MQ Message Channels

Overview

In a distributed environment WebSphereMQ utilizes Message Channels as the mechanism for transporting messages around the Queue Manager network.

WebSphereMQ's definition of a channel is nothing more than a 'pipeline' between queue managers. It is not a distinct physical connection; indeed, the Queue Manager is not concerned in the least about hardware, however, channel definitions must include the communication protocol being used by the channel (e.g., TCP/IP).

Message channels can be considered as logical connections between the nodes of the WebSphereMQ network. The logical connections are built upon the existing network infrastructure.

It is the Message Channel Agent (MCA) that is responsible for controlling the operation of a WebSphereMQ Channel.

Channels are usually defined in pairs as each channel has a single usage (it is either a 'sending' channel or a 'receiving' channel) this 'pairing' is what facilitates the bi-directional flow of messages between two Queue Managers

Channels come in four different types: sender, receiver, server, and requester. These four channel types can be combined to form one of five types of channel pairs.

- *Sender – Receiver*
- *Requester – Server*
- *Requester – Sender*
- *Server – Requester*
- *Server – Receiver*

While each combination has it own advantages and disadvantages, the *Sender - Receiver* matching is by far the most common pairing between Queue Managers.

It may seem an obvious statement but before any messages can flow between Queue Mangers, the channels must be started. For the MCA to start a channel the MCA receives the name of the channel to be started as parameter input from the RUNMQCHL command.

This command is usually issued 'under the covers' from one of three possible sources.

These are:

1. Entering control commands from the command line.
2. Entering MQSC commands from within the RUNMQSC utility
3. Using programmable command formats (PCF) on supported platforms.

The MCA then reads the channel definition file directly to obtain its attributes and start the channel. An appropriate instance of an MCA must be executing on each side of the communications link for the channel to start successfully.

Each MQSeries channel is associated with a transmission queue (XMITQ). A transmission queue is defined as a local queue with the usage attribute set to ***transmission***. Transmission queues contain messages that are destined for remote queue managers.
The channel reads messages from the transmission queue and transmits them to the MCA on the remote side of the link. The remote MCA then delivers the message to the destination queue (assuming the queue exists on the remote platform).

Triggering Channels

Channels can be started dynamically, based on the presence of a message in the associated transmit queue. This method is referred to as triggering channels. Triggered channels require the use of a channel initiator 'process'.

A channel initiator is basically a special purpose trigger monitor that starts the channel MCA based on the presence of a message on the associated transmit queue.

For triggering this XMITQ must have the **TRIGGER** parameter specified. Specifying this causes WebSphereMQ to place a trigger record into the corresponding initiation queue when the trigger criteria is met. This is the record that the channel initiator reads as input when starting a channel.

The triggering of channels offers some benefits that are not available with manually starting channels.

Reduced operational requirements

If channels are started manually it is reasonable to assume that this will be done one time, immediately after queue manager has been started (or at a set time). However, because channels rely upon external resources to function properly (i.e.TCP/IP), then error conditions like unavailable remote queue managers, underlying network problems, or operating system failures will prevent a channel from starting, or subsequently cause it to fail.

The required response to these problems depends upon which mechanism was chosen to start the channel. For manually started channels, operator intervention is required to reissue the **START CHANNEL** or **RUNMQCHL** command. This must be done after the underlying environmental or networking issues have been resolved. Coordinating these efforts may involve operations personnel at different geographic locations and different operating system disciplines.

Because a channel initiator starts the channel MCA based on the presence of a message in the transmission queue associated with the channel the process runs independently of any operator intervention.

As long as messages exist on the transmission queue (and all other trigger requirements are met), the channel initiator will try to start the channel.

If the required WebSphereMQ objects and external resources required for channel operation are available, the channel initiator can successfully start the channel.

If the required external resources needed for a channel to start are unavailable, the channel initiator's attempt to start the channel will fail.

Version 5 of MQSeries (now WebSphereMQ) introduced many significant performance enhancements in MCA operation. However, along with these enhancements came some changes to how a channel functioned.

With Version 5 channel status information is retained across invocations. The MCA considers some network errors as ‘permanent’. When a ‘permanent’ error occurs, the receiver channel retains a status of **STOPPED**. In these cases, normal triggering alone cannot successfully restart the channel. A START CHANNEL command must be explicitly issued on the receiver side of the channel to clear the "hard fail" condition that was recorded.

On the sender side of the channel, you are likely to encounter a channel status of **RETRYING**. This condition indicates that the MCA is attempting to restart the channel, but does not have the required resource available to it.

It is also possible that the XMITQ being serviced by the sender side MCA was placed in a "get inhibited" state when the original channel failure occurred. An ‘**ALTER QLOCAL**’ command must be issued to WebSphereMQ to reset the GET attribute of the XMITQ to **ENABLED**.

Disconnect Interval

The ‘Channel Disconnect Interval’ has much more of a role to play when channels make use of triggering.

The ‘Channel Disconnect Interval’ dictates how long a channel will stay in the **RUNNING** state while no messages are being transmitted.

When this timer interval is exceeded, WebSphereMQ sets the channel status to **INACTIVE**.

An **INACTIVE** channel can be started again with an operator command. Obviously, with triggering, this will take place ‘automatically’ when there are new messages on the transmission queue. When this happens the channel initiator restarts the channel.

Care must be taken to properly specify the disconnect interval.

If the disconnect interval is set too short, CPU resources will be wasted in needlessly shutting down and then quickly restarting channels.

Conversely, setting this parameter value too long defeats its useful purpose, in most cases.

Security ‘Exposure’

There is a ‘negotiation’ that takes place (called a **BIND**) when two Queue Mangers attempt to start a channel.

Part of this negotiation is to agree on various criteria that will exist for the duration of the channel connection.

One of these is (optionally) an encryption token. Channel encryption makes all message contents ‘unreadable’ as it flows across the channel.

The longer a channel remains active, the bigger the window of opportunity is to any subversive attempts to ‘hack’ the channel.

Setting a disconnect interval of 0 tells the MCA to never disconnect the channel. Doing this means that any encryption token that was negotiated at **BIND** will be used for the life cycle of this channel

execution instance. This could potentially be days, weeks or even months, giving a large window of opportunity to any potential subversive action.

Conclusion

Regardless of how you start channels, the simple fact remains that they must get started in order for messages to flow across them.
In a large WebSphereMQ network, automating the startup of channels becomes almost essential. Without using triggered channels, a significant effort in custom automation may be required. Given that WebSphereMQ may be running across many different platforms this may involve the use of several automation products to automate the starting of channels.

Coordinating this automation may also be a substantial undertaking.

Contrast this with triggered channels. Channel initiators exist on most MQSeries platforms and the definitions to control their execution are common to most MQSeries platforms.

Finally, and perhaps most significantly there is no custom coding required to implement triggering channels.