WebSphere MQ Everyplace V2.0.2

Contents

Configuring communications

Before any communications-specific configuration can take place, a number of decisions need to be taken from an application-design point of view. It is important to decide, in order of priority, what is important to the application. Typically, this will be a trade-off between performance, security and network-bandwidth usage.

Once these decisions have been made, it is important to create a prototype application to undertake empirical testing. This testing should use production data using a network sniffer tool in order to see timing and network usage information.

Operating system considerations

MQe uses TCP/IP sockets for communications and files to hold messages in persistent storage. On a system where a large number of messages are being flowed or a large number of clients are connecting (or both), operating system limits may be encountered. To ensure this does not cause a problem, the system administrator should ensure there will be enough file descriptors and TCP/IP sockets available. The method for doing this varies between operating systems. See the operating system documentation on how to set these limits.

Attributes

Attributes provide the ability to apply security and compression to data in the MQe network. Attributes are set either on a particular queue (available in Java™ only) or on a specific message. An MQeAttribute may contain one of each of the following in any combination:

MQeCompressor
Reduces the number of bytes across the wire; can impact performance.

MQe allows the application developer to select the best compression algorithm for the data being used by the application. In general, the compressor is best selected from empirical testing using data that will be generated by the application in production.

MQeCryptor
Potentially increases the number of bytes across the wire; provides security.

Typically when encrypting data, the amount of data in bytes is increased; this is no exception with MQe.

MQeAuthenticator
Adds bytes to the wire.

Used to provide authentication of the message sender.

Messages

An MQe message payload is held in one or more MQeFields objects. These are containers with a type, name and value. MQeFields may be recursive, so it is possible for an application developer to create messages containing multiple MQeFields objects, depending on the type of data required. The MQeFields object is self-describing, allowing MQe to dispense with a static header object to describe the data payload. Therefore, the amount of MQe information added to a message is partly dependent on the number of MQeFields objects in the MQeMsgObject. The best approach to minimize network usage is to put all the data into one MQeMsgObject with the smallest name appropriate for the application. This has two affects. Firstly, the amount of MQe data in the message payload is minimized, and secondly, the time

taken to serialize and de-serialize the message is also minimized. The down side of this approach is that your application becomes responsible for parsing the data in the message, rather than being able to use multiple MQeFields objects.

Every time data is sent across the network, additional bytes are added by the network protocol. For instance, TCP adds 20 bytes; then IP add an additional 20 bytes. The application developer can minimize this overhead by creating fewer large messages, rather than sending numerous small messages. For more information, see the various adapter settings throughout this section.

If your MQe messages are destined for an MQe queue, you can use the MQe API MQeMultiMsgObject, which allows you to put multiple messages into a single object. This object allows the application developer to wrap multiple messages into one large message with the corresponding support for retrieving the messages. MQeMultiMsgObject removes the responsibility of parsing a large message for imbedded smaller messages from the application.

Queue and queue manager names

The names of the queues and queue managers are sent across the network as part of the MQe information added to the message payload. The names for these resources should be kept as short as possible.

Communications

There are a number of objects instantiated as part of MQe communications, which although not exposed to the application developer, they can be configured to help optimize network usage. Many of these configuration values may be set using administration messages. The following diagram shows the MQe communications objects.

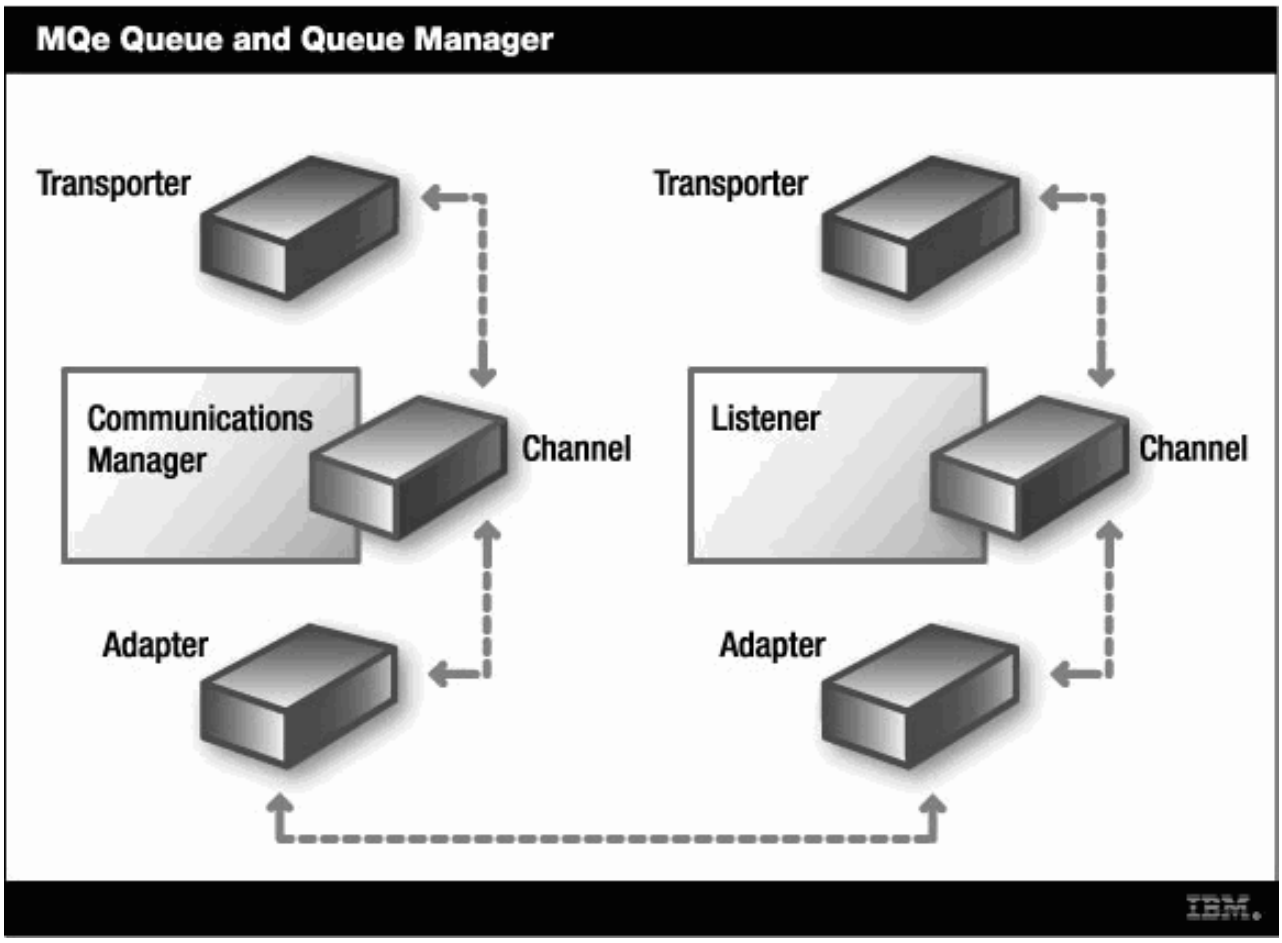

Figure 1. MQe queue and queue manager

MQeTransporter

The MQeTransporter class is not exposed to the API. This class is responsible for establishing a link from one queue to another. There is a one-to-one relationship between an MQeQueue and an MQeTransporter. The MQeTransporter is not configurable and is only included for completeness.

MQeChannel

The MQeChannel class is not exposed to the API. This class is responsible for establishing a link between queue managers. As can be seen from the diagram, the object holding a reference to an open channel alters, depending on whether the channel is incoming or outgoing.

The channel timeout value is given in milliseconds. A background thread is responsible for closing the channels if they have been idle for the period of time set by the channel timeout value. It is important to make the timeout values in the client and server compatible; ideally the client should timeout before the server. This allows the client to send the server the close command when the client channel is closing, thus both ends of the channel are closed. If the server closes before the client, no such command is sent. This will result in one of the following:

- The client socket being put into FINWAIT2 state as a result of it being closed when the *partner* socket is no longer able to respond with the required acknowledgement. State can be held either indefinitely or until the operating system times it out.

- The client wishes to send another message and receives a channel-ID error from the server. The client then closes the channel and has to recreate a new channel, thus taking up time and network resources.

The channel timeout value needs to be long enough for a client and server to have finished sending/receiving messages but not so long that channels take up valuable system resources.

For outgoing channels, the channel timeout is set on the queue manager. This has to be done once the queue manager is configured. Use the MQeQueueManagerAdminMsg to set the channel timeout on the queue manager.

To set the channel timeout on incoming channels, specify the channel timeout using the administration message MQeCommunicationsListenerAdminMsg for creating or altering a listener.

When setting the channel timeout on either the listener or the queue manager, the underlying thread responsible for the channel timeout is not notified. For testing purposes, it may be worthwhile stopping then restarting the queue manager once the channel timeouts have been sent. In a production environment, once the queue manager and listener have been configured, the values are kept in the MQe registry and are therefore used when the queue manager is started.

The default value is 5 minutes.

Communications adapters

The communications adapters provide the protocol support for sending the data across the network. The communications adapters have a number of configurable values; all of these are set using Java properties. The Java properties are set at the JVM level and are accessed by the communications adapter when it is instantiated. The adapter does not revisit these properties. This means if the JVM level properties are altered, they will not take affect until a new adapter is created.

Information about these properties may also be obtained from the *Java Programming Reference Manual* under the com.ibm.mqe.adapters.MQeCommunicationsAdapter class.

Packet size

To set the packet size value for the Native code base, an entry is required in the Windows® Registry.

The packet size has the most profound affect on the number of packets sent across the network. To determine the best packet size for your network, some empirical testing is required. You will need a sniffer tool for this.

An Ethernet LAN typically will have a maximum transmission unit (MTU) of 1500; however, this may be lowered by a router. In order to determine the effective MTU of your network, it is best to force some IP fragmentation; that is, forcing IP to split the data up. First of all, set the packet size to a value larger than the expected MTU of your network. Next, send messages from one MQe queue manager to another of a size greater than the packet size so the adapter has to split the data up. By looking at the results from the sniffer tool, you should see some IP fragmentation, which will determine the effective MTU of the network.

Having determined the MTU of the network, you now need to set the packet size allowing for the protocol headers:

- TCP/ IP has a total header length of 40 bytes (20 bytes for IP and 20 bytes for TCP).
- UDP/IP has a total header length of 44 bytes (12 bytes for MQe, 12 bytes for UDP and 20 bytes for IP). When creating the packets, the MQe UDP/IP adapter will take into account its own header information so all that needs to be taken into account are the UDP and IP headers, amounting to 32 bytes. If the MTU of the network is 532, the packet size should be set to 500. If the packet size is less then 30, the default packet size will be used.

- HTTP is a little more difficult to quantify as the amount of data put into the HTTP header by MQe will vary depending on whether a proxy is being used or if a servlet is being accessed and the length of data being sent. The minimum added by MQe is 116 bytes of data which assumes the length will fit into 3 bytes for its string representation. Additional bytes need to be allowed for if proxies or servlets are being used.
- Ethernet has a 14 byte header.

Java Default value TCP/IP 4096, UDP/IP 500

Set using the Java property com.ibm.mqe.adapters.MQeCommunicationsAdapter.packetSize

Native

Default value 500

Set using the Windows Registry: **HKEY_LOCAL_MACHINE** → **SOFTWARE** → **MQe** → **CurrentVersion** → **Communications** → **packetSize**. The value is a string and should represent the size in bytes.

Non-blocking timeout

The non-blocking timeout allows you to determine how long an adapter will wait on a socket before checking to see if the adapter has been closed. This close will probably be issued; either as a result of a channel timing out or the queue manager being closed. This value is used in conjunction with the Timeout value.

Java Default value is 1 second.

Set using Java property com.ibm.mqe.adapters.MQeCommunicationsAdapter.nonBlockingTimeout.

The value should represent the time in milliseconds.

Native

Not available.

Timeout

The adapter will attempt to read from a socket until the timeout value is exhausted. In order to allow the adapter to check for a shutdown, due to channel timeout or the queue manager being shut down, the actual socket timeout is set to the "Non-blocking timeout." The timeout value is then decremented by the non-blocking timeout until it is exhausted and is thus used as the overall timeout. Once the timeout value has been reached, the retries value is then decremented as described below.

The UDP/IP adapter uses the timeout value to determine the amount of time the adapter should wait for an acknowledgement. The timeout should be set to the time it takes a packet to flow across the network from initiator to responder. It is further suggested that multiple pings are undertaken with the packet size appropriate for the particular network. For instance if a + b is equal to the amount of time taken by the packet for the round trip, MQe assumes that a = b, the timeout should be set to the value of a. The UDP/IP adapter sends the value of the timeout across the network when setting up a conversation; the maximum value for the UDP/IP adapter is 4294967295.

Java Default value is TCP/IP 5 seconds, UDP/IP 10 seconds.

Set using Java property com.ibm.mqe.adapters.MQeCommunicationsAdapter.timeout.

The value should represent the time in milliseconds.

Native

Default value 10 seconds.

Set using Windows Registry: **HKEY_LOCAL_MACHINE** → **SOFTWARE** → **MQe** → **CurrentVersion** → **Communications** → **socketTimeout**. The value is a string and should represent the time in milliseconds.

Retries

The number of retries is used in conjunction with the "Timeout" on page 5 value. When an adapter exhausts the timeout value the number of retries is decremented and the adapter then starts the retry process again. This value is also used when an adapter experiences a problem, say, with obtaining a socket as well as reading from a socket.

Java Default value 3

Set using Java property com.ibm.mqe.adapters.MQeCommunicationsAdapter.retries

Native

Default value 3

Set using the Windows Registry: **HKEY_LOCAL_MACHINE → SOFTWARE → MQe → CurrentVersion → Communications → retry**. The value is a string and should represent the number of retries.

HTTP version

For use with the com.ibm.mqe.adapters.MqeTcpipHttpAdapter. In order to minimize the number of sockets being created and discarded, which can prove onerous for some operating systems, it is possible to specify the version of HTTP to be used. If the value for this property is set to 1.1 then sockets are not closed between individual messages but kept open for the life of the adapter. The listening adapter will use the version passed by the client.

Java Default value 1.0

Set using Java property com.ibm.mqe.adapters.MQeCommunicationsAdapter.httpVersion

The value should be either 1.0 or 1.1

Native

Default value 1.0

May not be set

Adapter group size – UDP/IP adapter only

UDP/IP is a connectionless protocol and it is therefore necessary for MQe to acknowledge which packets have been received. To help minimize the network traffic, the packets are flowed across the wire in groups. Once the number of packets in a group have been sent, the initiator will then wait for an acknowledgement from the responder before sending a request for acknowledgement.

- Set using Java property com.ibm.mqe.adapters.MQeCommunicationsAdapter.groupSize
- Default value 5
- Maximum value 255

Adapter variance – UDP/IP adapter only

This variable is used as a constant in conjunction with the timeout value. The variance value is used to provide a value that reflects possible variant behavior in a network. The value is set in milliseconds.

- Set using Java property com.ibm.mqe.adapters.MQeCommunicationsAdapter.variance
- Default value 1000
- Maximum value 4294967295

UDP/IP configurable value usage

As has been previously stated, the UDP/IP adapter is particularly sensitive to the values of the configurable parameters. The following information shows how these values are used internally:

- Time for responder to wait for next data packet = timeout + variance

- Time for responder to wait for next data packet after sending an acknowledgement = (timeout + variance) x 2
- Time for initiator to wait for acknowledgement after sending last packet in group = (timeout + variance) x (groupSize + 1)
- Time for initiator to wait for acknowledgement after sending a request for acknowledgement = (timeout + variance) x 2

Backlog

The backlog set on the listener is 128. Currently there is no way of altering this. Windows platforms will ignore this setting if the operating system is not a server operating system and will default it to 5.