Configuring an IBM MQ queue manager to use a dedicated Listener, Channel and Queue in Linux

https://www.ibm.com/support/pages/node/1135522

Date last updated: 13-May-2024

Angel Rivera
IBM MQ Support
https://www.ibm.com/products/mq/support
Find all the support you need for IBM MQ

+++ Objective

The purpose of this technical document is to show all the steps to configure an MQ 9.3 queue manager in host-1 (Linux) and create objects such as a Listener, a Server-Connection Channel with its Channel Authentication Records, a Queue, etc.
The steps in this tutorial can be applied also to a queue manager in Windows

The goal is for the user “fulano” (group “mquser”) in host-2 (Linux) to be able to put and get messages by using these dedicated objects and prevent other users from using them.
Host-1 name: stmichel1.fyre.ibm.com
Host-2 name: chamonix1.fyre.ibm.com

The queue manager will have a “normal” listener:
- Queue Manager name: QM93LNX
- Listener (such as LISTENER in the default port 1414)
- Channel Authentication Records (CHLAUTH) to allow the remote connection

The dedicated objects are:
- (Optional) Listener (such as MY.LISTENER in port 1420)
- Server-Connection Channel (such as MY.CHANNEL)
- Channel Authentication Record (CHLAUTH) for this server-connection channel that allows only the user “fulano” who is a member of the group “mquser”.
- Queue in Linux (such as MY.Q)
- Authority records for group “mquser” to display, put, get, browse, etc. for queue MY.Q.

The chapters are:
Chapter 1: User “root” adds the proper user and group in both host-1 and host-2.
Chapter 2: MQ administrator (user “mqm”) creates a queue manager with the basic/normal objects in host-1
Chapter 3: MQ administrator adds the dedicated objects.
Chapter 4: User “fulano” from host-2 puts and gets messages using the dedicated objects.
Chapter 5: User “fulano” from host-2 uses runmqsc to look at CURDEPTH on MY.Q
Chapter 6: User “bob” from host-2 fails to put/get messages using the dedicated objects.

++ Useful technotes

You can use the following tutorial to install the product:

https://www.ibm.com/support/pages/node/6988681
Installing MQ 9.3, applying Fix Pack 9.3.0.5, uninstalling in Linux RHEL

You can create an .mqsc file with the definitions and alterations of objects and then specify this file to be used during the creation of the queue manager.

https://www.ibm.com/support/pages/node/7028088
Example of using IBM MQ automatic configuration of MQSC and qm.ini attributes

https://www.ibm.com/support/pages/node/1288090
Example of using a back stop rule together with an MQ channel authentication rule that maps client user 'fred' to queue manager user 'fulano'

++ What is the default security behavior for remote connections?

By default, the queue manager has enabled several security features related to remote connections.

a) Feature: Channel Authentication Records

- NORMAL users …
  - are allowed to use non-SYSTEM channels (Of course, these users will need an explicit "connect" authority for the queue manager).
  - are NOT allowed to use SYSTEM channels:
    - the exception is: Normal users CAN use SYSTEM.ADMIN.SVRCONN

- The MQ Administrators CANNOT use ANY channel: SYSTEM and non-SYSTEM

b) Feature: Connection Authentication

By default, the queue manager will use the LOCAL users and groups (that is, no LDAP). This is specified via the CONNAUTH attribute: connection authentication

```
runmqsc QMGRNAME
display qmgr connauth
AMQ8408I: Display Queue Manager details.
   QMNAME(QM92TEST)
   CONNAUTH(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)
.
```

2) The default CONNAUTH is as follows:

```
- AUTHTYPE(IDPWOS) => Local users and groups
- CHCKCLNT(REQDADM) => MQ Administrators MUST provide a valid password
                  => Normal users are not required to provide a password,
                     but if they provide a password, it must be a valid one
```

```
display authinfo(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)
AMQ8566I: Display authentication information details.
   AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)
   AUTHTYPE(IDPWOS)                     ADOPTCTX(YES)
   DESCR( )                          CHCKCLNT(REQDADM)
   CHCKLOCL(OPTIONAL)                   FAILDLAY(1)
   AUTHENMD(OS)
.
```

3) Many customers like to make this security setting stronger, by changing
   CHCKCLNT

   ... in the AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) to the value of:
   CHCKCLNT(REQUIRED)

   which means that a valid password is required for ALL:
   => MQ Administrators MUST provide a valid password
   => Normal users MUST provide a valid password

.
For more details see:
https://www.ibm.com/docs/en/ibm-mq/9.3?topic=reference-define-authinfo-define-authentication-information-object
DEFINE AUTHINFO (define an authentication information object)
.
+ begin excerpt
.
REQUIRED
- All client applications must provide authentication credentials in the MQCSP structure.
- If an application provides a user ID and password, these credentials are authenticated by the queue manager against the password store indicated by the AUTHTYPE. The connection is only allowed to continue if the user ID and password are valid.

+ end excerpt
.
If your team wants to change the value of CHCKCLNT from REQDADM to REQUIRED, followed by either a Refresh Security or a restart of the queue manager.

Here are the 2 commands:

```
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) CHCKCLNT(REQUIRED)

REFRESH SECURITY TYPE(CONNAUTH)
```

+++ Chapter 1: User “root” adds the proper user and group in both hosts

The default authentication mechanism for the MQ queue manager is based on the local userid and local password.

When the MQ Client application connects from a remote host (such as host-2: chamonix1), the userid that started the MQ client application is passed to the queue manager.
In this case, the userid that will be passed is: fulano

The queue manager will check that there is a local user “fulano” in the same box in which the queue manager is running (such as host-1: stmichel1).
Then based on the group membership for this user the queue manager will determine if the user is authorized to connect, to open a queue and to put messages into a queue.

For the above reasons, it is necessary to create the user “fulano” in both hosts.
In both hosts the userid number might be different and the group membership might be different too. **But the same name “fulano” must be the same in both hosts.**

It is a best practice to use the same userid number and the same group number in both hosts, and therefore, the same actual commands which include a specific userid number and group number will be used in both hosts.

For each host do:

Login as user: root

Create group “mquser”:
```
  groupadd -g 1005 mquser
```

Create user “fulano”:
```
 useradd -u 1021 -g mquser -s /bin/bash -p passw0rd -d /home/fulano -m fulano
```

Confirm the creation of the user:
```
 id fulano
```
The result is:
```
    uid=1021(fulano) gid=1005(mquser) groups=1005(mquser)
```

Create user “bob”:
```
 useradd -u 1009 -g mquser -s /bin/bash -p passw0rd -d /home/bob -m bob
```

<u>+ Note to allow remote Windows user “Administrator” for TEST systems</u>

To facilitate the remote usage of MQ Explorer from a Windows system that is running under the default user “Administrator”, you could perform the following customization.

The Windows userid “Administrator” has 13 characters and it is longer than the normal maximum of 12 characters for MQ.

In the Linux server, you could create a userid “administrato” which is in lowercase and has 12 characters.
This userid could belong to the group “mqm” and thus, it could be an MQ administrator.

As user root:

```
**ROOT** stmichel1.fyre.ibm.com: /root
# useradd -u 603 -g mqm -s /bin/bash -d /home/administrato -m administrato

**ROOT** stmichel1.fyre.ibm.com: /root
# id administrato
uid=603(administrato) gid=500(mqm) groups=500(mqm)
```

When the Windows userid: Administrator
… tries to connect to the queue manager in Linux, the userid is processed as follows:
- it is mapped to ALL lowercase: administrator
- it is truncated to 12 characters: administrato
The end result is that the userid:
                                                            administrato
… is what the queue manager finally sees.

Then the queue manager will ask the OS the question:
  does the user "adminitrato" exist in this host?

+ Note to allow remote Windows user has only numbers

Some organizations have the bad practice of creating users in Windows that only have numbers, such as:
	12345678

The problem is that in Linux, it is NOT possible to create a user name with only numbers.

One approach is for the user root in Linux to create a use name that begins with the letter "u" followed by the number, such as:
	u12345678

Then later on create a channel authentication record that maps the remote Windows user "12345678" to the local Linux user "u12345678", such as:

```
DEFINE CHANNEL('MY.CHANNEL') CHLTYPE(SVRCONN)
SET CHLAUTH('MY.CHANNEL') TYPE(BLOCKUSER) USERLIST('nobody')
SET CHLAUTH('MY.CHANNEL') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS)
SET CHLAUTH('MY.CHANNEL') TYPE(USERMAP) CLNTUSER('594079897') USERSRC(MAP)
        MCAUSER('u594079897') ADDRESS('*') ACTION(ADD)
```

+++ Chapter 2: MQ administrator (user "mqm") creates a queue manager with the basic/normal objects in host-1

Host-1 name: stmichel1.fyre.ibm.com
"Normal" listener: LISTENER (using port 1414)

Login into host-1 as an MQ administrator (user: mqm)

Specify the MQ environment:

```
   .  /opt/mqm/bin/setmqenv -n Installation1
```

Create a queue manager named QM93 and to specify to use a Dead Letter Queue named "DLQ".
Caveat: You must define the DLQ. The above parameter does NOT actually create the queue, it just tells the queue manager which is the name of the dead letter queue.

```
     crtmqm -u DLQ QM93LNX
```

Instead of creating the queue DLQ, you use the following existing SYSTEM queue:

```
     crtmqm -u SYSTEM.DEAD.LETTER.QUEUE QM93LNX
```

You could specify the attribute "-p 1414" to tell crtmqm to create a Listener object with port 1414.

```
     crtmqm -u DLQ -p 1414 QM93LNX
```

… then the following Listener will be created:

```
     LISTENER(SYSTEM.LISTENER.TCP.1)          CONTROL(QMGR)
     TRPTYPE(TCP)                             PORT(1414)
```

Start the queue manager

```
strmqm QM93LNX
```

Start the administration tool and define typical objects for a TEST queue manager (such as allowing remote access by the MQ Administrator and not specifying a password).

```
runmqsc QM93LNX
```

## If you did not use the -p during crtmqm, you need to define a listener.
The default is port 1414.
## If you used -p, then there is no need to define a 2nd listener. One listener is enough.
define listener(LISTENER) trptype(tcp) control(qmgr) port(1414)
start listener(LISTENER)

## Define a channel to be used by a remote MQ Explorer
define channel(SYSTEM.ADMIN.SVRCONN) chltype(SVRCONN)

## Define the DLQ
define qlocal(DLQ) like(SYSTEM.DEAD.LETTER.QUEUE)

```
  ## Define test queue:
    define qlocal(Q1)
```

## Security note: the following ALTER is not suitable for PRODUCTION but might be suitable for TEST queue managers.

```
## For testing queue managers: disable userid/password for remote users
    ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)  +
      AUTHTYPE(IDPWOS) CHCKCLNT(OPTIONAL)
    REFRESH SECURITY TYPE(CONNAUTH)
```

## Security note: the following CHLAUTH records may not be suitable for PRODUCTION but might be suitable for TEST queue managers.

```
## For testing queue managers: allow MQ Administrator to remotely access the queue manager
  SET CHLAUTH(*) TYPE(BLOCKUSER) USERLIST('nobody','*MQADMIN')
  SET CHLAUTH(SYSTEM.ADMIN.*) TYPE(BLOCKUSER) USERLIST('nobody')
  SET CHLAUTH(SYSTEM.DEF.*) TYPE(BLOCKUSER) USERLIST('nobody')
  SET CHLAUTH(SYSTEM.DEF.SVRCONN) TYPE(ADDRESSMAP) ADDRESS(*) +
     USERSRC(CHANNEL)

## Exit runmqsc
  end
```

+ Some other optional commands for Test systems:

The following runmqsc commands might be suitable for Test/Development queue managers, but NOT for Production

```
** Do not ask for password for remote access
ALTER AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) +
    CHCKCLNT(OPTIONAL)
REFRESH SECURITY TYPE(CONNAUTH)

** Simplify the demo system by disabling channel and connection authentication
ALTER QMGR CHLAUTH(DISABLED) CONNAUTH(' ')
REFRESH SECURITY TYPE(CONNAUTH)
```

+++ Chapter 3: MQ administrator adds the dedicated objects.

There are 2 sections:
- the first one for the objects and
- the second one for the authorities for the user.

+ Section 1: Add objects

Login into host-1 as an MQ administrator (user: mqm)

The dedicated objects are:
- Listener (such as MY.LISTENER in port 1420)
- Server-Connection Channel (such as MY.CHANNEL)
- Queue in Linux (such as MY.Q)
- Channel Authentication Record (CHLAUTH) for this server-connection channel that allows only the user “fulano” who is a member of the group “mquser”.
- Authority records for group “mquser” to display, put, get, browse, etc. from the dedicated queue.

Start the administration tool:

```
runmqsc QM93LNX
```

```
  ## (optional) Define another listener, using port 1420.
    define listener(MY.LISTENER) trptype(tcp) control(qmgr) port(1420)
    start listener(MY.LISTENER)

  ## Define queue:
    define qlocal(MY.Q)

  ## Define a channel to be used by a remote MQ Explorer
    define channel(MY.CHANNEL) chltype(SVRCONN)

   ## At this point, the channel is open to anyone! Everything gets thru!
  ## 1st rule - back stop - blocks all users (it is like a wall)
     SET CHLAUTH('MY.CHANNEL') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS)

  ## At this point, the channel is NOT usable at all! Nothing gets thru!
  ## 2nd rule - allows one userid with password to connect (it is like a hole in the wall)
     SET CHLAUTH('MY.CHANNEL') TYPE(USERMAP) CLNTUSER('fulano')
USERSRC(CHANNEL) CHCKCLNT(REQUIRED) ACTION(ADD)
```

The following runmqsc commands might be suitable for Test/Development queue managers, but NOT for Production

```
## Allow the channel to be used by an MQ administrator
   SET CHLAUTH('MY.CHANNEL') TYPE(BLOCKUSER) USERLIST('nobody')

 end
```

+ Section 2: authorities for the group/user

The default authority for objects in the queue manager is based on the authority of the group membership for the user (based on the LOCAL configuration in the host where the queue manager).

The desired user “fulano” has the following group membership:

```
  id fulano
     uid=1021(fulano) gid=1005(mquser) groups=1005(mquser)
```

Notice that the primary group is “mquser”, thus, this is the group which needs to be specified in the setmqaut when using the -g flag.

The purpose of the following setmqaut commands is:
1. GENERAL: Grant authority to access the queue manager.
2. MQ EXPLORER: Grant authority to the client channel to get the command server reply messages.
3. MQ EXPLORER: Grant authority to put messages onto the command server input queue.
4. MQ EXPLORER: Grant authority to get the reply messages.
Issue these setmqaut commands to grant minimal authority to the Unix group (using the GroupName flag: -g)
5. For using runmqsc: Grant authority to use queue SYSTEM.MQSC.REPLY.QUEUE +inq +browse +get +dsp

```
  setmqaut -m QM93LNX -t qmgr -g mquser +connect +inq +dsp

  setmqaut -m QM93LNX -t q -n SYSTEM.DEFAULT.MODEL.QUEUE -g mquser +inq +browse
+get +dsp

  setmqaut -m QM93LNX -t q -n SYSTEM.ADMIN.COMMAND.QUEUE -g mquser +inq +put
+dsp

  setmqaut -m QM93LNX -t q -n SYSTEM.MQEXPLORER.REPLY.MODEL -g mquser +inq
+browse +get +dsp +put

  setmqaut -m QM93LNX -t q -n SYSTEM.MQSC.REPLY.QUEUE  -g mquser +inq +browse
+get +dsp
```

The following command gives additional put/get authority for queue MY.Q.

```
  setmqaut -m QM93LNX -t q -n MY.Q -g mquser +inq +browse +get +put +dsp
```

+++ Chapter 4: User “fulano” from host-2 puts and gets messages using the dedicated objects.

Host-2 name: chamonix1.fyre.ibm.com

Login into host-2 as user: fulano

Specify the MQ environment (you can add it into your .bashrc file)

```
  . /opt/mqm/bin/setmqenv -n Installation1
```

Unfortunately the setmqenv command does not add the directories for the samples. Thus, it is recommended that you add these directories.
You can add them into your .bashrc file

```
# Additional MQ directories for the PATH
export
PATH=$PATH:$MQ_INSTALLATION_PATH/java/bin:$MQ_INSTALLATION_PATH/samp/bin:$M
Q_INSTALLATION_PATH/samp/jms/samples:
# Add local directory for running Java/JMS programs
export CLASSPATH=$CLASSPATH:.
```

Export the MQSERVER environment variable to use the server-connection channel MY.CHANNEL, using port 1414 (or for the other listener, 1420) on the host of the queue manager. Notice that the actual name of the queue manager is not specified.
In Unix, you need to enclose the values within single quotes, due to the parenthesis around the port number.

```
export MQSERVER='MY.CHANNEL/TCP/stmichel1.fyre.ibm.com(1414)'
```

It is necessary to provide the password.
For more information see the following article:
http://www.ibm.com/support/docview.wss?uid=swg21680930
MQ 8.0: errors AMQ5540, AMQ5541 and AMQ5542, application did not supply a user ID and password, 2035 MQRC_NOT_AUTHORIZED

Section A of the about article indicates that for some of the MQ samples, it is necessary to setup an environment variable:

```
export MQSAMP_USER_ID=fulano
```

Issue the MQ sample to put messages, when connecting in network/client mode.
Because of the environment variable MQSAMP_USER_ID, you will be prompted for the password of use “fulano” as it is defined in host-1 (the host of the queue manager)

```
fulano@chamonix1.fyre.ibm.com: /home/fulano
$ amqsputc MY.Q QM93LNX
Sample AMQSPUT0 start
Enter password: *********
target queue is MY.Q
THIS IS A TEST MESSAGE

Sample AMQSPUT0 end
```

+ In host-1, confirm that message was received

To confirm that the message was received, you can login in host-1 as the user mqm and then issue the sample to browse the queue. Notice these values:

```
   UserIdentifier : 'fulano      '
  …
   PutApplName    : 'amqsputc                    '
  …
 00000000:  5448 4953 2049 5320 4120 5445 5354 204D           'THIS IS A TEST M'
 00000010:  4553 5341 4745                                    'ESSAGE          '
```

```
mqm@stmichel1.fyre.ibm.com: /home/mqm
$ amqsbcg MY.Q QM93LNX
AMQSBCG0 - starts here
**********************
 MQOPEN - 'MY.Q'
 MQGET of message number 1, CompCode:0 Reason:0
****Message descriptor****
  StrucId  : 'MD  '  Version : 2
   Report   : 0  MsgType : 8
   Expiry   : -1  Feedback : 0
   Encoding : 546  CodedCharSetId : 1208
   Format : 'MQSTR   '
   Priority : 0  Persistence : 0
   MsgId : X'414D5120514D39312020202020202020206198F25D02A37F20'
   CorrelId : X'000000000000000000000000000000000000000000000000'
   BackoutCount : 0
   ReplyToQ       : '                                                '
   ReplyToQMgr    : 'QM93LNX                                         '
   ** Identity Context
   UserIdentifier : 'fulano      '
   AccountingToken :
    X'0431303231000000000000000000000000000000000000000000000000000006'
   ApplIdentityData : '                                '
   ** Origin Context
   PutApplType    : '6'
   PutApplName    : 'amqsputc                    '
   PutDate  : '20191212'    PutTime  : '20140230'
```

```
ApplOriginData : '    '

GroupId : X'000000000000000000000000000000000000000000000000'
MsgSeqNumber   : '1'
Offset         : '0'
MsgFlags       : '0'
OriginalLength : '-1'


****   Message      ****
 length - 22 of 22 bytes
00000000:  5448 4953 2049 5320 4120 5445 5354 204D           'THIS IS A TEST M'
00000010:  4553 5341 4745                                    'ESSAGE          '

No more messages
MQCLOSE
MQDISC
```

+++ Chapter 5: User “fulano” from host-2 uses runmqsc to look at CURDEPTH on MY.Q

If you would like for user “fulano” from host-2 to use the runmqsc tool to monitor the current depth (CURDEPTH) of the queue MY.Q, then you could take a look at the following tutorial.

```
https://www.ibm.com/support/pages/node/616403
New features added to the runmqsc command in IBM MQ Version 8.0
```

Specifically:
Chapter 4: Flag -u to allow authentication
Chapter 5: Flag -c to allow remote access through a client connection

In host-1, as user mqm, issue the following, which is needed when using runmqsc by user who is not a member of the group "mqm":

```
setmqaut -m QM93LNX -t q -n SYSTEM.MQSC.REPLY.QUEUE  -g mquser +inq +browse +get +dsp
```

This example assumes that you have setup already:

```
export MQSERVER='MY.CHANNEL/TCP/stmichel1.fyre.ibm.com(1414)'
export MQSAMP_USER_ID=fulano
```

The following interaction uses 2 “display” commands.
The first one is successful because the user “fulano” has proper authorization to inquire the queue MY.Q.
But the second one fails, because the user does not have the authorization for queue Q1.

```
fulano@chamonix1.fyre.ibm.com: /home/fulano
$ runmqsc -c -u fulano QM93LNX
5724-H72 (C) Copyright IBM Corp. 1994, 2022.
Enter password:
**********
Starting MQSC for queue manager QM93LNX.

display QLOCAL(MY.Q) CURDEPTH
     1 : display QLOCAL(MY.Q) CURDEPTH
AMQ8409I: Display Queue details.
   QUEUE(MY.Q)                             TYPE(QLOCAL)
   CURDEPTH(1)

display QLOCAL(Q1) CURDEPTH
     2 : display QLOCAL(Q1) CURDEPTH
AMQ8135E: Not authorized.
```

+++ Chapter 6: User “bob” from host-2 fails to put/get messages using the dedicated objects.

Host-2 name: chamonix1.fyre.ibm.com

Login into host-2 as user: bob

Specify the MQ environment (you can add it into your .bashrc file)

```
. /opt/mqm/bin/setmqenv -n Installation1
```

Export the MQSERVER environment variable to use the server-connection channel MY.CHANNEL, using port 1414 on the host of the queue manager. Notice that the actual name of the queue manager is not specified.
In Unix, you need to enclose the values within single quotes, due to the parenthesis around the port number.

```
export MQSERVER='MY.CHANNEL/TCP/stmichel1.fyre.ibm.com(1414)'
```

For the MQ Samples, setup the user id:

```
export MQSAMP_USER_ID=bob
```

Issue the MQ sample to put messages, when connecting in network/client mode.
Because of the environment variable MQSAMP_USER_ID, you will be prompted for the password of use “bob” as it is defined in host-1 (the host of the queue manager)

```
bob@chamonix1.fyre.ibm.com: /home/fulano
$ amqsputc MY.Q QM93LNX
Sample AMQSPUT0 start
Enter password: **********
MQCONNX ended with reason code 2035
```

In host-1, as an MQ administrator take a look at the bottom of the error log for the queue manager:

```
mqm@stmichel1.fyre.ibm.com: /var/mqm/qmgrs
$ cd /var/mqm/qmgrs/QM93LNX/errors
mqm@stmichel1.fyre.ibm.com: /var/mqm/qmgrs/QM93LNX/errors
$ vi AMQERR01.LOG

01/16/2024 07:05:49 AM - Process(36275.89) User(mqm) Program(amqrmppa)
                    Host(stmichel1.fyre.ibm.com) Installation(Installation1)
                    VRMF(9.3.0.15) QMgr(QM93LNX)
                    Time(2024-01-16T15:05:49.193Z)
                    RemoteHost(9.46.110.220)
                    CommentInsert1(MY.CHANNEL)
                    CommentInsert2(chamonix1 (9.46.110.220))
                    CommentInsert3(CLNTUSER(bob) ADDRESS(chamonix1))

AMQ9777E: Channel was blocked

EXPLANATION:
The inbound channel 'MY.CHANNEL' was blocked from address 'chamonix1
(9.46.110.220)' because the active values of the channel matched a record
configured with USERSRC(NOACCESS). The active values of the channel were
'CLNTUSER(bob) ADDRESS(chamonix1)'.
```

Notice that the channel MY.CHANNEL was defined specifically to reject all users …

```
## At this point, the channel is open to anyone! Everything gets thru!
  ## 1st rule - back stop - blocks all users (it is like a wall)
    SET CHLAUTH(MY.CHANNEL) TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(NOACCESS)
```

… But allow user ‘fulano’ to pass, but to reject other users (back-stop rule)

```
  ## At this point, the channel is NOT usable at all! Nothing gets thru!
  ## 2nd rule - allows one userid with password to connect (it is like a hole in the wall)
    SET CHLAUTH('MY.CHANNEL') TYPE(USERMAP) CLNTUSER('fulano')
USERSRC(CHANNEL) CHCKCLNT(REQUIRED) ACTION(ADD)
```

+++ end