The interaction of CHLAUTH and CONNAUTH in IBM MQ

Mark\_P\_Wilson |Dec 4 2016 Updated

This article explains the interaction between channel authentication records (CHLAUTH) and connection authentication (CONNAUTH) in IBM MQ. When an application connects to MQ using client bindings, network connections have to be opened up, which can have security implications. In the last few MQ releases, these have been addressed with the introduction of the CHLAUTH and CONNAUTH features. This article will explain how they fit together when a receiving end of a channel is started.

**Different types of bindings**

IBM MQ supports two ways that an application can connect:

1. **Local bindings**: This is when the application and queue manager are on the same operating image. CHLAUTH is not relevant to this type of application connection.
2. **Client bindings**: This is when the application and queue manager use the network to communicate. The application and queue manager may be running on the same machine or they may be on different machines. In MQ a client connection is handled in the form of a server-connection (SVRCONN) channel. Both CONNAUTH and CHLAUTH are applicable, and it is this type of connection which is discussed.

**The binding steps of a receiving end of a channel**

When an application connects to a queue manager there is a lot of checking to perform to ensure that both ends of the channel understand what is supported by the other end. The receiving end does some extra checking to ensure that the client is allowed to connect. This checking involves CHLAUTH and CONNAUTH. This process may also include a security exit as this can affect the result. This channel connecting phase is also referred to as the binding phase.

In MQ version 7 SHARECNV was added to SVRCONN channels so multiple connections/conversations can share the same channel. This article does not cover what happens in terms of CHLAUTH and CONNAUTH when a second and following conversations share the same channel.

Figure 1 shows the steps that a SVRCONN channel goes through when the server end (at the queue manager) starts.



Figure 1: the steps that a SVRCONN channel goes through when starting up.

**Step 1: Receive a connection request** The channel initiator or listener receives a connection request from somewhere on the network.

**Step 2: Is address allowed to connect?** Before any data is read from the wire, MQ will check the partner's IP address against the CHLAUTH rules to see if the address is in the BLOCKADDR rule. If the address is not found and so not blocked the flow proceeds to the next step.

**Step 3: Read data from the channel** MQ can now read the data from the wire into a buffer and start to process the sent information.

**Step 4: Look up channel definition** In the first data flow MQ sends amongst other things the name of the channel which the sending end is trying to start. The receiving queue manager can then look up the channel definition, which has all the settings that are specified for the channel.

**Step 5: Pre CONNAUTH checks** when using CHLAUTH and CONNAUTH there is a situation where the user id of the channel rather than the asserted user id too be used in the CHLAUTH mapping. To give an example if CONNAUTH is using LDAP to map an email address that is provided by the application to a serial number you would properly want the result e.g. serial number to be used in the CHLAUTH mapping. By default this doesn't happen. An APAR has been done to add an ini parameter called ChlauthEarlyAdopt which performs an extra check before the CHLAUTH mapping. This APAR went into MQ 8.0.0.5, [IT12825: IBM MQV8: A CLIENT APPLICATION FAILS TO CONNECT TO A QUEUE MANAGER WITH ERROR AMQ9777: CHANNEL WAS BLOCKED](http://www-01.ibm.com/support/docview.wss?uid=swg1IT12825) details how to turn the option on. This was slightly altered at version 9.0 where the option is just Y which covers all the options described in the APAR.

**Step 6: CHLAUTH mapping** The CHLAUTH cache is inspected again to look for mapping rules (SSLPEERMAP, USERMAP, QMGRMAP and ADDRESSMAP). The rule that matches the incoming channel most specifically will be used. If the rule has USERSRC(CHANNEL) or USERSRC(MAP) the channel continues on binding. Rules that have USERSRC(NOACCESS) means that the channel would be blocked from connecting and the network connection is ended.

**Step 7: Call security exit** If the channel has a security exit (SCYEXIT) defined, this is called with the exit reason (MQCXP.ExitReason) set to MQXR\_SEC\_PARMS. If the client application has specified security credentials on a MQCONNX call (via MQCSP on the MQCNO) these will be passed in the exit parameters (a pointer to MQCSP will be present in the SecurityParms field of the MQCXP). The MQCSP structure has pointers to the user ID (MQCSP.CSPUserIdPtr) and password (MQCSP.CSPPasswordPtr). It is possible for the security exit to change these. Example 1 shows how a security exit would access the userId and password fields.

|  |
| --- |
| if (pMQCXP -> ExitReason == MQXR\_SEC\_PARMS){  /\* It is not a good idea for security reasons to print out the user ID \*/  /\* and password but the following is shown for demonstration reasons \*/  printf("User ID: %.\*s Password: %.\*s\n",         pMQCXP -> SecurityParms -> CSPUserIdLength,         pMQCXP -> SecurityParms -> CSPUserIdPtr,         pMQCXP -> SecurityParms -> CSPPasswordLength,         pMQCXP -> SecurityParms -> CSPPasswordPtr);}Example 1: accessing the user ID and password in an exit |

The exit could tell MQ to close the channel by returning MQXCC\_CLOSE\_CHANNEL or MQXCC\_FAILED in MQCXP.ExitResponse field. Otherwise, channel processing continues to the connection authentication phase.

**Step 8: Is user authenticated?** The authentication phase happens if CONNAUTH is enabled on the queue manager. To check this issue the MQSC command 'DISPLAY QMGR CONNAUTH'. Figures 2 and 3 show the output of this command on MQ for z/OS and distributed MQ.

|  |
| --- |
|  CSQM201I !MQ25 CSQMDRTC DISPLAY QMGR DETAILS QMNAME(MQ25) CONNAUTH(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)  END QMGR DETAILS CSQ9022I !MQ25 CSQMDRTC ' DISPLAY QMGR' NORMAL COMPLETIONFigure 2: output of the command DISPLAY QMGR CONNAUTH from a queue manager running on z/OS |

|  |
| --- |
|     1 : DISPLAY QMGR CONNAUTHAMQ8408: Display Queue Manager details.   QMNAME(DEMO)   CONNAUTH(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)Figure 3: output of the command DISPLAY QMGR CONNAUTH from a distributed MQ queue manager |

The CONNAUTH value is a name of an AUTHINFO MQ object. MQ version 8 supports two methods of authentication: using the operating system (AUTHTYPE(IDPWOS)) or using LDAP (not available on z/OS) (AUTHTYPE(IDPWLDAP)). This article concentrates on operating system authentication. In a future article. I will cover using LDAP for authentication. Figures 4 and 5 shows the shipped default object for AUTHINFO type(IDPWOS) in MQ for z/OS and distributed MQ.

|  |
| --- |
|  CSQM293I !MQ25 CSQMDRTC 1 AUTHINFO FOUND MATCHING REQUEST CRITERIA CSQM201I !MQ25 CSQMDRTC DISPLAY AUTHINFO DETAILS AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS) AUTHTYPE(IDPWOS) QSGDISP(QMGR) ADOPTCTX(NO) CHCKCLNT(NONE) CHCKLOCL(OPTIONAL) FAILDLAY(1) DESCR() ALTDATE(2015-06-04) ALTTIME(10.43.04)   END AUTHINFO DETAILS CSQ9022I !MQ25 CSQMDRTC ' DISPLAY AUTHINFO' NORMAL COMPLETIONFigure 4: displaying the shipped default object for AUTHINFO type(IDPWOS) from a queue manager running on z/OS |

|  |
| --- |
|     1 : display authinfo(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)AMQ8566: Display authentication information details.   AUTHINFO(SYSTEM.DEFAULT.AUTHINFO.IDPWOS)   AUTHTYPE(IDPWOS)                     ADOPTCTX(NO)   DESCR( )                             CHCKCLNT(REQDADM)   CHCKLOCL(OPTIONAL)                   FAILDLAY(1)   ALTDATE(2015-06-08)                  ALTTIME(16.35.16)Figure 5: displaying the shipped default object for AUTHINFO type(IDPWOS) from a distributed queue manager. |

The AUTHINFO TYPE(IDPWOS) has an attribute called CHCKCLNT. If the value is changed to REQUIRED all client applications would have to supply a valid user ID and password. While we are looking at the attributes involved with CONNAUTH, I must mention the adopt context (ADOPTCTX) option. The ADOPTCTX attribute controls whether the channel runs under MCAUSER or the user ID the application has supplied If ADOPTCTX is YES then the channel will adopt that user to run under (the active MCAUSER) and the object authorization will be done against this user. This will also be used in step 8 in the CONNAUTH check.

For example, say I don't have a MCAUSER set on the SVRCONN channel and my client is running under 'markw1' on my linux machine. My application specifies user 'fred' in the MQCSP the channel will start running with 'markw1' as the active MCAUSER. After the CONNAUTH check the user 'fred' will be adopted and the channel run with 'fred' as the active MCAUSER.

**Step 9: Is user allowed on this channel?** If the CONNAUTH checking is successful the CHLAUTH cache is then inspected again to check if the active MCAUSER is blocked by a BLOCKUSER rule. If the user is blocked the channel will be terminated.

**Steps 10 and 11: Object authorisation checks** The client application has now connected. As with locally bound applications any objects that the application opens e.g. a queue, requires a check will be made to ensure that the active MCAUSER has the appropriate authority for that object.

**Conclusion** In this article I have described what stages a channel goes through when connecting in terms of security checking. In the next article I will demonstrate how an application supplies user credentials.

**Resources**

The following links are provided to give more information the topics covered:

* [CHLAUTH](http://www-01.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.explorer.doc/e_chlauth.htm)
* [CONNAUTH](http://www-01.ibm.com/support/knowledgecenter/SSFKSJ_8.0.0/com.ibm.mq.sec.doc/q113250_.htm)
* [IT12825: IBM MQV8: A CLIENT APPLICATION FAILS TO CONNECT TO A QUEUE MANAGER WITH ERROR AMQ9777: CHANNEL WAS BLOCKED](http://www-01.ibm.com/support/docview.wss?uid=swg1IT12825)