WebSphere software

IBM

Redbooks Paper

Simon Davitt

WebSphere Application Server V6: Default Messaging Provider Problem Determination

This paper describes some of the problems that can be encountered when using the IBM® WebSphere Application Server V6 messaging component. Typically, this type of failure would manifest itself as an application that fails to start, an application hang, or an application that fails to receive an expected message. There might or might not be an accompanying error message from the application itself.

Although WebSphere Application Server V6 can use messaging mechanisms familiar to users of WebSphere® Application Server V4 and V5 (IBM WebSphere MQ provider and generic message providers), the default messaging provider included with V6 is very different to its predecessors. This paper confines itself to investigating messaging problems that are specific to the default messaging provider that is included with WebSphere Application Server.

Important: We recommend that you start your problem determination process by reading *Approach to Problem Determination in WebSphere Application Server V6*, REDP-4073, at:

http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf

Introduction

WebSphere Application Server V6 messaging is a general term for a group of components that provide the messaging functionality for applications. WebSphere Application Server provides a default messaging provider, as well as support for IBM WebSphere MQ or generic messaging providers.

The *default messaging provider* is the Java™ Message Service (JMS) API implementation for messaging (connection factories, JMS destinations, and so on). The concrete destinations (queues and topic spaces) behind the default messaging provider interface are implemented in a *service integration bus*. Similarly, the WebSphere MQ JMS provider is the JMS API implementation with WebSphere MQ (with queue managers, for example) implementing the real destinations for the JMS interface.

A service integration bus consists of one or more bus members. A bus member can be an application server or a cluster. Each bus member will have one (or possibly more in the case of clusters) *messaging engine* that manages connections to the bus and messages. A service integration bus can connect to other service integration buses and to WebSphere MQ.

Figure 1 on page 3 shows a simplified schematic of some of the components that make up the default messaging provider function. Although this is a simplified schematic, it allows us to see the many components involved: the service integration bus, messaging engines, links to remote messaging providers, data sources, destinations, queue points, and more.

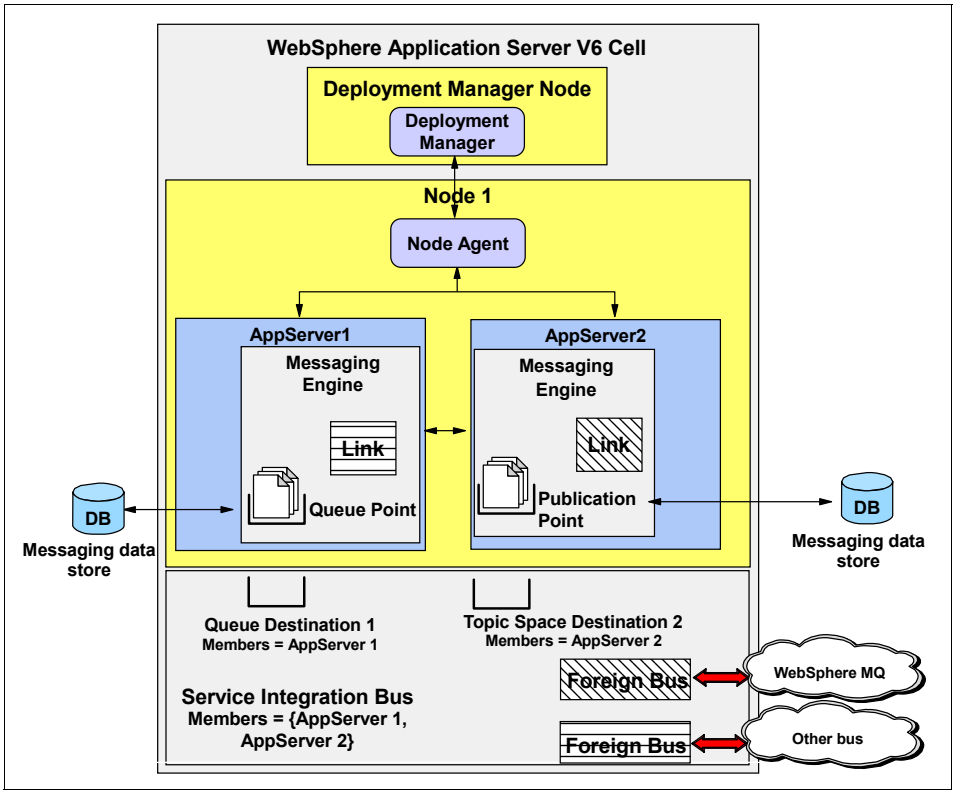

Figure 1 Default messaging provider structure

To gain a common understanding of the areas where potential problems can occur and before attempting to isolate and identify a messaging problem, it is useful to start with an overview of the various components using the diagram in Figure 1. These components include the following items:

- The service integration bus (referred to simply as the *bus*) is the primary architectural component on which the WebSphere Application Server V6 message functionality is based. It is used by the default messaging provider.
- The bus can hold references to other service integration buses or to WebSphere MQ. These are *foreign bus* definitions.
- *JMS-administered objects* encapsulate the information that is required within a JMS application to connect to the JMS provider and to access the *destinations* that are defined by the JMS provider. JMS-administered objects refer to objects on the bus.

- A destination defined on the bus can be a queue, a topic space, or a foreign bus destination. Each destination has a *reliability* property that defines a quality of service. The JMS interface can specify persistent or non-persistent for the messages sent, and the JMS-administered objects can map this JMS persistence setting to a preferred reliability in the bus.
- Destinations can have an associated *mediation*. A mediation allows the processing of messages at the destination before delivery to the consumer.
- Access to the bus is managed by the *messaging engine*. The messaging engine runs in an application server. When a cluster of application servers is added as a member of the bus, a single messaging engine is automatically created and associated with the application server cluster, regardless of the number of application servers defined as members of the cluster. At run time, this messaging engine is activated within a single application server within the cluster. The application server that is chosen to host the messaging engine will be the first cluster member to start.
- A queue is assigned to one bus member. The messaging engine in the bus member hosts the message point for the queue, known as a *queue point*. The queue point is the location where messages for the queue are stored and processed. If the bus member has more than one messaging engine, the queue is partitioned across the messaging engines. Each messaging engine has a separate queue point for the queue and handles a share of the messages arriving at the destination.
- Each messaging engine has its own data store where it stores persistent messages. Non-persist messages can be spilled to the data store if the messaging engine decides to do so (for example, to reduce the memory footprint of messages).
- A bus can be connected to other service integration buses or to WebSphere MQ. These connections are made by a *link* definition on a messaging engine within the bus, with the type of link depending on the target. A messaging engine can also be configured to emulate the WebSphere Application Server V5 embedded messaging provider to allow V5 embedded messaging clients to connect to the bus as though they were connected to the V5 embedded provider.

As you can see, there are many different components and many opportunities for configuration errors. These might be with the local components, such as the messaging engine or data store, or with WebSphere MQ.

For a comprehensive description of the messaging features, terminology, and topologies available, see Chapters 10 and 11 in the IBM Redbook *WebSphere Application Server V6 System Management and Configuration Handbook*, SG24-6451.

Problem categories

Messaging problems fall broadly into the following categories:

- Messaging engine startup problems
- Message flow problems, including interfaces to foreign buses and client applications
- Application configuration and resource problems

Note: Although this paper does not specifically address performance problems related to messaging, you should be aware that references to large message objects must be maintained by many components because they are delivered around the network. This can have severe implications for the Java Virtual Machine (JVM™). If you suspect messaging applications are causing memory problems, this is something to consider.

Work the problem

You begin the problem determination (PD) process by collecting the appropriate data that is required to diagnose the problem. We first provide a list of all the documentation that might be required.

Next, you go through a series of questions and actions that will help you define the high-level symptoms that you are experiencing. Each of these steps leads to a more detailed procedure that is designed to take you through the process of analyzing data to determine the most likely source of the problem. While it is impossible to enumerate every possible failure scenario, we describe various use cases that have occurred during testing and real customer use as examples for these categories. We also provide a strategy for analyzing and resolving these problems.

And lastly, we provide guidance on the next step to take for resolution, whether it be a support site, contacting IBM, information about configuration, or some other suggestion as to how to proceed.

Before going further with the investigation, it is worth asking some basic questions about the nature of the problem in an attempt to isolate why the problem is occurring. Basic questions include:

- What is the error message received or the scenario that leads you to believe you have a messaging problem? Any error or warning messages with a prefix of CWSI or CESJ can be an indicator of the problem. If your application has a message-driven bean that uses WebSphere Application Server V5 JMS resources, look for the prefixes MSGS and WMSG.

For a complete list of messaging message prefixes, see:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.nd.doc/info/ae/ae/welc_ref_trb_msg.html

You can find explanations of individual messages in the WebSphere Information Center. When searching for message prefixes, enclose the prefix in double quotation marks.

- What changes have been made to the application, configuration, WebSphere Application Server maintenance level, operating system, network, or hardware?
- What are the implications of backing out any changes that were made to verify whether that the change precipitated the problem?
- Do the basic functions of the application work, or is this some new scenario (for example, some code that has been added or a newly tested function)?
- Have you tested the IVT applications and the samples to ensure that they work OK? This is extremely important, because it establishes the basic functionality of the application server.
- Have you validated the WebSphere Application Server configuration using **`$AdminConfig validate`**? Although this is not infallible, it does provide a check of the configuration changes that you might have made. Configuration problems can also be viewed from the administrative console (**Troubleshooting** → **Configuration Problems**).
- Has there been a change to the data being used? A commonly encountered problem is where a new data format (long messages, object messages, or null data) is used for the first time and this precipitates an error in the processing of that data.

Having established the type of change that might have precipitated the problem, it might be expedient to back out the change to allow the system to continue running.

Collect the data

There is a minimum set of documentation that is required for investigating any problem. This information should be gathered before starting to examine any messaging problems. Then, depending on the type and the complexity of the problem, additional documentation might be required. For our initial investigations, the minimum set of documentation that you should gather is:

- SystemOut.log
- SystemErr.log
- Exception logs in FFDC directory
- Exceptions received by any client applications

You can find information about finding and reviewing logs and traces in *WebSphere Application Server V6: Diagnostic Data*, REDP-4085, at:

http://www.redbooks.ibm.com/redpapers/pdfs/redp4085.pdf

If the problem is difficult to recreate or disruptive to business operations, see "The next step" on page 50 for a complete list of documentation to collect before continuing. In particular, you should review the MustGather document for service integration technologies for a complete list of documentation that is required by IBM support.

Analyze the high-level symptoms

At this point, the problem could be any of the categories that we have discussed. The following analysis strategy should help you define the problem.

Begin by examining the documentation that you have gathered. Look for any error codes, exception codes, or stack traces. Use these as search arguments in the appropriate IBM support sites and the WebSphere Information Center.

It is also a useful technique to use an Internet search engine to search the IBM Web site for matching symptoms and message codes:

http://www.ibm.com/software/webservers/appserv/was/support/

Often, the explanatory notes in the WebSphere Information Center will be sufficient to resolve a problem.

Ensure that the messaging engine has started

The first step in diagnosing any messaging problem is to ensure that all the messaging engines on the bus are active. To check the status of the messaging engines from the administrative console:

1. Select **Service integration** → **Buses**.
2. Click the bus name to open the details page.
3. Click **Bus members**.
4. Click the bus member.

This shows a list of messaging engines and their status, as shown in Figure 2 on page 8.

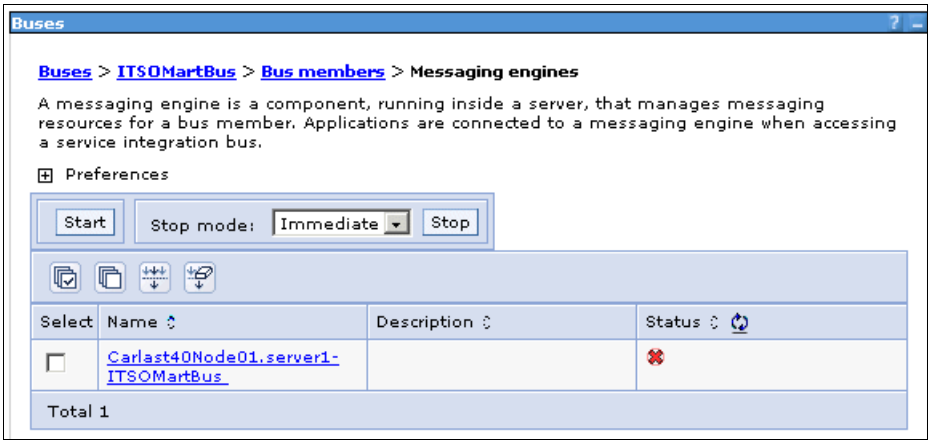

Figure 2 Checking the messaging engine status

If a messaging engine is stopped, go to “Messaging engine startup problems” on page 10.

This should always be the first area to isolate and eliminate from your investigations. It might be that you iterate through this process quickly for every problem simply to ensure that the underlying messaging components have started correctly and are active.

Check the message flow to the destination

If the application has been running successfully and then fails, it would appear to rule out a startup problem. So, you should continue by investigating message delivery and JMS application problems.

The next thing to do is see if there is a message available for the application to consume. This can be done easily by using the administrative console and looking at the runtime values for the queue. To do this, select **Service integration** → **Buses** → ***<busname>*** → **Destination** → ***<destinationame>*** → **Queue Point** → ***<qname>***.

Figure 3 on page 9 shows an example.

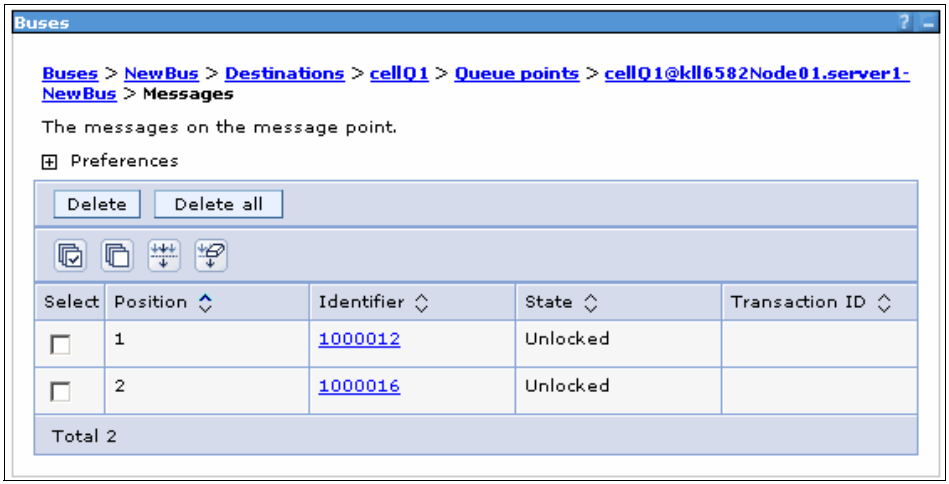

Figure 3 Viewing messages on the queue

If the messaging engine started correctly and there are messages on the destination queue point that you believe your application should be processing, go to “Message flow problems” on page 24.

If there are no messages, continue to the next section.

Ensure that the application is producing messages

If you have read through thus far, then you have now confirmed that the messaging engine has started but that there are no messages available for processing on the destination queue point.

At this point, you could still be examining a message flow problem. Why are there none of the expected messages on the destination? This could be a problem with the message producer application. Has the message producer previously delivered messages? You have checked the messaging engine and queues for the message consumer in an earlier step. Now, it is time to perform the same checks for the message producer.

Assuming that the message producer’s messaging engine is working correctly, you should next ensure that the application that is able to access the default messaging provider and the resources have been configured properly. For information about this, go to “Application configuration and resource problems” on page 44.

Even assuming the message producer has been executing correctly up to now, this does not rule out an application problem, for example when processing a

specific message or data type. You should next look at the possibility of a message flow problem as described in “Message flow problems” on page 24.

What to do if your symptom is not listed here

As stated previously, this paper cannot cover all possible scenarios that might occur. It is quite possible that your problem does not fall neatly into one of the categories that we have mentioned. If you do not see your symptom listed here, go to “The next step” on page 50 for more information about how to proceed.

Analyzing problem areas

This section explores the following problem areas:

- Messaging engine startup problems
- Message flow problems
- Application configuration and resource problems

Messaging engine startup problems

This type of error is not usually apparent at startup. Few applications will fail to start due to being unable to connect to the bus. Only message-driven bean (MDB) applications attempt to connect to the bus during startup, and they will start successfully as long as the bus is defined, even if the messaging engine they are connecting to is not available yet. This is by design, because applications start before the messaging engine in the server startup sequence. MDBs will fail to start if there is no ActivationSpec for them, or if the bus they are configured to connect to does not exist.

EJB™ applications generally do not connect to the bus until a client connects to them and tells them to. At this point, they will get a JMS exception on connect, but the application will have started and probably will not stop due to the failure to connect.

Becoming familiar with a normal startup

To evaluate a messaging engine startup problem, it is advisable to familiarize yourself with how a normal startup appears in the log files. Example 1 on page 11 is an extract from a SystemOut.log file that shows a normal messaging engine startup.

Note: In the following examples, the date and time that is displayed in the log file output has been replaced by the sequence **[...]**. This has been done for brevity. In some examples, the messaging engine name has been replaced by `mename`.

Example 1 Messaging engine startup

```
:
[...] 0000000a SibMessage    I   [:] CWSIU0000I: Release: WAS601.SIB Level: o0518.09
:
:
[...] 0000000a ResourceMgrIm I   WSVR0049I: Binding _kll6582Node01.server1-TestSIBus as
jdbc/com.ibm.ws.sib/kll6582Node01.server1-TestSIBus
:
:
[...] 0000000a TCPChannel    A   TCPC0001I: TCP Channel SIB_TCP_JFAP is listening on host *
(IPv4) port 7276.
[...] 0000000a WSChannelFram A   CHFW0019I: The Transport Channel Service has started chain
InboundBasicMessaging.
[...] 0000000a TCPChannel    A   TCPC0001I: TCP Channel SIB_TCP_JFAP_SSL is listening on host *
(IPv4) port 7286.
[...] 0000000a WSChannelFram A   CHFW0019I: The Transport Channel Service has started chain
InboundSecureMessaging.
[...] 0000000a SibMessage    A   [:] CWSIC2001I: Messaging connections are being accepted.
[...] 0000000f SibMessage    I   [TestSIBus:kll6582Node01.server1-TestSIBus] CWSID0016I:
Messaging engine kll6582Node01.server1-TestSIBus is in state Joined.
[...] 00000014 SibMessage    I   [TestSIBus:kll6582Node01.server1-TestSIBus] CWSID0016I:
Messaging engine kll6582Node01.server1-TestSIBus is in state Starting.
:
:
[...] 00000014 InternalGener I   DSRA8203I: Database product name : DBMS:db2j
[...] 00000014 InternalGener I   DSRA8204I: Database product version : 5.1.60.17
[...] 00000014 InternalGener I   DSRA8205I: JDBC driver name  : Cloudscape Embedded JDBC Driver
[...] 00000014 InternalGener I   DSRA8206I: JDBC driver version  : 5.1.60.17
[...] 0000000a WsServerImpl  A   WSVR0001I: Server server1 open for e-business
:
:
[...] 00000014 SibMessage    I   [TestSIBus:kll6582Node01.server1-TestSIBus] CWSIS1538I: The
messaging engine is attempting to obtain an exclusive lock on the data store.
[...] 00000033 SibMessage    I   [TestSIBus:kll6582Node01.server1-TestSIBus] CWSIS1537I: The
messaging engine has acquired an exclusive lock on the data store.
[...] 00000014 SibMessage    I   [TestSIBus:kll6582Node01.server1-TestSIBus] CWSID0016I:
Messaging engine kll6582Node01.server1-TestSIBus is in state Started.
```

Becoming familiar with your own system's normal startup messages will help you to recognize irregularities in the log files.

Note that messaging engines frequently only get to state Started after the “Open for e-business” message in the SystemOut.log.

What to look for

If the messaging engine is stopped, check the log for the following message:

```
Message CWSIS0002E
```

If you find this message go to “Symptom: CWSIS0002E (messaging engine exception)” on page 12.

Otherwise, the messaging engine has not started, but there does not appear to be a relevant message code in the SystemOut.log. In this case, go to “The next step” on page 50.

Symptom: CWSIS0002E (messaging engine exception)

Exceptions received during messaging engine startup are reported using CWSIS0002E. This message might contain the information that tells you the cause of the problem, or you might have to scan back through the log to find more information.

Message CWSIS0002E is often followed by message CWSID0027I, which indicates that a serious error has occurred while restarting the messaging engine, and message CWSID00016I, which indicates that the messaging engine is in a stopped state. These messages are generic in nature and are not indications of the underlying cause.

To begin, focus on the information that is contained in CWSIS0002E. The following are examples of common configuration problems that can cause the messaging engine to fail during startup:

- JNDI name error when accessing the messaging data store.
- Invalid authentication alias information.
- Invalid database tables or tables with invalid data. For example, tables for another messaging engine exist in the data store after the messaging engine has been deleted and recreated.

For information about the messaging data store, see “Example: Incorrect data store for the messaging engine” on page 18.

Example: JNDI error for the messaging data store

In Example 2 on page 13, the original error is contained in the CWSIS0002E message. In this case, it is a configuration error that has occurred while processing the JNDI name of the data source. The message itself contains an

embedded exception code, CWSIS1524E, and looking at the data source name, you can see the error in the name.

Note that you also see CWSID0035E. CWSID0035E is an internal message indicating which Java class and method experienced the problem. This is not directly helpful to you. However, it might be helpful to the IBM service team in the event that you need to contact them.

Example 2 Messaging engine startup failure: JNDI error

```
[...] 00000010 SibMessage    I   [mename] CWSID0016I: Messaging engine <mename> is in state
Joined.
[...] 00000014 SibMessage    I   [mename] CWSID0016I: Messaging engine <mename> is in state
Starting.
[...] 00000014 SibMessage    E   [mename] CWSIS0002E: The messaging engine encountered an
exception while starting. Exception: com.ibm.ws.sib.msgstore.MessageStoreRuntimeException:
CWSIS1524E: Data source, jdbc/com.ibm.ws.sib/xxxxx, not found.
[...] 00000014 SibMessage    E   [mename] CWSID0035E: Messaging engine <mename> cannot be
started; detected error reported during com.ibm.ws.sib.msgstore.impl.MessageStoreImpl start()
[...] 00000014 SibMessage    E   [mename] CWSID0027I: Messaging engine <mename> cannot be
restarted because a serious error has been reported.
[...] 00000014 SibMessage    I   [mename] CWSID0016I: Messaging engine <mename> is in state
Stopped.
[...] 00000014 SibMessage    I   [mename] CWSID0016I: Messaging engine <mename> is in state
Joined.
```

In this specific example, the problem was caused because an incorrect value was specified in the data source JNDI name of the definition. This type of error most likely occurs when you are setting up an application server or cluster as a bus member for the first time.

You can verify or alter the setting from the administrative console by doing the following:

1. Select **Service integration** → **Buses**.
2. Click the bus name.
3. Under Additional Properties, click **Messaging engines**.
4. Click the messaging engine name.
5. Under Additional Properties, click **Data store**.

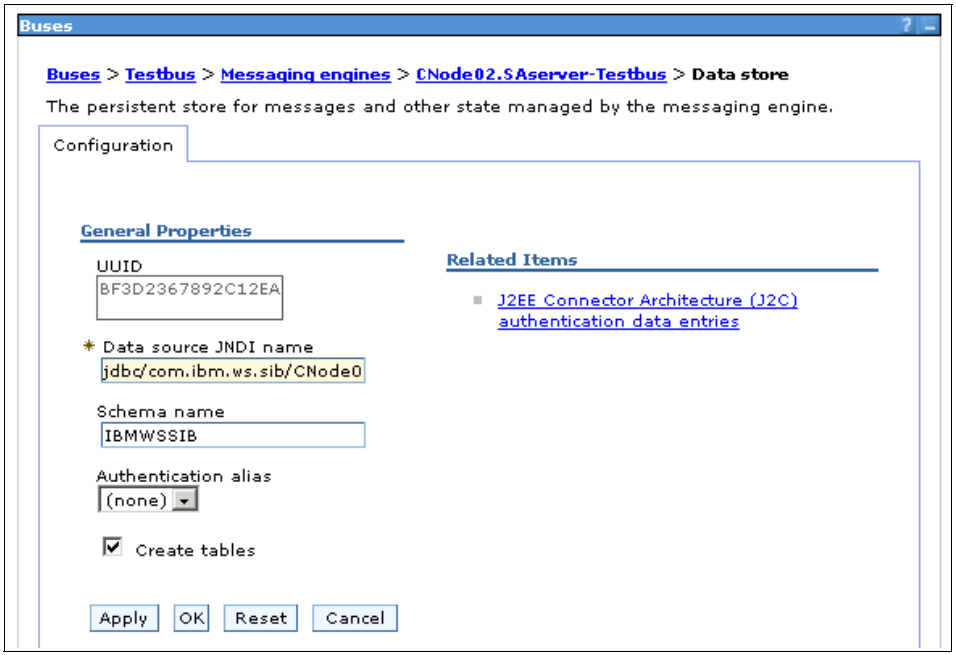

Figure 4 Messaging engine data store properties

The JNDI name specified here has to match the JNDI name that is specified for the data store. For an example of finding the data source for the messaging data store, see “Example: Authentication error for the messaging data store” on page 14.

Example: Authentication error for the messaging data store

A common startup problem is caused when a failure to access the messaging data store occurs due to an authentication problem. In the extract from the SystemOut.log shown in Example 3, you can see the CWSIS0002E message. The message indicates an authentication error has occurred when trying to access the messaging data store. Note that the J2CA0046E and SQL30082N messages provide supporting information.

Example 3 Messaging engine startup failure: authentication error

```
[...] 0000002f SibMessage    I   [NewBus:kll6582Node03.server1-NewBus] CWSID0016I: Messaging
engine kll6582Node03.server1-NewBus is in state Starting.
[...] 0000002f FreePool      E   J2CA0046E: Method createManagedConnectionWithMCWrapper caught
an exception during creation of the ManagedConnection for resource jdbc/JDBCDataSource,
throwing ResourceAllocationException.  Original exception: com.ibm.ws.exception.WsException:
DSRA8100E: Unable to get a PooledConnection from the DataSource. with SQL State : 08001 SQL
Code : -30082
:
```

```
:
Caused by: java.sql.SQLException: SQL30082N  Attempt to establish connection failed with
security reason "24" ("USERNAME AND/OR PASSWORD INVALID").  SQLSTATE=08001
DSRA0010E: SQL State = 08001, Error Code = -30,082DSRA0010E: SQL State = 08001, Error Code =
-30,082
   ... 23 more
Next Linked Exception:
java.sql.SQLException: SQL30082N  Attempt to establish connection failed with security reason
"24" ("USERNAME AND/OR PASSWORD INVALID").  SQLSTATE=08001
DSRA0010E: SQL State = 08001, Error Code = -30,082DSRA0010E: SQL State = 08001, Error Code =
-30,082
:
:
[...] 0000002f SibMessage    E   [NewBus:kll6582Node03.server1-NewBus] CWSIS0002E: The
messaging engine encountered an exception while starting. Exception:
com.ibm.ws.sib.msgstore.PersistenceException: CWSIS1501E: The data source has produced an
unexpected exception: java.sql.SQLException: SQL30082N  Attempt to establish connection failed
with security reason "24" ("USERNAME AND/OR PASSWORD INVALID").  SQLSTATE=08001
DSRA0010E: SQL State = 08001, Error Code = -30,082DSRA0010E: SQL State = 08001, Error Code =
-30,082
[...] 0000002f FreePool      E   J2CA0046E: Method createManagedConnectionWithMCWrapper caught
an exception during creation of the ManagedConnection for resource jdbc/JDBCDataSource,
throwing ResourceAllocationException.  Original exception: com.ibm.ws.exception.WsException:
DSRA8100E: Unable to get a PooledConnection from the DataSource. with SQL State : 08001 SQL
Code : -30082
:
:
Caused by: java.sql.SQLException: SQL30082N  Attempt to establish connection failed with
security reason "24" ("USERNAME AND/OR PASSWORD INVALID").  SQLSTATE=08001
DSRA0010E: SQL State = 08001, Error Code = -30,082DSRA0010E: SQL State = 08001, Error Code =
-30,082
   ... 23 more
Next Linked Exception:
java.sql.SQLException: SQL30082N  Attempt to establish connection failed with security reason
"24" ("USERNAME AND/OR PASSWORD INVALID").  SQLSTATE=08001
DSRA0010E: SQL State = 08001, Error Code = -30,082DSRA0010E: SQL State = 08001, Error Code =
-30,082
:
:
[...] 0000002f SibMessage    E   [NewBus:kll6582Node03.server1-NewBus] CWSID0035E: Messaging
engine kll6582Node03.server1-NewBus cannot be started; detected error reported during
com.ibm.ws.sib.msgstore.impl.MessageStoreImpl start()
[...] 0000002f SibMessage    E   [NewBus:kll6582Node03.server1-NewBus] CWSID0027I: Messaging
engine kll6582Node03.server1-NewBus cannot be restarted because a serious error has been
reported.
[...] 0000002f SibMessage    I   [NewBus:kll6582Node03.server1-NewBus] CWSID0016I: Messaging
engine kll6582Node03.server1-NewBus is in state Stopped.
```

The solution to this problem is to verify the component managed authentication alias. This alias can be specified for the data source definition, or for the messaging engine definition.

Note: The recommended (and more secure) method of specifying the authentication alias is to do it in the messaging engine data store definition. Be aware, that if you specify the authentication alias in this way, testing the data source connection in the administrative console will fail.

If the authentication alias is specified in both places (data source and messaging engine), the alias specified on the messaging engine will be used.

If specified at the messaging engine, you can find the authentication alias specification by performing the following steps in the administrative console:

1. Select **Service integration** → **Buses**.
2. Click the bus name.
3. Under Additional Properties, click **Messaging engines**.
4. Click the messaging engine name.
5. Under Additional Properties, click **Data store**.

If specified at the data source, you can find it by performing the following steps:

1. Select **Resources** → **JDBC™ providers**.
2. Select the proper scope.
3. Select the JDBC provider from the list.
4. Under Additional Properties, click **Data sources**.
5. Select the data source name.

The authentication alias must contain a valid user name and password combination that has appropriate access privileges for the database. Refer to Section 11.8.4 of *WebSphere Application Server V6 System Management and Configuration Handbook*, SG24-6451.

Tip: Defining the data source for the data store at the wrong scope is a common error.

The problem generally lies with an administrator that does not notice that the default scope for creating resources in the administrative console is set to the deployment manager node level (Network Deployment only). While defining the data source at the node level will work, using the deployment manager node will not. The node where the application server is defined needs to be selected.

In general, we recommend that, for a server bus member, you define the data source at the server scope, and for a cluster, that you define it at the cluster scope.

In Figure 5 on page 18, you can see an incorrectly specified authentication alias for this database. The resolution for this problem is to specify a correct authentication alias.

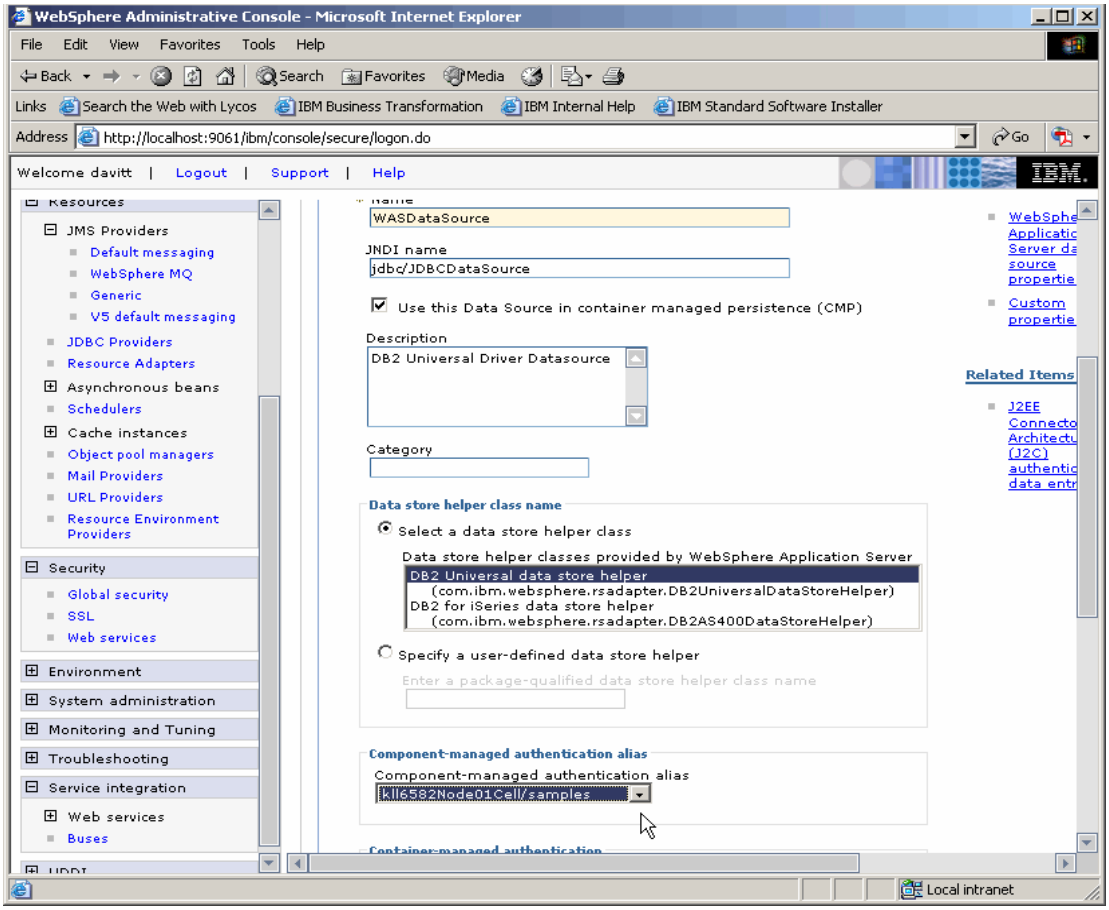

Figure 5 Specifying the authentication alias for the messaging data store

Example: Incorrect data store for the messaging engine

In this example the messaging engine again fails during start up with message code CWSIS0002E. However, in this case, the information that is included in the message does not make it clear what happened. Looking at the extract from SystemOut.log, as shown in Example 4 on page 19, you can see other errors that appear to precipitate that message code.

Immediately after attempting to take an exclusive lock on the data store you can see the CWSIS1535E message code.

Example 4 Messaging engine startup failure: incorrect data store

```
[...] 0000002d InternalGener I   DSRA8203I: Database product name : DB2/NT
[...] 0000002d InternalGener I   DSRA8204I: Database product version : SQL08020
[...] 0000002d InternalGener I   DSRA8205I: JDBC driver name  : IBM DB2 JDBC Universal Driver
Architecture
[...] 0000002d InternalGener I   DSRA8206I: JDBC driver version  : 2.3.63
[...] 0000002d WSRdbDataSour I   DSRA8208I: JDBC driver type  : 2
[...] 0000002d SibMessage    I   [NewBus:kll6582Node01.server1-NewBus] CWSIS1538I: The
messaging engine is attempting to obtain an exclusive lock on the data store.
[...] 0000002e SibMessage    I   [NewBus:kll6582Node01.server1-NewBus] CWSIS1535E: The
messaging engine's unique id does not match that found in the data store.
ME_UUID=884177EB370F543C, ME_UUID(DB)=7E9CB093BBD8794E
[...] 0000002f SibMessage    I   [NewBus:kll6582Node01.server1-NewBus] CWSIS1519E: Messaging
engine kll6582Node01.server1-NewBus cannot obtain the lock on its data store, which ensures it
has exclusive access to the data.
[...] 0000002d SibMessage    E   [NewBus:kll6582Node01.server1-NewBus] CWSIS0002E: The
messaging engine encountered an exception while starting. Exception:
com.ibm.ws.sib.msgstore.PersistenceException: CWSIS1501E: The data source has produced an
unexpected exception: com.ibm.ws.sib.msgstore.persistence.DatasourceWrapperStateException: New
connections cannot be provided because the persistence layer has been stopped
[...] 0000002d SibMessage    E   [NewBus:kll6582Node01.server1-NewBus] CWSID0035E: Messaging
engine kll6582Node01.server1-NewBus cannot be started; detected error reported during
com.ibm.ws.sib.msgstore.impl.MessageStoreImpl start()
[...] 0000002d SibMessage    E   [NewBus:kll6582Node01.server1-NewBus] CWSID0027I: Messaging
engine kll6582Node01.server1-NewBus cannot be restarted because a serious error has been
reported.
[...] 0000002d SibMessage    I   [NewBus:kll6582Node01.server1-NewBus] CWSID0016I: Messaging
engine kll6582Node01.server1-NewBus is in state Stopped.
[...] 0000002d SibMessage    I   [NewBus:kll6582Node01.server1-NewBus] CWSID0016I: Messaging
engine kll6582Node01.server1-NewBus is in state Joined.
[...] 0000002d SibMessage    E   [NewBus:kll6582Node01.server1-NewBus] CWSID0039E:
HAManager-initiated activation has failed, messaging engine kll6582Node01.server1-NewBus will
be disabled.
```

Each messaging object has a unique identifier associated with it that is called a *UUID*. When a messaging engine is created and first accesses the database, the messaging engine registers its UUID in the database SIBOWNER table. When a database has been marked in such a way, it cannot be used by another messaging engine.

The error shown in Example 4 was caused by the WebSphere administrator modifying the database name in the messaging data store data source to a name that is already in use by another messaging engine. In this case, the solution is to correct the data source configuration.

Other common scenarios for this type of error include:

- An administrator deletes an application server or removes it from the bus, and then tries to use the same database when adding a new server to the bus.
- An administrator creates a bus, deletes it, and then creates a new bus with the same name and configuration, especially if using the default data store (IBM Cloudscape™). This often occurs during testing or initial setup.

The solution for these cases is to drop and recreate the database (or tables). If you are using Cloudscape, a quick fix for this is to stop the server, delete the Cloudscape database folder found at *<WAS_install_root>*/profiles/*<profile_name>*/databases/com.ibm.ws.sib/*<messaging_engine_name>*, and then restart the server.

You should only do this if you do not need to keep any persistent messages currently stored in the data store.

Example: messaging data store not available J2CS0046E

This example shows what happens when you start an application server and the database that is associated with the messaging engine data source is not available. The error that is reported by the messaging engine is CWSIS0002E. However, from the abbreviated extract from SystemOut.log shown in Example 5, you can see there are multiple messages that are reported from the J2C component.

Example 5 Messaging engine startup failure: data store is not available

```
[7/8/05 10:49:21:283 EDT] 00000028 SibMessage    I   [NewBus:kll6582Node03.server1-NewBus]
CWSID0016I: Messaging engine kll6582Node03.server1-NewBus is in state Joined.
[7/8/05 10:49:21:623 EDT] 0000002a SibMessage    I   [NewBus:kll6582Node03.server1-NewBus]
CWSID0016I: Messaging engine kll6582Node03.server1-NewBus is in state Starting.
[7/8/05 10:49:26:390 EDT] 0000002a FreePool      E   J2CA0046E: Method
createManagedConnectionWithMCWrapper caught an exception during creation of the
ManagedConnection for resource jdbc/JDBCDataSource, throwing ResourceAllocationException.
Original exception: com.ibm.ws.exception.WsException: DSRA8100E: Unable to get a
PooledConnection from the DataSource. with SQL State : 57019 SQL Code : -1032
:
:
Caused by: java.sql.SQLException: SQL1032N  No start database manager command was issued.
SQLSTATE=57019
DSRA0010E: SQL State = 57019, Error Code = -1,032DSRA0010E: SQL State = 57019, Error Code =
-1,032
   ... 28 more
Next Linked Exception:
java.sql.SQLException: SQL1032N  No start database manager command was issued.  SQLSTATE=57019
DSRA0010E: SQL State = 57019, Error Code = -1,032DSRA0010E: SQL State = 57019, Error Code =
-1,032
:
```

```
:
[7/8/05 10:49:26:631 EDT] 0000002a SibMessage    E   [NewBus:kll6582Node03.server1-NewBus]
CWSIS0002E: The messaging engine encountered an exception while starting. Exception:
com.ibm.ws.sib.msgstore.PersistenceException: CWSIS1501E: The data source has produced an
unexpected exception: java.sql.SQLException: SQL1032N  No start database manager command was
issued.  SQLSTATE=57019
DSRA0010E: SQL State = 57019, Error Code = -1,032DSRA0010E: SQL State = 57019, Error Code =
-1,032
[7/8/05 10:49:26:651 EDT] 0000002a FreePool      E   J2CA0046E: Method
createManagedConnectionWithMCWrapper caught an exception during creation of the
ManagedConnection for resource jdbc/JDBCDataSource, throwing ResourceAllocationException.
Original exception: com.ibm.ws.exception.WsException: DSRA8100E: Unable to get a
PooledConnection from the DataSource. with SQL State : 57019 SQL Code : -1032
:
:
Caused by: java.sql.SQLException: SQL1032N  No start database manager command was issued.
SQLSTATE=57019
DSRA0010E: SQL State = 57019, Error Code = -1,032DSRA0010E: SQL State = 57019, Error Code =
-1,032
```

To resolve this problem:

1. Ensure that the database is started correctly.
2. Attempt to restart the messaging engine.
3. If the messaging engine will not start, stop and restart the application server.
4. Test connectivity to the messaging data store database. (Note that this will not work if the database is secured and the authentication alias was defined at the messaging engine data store level.)

   To test using the administrative console:

   a. Under Resources, select J**DBC providers** and click the JDBC provider name.

   b. Under Additional Properties, select **Data sources**.

   c. Select the box to the left of the data source.

   d. Select **Test Connectivity**.

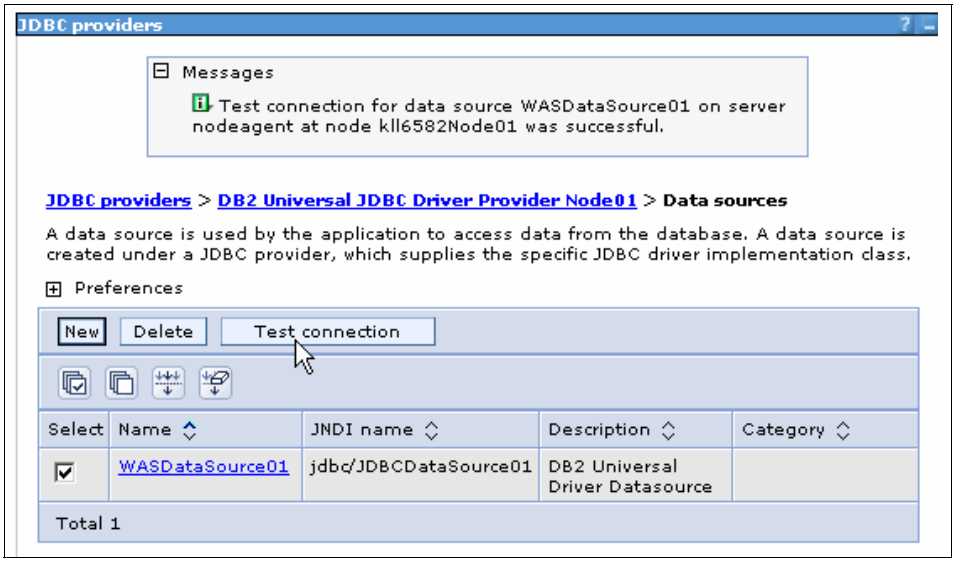

Figure 6 Test the data source connection

Example: Data store tables missing using DB2 for z/OS

It is possible to configure WebSphere Application Server on a distributed system to use IBM DB2® for z/OS® as the messaging data store. If so, you might see the message shown in Example 6 during the first attempt to access the data store.

Example 6 Messaging engine startup failure: DB2 for z/OS data store tables missing

```
Exception: com.ibm.ws.sib.msgstore.PersistenceException: CWSIS1501E: The data source has
produced an unexpected exception:java.lang.IllegalStateException: CWSIS1523E: Dynamic
allocation of database objects in DB2 for z/OS is not allowed.
com.ibm.ws.sib.utils.ras.SibMessage
```

When using a DB2 for z/OS messaging data store, the tables are not created dynamically. These tables must be created manually before starting the messaging engine.

The rsibDDLgenerator utility assists in resolving this problem by producing the syntactically correct DDL statements. You can find information about this utility in the WebSphere Information Center:

http://publib.boulder.ibm.com/infocenter/ws60help/index.jsp?topic=/com.ibm.websphere.pmc.zseries.doc\ref\rjm0630_.html

You can redirect the output from this command into a file to submit at to DB2.

Finally, you must ensure that you deselect **Create tables** when specifying the messaging engine data store, as shown in Figure 7.

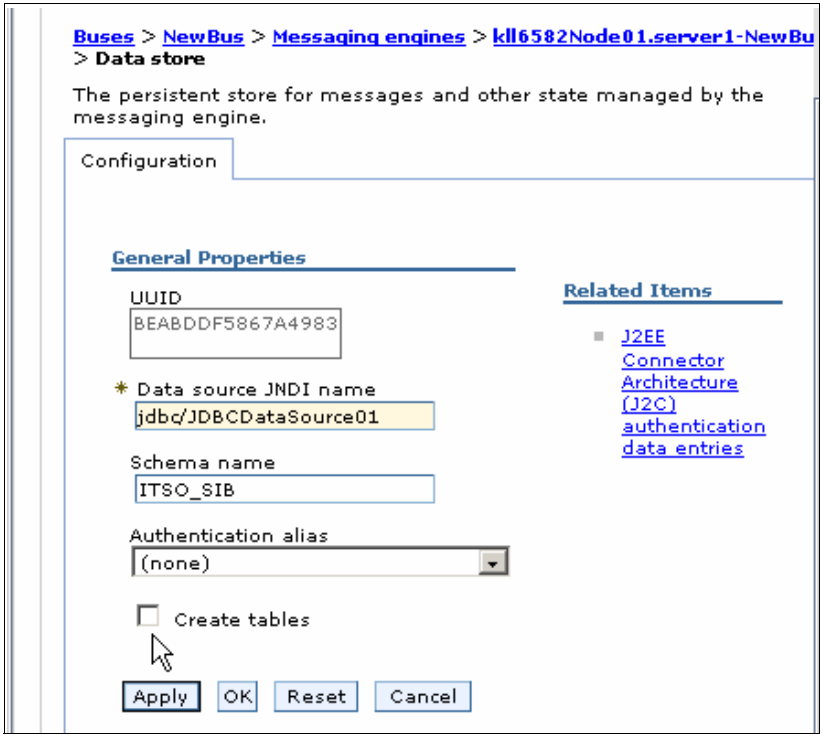

Figure 7 Deselect the Create tables field

Example: problem is not clear, message CWSID0003E

In Example 7, another error has occurred with starting the messaging engine.

Example 7 Messaging engine startup failure: no clear indicator of problem

```
[...] 0000001b SibMessage    I   [mename] CWSID0016I: Messaging engine <mename> is in state Joined.
[...] 0000001c SibMessage    I   [mename] CWSID0016I: Messaging engine <mename> is in state Starting.
[...] 0000001c SibMessage    I   [mename] CWSIS9999E: Attempting to obtain an exclusive lock on the data
store.
[...] 00000034 SibMessage    I   [mename] CWSIS9999E: Obtained an exclusive lock on the data store
[...] 0000001c SibMessage    E   [mename] CWSID0003E: An internal error occurred; reason: Messaging engine
<mename> cannot be started; detected error reported during com.ibm.ws.sib.msgstore.impl.MessageStoreImpl
start()
[...] 0000001c SibMessage    E   [mename] CWSID0027I: Messaging engine <mename> cannot be restarted
because a serious error has been reported.
[...] 0000001c SibMessage    I   [mename] CWSID0016I: Messaging engine <mename> is in state Stopped.
[...] 0000001c SibMessage    I   [mename] CWSID0016I: Messaging engine <mename> is in state Joined.
```

At first sight, this appears to be similar to the error seen in “Example: JNDI error for the messaging data store” on page 12. However, looking prior to that message in the example log, you can also see a message CWSID0003E with further details that indicates a problem with a Java class, as follows:

```
com.ibm.ws.sib.msgstore.impl.MessageStoreImpl start()
```

In this example, there is no clear indication of the exact nature of the problem. So, at this point, it is appropriate to gather suitable documentation and contact the IBM service organization. To determine the documentation that you need to collect, see “The next step” on page 50. Where possible, consider obtaining trace information that might be useful to the service team (see Table 2 on page 52).

Message flow problems

A variety of things can cause message flow problems. In fact, a failure in any component will stop the flow of messages. This section addresses the following problem symptoms:

- Symptom: Message consumer fails, possible CWSIA0144E
- Problems with messages flowing between the bus and WebSphere MQ:
  - Symptom: Messages disappear when using an MQ link
  - Symptom: CWSIC3098I receiving msgs from WebSphere MQ
  - Symptom: Messages not sent across MQ link
- Symptom: Unrecoverable error from data source DSRA0080E
- Problems with message-driven beans:
  - Symptom: A message-driven bean has not started
  - Symptom: MDB is not receiving messages
  - Symptom: MDB is causing CNTR0020E message
  - Symptom: An MDB fails and is invoked in an infinite loop
  - Symptom: Message not restored to queue after MDB failure
- Problems with mediation:
  - Symptom: Mediation of a destination not working
  - Symptom: Mediation fails with CWSIZ0002E, messages disappear
  - Symptom: Mediation implemented but messages disappear

Symptom: Message consumer fails, possible CWSIA0144E

A common error that can occur is in multi-messaging engine environment where a message consumer appears to be failing to receive messages. This can occur when the messaging engine where the queue has its queue point has stopped. Figure 8 on page 25 illustrates a message producer and a message consumer that are connected to different messaging engines but that use the same destination.

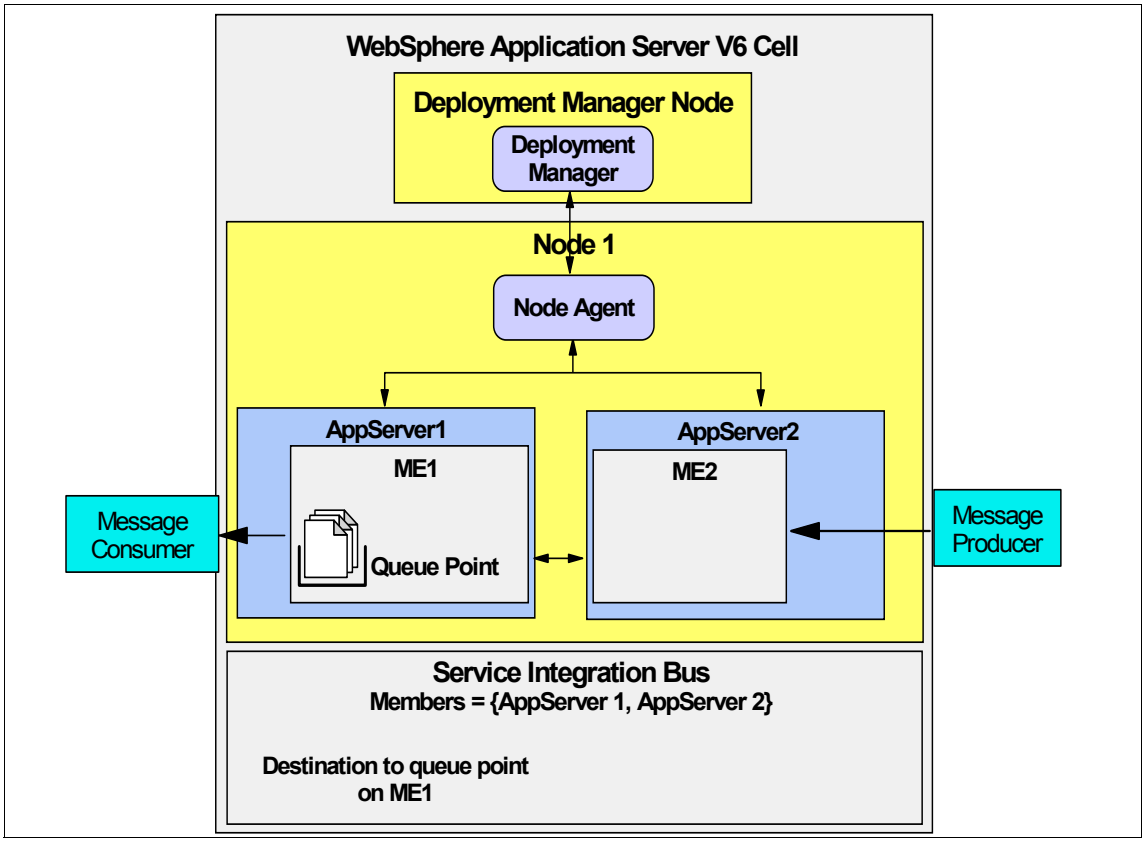

Figure 8 Multiple messaging engines using the same destination

In this problem scenario, the message consumer attempts to get messages from a destination and gets an JMSException code (Example 8). A message producer attached to a second messaging engine sending messages to the same destination appears unaffected.

Example 8 shows the exception code caught by the application.

Example 8 JMSException when a consumer tries to get a message

```
JMSException caught
Stack Trace:
javax.jms.JMSException: CWSIA0144E: An exception was received during the call to the method
createBrowserSession: com.ibm.websphere.sib.exception.SIResourceException: CWSIC8007E: An
exception was caught from the remote server with Probe Id 3-023-0003. Exception: CWSIP0002E: An
internal messaging error occurred in com.ibm.ws.sib.processor.impl.BrowserSessionImpl, 10,
com.ibm.ws.sib.msgstore.MessageStoreException: CWSIP0532E: A timeout occurred while remotely
browsing destination cellQ2...
```

```
   at
com.ibm.ws.sib.api.jms.impl.JmsQueueBrowserImpl.instantiateBrowser(JmsQueueBrowserImpl.java:575
)
   at com.ibm.ws.sib.api.jms.impl.JmsQueueBrowserImpl.<init>(JmsQueueBrowserImpl.java:183)
   at com.ibm.ws.sib.api.jms.impl.JmsSessionImpl.createBrowser(JmsSessionImpl.java:1489)
   at com.ibm.ws.sib.api.jms.impl.JmsSessionImpl.createBrowser(JmsSessionImpl.java:1439)
```

Example 9 shows the stack trace seen from the same error.

Example 9 Stack trace

```
[...] 00000012  E UOW=null source=com.ibm.ws.sib.utils.ras.SibMessage org=IBM prod=WebSphere
component=Application Server thread=[JFAP TCP Channel : 6]
[:] CWSIC1010E: An internal error occurred. A protocol error has occurred. Unexpected data was
received from the server (kll6582.itso.ral.ibm.com:7276 - BootstrapBasicMessaging). The data ID
was 223
(0xDF).
```

In this case, the messaging engine that holds the queue point for the destination has stopped. This should be relatively easy to see in the administrative console by looking at the status of the messaging engine associated with the failing server, as shown in Figure 9.

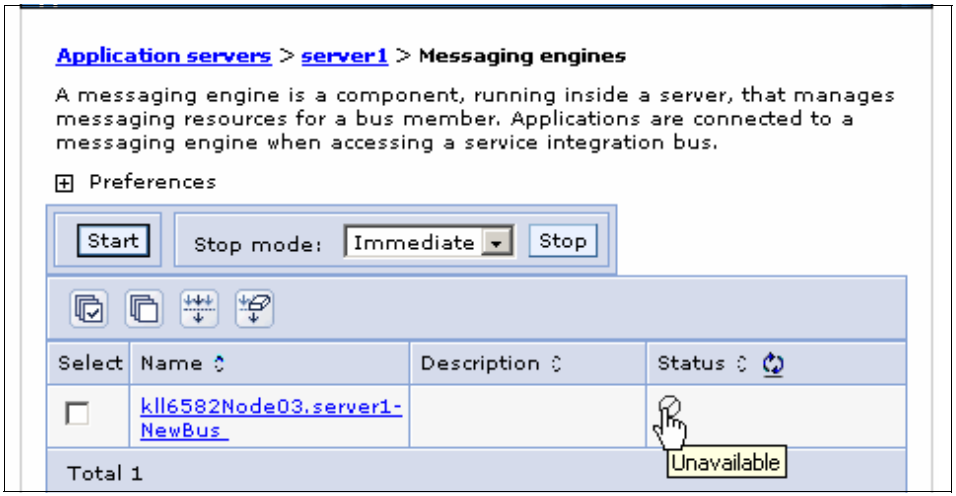

Figure 9 Check the status of the messaging engine

Fortunately, the resolution for this problem is relatively easy. Ensure that the messaging engine that is holding the queue point for the failing destination has been started.

Symptom: Messages disappear when using an MQ link

In this problem scenario, you have defined a connection to WebSphere MQ as a foreign bus. You have two application servers and two messaging engines and a foreign bus and MQLink to connect to WebSphere MQ. You are expecting messages to flow to and from the MQ queue manager but this is not happening.

See Figure 10 as an example of the topology.

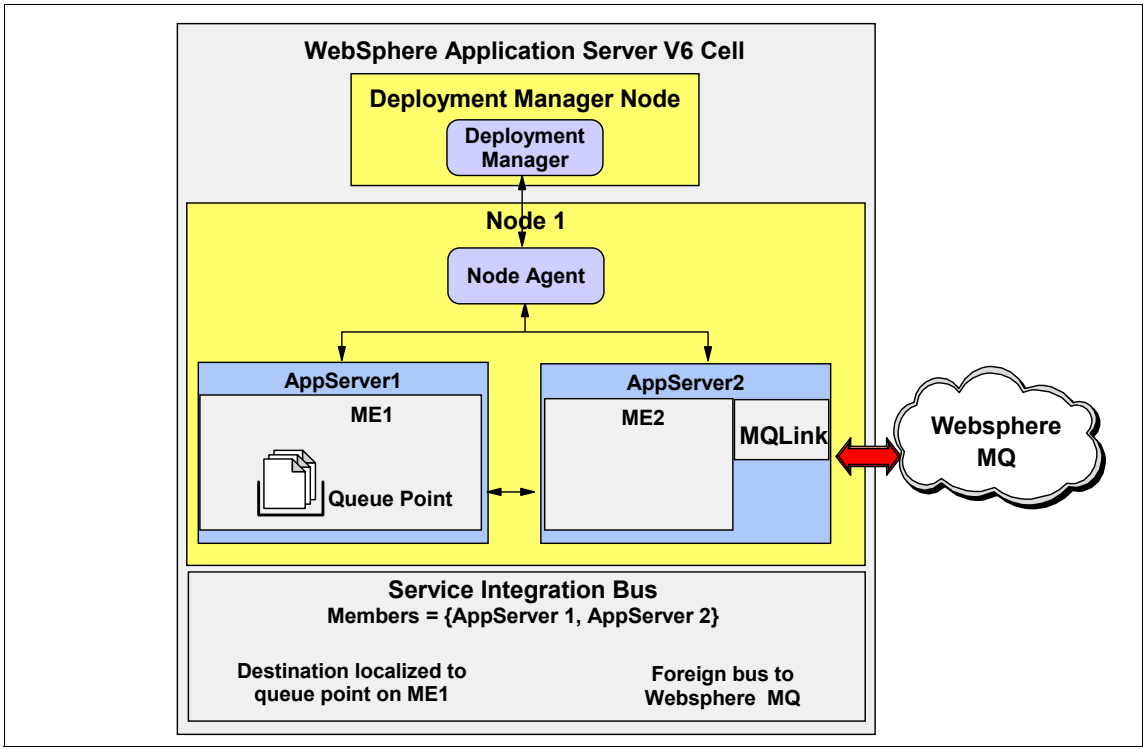

Figure 10 Topology of bus connection to WebSphere MQ

Note: Be careful when defining the WebSphere MQ link definitions. The Queue manager name field can be confusing. This defines a virtual name given to the current bus so that the remote queue manager can address messages to it.

In this scenario, the basic cause of the problem is that ME1 was restarted. The message producer sends messages using WebSphere MQ to the application on Appserver1. The message producer, WebSphere MQ, and the bus appear to be working fine. However, the messages are not arriving at ME1. Using the

administrative console, you view the SYSTEM.Exception.Queue, but there are no messages. They have apparently disappeared.

The solution to this problem is to stop and restart AppServer2. When this occurs, any messages that are waiting to be transmitted to the target system are sent. After restarting the AppServer2, you can view the destination queue point using the administrative console. Looking at Figure 11, you can see that there is a considerable discrepancy between the message time stamp and the current messaging arrival time. This discrepancy was caused while the messages were waiting for AppServer2 to be restarted.

Figure 11 Checking the time stamp of a message

This problem occurs in the early versions of the code and should be fixed in a future version.

Symptom: CWSIC3098I receiving msgs from WebSphere MQ

In this problem scenario, messages are not being delivered to the application server from WebSphere MQ. No error is seen from the WebSphere MQ side.

However, a message code is seen in the application server SystemOut.log (Example 10).

Example 10 Errors receiving messages from WebSphere MQ

```
[...] 00000018 SibMessage    I   [:] CWSIC3098I: While receiving message from queue manager
QM_MQ53_QMGR down WebSphere MQ link MyMQLink one or more messages were put to the exception
destination.
[...] 00000018 SibMessage    W   [:] CWSIP0291W: An attempt to send a message to exception
destination of REMOTE.SIB.Q on messaging engine kll6582Node02.server1-NewBus failed.
```

In this case, the remote queue manager is attempting to send messages to a destination called REMOTE.SIB.Q. Unfortunately, no such destination exists on the bus, and the messages are routed to the SYSTEM.Exception.Destination queue for this messaging engine.

The resolution is to either correct the WebSphere MQ definitions to reference a valid destination on the bus, or to create a destination and queue point that match that which is used by WebSphere MQ. Another option is to create an alias destination in the bus with that name to redirect the messages to the proper bus destination.

Symptom: Messages not sent across MQ link

In a topology similar to that shown in Figure 10 on page 27, problems can arise when sending messages from a bus to a WebSphere MQ provider. One such case would be when the network connection between the two systems is lost.

Unfortunately, there are no messages in the SystemOut.log to indicate this event, and the WebSphere administrative console does not show any change of status for the MQ link sender channel (see Figure 12 on page 30).

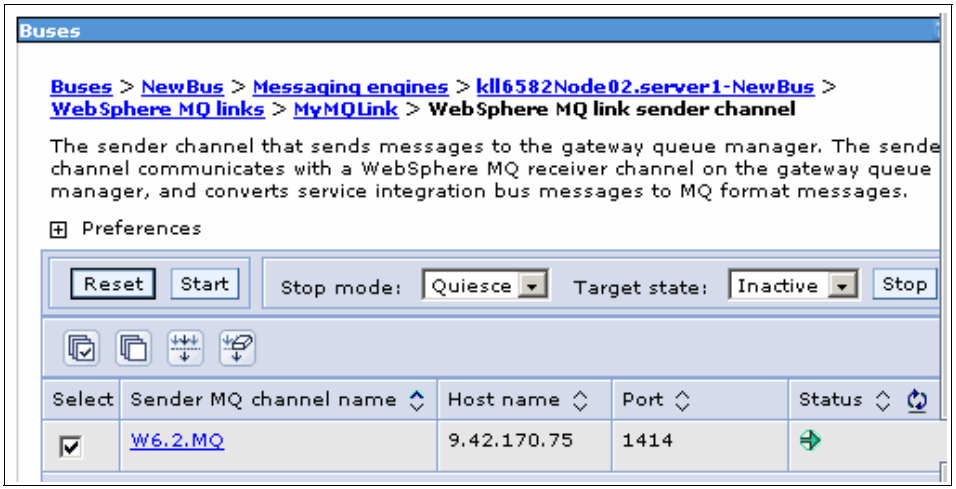

Figure 12 Sender MQ channel status

It is possible for you to check the link from WebSphere MQ by viewing the status of the MQ channel.

The resolution from the WebSphere Application Server perspective is to:

1. Correct the network or link error that is causing the problem (this might involve obtaining TCP/IP documentation tom see what causes the network to fail).
2. Stop and start the messaging engine that uses the MQ link (it is not sufficient to stop and start the sender channel or the MQ link).

Symptom: Unrecoverable error from data source (DSRA0080E)

The messaging engines might use one of several different database providers for the storage of message data. This dependence on the database can cause a problem if there is an unrecoverable error from the database. For example:

- Network access error
- Catastrophic media error
- Accidental database closure

Example 11 on page 31 shows an extract from SystemOut.log that shows such an error.

Initially, the error DSRA0080E is reported from the message data store component followed by J2CA0056I. Each of these messages is repeated several times. This sequence is then followed by message codes CWSIS1546I and

CWSIS1538I, which indicate that the lock on the database has been lost and repeated attempts are being made to recover the lock.

Finally the application server JVM is ended to avoid any possible data loss. See message HMGR0130I. The application server is terminated in a orderly way.

Example 11 Message data store errors

```
:
:
[...] 00000038 WSRdbManagedC W   DSRA0080E: An exception was received by the Data Store
Adapter. See original exception message: com.ibm.db2.jcc.b.SqlException: invalid operation:
connection closed
   at com.ibm.db2.jcc.b.o.yb(o.java:3433)
   at com.ibm.db2.jcc.b.o.getAutoCommit(o.java:1006)
   at com.ibm.db2.jcc.b.wb.getAutoCommit(wb.java:154)
   at
com.ibm.ws.rsadapter.spi.WSRdbManagedConnectionImpl.introspectSelf(WSRdbManagedConnectionImpl.j
ava:1108)
   at com.ibm.ws.rsadapter.FFDCLogger.introspect(FFDCLogger.java:169)
   at com.ibm.ws.rsadapter.jdbc.WSJdbcConnection.introspectSelf(WSJdbcConnection.java:1511)
   at com.ibm.ws.rsadapter.jdbc.WSJdbcObject.introspectSelf(WSJdbcObject.java:355)
   at
com.ibm.ws.ffdc.IntrospectionLevelMember.getNextMembers(IntrospectionLevelMember.java:451)
   at com.ibm.ws.ffdc.IntrospectionLevel.getNextLevel(IntrospectionLevel.java:181)
   at com.ibm.ws.ffdc.ObjectIntrospectorImpl.dumpContents(ObjectIntrospectorImpl.java:67)
   at com.ibm.ws.ffdc.ObjectIntrospectorImpl.dumpContents(ObjectIntrospectorImpl.java:51)
   at com.ibm.ws.ffdc.IncidentStreamImpl.introspectAndWrite(IncidentStreamImpl.java:427)
   at com.ibm.ws.ffdc.IncidentStreamImpl.introspectAndWriteLine(IncidentStreamImpl.java:663)
   at com.ibm.ws.ffdc.DiagnosticEngine.dumpObjectAndStack(DiagnosticEngine.java:311)
   at com.ibm.ws.ffdc.DiagnosticEngine.processIncident(DiagnosticEngine.java:152)
   at com.ibm.ws.ffdc.FFDCFilter.filterEngine(FFDCFilter.java(Compiled Code))
   at com.ibm.ws.ffdc.FFDCFilter.processException(FFDCFilter.java(Inlined Compiled Code))
   at
com.ibm.ws.rsadapter.jdbc.WSJdbcPreparedStatement.executeQuery(WSJdbcPreparedStatement.java(Com
piled Code))
   at
com.ibm.ws.sib.msgstore.persistence.impl.MEOwnerTable.readOwningME(MEOwnerTable.java(Compiled
Code))
   at
com.ibm.ws.sib.msgstore.persistence.lock.DBLockingThread.waitAndRefreshLock(DBLockingThread.jav
a(Compiled Code))
   at
com.ibm.ws.sib.msgstore.persistence.lock.DBLockingThread.run(DBLockingThread.java(Compiled
Code))
.
:
[...] 00000038 ConnectionEve A   J2CA0056I: The Connection Manager received a fatal connection
error from the Resource Adaptor for resource jdbc/JDBCDataSource.  The exception which was
```

```
received is com.ibm.websphere.ce.cm.StaleConnectionException: A communication error has been
detected. Communication protocol being used: UWReply.fill(). Communication API being used:
recvBuff(). Location where the error was detected: ReasonCode=71. Communication function
detecting the error: *. Protocol specific error codes(s) TCP/IP SOCKETS
[...] 00000038 MCWrapper     E   J2CA0081E: Method cleanup failed while trying to execute
method cleanup on ManagedConnection WSRdbManagedConnectionImpl@5915eba0 from resource No longer
available. Caught exception: com.ibm.ws.exception.WsException: DSRA1130E: A fatal connection
error occurred on another connection while this connection was active. This connection cannot
be reset to a usable state.
   at
com.ibm.ws.rsadapter.exceptions.DataStoreAdapterException.<init>(DataStoreAdapterException.java
:226)
   at
com.ibm.ws.rsadapter.exceptions.DataStoreAdapterException.<init>(DataStoreAdapterException.java
:177)
   at com.ibm.ws.rsadapter.AdapterUtil.createDataStoreAdapterException(AdapterUtil.java:232)
   at
com.ibm.ws.rsadapter.spi.WSRdbManagedConnectionImpl.cleanup(WSRdbManagedConnectionImpl.java:301
2)
   at com.ibm.ejs.j2c.MCWrapper.cleanup(MCWrapper.java:1343)
   at com.ibm.ejs.j2c.poolmanager.FreePool.cleanupAndDestroyMCWrapper(FreePool.java:627)
   at
com.ibm.ejs.j2c.poolmanager.FreePool.removeCleanupAndDestroyAllFreeConnections(FreePool.java:18
17)
   at com.ibm.ejs.j2c.poolmanager.PoolManager.fatalErrorNotification(PoolManager.java:1082)
   at com.ibm.ejs.j2c.MCWrapper.connectionErrorOccurred(MCWrapper.java:1981)
   at
com.ibm.ejs.j2c.ConnectionEventListener.connectionErrorOccurred(ConnectionEventListener.java:31
3)
   at
com.ibm.ws.rsadapter.spi.WSRdbManagedConnectionImpl.processConnectionErrorOccurredEvent(WSRdbMa
nagedConnectionImpl.java:1989)
   at
com.ibm.ws.rsadapter.jdbc.WSJdbcConnection.fireConnectionErrorEvent(WSJdbcConnection.java:1253)
   at com.ibm.ws.rsadapter.jdbc.WSJdbcUtil.mapException(WSJdbcUtil.java:874)
   at
com.ibm.ws.rsadapter.jdbc.WSJdbcPreparedStatement.executeQuery(WSJdbcPreparedStatement.java(Com
piled Code))
   at
com.ibm.ws.sib.msgstore.persistence.impl.MEOwnerTable.readOwningME(MEOwnerTable.java(Compiled
Code))
   at
com.ibm.ws.sib.msgstore.persistence.lock.DBLockingThread.waitAndRefreshLock(DBLockingThread.jav
a(Compiled Code))
   at
com.ibm.ws.sib.msgstore.persistence.lock.DBLockingThread.run(DBLockingThread.java(Compiled
Code))
:
:
```

```
[...] 00000037 SibMessage    I   [NewBus:kll6582Node03.server1-NewBus] CWSIS1546I: The
messaging engine, ME_UUID=7E9CB093BBD8794E, INC_UUID=0ee96bb71181170a, has lost an existing
lock or failed to gain an initial lock on the data store.
[...] 00000037 SibMessage    I   [NewBus:kll6582Node03.server1-NewBus] CWSIS1546I: The
messaging engine, ME_UUID=7E9CB093BBD8794E, INC_UUID=0ee96bb71181170a, has lost an existing
lock or failed to gain an initial lock on the data store.
[...] 00000037 SibMessage    I   [NewBus:kll6582Node03.server1-NewBus] CWSIS1538I: The
messaging engine, ME_UUID=7E9CB093BBD8794E, INC_UUID=0ee96bb71181170a, is attempting to obtain
an exclusive lock on the data store.
:
:
[...] 00000037 SibMessage    I   [NewBus:kll6582Node03.server1-NewBus] CWSIS1538I: The
messaging engine, ME_UUID=7E9CB093BBD8794E, INC_UUID=0ee96bb71181170a, is attempting to obtain
an exclusive lock on the data store.
[...] 00000032 SibMessage    E   [NewBus:kll6582Node03.server1-NewBus] CWSID0046E: Messaging
engine kll6582Node03.server1-NewBus detected an error and cannot continue to run in this
server.
[...] 00000032 HAGroupImpl   I   HMGR0130I: The local member of group
WSAF_SIB_BUS=NewBus,WSAF_SIB_MESSAGING_ENGINE=kll6582Node03.server1-NewBus,type=WSAF_SIB has
indicated that is it not alive. The JVM will be terminated.
[...] 00000032 SystemOut     O Panic:component requested panic from isAlive
:
:
[...] 000009fa ServerCollabo A   WSVR0024I: Server server1 stopped
```

Naturally, such errors should be rare, and the resolution is to investigate and correct the error with the database.

Problems with message-driven beans

Message-driven beans (MDBs) are EJBs that act as message consumers. When a message producer is putting messages on a queue, but those messages are not taken off the queue by the MDB, you will see the messages collect on the queue.

Symptom: A message-driven bean has not started

When an MDB does not start, verify that the configuration is correct. For a sample of how to set up and configure an MDB, see:

http://www.ibm.com/developerworks/websphere/techjournal/0504_reinitz/0504_reinitz.html

One indication that the MDB is correctly configured and that the messaging engine has started is the message CWSIV0764I in the SystemOut.log.

Example 12 MDB startup messages in SystemOut.log

```
[...] 00000033 SibMessage    I   [NewBus:kll6582Node03.server1-NewBus] CWSIS1537I: The
messaging engine, ME_UUID=7E9CB093BBD8794E, INC_UUID=52e3ef4111668436, has acquired an
exclusive lock on the data store.
[...] 0000002e SibMessage    I   [NewBus:kll6582Node03.server1-NewBus] CWSIP0212I: messaging
engine kll6582Node03.server1-NewBus on bus NewBus is starting to reconcile the WCCM destination
and link configuration.
[...] 0000002e SibMessage    I   [NewBus:kll6582Node03.server1-NewBus] CWSIP0213I: messaging
engine kll6582Node03.server1-NewBus on bus NewBus has finished reconciling the WCCM destination
and link configuration.
[...] 0000002e SibMessage    I   [NewBus:kll6582Node03.server1-NewBus] CWSIV0764I: A consumer
has been created for a message-driven bean against destination [destinationname] on bus NewBus
following the activation of messaging engine kll6582Node03.server1-NewBus.
```

Note that the CWSIV0764I message is only seen if the messaging engine starts after the application that contains the MDB (there will have been a corresponding CWSIV0759W when the application started). If the messaging engine is started before the application, you will not see the informational message even if there are no configuration problems.

Symptom: MDB is not receiving messages

By default, no SystemOut.log messages are produced when an MDB handles a received message. With this in mind, it is advisable that you add some form of logging to an application so that you can recognize that messages are being processed successfully.

If the MDB fails to handle a message, the messaging engine attempts to redeliver the message based on the destination properties of the queue. These can be set or displayed by selecting **Service Integration** → **Buses**. Click the bus name to display its properties. Then, click **Destinations** to display a list of queue and topic destinations. Finally, click the queue destination. See Figure 13 on page 35.

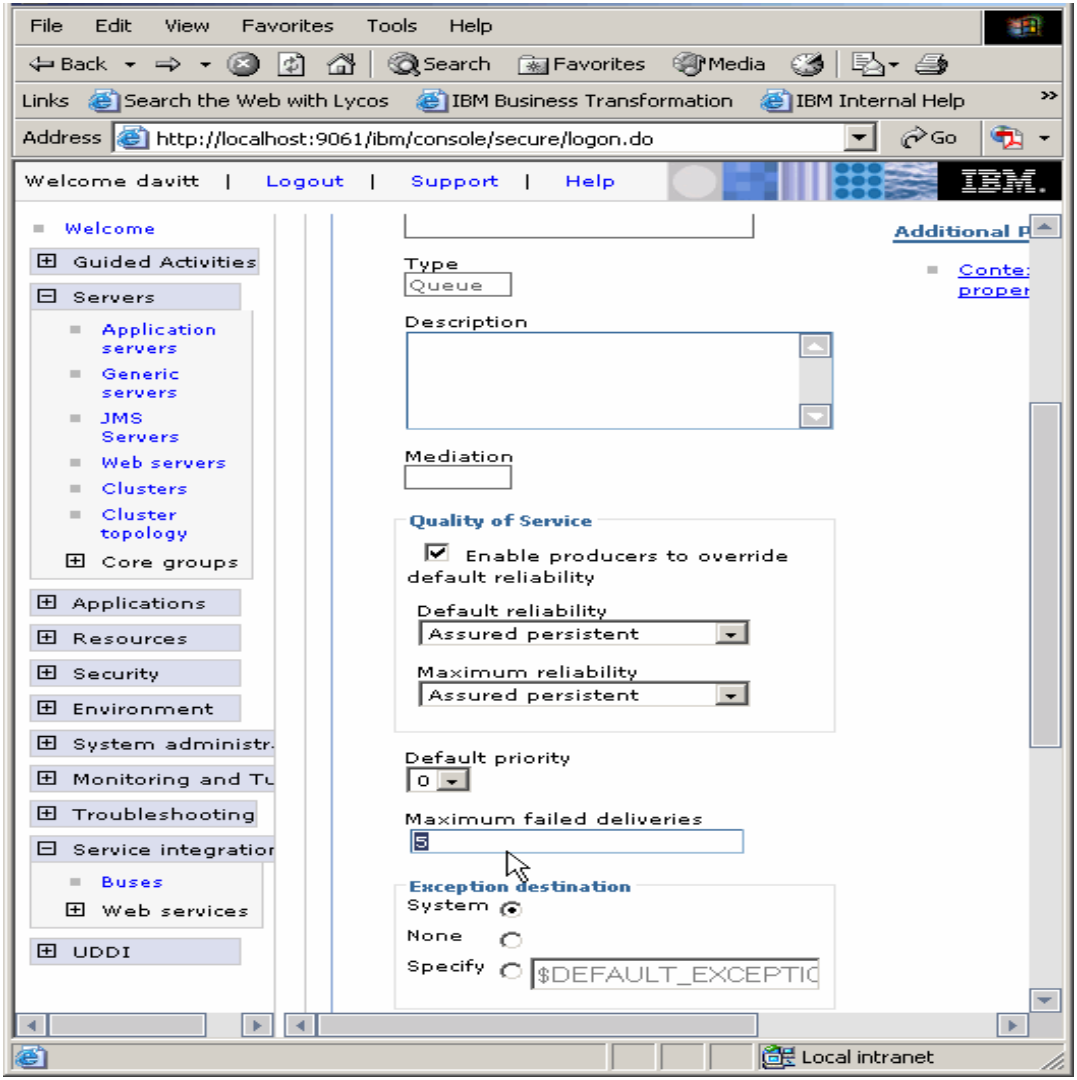

Figure 13 Queue destination properties

If the message cannot be processed, it is redelivered based on the *maximum failed deliveries* value.

If this value is exceeded, the messaging engine attempts to deliver the message to the *exception destination* address. By default, this address uses a destination that is generic for that messaging engine. The format of the exception destination is as follows:

`SYSTEM.Exception.Destination.<messaging_engine_name>`

If required, you can define your own exception destination specific for the queue that is processing the MDB messages.

The resolution in this case is to make sure the destination definition is correct. If it is, check the exception destination queue in case the message has been delivered there.

Symptom: MDB is causing CNTR0020E message

When an MDB experiences an unexpected exception code this is recognized by the application server. The exception is captured and a message code added to SystemOut.log (see Example 13).

Example 13 MDB exception

```
[...] 00000066 ExceptionUtil E   CNTR0020E: EJB threw an unexpected (non-declared) exception
during invocation of method "onMessage" on bean
"BeanId(PackageReceivedEAR#PackageReceived.jar#PackageReceived, null)". Exception data:
java.lang.RuntimeException: MDB error
   at receiver.PackageReceivedBean.onMessage(Unknown Source)
   at
com.ibm.ejs.container.MessageEndpointHandler.invokeMdbMethod(MessageEndpointHandler.java:990)
   at com.ibm.ejs.container.MessageEndpointHandler.invoke(MessageEndpointHandler.java:723)
   at $Proxy0.onMessage(Unknown Source)
   at
com.ibm.ws.sib.api.jmsra.impl.JmsJcaEndpointInvokerImpl.invokeEndpoint(JmsJcaEndpointInvokerImp
l.java:201)
   at com.ibm.ws.sib.ra.inbound.impl.SibRaDispatcher.dispatch(SibRaDispatcher.java:544)
   at
com.ibm.ws.sib.ra.inbound.impl.SibRaSingleProcessListener$SibRaWork.run(SibRaSingleProcessListe
ner.java:403)
   at com.ibm.ejs.j2c.work.WorkProxy.run(WorkProxy.java:434)
   at com.ibm.ws.util.ThreadPool$Worker.run(ThreadPool.java:1455)
```

It is the responsibility of the MDB application to handle exceptions. The solution in this case is for the MDB to gracefully handle exceptions.

Symptom: An MDB fails and is invoked in an infinite loop

An exception destination is used to handle messages that cannot be delivered to their intended bus destination. In this scenario, you have defined that no exception destination be associated with the bus destination (see Figure 14 on page 37).

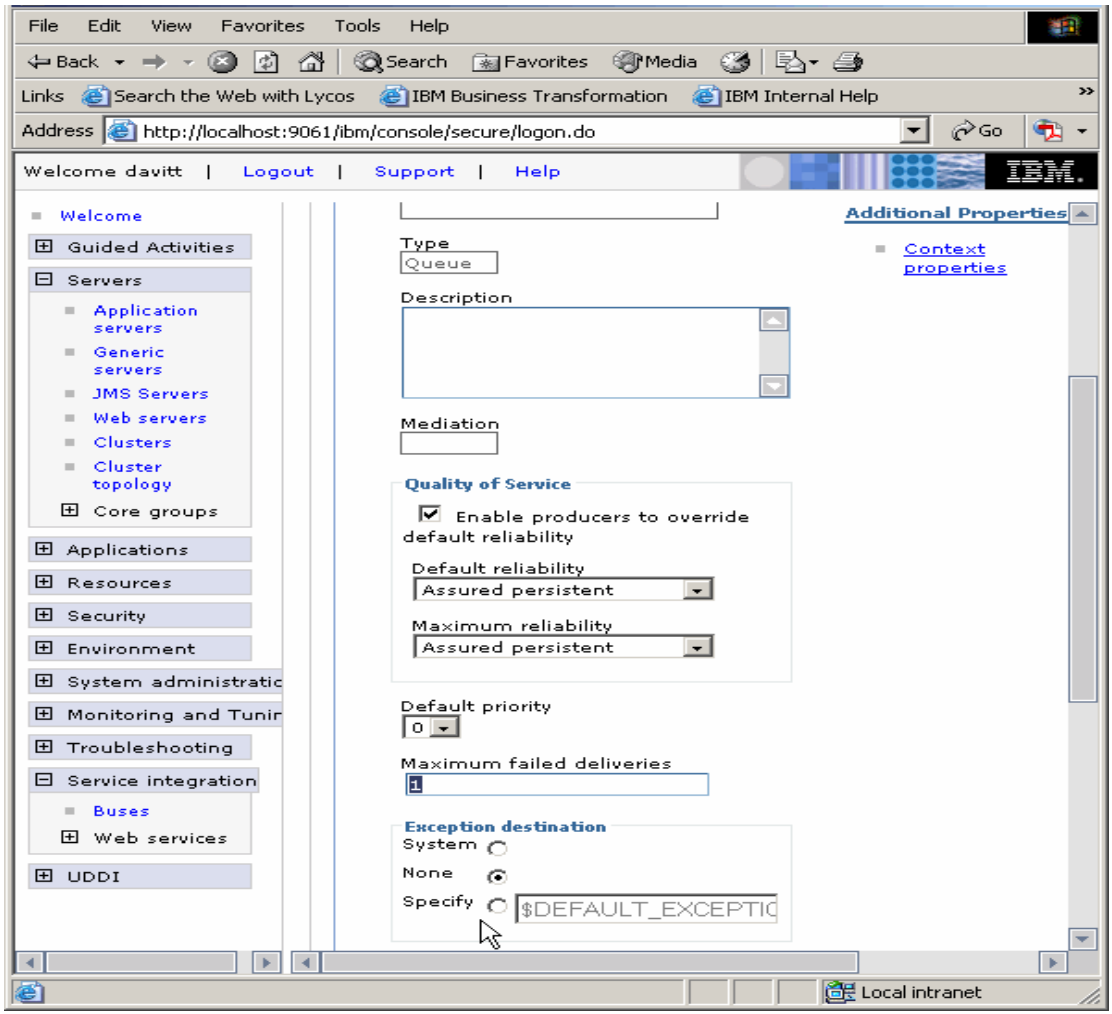

Figure 14 Exception destination is set to None

In many customer environments, defining an MDB on a queue with no exception destination is intentional. It is done to ensure that the message order is preserved for delivery to the MDB from the queue (if one message cannot be processed, they do not want any following messages to be processed).

When no exception destination is used and a message cannot be delivered (for example, the MDB fails), the *Maximum failed deliveries* limit specified for the destination is ignored, and the MDB is repeatedly driven to consume the message. This situation continues until either the message is removed from the destination (for example, by an administrator using the WebSphere

administrative console), or the MDB is able to handle the message. The problem cannot be corrected by setting *Discard messages*.

Be sure that the MDB is performing as expected. Also consider modifying the modify the destination properties to use another exception destination mechanism.

Symptom: Message not restored to queue after MDB failure

The MDB transaction-type, set in the EJB module deployment descriptor, can be container-managed or bean-managed. A bean-managed transaction is responsible for committing or rolling back the transaction. It does this by the use of the UserTransaction interface. If the MDB is bean managed and does not correctly cope with transactions and exceptions, messages can end up “lost.”

Refer to Chapter 10 of *WebSphere Application Server V6 System Management and Configuration Handbook*, SG24-6451, for details.

Problems with mediation

You can configure mediations in two ways:

- A single mediation handler associated with a destination
- A mediation list associated with a destination

If you have implemented a mediation list (a sequence of mediations), you should ensure that:

- They are called in the correct sequence.
- Each mediation returns correctly.

A single mediation or a mediation handler list can be associated with multiple destinations. You should ensure that only those destinations that are required are associated with the mediation. Otherwise, the mediation handler might fail to recognize the message format for that particular destination. Finally, each of the mediations might require its own diagnostic output to ensure that they are working correctly.

Figure 15 on page 39 illustrates how mediations are associated with destinations.

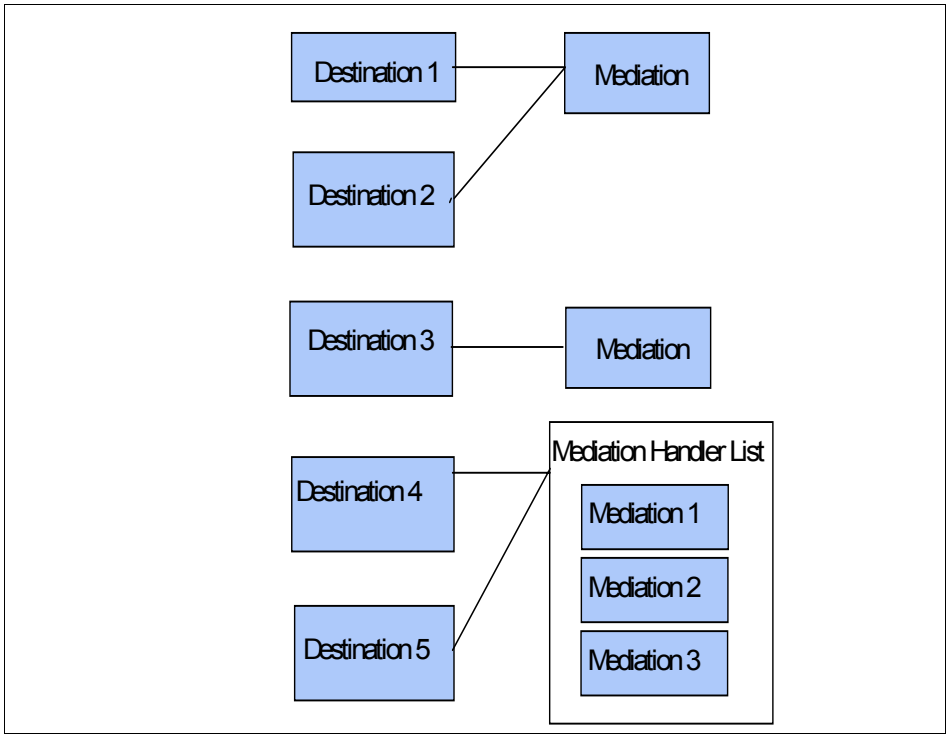

Figure 15 Mediation

Symptom: Mediation of a destination not working

Mediation is implemented using a stateless EJB. Before a mediation will work properly, the infrastructure must be set up correctly. To check the mediation EJB status from the administrative console, select **Applications** → **Enterprise applications** and ensure that any mediation EJBs are started correctly.

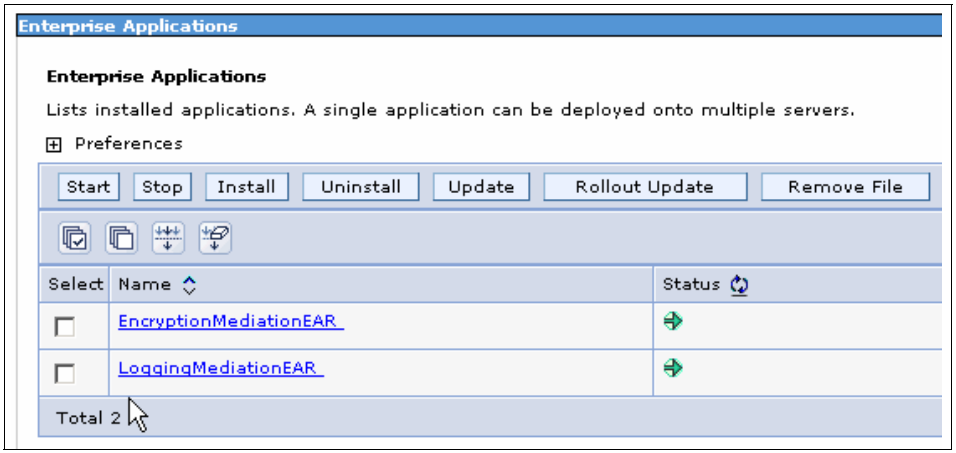

Figure 16 Check the mediation application

If the mediations are started correctly you must ensure they have started on the correct node. This is a common error. Examine the SystemOut.log for the node where the mediation is implemented.

You should see messages WSVR0200I and WSVR0221I for each mediation that is started. See Example 14.

Example 14 Log showing mediations starting

```
[...] 0000001d ApplicationMg A   WSVR0200I: Starting application: LoggingMediationEAR
[...] 0000001e ApplicationMg A   WSVR0200I: Starting application: EncryptionMediationEAR
[...] 0000001e EJBContainerI I   WSVR0207I: Preparing to start EJB jar: EncryptionMediation.jar
[...] 0000001d EJBContainerI I   WSVR0207I: Preparing to start EJB jar: LoggingMediation.jar
[...] 0000001e EJBContainerI I   WSVR0037I: Starting EJB jar: EncryptionMediation.jar
[...] 0000001d EJBContainerI I   WSVR0037I: Starting EJB jar: LoggingMediation.jar
[...] 0000001e ApplicationMg A   WSVR0221I: Application started: EncryptionMediationEAR
[...] 0000001d ApplicationMg A   WSVR0221I: Application started: LoggingMediationEAR
```

If these do not appear, then you should verify that the mediation is configured to the correct node.

When installing the mediation EJB with the administrative console:

1. Select **Applications** → **Enterprise applications**.
2. Select the **Install** option.
3. Browse and select the correct **Local or Remote file system** for the mediation ear file.
4. Continue to **Map modules to servers**.

5. Ensure that **Select Module name** is selected.
6. Also ensure that in the **Clusters and Servers** drop down the correct node name and server have been selected.
7. Apply the correct options for both. See Figure 17 for details.
8. Continue and **Finish** the installation of the EJB.

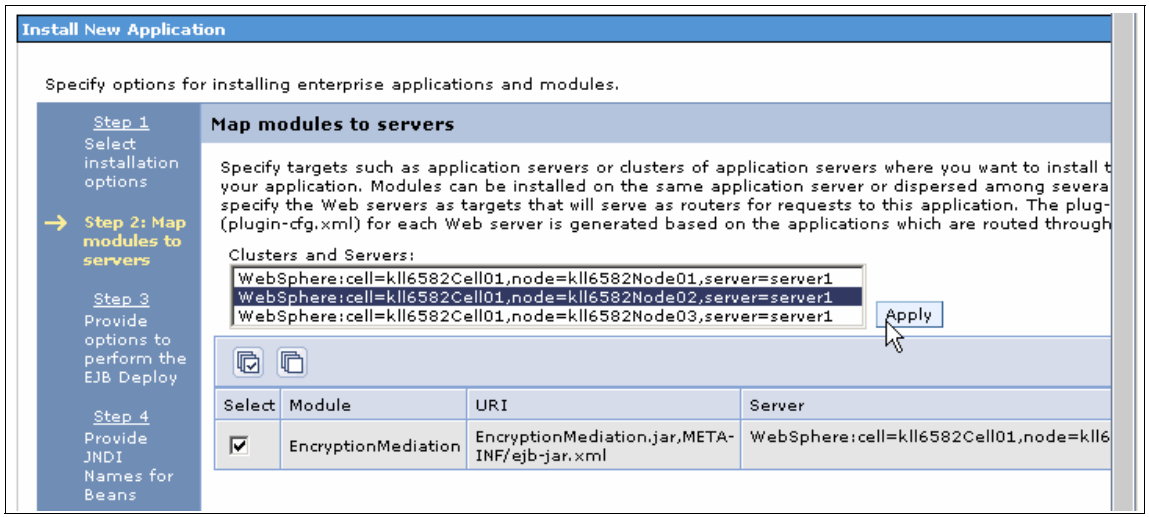

Figure 17 Map module to server

After the EJB has been correctly defined, start the EJB from the **Applications** → **Enterprise applications** panel and verify in the SystemOut.log of the target application server that it has started correctly. Refer again to Example 14 on page 40.

Note: A common error made when implementing mediations is incorrectly specifying the mediation handler list name when defining the mediation in the administrative console. The mediation handler list is defined in the deployment descriptor for the EJB and must be typed in when you create the mediation.

The following messages can be seen when in the log when the names do not match:

```
CWSIP0655E: The local mediation point for destination firstDestination
failed to start because an error occurred. Mediation MyFirstMediation was
not found. The error is CWSIZ0002E: The mediation named MyFirstMediation
that is attached to destination firstDestination is defined to use mediation
handler list RouterHandler. However this handler list does not exist.
```

Symptom: Mediation fails with CWSIZ0002E, messages disappear

Messages can apparently disappear for a variety of reasons. One possibility is that a mediation has failed to start correctly. Typically, this can be seen in the SystemOut.log as CWSIZ0002E message codes. See Example 15.

Example 15 A mediation fails to start

```
[...] 0000002f SibMessage    I   [busname] CWSID0016I: Messaging engine
kll6582Node02.server1-NewBus is in state Started.
[...] 0000002f SibMessage    E   [busname] CWSIZ0002E: The mediation named compressionMediation
that is attached to destination [DestinatioName] is defined to use mediation handler list
[mediationListName]. However this handler list does not exist.
[...] 0000002f SibMessage    I   [busname] CWSIZ0052I: Mediation handler lists defined in the
server are [].
```

In this scenario, the mediation was defined correctly, and the destination was correctly modified to be mediated. However, the mediation handler was not correctly loaded. Messages directed to the destination will not be delivered but will wait at the mediation point that holds messages until they have been mediated. You can confirm this by viewing the mediation point.

In this example, destination cellQ2 is mediated by handler compressionMediation. However, messages sent to that destination are not arriving as expected. You would now suspect messages should be held prior to mediation at the mediation point. You can verify that in the administrative console by doing the following.

1. Select **Buses** and click the bus name.
2. Under the Additional Properties section, select **Destinations**.
3. Select the mediated destination in this example **cellQ2**.
4. Select **Mediation point**.
5. Select the mediation point name.
6. Select the **Runtime** tab.
7. Select **Messages**.

Figure 18 on page 43 shows a list of any messages awaiting mediation before possible delivery to the destination.

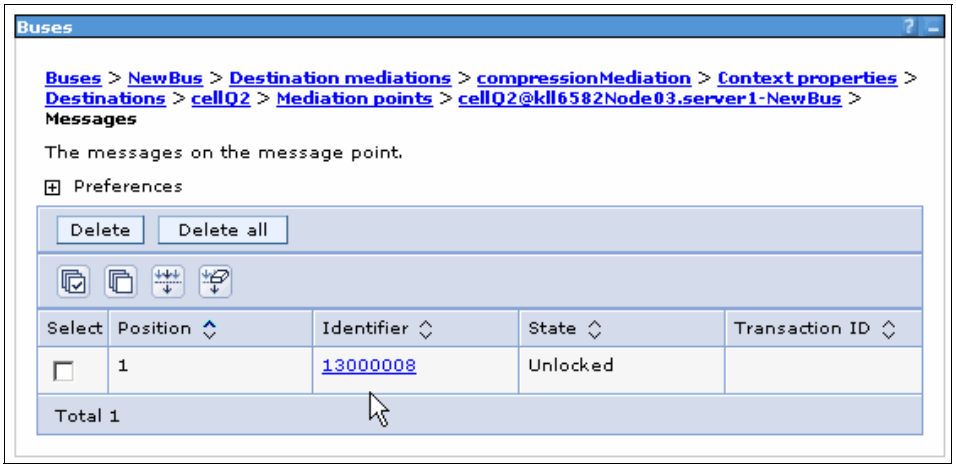

Figure 18 Messages waiting for mediation

This scenario is only one in which messages seem to disappear. There are others which you should also investigate, such as the following:

- The message has expired.
- The mediation might have discarded the message (see “Symptom: Mediation implemented but messages disappear” on page 43).
- The messaging engine might have discarded the message based on its reliability.
- The message might have been directed to an exception destination.

Symptom: Mediation implemented but messages disappear

Mediation depends on user written code and sometimes puts the reliability of message delivery outside the control of the messaging engine. If the mediation has been correctly implemented and started, it is quite possible that the mediation code processes and removes a message from the target destination.

The mediation code is implemented using the MediationHandler interface that uses a handle method.

Example 16 Mediation handler that discards a message

```
public class LoggingMediation implements MediationHandler {
      /* (non-Javadoc)
      * @see
com.ibm.websphere.sib.mediation.handler.MediationHandler#handle(javax.xml.rpc.handler.MessageCo
ntext)
      */
      public boolean handle(MessageContext arg0) throws MessageContextException {
```

```
:
:
:
        return false;// FALSE indicates that the message should be discarded!!
        }
```

By default, there are no messages written to SystemOut.log to indicate that the message has been discarded. You should carefully implement your mediation handlers to output appropriate diagnostic messages when it returns false.

Application configuration and resource problems

In this section, we look at some of the errors that can happen when an application attempts to use the default messaging provider. These errors include configuration errors and resource problems as follows:

Symptom: V5 application gets JMSException MQJMS2005

It is common to find configuration problems during the migration of WebSphere Application Server V5 and WebSphere MQ JMS V5.3 applications to WebSphere Application Server V6 messaging. An existing WebSphere MQ JMS client application would be configured to use the WebSphere MQ V5.3 queue manager name, channel name, host name, and port number to access the JMS objects.

Example 17 shows an application that is migrated to V6 and has experienced an exception MQJMS2005.

Example 17 JMSException MQJMS2005

```
javax.jms.JMSException: MQJMS2005: failed to create MQQueueManager for 'hostname:servername'
        at com.ibm.mq.jms.services.ConfigEnvironment.newException(ConfigEnvironm
ent.java:569)
        at com.ibm.mq.jms.MQConnection.createQM(MQConnection.java:2311)
        at com.ibm.mq.jms.MQConnection.createQMNonXA(MQConnection.java:1739)
        at com.ibm.mq.jms.MQQueueConnection.<init>(MQQueueConnection.java:144)
        at com.ibm.mq.jms.MQQueueConnection.<init>(MQQueueConnection.java:54)
        at com.ibm.mq.jms.MQQueueConnectionFactory.createQueueConnection(MQQueue
ConnectionFactory.java:106)
        at JMSPutClient.main(JMSPutClient.java:79)
```

To connect correctly to the bus, the application should present correct values to the application server. Table 1 shows the connection parameters to check and where you can find them.

Table 1 Connection parameters

Property	Location
Queue manager name	WebSphere MQ Client link
Host name / IP address	Network configuration
Port number	SIB_MQ_ENDPOINT_ADDRESS or SIB_MQ_ENDPOINT_SECURE_ADDRESS
Channel name	WebSphere MQ Client link
Queue / Topic name (destination)	Messaging engine / endpoint definition

These values can be configured in the connection factory, queue, and topic objects that are looked up through JNDI. The JNDI values or those used by the application code should be cross-referenced with those that are used by the application server.

When the JMS client successfully connects to the application server, then message CWSIC3704I is displayed in the SystemOut.log file as shown in Example 18.

Example 18 Client successfully connects to the application server

```
[...] 00000034 SibMessage    I   [:] CWSIC3704I: A WebSphere MQ client application has
connected from host 127.0.0.1:1164 on transport chain InboundBasicMQLink.
```

When a client is connected, then you can confirm this by viewing the connections:

1. Select **Service Integration** → **Buses** and click the bus name.
2. Click **Messaging Engines** and then select the engine name.
3. Click **WebSphere MQ client links**, then the link name.
4. Click **Connections**.

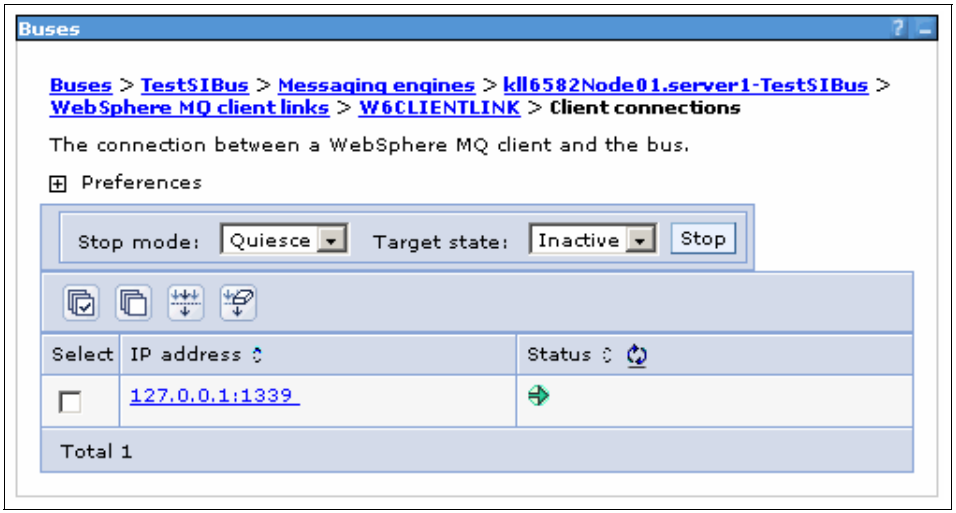

Figure 19 Client connections

Note: WebSphere MQ client links are intended to be a used as a temporary migration path. Longer term, all migrated applications should have their JMS resources migrated to V6 default messaging provider JMS resources. This means that you need to delete the V5 embedded messaging provider JMS resources and create, in their place, V6 default messaging provider JMS resources with the same JNDI names.

Symptom: ServiceUnavailableException on bus connect

Possibly one of the most common errors when starting client applications in the client container is that the application servers have not been started. Naturally, there will not be a message in the application server log. So, the application stack trace must be investigated. These can be very lengthy, but fortunately, the most appropriate error usually appears at the start of the stack trace. Example 19 shows that the first exception code displayed is ServiceUnavailableException, and the accompanying text indicates the actual node and port number that the application was attempting to use.

Example 19 ServiceUnavailableException

```
ServiceUnavailableException caught
Stack Trace:
javax.naming.ServiceUnavailableException: A communication failure occurred while attempting to
obtain an initial context with the provider URL: "corbaloc:iiop:kll6582.itso.ral.ibm.com".
Make sure that any bootstrap address information in the URL is correct and that the target name
server is running.  A bootstrap address with no port specification defaults to port 2809.
Possible causes other than an incorrect bootstrap address or unavailable name server include
```

```
the network environment and workstation network configuration.  Root exception is
org.omg.CORBA.TRANSIENT: java.net.ConnectException: Connection refused:
connect:host=9.42.171.161,port=2809  vmcid: IBM  minor code: E02  completed: No
   at
com.ibm.CORBA.transport.TransportConnectionBase.connect(TransportConnectionBase.java:443)
   at com.ibm.ws.orbimpl.transport.WSTransport.getConnection(WSTransport.java:437)
```

You need to check the following:

- The node address and port are correct.
- The bootstrap server are running on that specific URL and port.

Symptom: JMSException on bus connect, msg CWSIT0006E

Another common example is a JMSException while trying to connect to the bus. This can be caused by a stopped messaging engine, or more commonly, a wrong or misspelled bus name (case-sensitive) in the connection factory.

Example 20 shows an extract from an application stack trace showing such a problem. In this case, the messaging engine is stopped.

Example 20 JMSException: Messaging engine stopped

```
JMSException caught
Stack Trace:
javax.jms.JMSException: CWSIA0241E: An exception was received during the call to the method
JmsManagedConnectionFactoryImpl.createConnection:
com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E: It is not possible to contact
a messaging engine in bus TestSIBus..
   at
com.ibm.ws.sib.api.jms.impl.JmsManagedConnectionFactoryImpl.createConnection(JmsManagedConnecti
onFactoryImpl.java:225)
   at
com.ibm.ws.sib.api.jms.impl.JmsManagedConnectionFactoryImpl.createConnection(JmsManagedConnecti
onFactoryImpl.java:148)
```

You can see the generalized JMSException code CWSIA0241E and the CWSIT0006E message, which indicates that when the connection was made, there was no messaging engine active for the specified bus. To see why the messaging engine was stopped, you need to check the SystemOut.log for applicable messages. Examine the log file starting at the time period when the client experienced the problem and search backward.

Example 21 SystemOut.log messages for a stopped messaging engine

```
:
[...] 00000033 SibMessage    I   [:] CWSID0016I: Messaging engine [mename]is in state Stopping.
[...] 00000033 SibMessage    I   [:] CWSID0016I: Messaging engine [mename]is in state Stopped.
:
```

Example 22 shows another example of an application failing to connect to a bus. This shows a stack trace from an application that has failed to create a connection to a named bus. As before, you see the JMSException message, CWSIA0241E and CWSIT0006E.

Example 22 Bus connect fails, msg CWSIT0006E

```
JMSException caught
Stack Trace:
javax.jms.JMSException: CWSIA0241E: An exception was received during the call to the method
JmsManagedConnectionFactoryImpl.createConnection:
com.ibm.websphere.sib.exception.SIResourceException: CWSIT0006E: It is not possible to contact
a messaging engine in bus AnyBus..
   at
com.ibm.ws.sib.api.jms.impl.JmsManagedConnectionFactoryImpl.createConnection(JmsManagedConnecti
onFactoryImpl.java:225)
```

Again, review the SystemOut.log file for messages related to the external bus. Example 23 illustrates a problem with the bus.

Example 23 Bus connect fails, msg CWSIT0018W

```
[...] 0000001a SibMessage    W   [BusName] CWSIT0018W: It is not possible to create the
inter-bus connection AnyBus.Link to messaging engine AnyBus.ME in bus AnyBus, on host
nn.nn.nn.nn port nnn using protocol BootstrapBasicMessaging.
```

If you were expecting the application to continue running on any available messaging engine, look at the JMS connection factory definitions to ensure the appropriate target name, target type, and proximity values have been matched.

Symptom: Bus connect fails, message CWSIT0019E

There are many other reasons for the an application being unable to connect to the bus, for example:

```
CWSIT0019E: No suitable messaging engine is available in bus
```

If you see this message, make sure the appropriate messaging engines have started correctly. Look for the following message in SystemOut.log before the exception occurred:

```
CWSID0016I: Messaging engine [mename] is in state Started.
```

If the messaging engines have started, then you could have a configuration problem.

In the JMS connection factory, you can define a list of provider endpoints that govern which servers the application will attempt to connect to. If you have defined these endpoints, ensure that the messaging engines for these servers have all started and that the port and transport chains defined in the endpoint list are correct. Also, ensure that the values for the Connection proximity (bus, cluster, host or server) and the Target groups and Target significance settings are appropriate. For example, if the Target significance value is Required and no suitable messaging engine could be located, then this would cause a connection failure.

For more information about configuring the JMS connection factory and controlling messaging engine selection, refer to Chapter 10 of *WebSphere Application Server V6 System Management and Configuration Handbook*, SG24-6451.

If this solution does not resolved the problem, go to “The next step” on page 50.

Product errors

As with any large and complex software application, there is the possibility of products errors. The following are examples that arose during the writing of this paper. If you believe that you have discovered others, go to “The next step” on page 50.

Symptom: JMS application returns MQJMS2007

In Example 24, a JMS application is sending a message via the WebSphere MQ client link. During testing, all appears well. However, the application suddenly produces a stack trace with MQJMS2007.

Example 24 Stack trace with message MQJMS2007

```
javax.jms.JMSException: MQJMS2007: failed to send message to MQ queue
        at com.ibm.mq.jms.services.ConfigEnvironment.newException(ConfigEnvironm
ent.java:553)
        at com.ibm.mq.jms.MQMessageProducer.sendInternal(MQMessageProducer.java:
1598)
        at com.ibm.mq.jms.MQMessageProducer.send(MQMessageProducer.java:1022)
        at com.ibm.mq.jms.MQMessageProducer.send(MQMessageProducer.java:1056)
        at JMSPutClient.main(JMSPutClient.java:94)
```

By further investigation of the JMS application, it was noticed that then problem only seems to occur when sending a JMS TextMessage greater than or equal to

32079 bytes in size. The WebSphere MQ client link configuration is checked and all appears correct.

Ensure that you are using the latest version of the WebSphere MQ client code by checking the jar files that are used by the JMS application and by verifying that they are at the current service level. See the following link for additional information:

http://www.ibm.com/software/integration/mqfamily/support/summary

Having established what we believe to be a product defect, we refer to “The next step” on page 50 for what to do next.

Note: This problem has been resolved in the 6.0.2 refresh pack.

Message CWSID0032W

One minor problem found during the writing of this book was the message code CWSID0032W during startup. See Example 25.

Example 25 CWSID0032W during startup

```
SibMessage    E   [busname] CWSID0032W: An inconsistency in the WCCM document sib-engines.xml
was detected; reason uuid=B77F76AAB6EC0BC35414B268 targetUuid=B77F76AAB6EC0BC35414B268
```

The error did not appear to cause any problems to the runtime and was resolved by the installation of refresh pack 6.0.2.

The next step

The symptoms and problem areas included in this paper are some that you are more likely to experience. However, there are other things that can go wrong, or the cause of the problem might be related to a component other than the default messaging provider.

If, after going through this process, you still have an undiagnosed problem, we recommend that you go back to *Approach to Problem Determination in WebSphere Application Server V6*, REDP-4073, at:

http://www.redbooks.ibm.com/redpapers/pdfs/redp4073.pdf

Review the “Classify the problem and determine the root cause” section to see if there are any other components that might be causing the problem.

If you feel sure you have a messaging problem, there are things you can do before contacting IBM support. First, you should review the documentation you

have gathered for errors related to the problem that were not addressed in this paper and search support sites for information or fixes.

The WebSphere Information Center contains problem determination information for troubleshooting service integration technologies. Take the time to review this information to see if there are any tips that will help you. You can find this guide at:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r0/topic/com.ibm.websphere.pmc.nd.doc/ref/rjk_prob0.html

To assist the service organization it is useful to gather appropriate documentation. As we have mentioned, there are some basics such as SystemOut.log, SystemErr.log, server logs, and the exception logs from the FFDC directory. From your own analysis, you might have determined that traces would also be required. Before doing so, remember the following:

- Make the documentation relevant. Ensure that the documentation is recent and that it shows the problem.
- Provide a clear and concise description of the problem. By now, you have investigated the problem yourself, and you have a good understanding of how the problem occurs.
- Is the problem easily recreated? If possible, describe the steps that are needed to reproduce the problem or provide a small test case to demonstrate the problem. It is easier to diagnose and fix a problem with a suitable small test case that does not require a large configuration (for example using databases or network configurations). The service organization is aware that this is not always possible.
- Can you gather a suitable trace for the problem? Because you have done some preliminary analysis, you might have an understanding of which component you believe is causing the problem. See Table 2 on page 52 for a listing of the trace strings that might be appropriate for the components you believe are responsible.

For a list of documentation required to report a problem to IBM support, see *Mustgather: Service Integration Technology*, at:

http://www.ibm.com/support/docview.wss?uid=swg21199330

You can also find general information about what documentation to gather in the *Troubleshooting Guide for WebSphere Application Server* at:

http://www.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg27005324

If you believe you have identified a messaging component area where the problem exists but have not identified the particular problem, you could attempt

to identify it further or assist the support organization by providing an appropriate trace. To do this, you need to set one or more trace strings.

Table 2 Component trace strings

Component	Trace string or strings
Data store	SIBMessageStore=all
MQ link	SIBCommunications=all=enabled SIBMfp=all=enabled SIBMqFapChannel=all=enabled
Security	SIBSecurity=all=enabled
Mediation	SIBMessageTrace=all SIBMediations*=all=enabled
JMS applications	SIBJms*=all=enabled
Bus message routing This can include client connections, ME to ME connections, and interbus connections.	SIBTrm=all=enabled

If your problem relates to the interoperability with WebSphere MQ, it might be necessary to gather trace for that system.

For instructions about gathering traces for WebSphere MQ systems, refer to the platform-specific *Administration Guide* at:

http://www.ibm.com/software/integration/mqfamily/library/manualsa/

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662, P.O. Box 12195
  Research Triangle Park, NC 27709-2195 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Cloudscape™	Redbooks (logo) ™	z/OS®
DB2®	Redbooks™	
IBM®	WebSphere®	

The following terms are trademarks of other companies:

EJB, Java, JDBC, JVM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.