFile with the Group keywords:

```
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
```

z/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 z/MQAUSX will not allow the connection.

Example:

```
UseGroups=Y
Groups=grp1;grp2;grpB
UseGroupFile=Y
GroupFile=GROUP
```

5.7 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 character 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.

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

5.8 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 character 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
```

5.9 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 z/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 z/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 use 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
```

5.10 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 character 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
```

5.11 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 character 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.

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

5.12 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 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 character 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.

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

5.13 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 character 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
```

5.14 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 z/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 z/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
```

5.15 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 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 character 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*;
```

5.16 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 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 character 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]
```

5.17 Excessive Client Connections

This section describes the necessary entries to configure Excessive Client Connections (ECC) alert system in z/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:

- **UseECC** enables excessive client connections feature
- **ECCWarnCount** specifies a count which, when exceeded, will cause an alert to be generated. The default value is 5000.
- **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
```

5.18 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 10 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.
6. **MCCRedoCount** specifies how often the 'display channel status' command should be issued.
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.
8. **ModelQueueName** is the name of the system model reply queue
9. **CommandQueueName** is the name of the command queue used by the Queue Manager's Command Server
10. **TempDynPrefix** is the queue name prefix that will be used when the Queue Manager creates the temporary dynamic queue

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=6000
MCCGetTimeOut=5
#
CommandQueueName=SYSTEM.COMMAND.INPUT
ModelQueueName=SYSTEM.COMMAND.REPLY.MODEL
TempDynPrefix=SYSTEM.MQAUSX.*
```

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
```

5.19 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 DDName of the file to do alternate UserID look-up

```
UseProxy=Y
ProxyFile=PROXY
```

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
```

5.20 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).

```
AllowPlainTextCredentials=Y
```

5.21 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=A
```

5.22 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
```

5.23 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
```

5.24 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
```

5.25 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
```

5.26 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
```

5.27 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.

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

```
AllowSSLSSCert=Y
```

5.28 SSLCertUserID Field

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

- UseSSLCertUserID enables the connection to use the UserID value specified in the channel's SSLCertUserID field

```
UseSSLCertUserID=Y
```

5.29 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:

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

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

5.30 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
```

5.31 LicenseFile

This section will describe how to have a file that contains all of the user's z/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 z/OS DD:

```
LicenseFile=AUSXFILE
```

5.32 License Key

This section will describe how to license MQ Authenticate User Security Exit for z/OS 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 Server-side Log File

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

6.1 z/OS

The log file is located at the following (assuming a default install of SYSPRINT):

CHIN Started-task JES-log

All log entries will be marked with either **INFO** or **ERROR** in columns 21 to 26.

6.1.1 Authentication Log Sample

```
2004/11/18 00:47:53 INFO    MQAUSX #00881: Connection accepted for UserID='rlacroix'
UserSpecifiedServer='roger-1525ca' QMgr='MQA1' ChlName='MY.TEST.EXIT' ConName='127.0.0.1' Server='roger-
1525ca' RemoteUserID=''
2004/11/18 00:49:12 INFO    MQAUSX #00881: Connection accepted for UserID='rlacroix'
UserSpecifiedServer='roger-1525ca' QMgr='MQA1' ChlName='MY.TEST.EXIT' ConName='127.0.0.1' Server='roger-
1525ca' RemoteUserID=''
2004/11/18 00:50:11 INFO    MQAUSX #00881: Connection accepted for UserID='rlacroix'
UserSpecifiedServer='roger-1525ca' QMgr='MQA1' ChlName='MY.TEST.EXIT' ConName='127.0.0.1' Server='roger-
1525ca' RemoteUserID=''
```

6.1.2 Non-Authentication Log Sample

```
2004/11/18 00:47:53 INFO    MQAUSX #00881: Connection accepted for QMgr='MQA1' ChlName='MY.TEST.EXIT'
ConName='127.0.0.1' RemoteUserID=''
2004/11/18 00:49:12 INFO    MQAUSX #00881: Connection accepted for QMgr='MQA1' ChlName='MY.TEST.EXIT'
ConName='127.0.0.1' RemoteUserID=''
2004/11/18 00:50:11 INFO    MQAUSX #00881: Connection accepted for QMgr='MQA1' ChlName='MY.TEST.EXIT'
ConName='127.0.0.1' RemoteUserID=''
```

7 Appendix A – z/MQAUSX IniFile (Server-side)

The sample IniFile below is the z/MQAUSX IniFile supplied for z/OS. The IniFile supports the following keywords and their values:

```
LogMode=N
LogFile=SYSPRINT
UseFBA = N
Allowmqm=N
UseMCC=N
UseAllowIP=N
UseProxy=N
SequenceNumberFlag=N
```

Note: Keywords are case sensitive.

Keyword	Description of Server-side keywords
AllowBlankUserID	**AllowBlankUserID** specifies where 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 z/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'.
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'.

Keyword	Description of Server-side keywords
AllowIP	**AllowIP** specifies a set of regular expression patterns that the incoming channel's IP address will be parsed 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'.
AllowPlainTextCredentials	**AllowPlainTextCredentials** allows the z/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 N. e.g. AllowPlainTextCredentials=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
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'.
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

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 2 supported values: files and mqausx. e.g. AuthOrder= files mqausx Note: Only used if UseAuthOrder is set to 'Y'.
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'.
CommandQueueName	**CommandQueueName** specifies the queue used by the queue manager's Command Server to process MQSC commands. The default value is 'SYSTEM.COMMAND.INPUT'. e.g. CommandQueueName=SYSTEM.COMMAND.INPUT Note: Only used if UseMCC is 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

Keyword	Description of Server-side keywords
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
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'.
FBAFile	**FBAFile** specifies the DDName of the file that will be used for UserId and Password authentication. The default value is 'FBAFILE'. e.g. FBAFile=FBAFILE 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
GroupFile	**GroupFile** specifies the DDName of the file to look up the UserID against a group. The default value is 'GROUP'. e.g. GroupFile=GROUP Note: Only used if UseGroups is set to 'Y'.
License	**License** specifies the queue manager's license key. Your license will look something like: 0000-AAAA-BBBBBBBB (Note: This is a sample license only and will NOT work). e.g. License=0000-AAAA-BBBBBBBB
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 z/OS DD: LicenseFile=AUSXFILE e.g. LicenseFile=AUSXFILE
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 z/OS: LogFile=SYSPRINT

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
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 connection 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 'display current channel status' is issued. The default value is 360 minutes. e.g. MCCRedoMinutes=360 Note: Only used if UseMCC is set to 'Y'.
ModelQueueName	**ModelQueueName** specifies the model queue to be used when z/MQAUSX creates a temporary reply queue. The default value is 'SYSTEM.COMMAND.REPLY.MODEL'. e.g. ModelQueueName=SYSTEM.COMMAND.REPLY.MODEL 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 they have the client-side security exit enabled.*** NoAuth supports 2 values [Y / N]. The default value is N. e.g. NoAuth=Y
ProxyFile	**ProxyFile** specifies the DDName of the file to do alternate UserID look-up. The default value is 'PROXY'. e.g. ProxyFile=PROXY 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
RejectHostByName	**RejectHostByName** specifies a list of hostnames that z/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'.
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'.

Keyword	Description of Server-side keywords
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'.
SequenceNumberFlag	**SequenceNumberFlag** is a z/OS (OS/390) only flag. It states whether or not there are sequence numbers in columns 72 to 80. SequnceNumberFlag supports 2 values [Y / N]. The default value is N. e.g. SequenceNumberFlag = 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'.
TempDynPrefix	**TempDynPrefix** specifies a prefix name for the temporary dymanic queue. The default value is 'SYSTEM.MQAUSX.*'. e.g. TempDynPrefix= SYSTEM.MQAUSX.* Note: Only used if UseMCC is set to '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 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
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
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
UseFormFeed	**UseFormFeed** specifies that a FormFeed command be issued once a day at midnight. UseFormFeed supports 2 values [Y / N]. The default value is N. e.g. UseFormFeed=Y
UseGroups	**UseGroups** allows or restricts the incoming UserID against a group file. UseGroups supports 2 values [Y / N]. The default value is N. e.g. UseGroups=Y

Keyword	Description of Server-side keywords
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
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
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
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
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

Keyword	Description of Server-side keywords
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 z/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
UseSSLCertUserID	**UseSSLCertUserID** allows the connection to use the UserID value specified in the channel's SSLCertUserID field. UseSSLCertUserID supports 2 values [Y / N]. The default value is N. e.g. UseSSLCertUserID=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 that MQAUSX write a log entry to the server's 'logging system'. On z/OS, the server's 'logging system' is JES. WriteToSystemLog supports 2 values [Y / N]. The default value is N. e.g. WriteToSystemLog =Y

`

8 Appendix B – z/MQAUSX Upgrade Procedures

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

1. Stop all of the channels using the z/MQAUSX server-side security exit or stop the queue manager's CHIN (channel initiator).

2. ftp the z/OS XMIT prepared datasets to the z/OS LPAR.

```
ftp -s:mqausx.ftp  z/OS_hostname
```

```
your-z/OS-userid
your-z/OS-password

binary
quote SITE recfm=fb lrecl=80 blksize=3120
put MQAUSX.LOAD.ZOS
quit
```

If the user receives the following error message then they will need to pre-allocate the z/OS datasets:

```
ftp> put MQAUSX.LOAD.ZOS
200 Port request OK.
550-SVC99 RETURN CODE=4 S99INFO=0 S99ERROR=38656 HEX=9700 S99ERSN code X'000003F3'.
550 Unable to create data set xxxxx.MQAUSX.LOAD.ZOS for STOR command.
ftp> put MQAUSX.SYSIN.ZOS
200 Port request OK.
550-SVC99 RETURN CODE=4 S99INFO=0 S99ERROR=38656 HEX=9700 S99ERSN code X'000003F3'.
550 Unable to create data set xxxxx.MQAUSX.SYSIN.ZOS for STOR command.
```

To pre-allocating the XMIT datasets go to option 3.2 of ISPF and allocate both dataset:
MQAUSX.LOAD.ZOS

Use the following dataset attributes when allocating the dataset:

Space	
Units	BLOCKS
Primary Quantity	40
Secondary Quantity	40
Directory Blocks	0
DCB Parameters	
RECFM	FB
LRECL	80
BLKSIZE	3120
DsnType	Blank

After the user has pre-allocated the dataset, the user can redo the ftp commands.

3. Log on to z/OS LPAR and issue the following TSO RECEIVE command:

```
TSO RECEIVE INDATASET(MQAUSX.LOAD.ZOS)
```

After issuing the above command, the following product dataset will appear:

- **+HLQ+.CPTLWARE.MQAUSX.LOAD** is the dataset that contains the z/OS load-module.

4. Start all of the channels using the z/MQAUSX server-side security exit or restart the queue manager's CHIN.

9 Appendix C - FBA Encrypted File

The user can create a file that will contain the UserID and encrypted Password. The ***encsrvr*** 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. encsrvr uses the same Unix crypt method as the Unix ***passwd*** program to encrypt the password. The encsrvr's file format is very similar to the Unix ***/etc/shadow*** password file.

Syntax:

```
ENCSRVR {-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 DD name (optional)

9.1 Examples

9.1.1 z/OS

Add a UserId & Password:

```
//ENCSRVR  EXEC PGM=ENCSRVR,PARM='-a -u barney -p bedrock -f FBAENC'
//SYSPRINT DD  SYSOUT=*
//FBAENC   DD  DISP=SHR,DSN=MY.SEQ.DATA
```

Delete a UserId & Password:

```
//ENCSRVR  EXEC PGM=ENCSRVR,PARM='-d -u barney -f FBAENC'
//SYSPRINT DD  SYSOUT=*
//FBAENC   DD  DISP=SHR,DSN=MY.SEQ.DATA
```

Replace a UserId & Password:

```
//ENCSRVR  EXEC PGM=ENCSRVR,PARM='-r -u barney -p bedrock -f FBAENC'
//SYSPRINT DD  SYSOUT=*
//FBAENC   DD  DISP=SHR,DSN=MY.SEQ.DATA
```

9.2 Password Restrictions

The following are the restrictions the MQAdmin must follow when creating FBA passwords with ENCSRVR 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 – Capitalware Product Display Version

z/MQAUSX includes a program to display the product version number.

10.1 Examples

10.1.1 z/OS

To use the CWDSPVER program on z/OS, use the following JCL:

```
//CWDSPVER EXEC PGM=CWDSPVER
//SYSPRINT DD  SYSOUT=*
//STEPLIB  DD  DISP=SHR,DSN=+HLQ+.CPTLWARE.MQAUSX.LOAD
```

11 Appendix E – Encryption

MQ Authenticate User Security Exit for z/OS 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*

12 Appendix F – Support

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

By email at:
support@capitalware.com

By regular mail at:

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

13 Appendix G – Summary of Changes

- MQ Authenticate User Security Exit v3.5.0
  - 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
  - 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
  - Added code to check all cache entries if they have expired.
  - Fixed an issue in the logging framework where a constant was being modified.

- MQ Authenticate User Security Exit v3.2.0
  - 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)
  - 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
  - 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 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"

- MQ Authenticate User Security Exit v3.0.0
  - Added support for Queue Manager Password Authentication (new keywords: UseQMgrPwd & QMgrPwd)
  - Ability to monitor for excessive client connections (ECC) and then generate an alert (new keywords: UseECC, ECCWarnCount & ECCInterval)

- MQ Authenticate User Security Exit v2.1.0

- Added new CheckFinalUserID keyword. It will 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
- Fixed a bug with UseGroupFile on z/OS
- 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

➢ MQ Authenticate User Security Exit v2.0.1

- Added UseMCCRedo flag to control MCCRedoCount, MCCRedoMinutes and MCCGetTimeOut
- Renamed UppercaseUserID flag to UserIDFormatting. UserIDFormatting supports 3 values: A/U/L (As Is/Uppercase/Lowercase)
- Renamed AllowMQCSPAuth flag to AllowPlainTextCredentials
- Fixed a bug with MQAUSXCL handling the DD for encrypted credential file

➢ MQ Authenticate User Security Exit v2.0.0

- z/MQAUSX server-side security exit defaults to use AES 256-bit encryption for user credentials
- 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 & GroupFile
- Added program CWDSPVER to display the product version number
- Added code in the Ini parser to distinguish between 'ABC' and 'ABCDEF' keywords
- ENCPWD program defaults to use AES 256-bit encryption
- Added keyword UseFormFeed (z/OS only) to issue a FormFeed command once a day at midnight
- Increased the accepted IniFile parameter length from 1024 to 2048 characters
- 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 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 with SSLPeerNamePtr field.
- Tested with MQ v7.1

➢ MQ Authenticate User Security Exit v1.5.0

- Added UseSSLCertUserID IniFile keyword to enable the use of the UserID from the channel's SSLCertUserID field
- 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 LicenseFile to support multiple license keys in a single file
- Fixed a bug with Proxy file processing

➢ MQ Authenticate User Security Exit v1.4.0

- Major performance and tuning to many modules - a 7% - 12% improvement in speed depending on features used
- Added ***encsrvr*** - it is used to create an encrypted server-side FBA file. encsrvr is similar/combination to the Unix programs: useradd, userdel and passwd including Unix crypt.
- 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 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.

➢ MQ Authenticate User Security Exit v1.3.0

- Created new CHAD (Channel Auto-Definition) exit so that z/MQAUSX can work with cluster channels.
- 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
- Successfully tested with MQ v7.0
- Fixed a bug related to uppercasing a very long UserId.

- MQ Authenticate User Security Exit for z/OS v1.2.3
  - Initial release.

14 Appendix H – 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.

15 Appendix I – Notices

Trademarks:

AIX, IBM, MQSeries, OS/2 Warp, OS/400, iSeries, MVS, OS/390, REXX, ISPF, TSO, 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.