This article describes the IBM MQ messaging provider activation specifications in WebSphere Application Server. It also describes the mechanisms that are used to connect to IBM MQ queue managers and monitor destinations for messages. Then, it shows how the application server delivers suitable messages to message-driven beans for processing. The article is for WebSphere Application Server administrators and developers who need to tune their systems and for IBM MQ administrators who want to understand how the application server interacts with IBM MQ.

**Introduction**

*Editor's note:*
This article is based on IBM WebSphere Application Server V7 and V8 and IBM MQ V7. The information and guidance in the article also apply to WebSphere Application Server V9 and IBM MQ V8 and V9.

IBM® WebSphere® Application Server V7 and V8 support asynchronous messaging based on the Java Message Service (JMS) specification. By using the IBM MQ messaging provider, you can write message-driven beans that listen on an IBM MQ destination (either a message queue or a topic). When a message arrives on the destination, the `onMessage()` method of the message-driven bean is started to process the message.

In WebSphere Application Server V7 and V8, the IBM MQ messaging provider supports the use of activation specifications to monitor destinations that are hosted by IBM MQ queue managers. This article shows how activation specifications connect to MQ on distributed platforms. It describes the mechanism that is used to monitor the destinations that are looking for messages. Then, it shows how message-driven beans are started after a suitable message is detected. The article assumes a basic knowledge of JMS and IBM MQ.

**Activation specifications**

In general terms, J2C activation specifications are administered objects that contain information about how to connect to a JMS provider, along with details of the destination on that JMS provider.
that will be monitored for messages. When you deploy an application that contains a message-driven bean, you need to specify the activation specification that the message-driven beans will use. When the activation specification starts, it connects to a JMS provider, opens the JMS destination, and then monitors it looking for messages.

Figures 1 and 2 show a sample MQ activation specification that is defined by using the activation specification window in the WebSphere Application Server Integrated Solutions Console. When this activation specification starts, it makes a BINDINGS mode connection to a local MQ queue manager called pault, opens the destination jms/TestQueue, and then starts monitoring this destination for messages.

**Figure 1. Specifying the queue manager name and transport type**

Activation specifications can be configured to use message selectors, which enable them to pass only messages that meet the selection criteria to message-driven beans. In Figure 2, no message selector is specified. Therefore, all messages that arrive on the destination are suitable for processing by this activation specification.
Figure 2. Specifying the JMS destination that an activation specification will monitor

When an activation specification finds a suitable message, it schedules a piece of work within the application server to process it. Each message requires a JMS server session to run, and multiple messages can be processed at the same time.

Each activation specification has an associated server session pool, and its size controls the number of messages that can be processed concurrently by an activation specification. The default size of the server session pool is 10, which means that up to 10 messages can be processed at the same time by a single activation specification. To change the server session pool size, modify the activation specification advanced property **Maximum server sessions**, as shown in Figure 3.
Detecting messages

The mechanism that an activation specification uses to detect messages on JMS destinations that are hosted on an MQ queue manager varies depending on the MQ messaging provider mode that is being used, as described below.

**MQ messaging provider normal mode**

Activation specifications use the MQ messaging provider normal mode if they are connecting to an MQ V7 queue manager and they have the **Provider version** property set to either unspecified (the default value) or 7. In this mode of operation, the activation specification takes advantage of several features of IBM MQ V7 when it connects to a queue manager and gets messages. When it starts, the activation specification:

1. The activation specification creates a connection to the MQ queue manager that it is set up to use.
2. If the activation specification is configured to use a Queue Destination, the activation specification opens the queue by using the MQ API call MQOPEN.
3. If the activation specification is configured to use a Topic Destination, the activation specification issues an MQ API MQSUB call to subscribe to the appropriate topic.
4. After the queue is opened or the topic is subscribed to, the activation specification uses the MQ API call MQCB to register a callback. The callback is set up with the following MQ GetMessageOptions:
   - MQGMO_BROWSE_FIRST
   - MQGMO_UNMARKED_BROWSE_MSG
5. After the callback is registered, the activation specification issues an MQ MQCTL API call, which tells the queue manager that the activation specification is ready to start receiving messages.

Now, when a suitable message arrives on the queue that the activation specification is monitoring, or is published on the topic that the activation specification is subscribed to, the queue manager marks the message to prevent other activation specifications from detecting it. It then passes details of the message to the activation specification by using the callback that was set up earlier.

### MQ messaging provider migration mode

The other way that activation specifications can connect to an MQ queue manager is by using the MQ messaging provider migration mode. This mode is used if one of the following conditions is true:

- The activation specification is configured to connect to an IBM MQ V6 queue manager.
- The activation specification is configured to connect to an IBM MQ V7 queue manager, and has the `Provider Version` property set to 6.
- The activation specification is configured to connect to an IBM MQ V7 queue manager by using the CLIENT transport, and is using an MQ channel that has the `Sharing Conversations` (SHARECNV) property set to 0.

When the activation specification starts in migration mode, it:

1. Creates a connection to the MQ queue manager that it is set up to use.
2. If the activation specification is configured to monitor a Queue Destination, it issues an MQOPEN API call to open the queue.
3. If the activation specification is configured to use a Topic Destination, it:
   - Opens a subscription for the topic.
   - Checks the values of the activation specification Broker Properties `Broker connection consumer subscription queue` and `Broker durable subscriber connection consumer queue` to see which WebSphere MQ queue the Broker will publish messages for this activation specification to.
   - Calls the MQ API MQOPEN to open the appropriate subscription queue.
4. After the queue is opened on the queue manager, the activation specification browses the queue that is looking for messages by issuing MQGET API calls. The activation specification uses a combination of the MQ GetMessageOptions MQGMO_BROWSE_FIRST and MQGMO_BROWSE_NEXT to scan the queue from top to bottom.

### Processing messages

When an activation specification is detected, a message on a destination (either because an IBM MQ V7 queue manager passed back information about a message by using a callback, or because the activation specification browsed a suitable message), it:

1. Constructs a message reference that represents the message.
2. Gets a server session from the activation specification server session pool.
3. Loads the server session with the message reference.
4. Schedules a piece of work with the application server Work Manager.

The activation specification then goes back to looking for more messages.

Getting server sessions
As mentioned earlier, activation specifications will process up to 10 messages concurrently by default. What happens if an activation specification tries to process a message and all 10 server sessions are already busy processing messages? In this situation, the activation specification will block until a server session becomes free. As soon as a server session is available, the activation specification loads it with the message reference, and then schedules a new piece of work so the server session can run again.

Scheduling work
When the activation specification loads a server session with a message reference, it schedules some work so that the message can be processed. What happens to the work? The Work Manager:

1. Gets a thread from the MQ messaging provider Resource Adapter thread pool in WebSphere Application Server. The name of this thread pool is WMQJCAResourceAdapter.
2. Runs the piece of work on this thread.

After the work is scheduled, the application server Work Manager runs this piece of work at some point in the future. The work, when started:

1. Starts either a local or global (XA) transaction, depending on whether the message-driven bean requires XA transactions or not (specified in the message-driven bean's deployment descriptor).
2. If this time is the first time the server session is used, it:
   • Creates a connection to MQ.
   • Issues an MQOPEN API call to open the queue where the message is.
3. Gets the message from MQ by issuing a destructive MQGET API call.
4. Runs the message-driven bean's onMessage() method.
5. When onMessage() has completed, the server session completes the local or global transaction before it exits.

To improve performance, the connection to the queue manager that the server session uses is left open after the message is processed and the work completed. Then, the next time the server session is used to process a message, it does not need to reconnect to MQ and reopen the queue that contains the message. By default, unused server sessions that are associated with activation specifications are left open for 30 minutes before they are closed off. You can extend this timeout period by modifying the value of the activation specification advanced property Server session pool timeout, as shown in Figure 4.

On a lightly loaded system, the time between the piece of work that is being scheduled and the Work Manager starting the work can be just a few milliseconds. On busy systems, there might be a lengthy delay before the work is started. Here are two possible reasons for a delay:
• There were no free threads in the WMQJCAResourceAdapter thread pool to run the work.
• The Work Manager was able to get a thread from the thread pool, and but then cannot start
  the work because the application server was too busy.

The Work Manager records when a piece of work was scheduled, and when it starts the work, it
checks how much time elapsed since the activation specification scheduled the work. By default,
the activation specification expects the work to be started within 10 seconds of it being scheduled.
If more than 10 seconds elapse before the Work Manager starts the work, then a WorkRejected
exception is returned back to the activation specification, causing exceptions similar to the one
below to appear in the application server SystemErr.log:

```
Exception in thread "WMQJCAResourceAdapter : 1" java.lang.RuntimeException:
javax.resource.spi.work.WorkRejectedException: Work timed out (id=4), error code: 1
Caused by: javax.resource.spi.work.WorkRejectedException: Work timed out
  (id=4), error code: 1
```

When an exception like this one occurs, the message in the Message Reference will be
"unmarked" by the queue manager so that it can be reprocessed. You can change this 10-second
time limit on the activation specification Advanced properties panel by using **Start timeout**, as
shown in Figure 4.

**Figure 4. Modifying the server session timeout and the amount of time to wait
for work to start**
Earlier, it was mentioned that a piece of work might get delayed if the WMQJCAResourceAdapter thread pool does not have enough threads. This leads to the obvious question, "What should the size of this thread pool be?"

One thread pool per application server is used by activation specifications to run server sessions. Each activation specification has an advanced property that is called **Maximum server sessions**, which defines the maximum number of concurrent server sessions that can be running at the same time. Because each server session is used to process messages, this property essentially says how many messages can be processed concurrently by message-driven beans that use this activation specification.

To determine what the size of the WMQJCAResourceAdapter thread pool should be, add the values of the **Maximum server sessions** properties for each MQ messaging provider activation specification on the application server. For example, suppose that you have 25 activation specifications that are defined, each with the **Maximum server sessions** property set to 3. In this situation, up to 75 server sessions can run concurrently, each of them using a thread from the WMQJCAResourceAdapter thread pool. Therefore, set the maximum size of this thread pool to 75. Figure 5 shows the WMQJCAResourceAdapter thread pool panel in the WebSphere Integrated Solutions Console, where you can change the size of this thread pool.

**Figure 5. Changing max number of threads available to all activation specifications defined in the application server**

![Figure 5: Changing max number of threads available to all activation specifications defined in the application server](image-url)
If WorkRejected errors start appearing in the application server SystemOut.log file, ensure that the WMQJCAResourceAdapter thread pool is large enough to handle all of the server sessions that are needed by your activation specifications. If the thread pool is the right size, then the errors are caused by the Work Manager being unable to start the work request within the specified time period. In this situation, either increase the value of the activation specification advanced property Start Timeout, or investigate reducing the load on your application server system.

**Using MQ messaging provider normal mode**

As described previously, a delay can occur between a message that is being detected and that message is being processed by a message-driven bean in the following situations:

- All server sessions that are associated with an activation specification are being used.
- All threads in the WMQJCAResourceAdapter thread pool are being used to process messages.
- A delay occurs between work that is being scheduled and the Work Manager starting the work.

If the activation specification is running in MQ messaging provider normal mode, the queue manager marks messages before it passes their details back to the activation specification. Marking the message means that no other activation specification (or WebSphere Application Server Listener Port), either running in the same application server or on a different application server, can see the message. This method prevents another message-driven bean from getting the message before a server session has time to process it.

By default, messages are marked for 5 seconds. To change this time period, modify the MQ queue manager property Message mark browse interval (MARKINT).

After MQ passes details of a message to process to an activation specification, the 5-second timer starts. During these 5 seconds:

1. The activation specification must get a server session from the server session pool.
2. The server session must be loaded with details of the message to process.
3. The work must be scheduled.
4. The Work Manager must start the work request.

A delay can occur in getting a server session or a thread from the WMQJCAResourceAdapter thread pool. Or, if the system is busy, it can take a long time for the Work Manager to schedule the work. The time between MQ passing details of the message and it actually being consumed might be longer than 5 seconds. What happens in this situation?

Well, if the message is on the queue for longer than 5 seconds, the queue manager will unmark it, and another activation specification or listener port is then free to come along and get the message. If this situation happens, when the server session that previously was given details about this message tries to get it, it will find that the message is no longer on the destination, and write the following message to the application server SystemOut.log:
CWSJY0003W: JMSCC0108: WebSphere classes for JMS attempted to get a message for delivery to an message listener that had previously been marked using browse-with-mark, but the message was not there.

If you see this message, you have three options:

- Increase the value of the MQ queue manager property Message mark browse interval (MARKINT) to give the activation specification more time to get the message. If you have multiple applications monitoring the same destination and want the messages to be processed quickly, consider adopting this approach because increasing the amount of time the message is marked will prevent other applications from getting it.
- Tune the application server so that it does not block either waiting for a server session or waiting for a thread from the WMQJCAResourceAdapter thread pool. To do this, increase the size of both the server session pool and the thread pool. This change means that messages can be processed within the default message browse mark interval, although more resources will be used within the application server as it will be able to process more messages concurrently.
- Do nothing. This option is not recommended because the activation specification will waste time and resources from trying to get messages that are already picked up and processed by another application.

**Conclusion**

This article described the mechanisms that activation specifications use to get messages from MQ queue managers. It explained how activation specifications create a connection to a queue manager and the mechanisms they use to monitor JMS destinations that are looking for suitable messages to process. This article also described how the application server schedules the processing of messages when a suitable message is found.
Related topics

- IBM WebSphere Application Server on Cloud product page
- Free trial of WebSphere Application Server
- WebSphere documentation in the IBM Knowledge Center
- IBM MQ product page
- Free trial of MQ
- MQ documentation in the IBM Knowledge Center
- IBM Messaging developer center
- IBM Middleware User Community
- JCA 1.5, Part 2: Work management and transaction inflow

© Copyright IBM Corporation 2011, 2016
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)