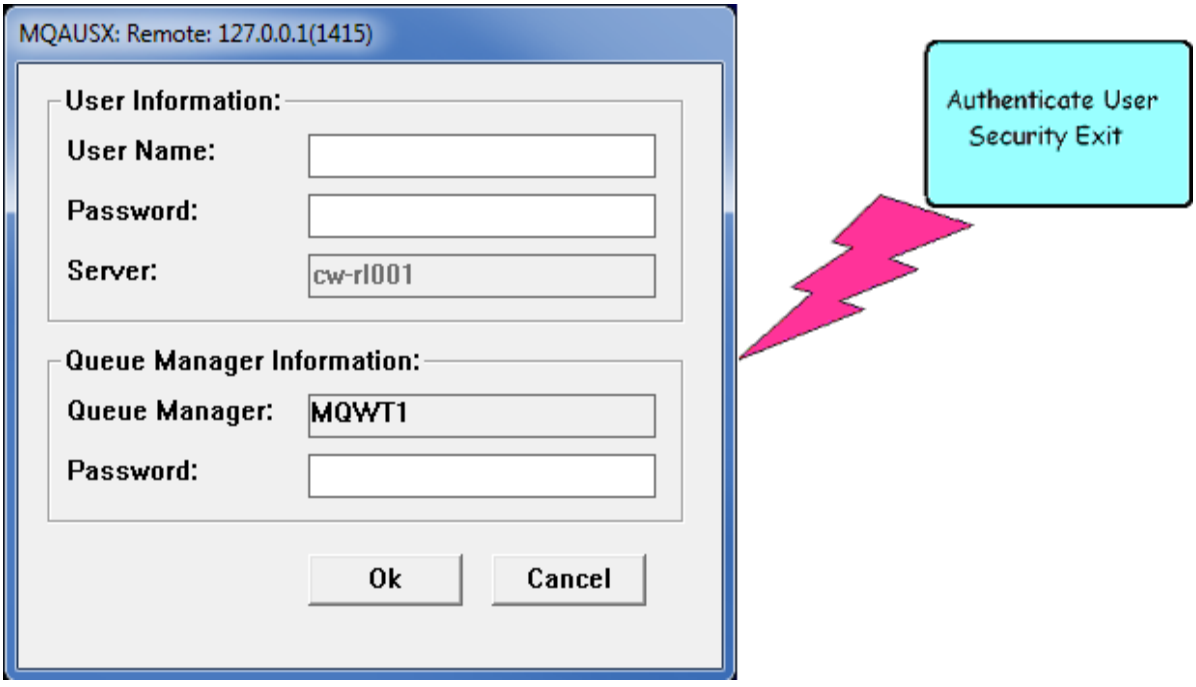
MQAUSX for z/OS Queue Manager To Queue Manager Configuration Manual



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.

2 Queue Manager To Queue Manager Overview

This section provides an overview of how z/MQAUSX can authenticate the UserId and Password of the connection request from one queue manager to any queue manager.

As mentioned in Chapter 1, z/MQAUSX is comprised of 2 MQ security exits: client-side security exit and server-side security exit.

2.1 Sender and Receiver Channel Pair

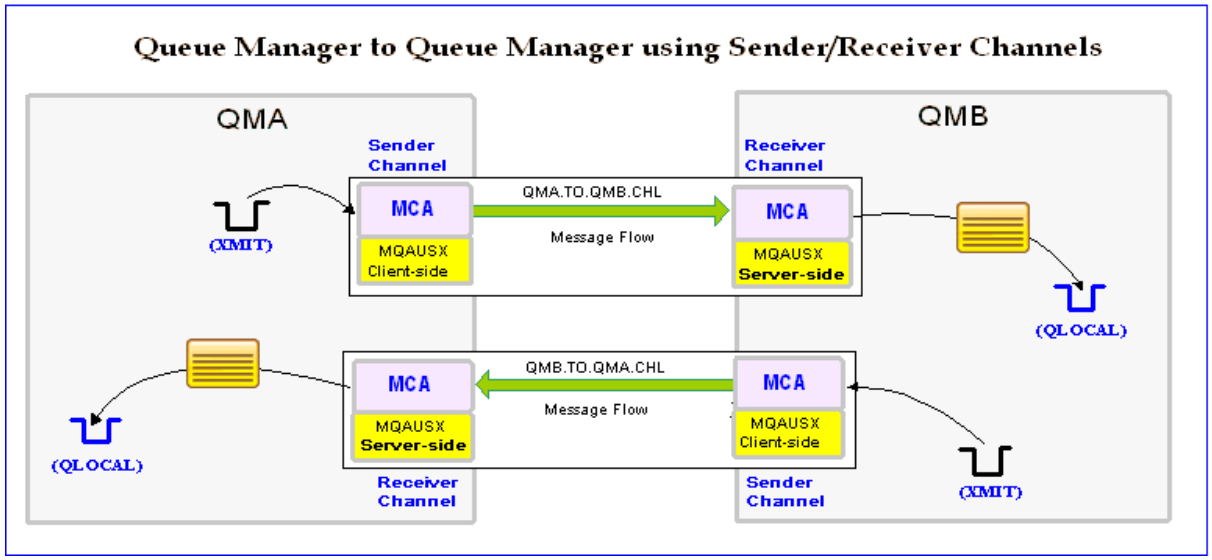
As noted below (in yellow) in the diagram, the z/MQAUSX client-side security exit works with the Sender (SDR) channel and the z/MQAUSX server-side security exit works with the Receiver (RCVR) channel.

There is a Message Channel Agent (MCA) at each end of the channel. The MCA is a component that handles the sending and receiving of messages between queue managers. Before the MCA can send and receive messages, the UserId and Password must be authenticated as detailed below:

- The MCA that is running the Sender channel will call z/MQAUSX client-side security exit to send a security message that contains the UserId and encrypted Password across the channel to the Receiver channel.
- The MCA that is running the Receiver channel will call z/MQAUSX server-side security exit to authenticate the incoming UserId and encrypted Password.

After the UserId and Password has been successfully authenticated, the channel will go to a 'Running' state and the messages will flow along the channel.

The following diagram highlights security exits in an MQ environment:



2.2 Server and Requester Channel Pair

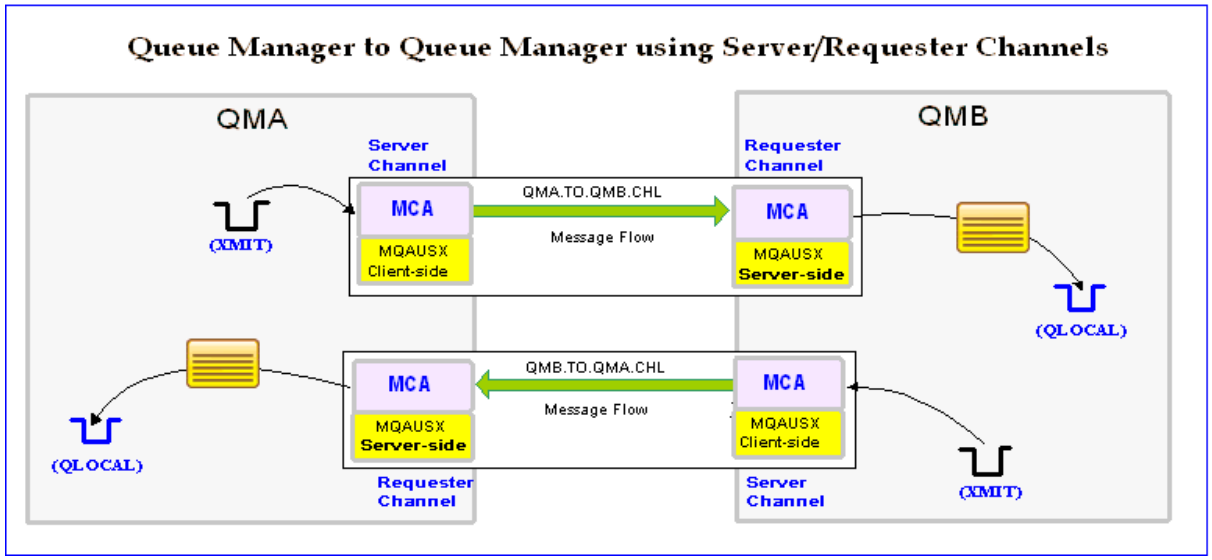
As noted below (in yellow) in the diagram, the z/MQAUSX client-side security exit works with the Server (SVR) channel and the z/MQAUSX server-side security exit works with the Requester (RQSTR) channel.

There is a Message Channel Agent (MCA) at each end of the channel. The MCA is a component that handles the sending and receiving of messages between queue managers. Before the MCA can send and receive messages, the UserId and Password must be authenticated as detailed below:

- The MCA that is running the Server channel will call z/MQAUSX client-side security exit to send a security message that contains the UserId and encrypted Password across the channel to the Requester channel.
- The MCA that is running the Requester channel will call z/MQAUSX server-side security exit to authenticate the incoming UserId and encrypted Password.

After the UserId and Password has been successfully authenticated, the channel will go to a 'Running' state and the messages will flow along the channel.

The following diagram highlights security exits in an MQ environment:



3 Configuring a Sender Channel

This section describes the necessary entries to enable the client-side security exit on a Sender Channel. The client-side security exit and its data will be applied to 2 fields of the Sender Channel. The MQ Administrator will need to update these 2 fields of the Sender Channel.

For more information on client-side IniFile parameters, please review *Appendix A* and for more information on client-side encrypted file, review *Appendix B* of the ***MQAUSX Client-side Configuration*** manual.

3.1 z/OS

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

- SCYEXIT
  MQAUSXCL

- SCYDATA - There are 3 ways to specify the UserId and Password:

  1. By explicitly setting them in the SCYDATA
     u=fred;p=abcdef

  2. Use a plain text IniFile by setting them in a dataset and specifying a DD name
     CLNTINI

  3. Use an encrypted client-side by setting them in a dataset and specifying a DD name
     CLNTENC

The following is an example of an MQSC command using a DDName for SCYDATA:

```
DEFINE CHANNEL ('QMA.TO.QMB.CHL') CHLTYPE(SDR) +
       TRPTYPE(TCP) +
       CONNAME(127.0.0.1(1415) +
       XMITQ(QMB.XMIT) +
       SCYEXIT('MQAUSXCL') +
       SCYDATA('CLNTENC') +
       REPLACE
```

Note: To use an encrypted client-side file, the 'DD' must end with 'ENC', otherwise the MQAUSX client-side component will assume the 'DD' points to a plain text IniFile.

4 Configuring a Receiver Channel

This section describes the necessary entries to enable the server-side security exit on a Receiver Channel. The server-side security exit and its data will be applied to 2 fields of the Receiver Channel. The MQ Administrator will need to update these 2 fields of the Receiver Channel.

For more information on server-side IniFile parameters, please review *Appendix A* of the ***MQAUSX for z/OS Server-side Installation and Operation*** manual.

4.1 z/OS

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

- SCYEXIT
  MQAUSX
- SCYDATA
  MQAUSXIN

The following is an example of an MQSC command for creating a Receiver Channel with the server-side security exit and its data:

```
DEFINE CHANNEL ('QMA.TO.QMB.CHL') CHLTYPE(RCVR) +
       TRPTYPE(TCP) +
       SCYEXIT('MQAUSX') +
       SCYDATA('MQAUSXIN') +
       REPLACE
```

5 Configuring a Server Channel

This section describes the necessary entries to enable the client-side security exit on a Server Channel. The client-side security exit and its data will be applied to 2 fields of the Server Channel. The MQ Administrator will need to update these 2 fields of the Server Channel.

For more information on client-side IniFile parameters, please review *Appendix A* and for more information on client-side encrypted file, review *Appendix B* of the ***MQAUSX Client-side Configuration*** manual.

5.1 z/OS

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

- SCYEXIT
  MQAUSXCL

- SCYDATA - There are 2 ways to specify the UserId and Password:

  1. By explicitly setting them in the SCYDATA
     u=fred;p=abcdef

  2. Use a plain text IniFile by setting them in a dataset and specifying a DD name
     CLNTINI

  3. Use an encrypted client-side by setting them in a dataset and specifying a DD name
     CLNTENC

The following is an example of an MQSC command using a DDName for SCYDATA:

```
DEFINE CHANNEL ('QMA.TO.QMB.CHL') CHLTYPE(SVR) +
       TRPTYPE(TCP) +
       CONNAME(127.0.0.1(1415) +
       XMITQ(QMB.XMIT) +
       SCYEXIT('MQAUSXCL') +
       SCYDATA('CLNTENC') +
       REPLACE
```

Note: To use an encrypted client-side file, the 'DD' must end with 'ENC', otherwise the MQAUSX client-side component will assume the 'DD' points to a plain text IniFile.

6 Configuring a Requester Channel

This section describes the necessary entries to enable the server-side security exit on a Requester Channel. The server-side security exit and its data will be applied to 2 fields of the Requester Channel. The MQ Administrator will need to update these 2 fields of the Requester Channel.

For more information on server-side IniFile parameters, please review *Appendix A* of the ***MQAUSX for z/OS Server-side Installation and Operation*** manual.

6.1 z/OS

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

- SCYEXIT
  MQAUSX
- SCYDATA
  MQAUSXIN

The following is an example of an MQSC command for creating a Receiver Channel with the server-side security exit and its data:

```
DEFINE CHANNEL ('QMA.TO.QMB.CHL') CHLTYPE(RQSTR) +
       TRPTYPE(TCP) +
       SCYEXIT('z/MQAUSX') +
       SCYDATA('z/MQAUSXIN') +
       REPLACE
```

7 Appendix A– 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*

8 Appendix B – 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 user, or group of users for Muliple User Licenses and Site Licenses. This Agreement requires that each user of the SOFTWARE be Licensed, either individually, or as part of a group. A Multi-User License provides for a specified number of users to use this SOFTWARE at any time. This does not provide for concurrent user Licensing. Each user of this SOFTWARE must be covered either individually, or as part of a group Multi-User 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 users only.

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 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.

9 Appendix C – Notices

Trademarks:

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

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

Intel is a registered trademark of Intel Corporation.

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

Linux is a trademark of Linus Torvalds.

Mac OS X is a trademark of Apple Computer Inc.

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

UNIX is a registered trademark of the Open Group.

WebLogic is a trademark of BEA Systems Inc.