IBM Software Group

Understanding Stuck Messages

Ty Shrake (tyshrake@us.ibm.com)
Paul O’Donnell (paulod@us.ibm.com)

WebSphere® Support Technical Exchange

Agenda

- Introduction
- Overview of Message Flow
- Message Producers and Consumers
- Message Driven Bean Basics
- MDB Transactionality
- How Message Driven Beans Work
- Resource Adapters
- Inside onMessage()
- Message Accumulation
- Identifying the Problem
- Resolving the Problem
- Javacores
- Message States
- Summary

Introduction

Messages “stuck” on a destination is one of the most common problems customers encounter in messaging systems. In most cases the symptoms include messages stacking up or accumulating on a destination and a failure to process business data. This presentation will explain why message accumulation occurs and how to troubleshoot this type of problem.

This presentation is provided in two parts:

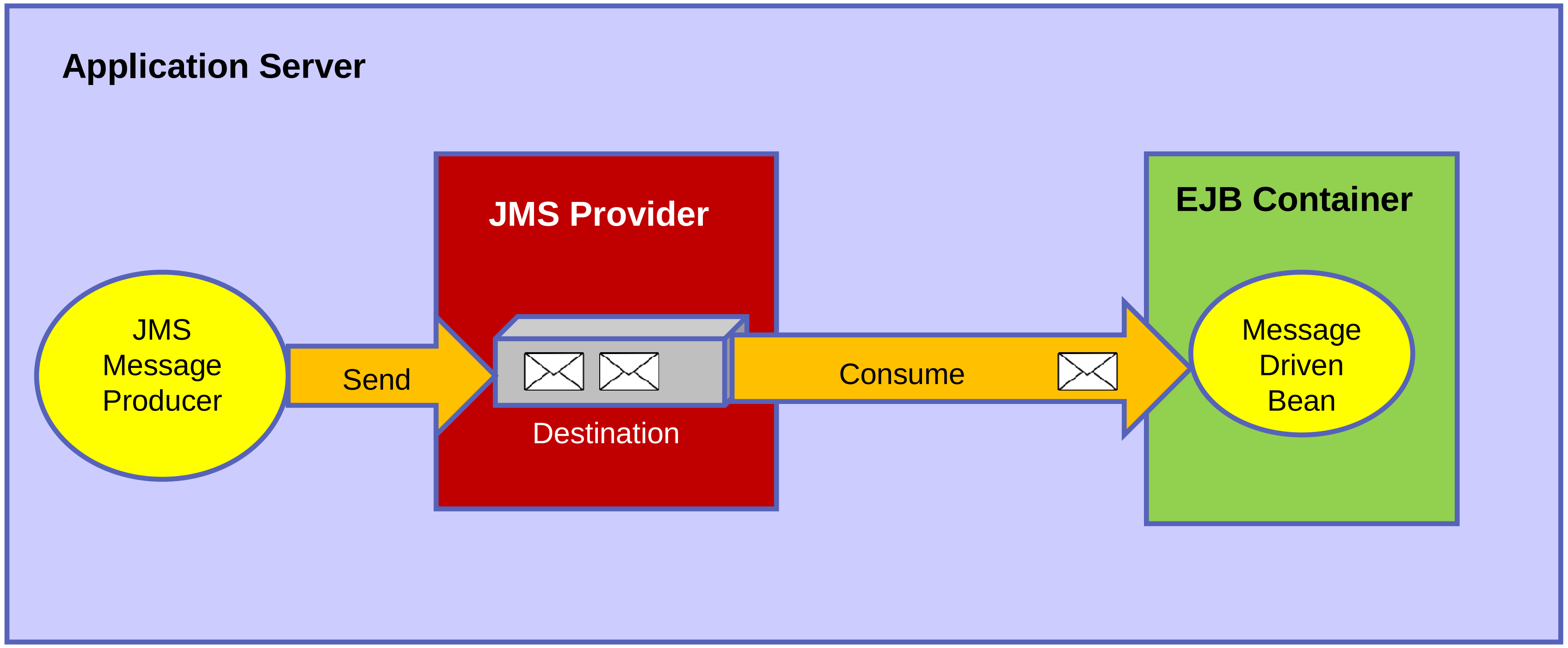
- Main Presentation
- Detailed Addendum

The Detailed Addendum provides additional details and analysis on Javacores, how to gather and read them, and all possible message states.

The terms “queue” and “destination” are used interchangeably throughout the presentation.

Overview of Message Flow

When messages are stuck on a queue it is important to have a basic understanding of message flow in your environment. In 99% of all cases the message flow is very similar to the flow shown below:



Message Producers and Consumers

Messages are created in Producer applications. Once the message is created the application sends the message to a destination. This process usually works fine and is rarely problematic.

Most problems with messages involve the message Consumer application, which is supposed to ‘consume’ (remove) the message from the destination and process the business data in the message. Most investigations into message ‘stuck’ on a queue/destination should start with the Consumer application, not the Producer**.** In Websphere Application Server message consumers are almost always **Message Driven Beans.**

Message Driven Bean Basics

When investigating messages that are stuck on a queue or are accumulating on a queue it helps a great deal if you understand what Message Driven Beans are and how they work.

There are three types of Enterprise Java Beans: Session Beans, Entity Beans and Message Driven beans.

- **Message Driven Bean**: A Message Driven Bean (**MDB**) is a special case of EJB. It is usually a fairly small application whose purpose is to receive (consume) messages from a messaging system.
- Session Beans
- Entity Beans

Note: All EJBs run inside the EJB Container. The container manages each bean and ensures the bean can access the services provided by the application server environment.

Message Driven Bean Basics

An MDB is an application whose purpose is to receive (consume) messages from a messaging system.

Important Features of MDBs:

- MDBs do not need any connection code to consume messages.
- All MDBs have a method named **onMessage()**.
- MDBs run inside an EJB Container

MDB Transactionality

The Deployment Descriptor of an MDB (**ejb-jar.xml**) holds important information about how the bean behaves and what resources it will use. One of the items in the DD is the transactionality the bean will use. There are 2 basic options:

- **Container Managed** - Preferred
- **Component or Bean Managed**

How Message Driven Beans Work

When we talk about Message Driven Beans we often say that they “consume messages”. This is true but the situation is a bit more complex than this.

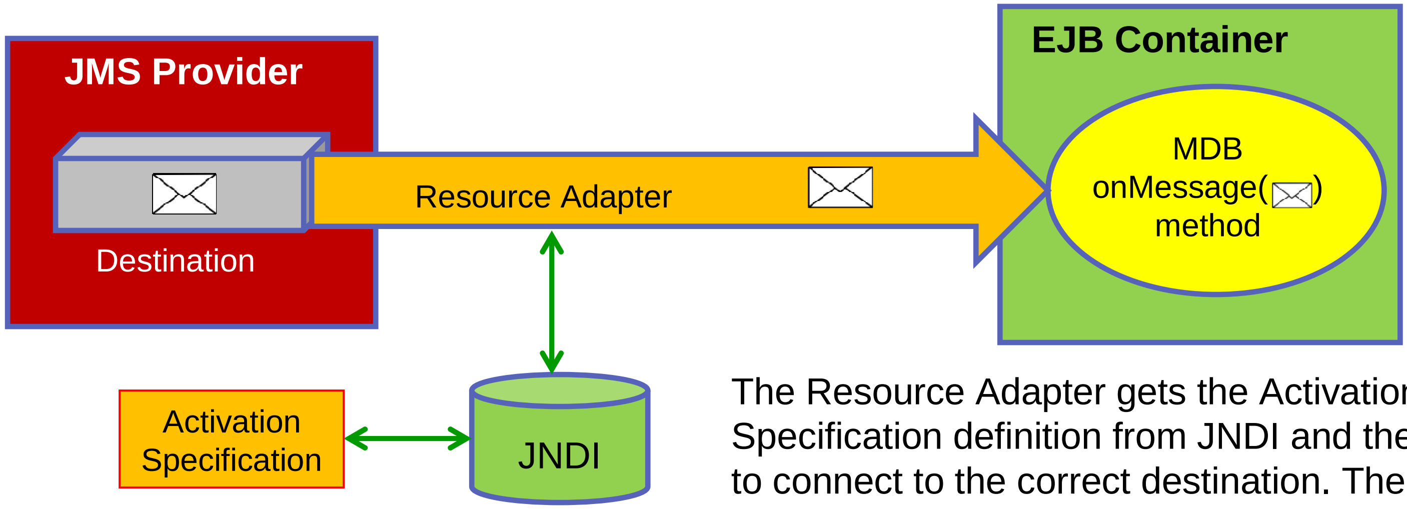
When discussing MDB message consumption we must break the process down into three parts:

- The Destination
- The Resource Adapter
- The MDB

Resource Adapters

A **Resource Adapter** (**RA**) is *a program* that connects to a JMS destination, listens for messages arriving on that destination, and then passes the messages to the onMessage() method of the MDB using that RA.

The RA is configured by an **Activation Specification** (**AS**). The AS is just a configuration object that tells the RA what destination to connect to and how to manage message delivery to the MDB. The AS is stored in JNDI. When the Resource Adapter starts it reads the AS connection information and uses that to actually connect to the destination.



The Resource Adapter gets the Activation Specification definition from JNDI and then uses that to connect to the correct destination. The AS also configures the message delivery options in the RA .

Inside onMessage()

MDB's that consume JMS messages have a method named **onMessage()** in their code. There are other methods as well (beyond the scope of this education) but onMessage() is the only method we are concerned with. The onMessage() method takes a Message object as its argument, as follows:

onMessage(your_msg)

In the MDB code the basic onMessage() method looks like this:

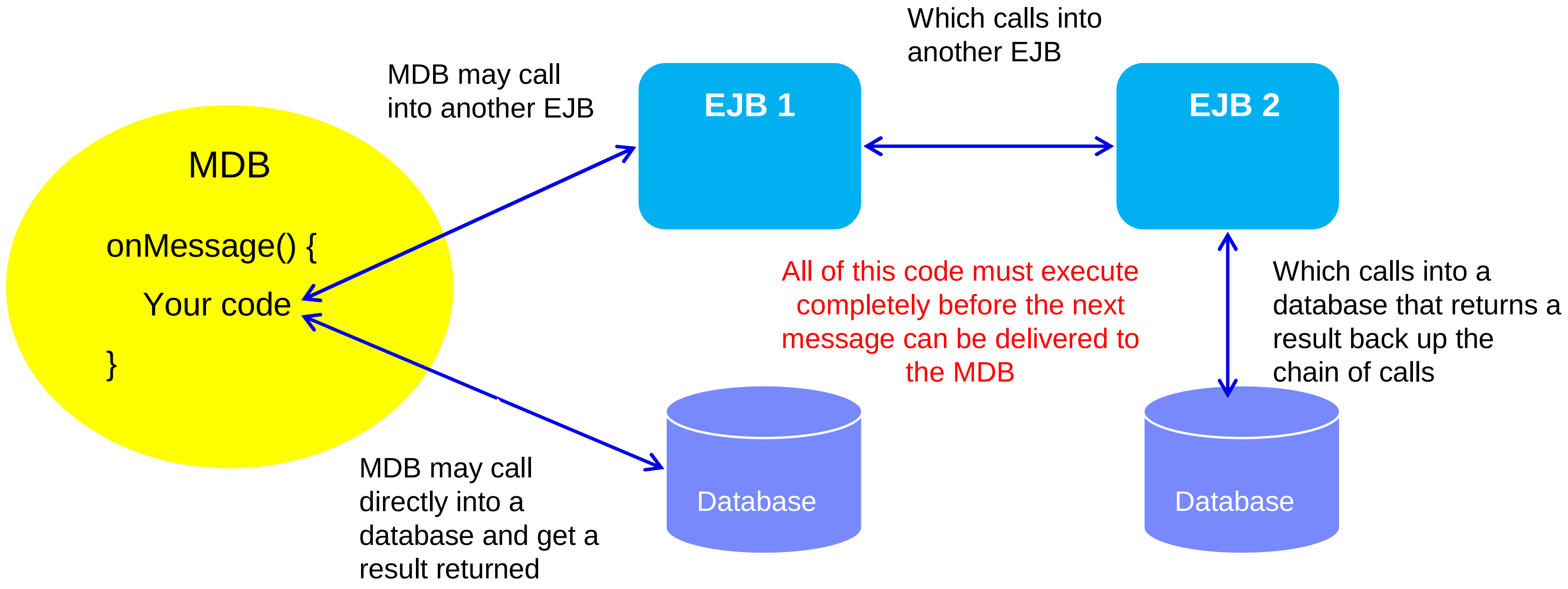
```
public void onMessage(javax.jms.Message  msg) {

    Your business code goes here…

} //end onMessage()
```

Inside onMessage()

The code inside the onMessage() method processes the message data. In most cases the data inside the message goes through a little processing and then other business objects (EJBs) are called for additional processing. For example…



Which calls into another EJB
MDB may call into another EJB
EJB 1
EJB 2
MDB
onMessage() {
Your code
}
All of this code must execute completely before the next message can be delivered to the MDB
Which calls into a database that returns a result back up the chain of calls
MDB may call directly into a database and get a result returned
Database
Database

Message Accumulation

But what if messages start to accumulate or stack up on the queue? *This is usually the first hint to a customer that something isn't working correctly.* There are generally 2 causes of message accumulation:

- Message consumption by the MDB is slower than message production to the destination. In other words, messages are sent to the destination faster than they can be removed and processed by the MDB.
- The MDB is having trouble processing a message and is effectively stopped.

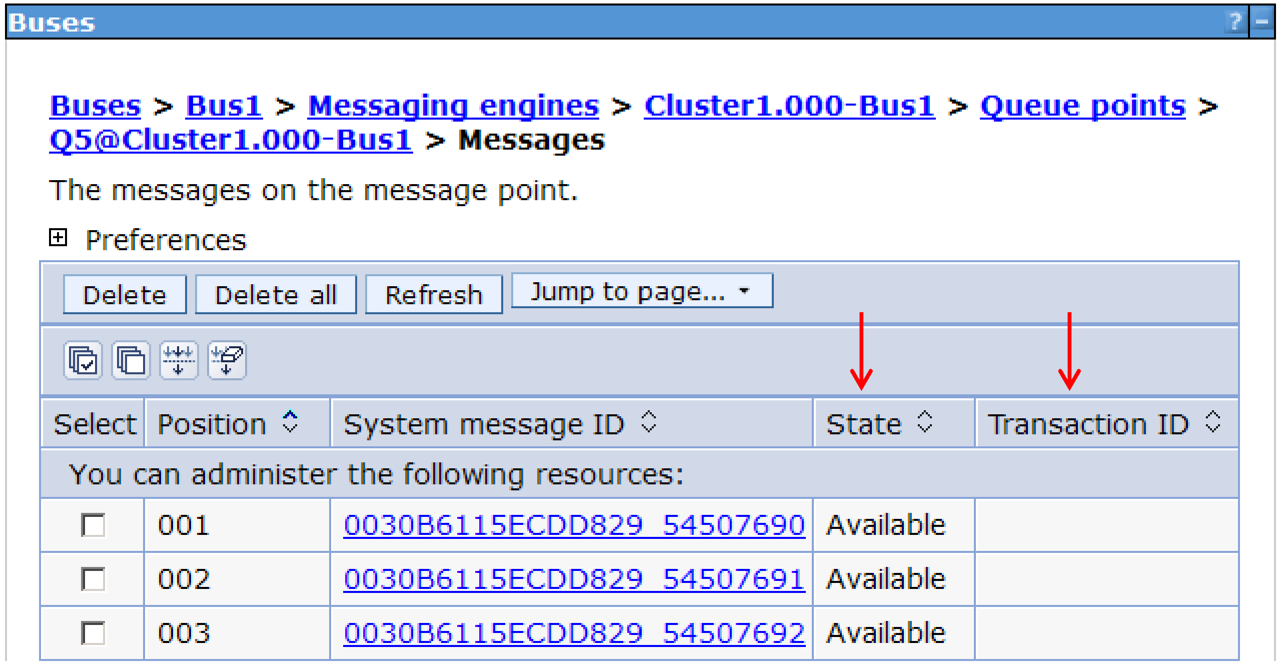
Identifying the Problem

In order to determine which of these two conditions is happening you must first view the messages on the destination. You can do this in the WAS Administration Console by navigating as follows:

Service Integration > Buses > YOUR_BUS > Messaging engines > YOUR_MESSAGING_ENGINE > Queue points > YOUR_QUEUE_POINT (Runtime tab) > Messages

Identifying the Problem

In this example notice that the messages have a State of **Available** and there is **no Transaction ID**. This means the messages are available to be delivered to an MDB.


Buses

Buses > Bus1 > Messaging engines > Cluster1.000-Bus1 > Queue points > Q5@Cluster1.000-Bus1 > **Messages**

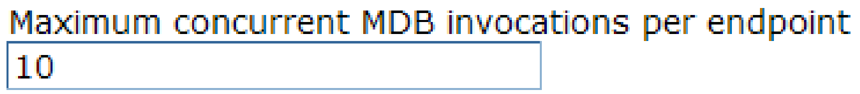
The messages on the message point.

⊞ Preferences

Select	Position	System message ID	State	Transaction ID
You can administer the following resources:				
☐	001	0030B6115ECDD829_54507690	Available	
☐	002	0030B6115ECDD829_54507691	Available	
☐	003	0030B6115ECDD829_54507692	Available	

Identifying the Problem

If the list is constantly changing (messages are being delivered to an MDB) but the list continues to grow in size (the total number of messages on the destination is increasing) then this indicates that messages are arriving on the destination faster than they can be removed. This can often be corrected by increasing the **Maximum concurrent MDB invocations per endpoint** in the Activation Specification used by the MDB**.** The default value is **10**.

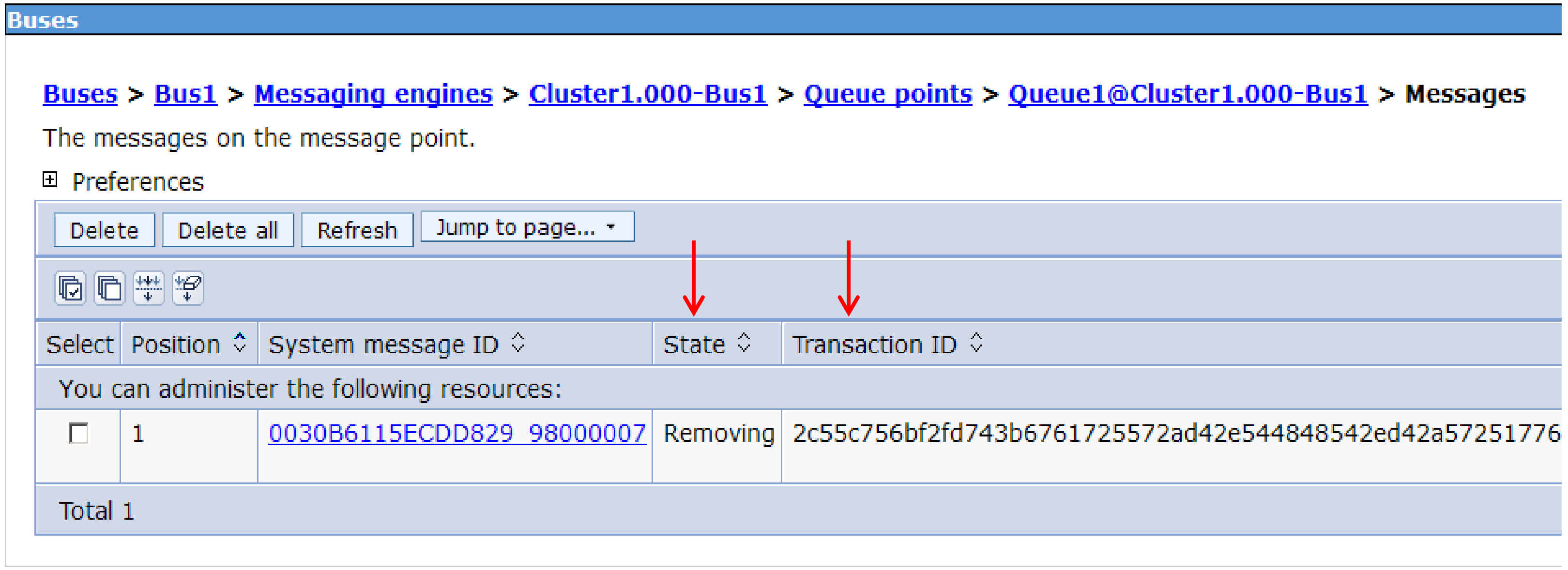

Maximum concurrent MDB invocations per endpoint
10

Note: Changing this value will require a restart of your messaging cluster.

This means that up to 10 instances (copies) of the same MDB can run simultaneously. Using more than one MDB instance is sometimes called **MDB Throttling**.

Identifying the Problem

Many times, when an MDB is Started but messages are not being consumed from the destination, the list of messages will show at least 1 message in a **Removing** state and it will have a Transaction ID, as follows:



Buses

Buses > Bus1 > Messaging engines > Cluster1.000-Bus1 > Queue points > Queue1@Cluster1.000-Bus1 > Messages

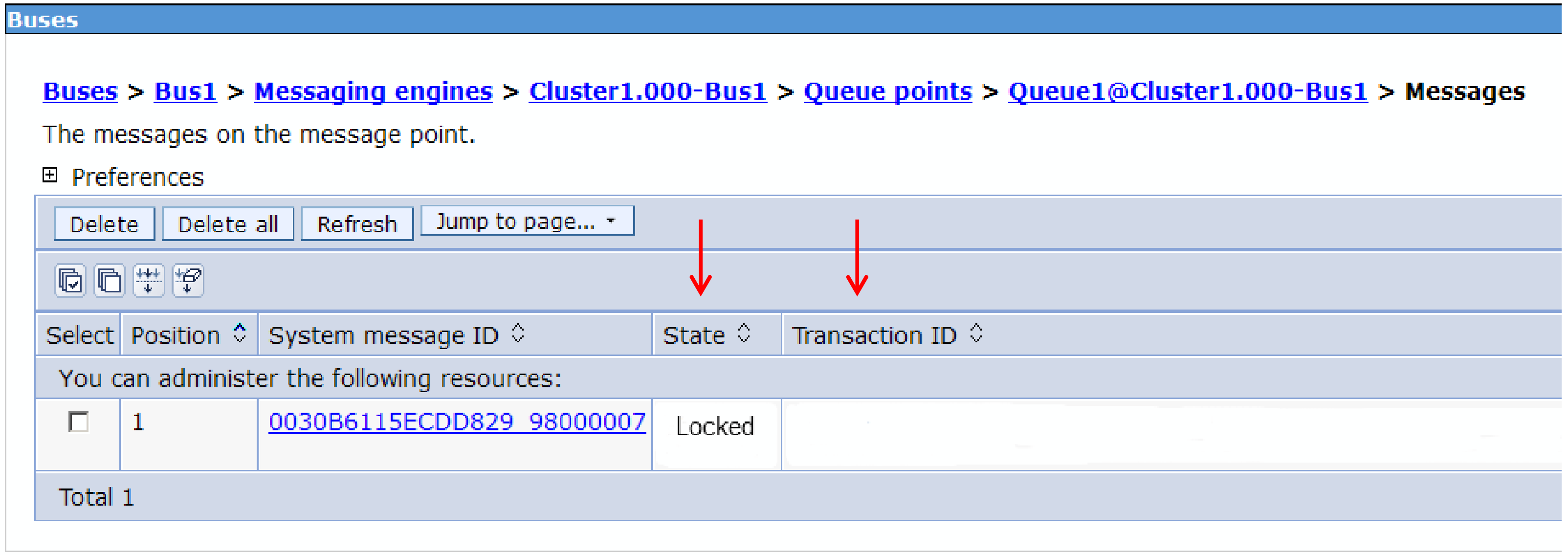
The messages on the message point.

Select	Position	System message ID	State	Transaction ID
You can administer the following resources:				
☐	1	0030B6115ECDD829_98000007	Removing	2c55c756bf2fd743b6761725572ad42e544848542ed42a57251776
Total 1				

This means that the message is currently being removed from the destination under a transaction that hasn't completed. **This often indicates a problem with the MDB**.

Identifying the Problem

Sometimes the list of messages will show at least 1 message in a **Locked** state with no Transaction ID, as follows:



Buses > Bus1 > Messaging engines > Cluster1.000-Bus1 > Queue points > Queue1@Cluster1.000-Bus1 > Messages

The messages on the message point.

Select	Position	System message ID	State	Transaction ID
You can administer the following resources:				
☐	1	0030B6115ECDD829_98000007	Locked	
Total 1				

This message is most likely being consumed by a non-transacted MDB**. If this message stays in this state for a long period the MDB is the most likely cause.**

Resolving the Problem

The following slides will discuss solving stuck messages in eight common scenarios:

- Messages are **Available** but **moving slowly**
- Messages are **Available** and **not moving**

- Messages are in a **Removing** state **with Transaction ID but moving slowly**
- Messages are in a **Removing** state **with Transaction ID but not moving**

- Messages are in a **Locked** state with **no Transaction ID but moving slowly**
- Messages are in a **Locked** state with **no Transaction ID but not moving**
- Messages are in a **Locked** state with **Transaction ID but moving slowly**

- Messages are in an **Unlocked** state with **no Transaction ID but not moving**

Resolving the Problem

- Messages are **Available** but moving slowly:

Interpretation: Messages are being consumed by the MDB but at a slow rate.

Possible Solution:

- Try increasing Maximum MDB Concurrency (see slide 16)
- Check MDB performance by adding timestamps to the MDB source code or gather three Javacores at about 30 second intervals to see where the MDB is spending time.

Resolving the Problem

- Messages are **Available** and not moving:

  Interpretation: The MDB may not be started.

  Possible Solution:

  - Check the state of your MDB and make sure it is in a Started state.
  - Manually stop and restart your MDB
  - If the procedure above does not work and the MDB is Started, gather three Javacores against the PID of the server hosting the MDB.

Resolving the Problem

- Messages are in a **Removing** state **with Transaction ID** but moving slowly:

Interpretation: Messages are being consumed by the MDB but at a slow rate.

Possible Solution:

- Increase Maximum MDB Concurrency (see slide 16)
- Manually stop and restart the MDB
- Check MDB performance by adding timestamps to the MDB source code or gather three Javacores at about 30 second intervals to see where the MDB is spending time.

Resolving the Problem

- Messages are in a **Removing** state **with Transaction ID** but not moving:

  Interpretation: Messages are being consumed by the MDB but at a slow rate.

  Possible Solution:

  - Increase Maximum MDB Concurrency (see slide 16)
  - Manually stop and restart the MDB
  - Check the state of the transaction and try to manually complete the transaction by navigating in the WAS Administration Console as follows:

    Servers > Server Types > WebSphere application servers > [Content pane] *server_name* > [Container Settings] Container Services > Transaction Service > Runtime > Manual transactions - Review

Resolving the Problem

To list the resources used by a transaction, click the transaction local ID in the list displayed.

To act on one or more of the transactions listed, select the check boxes next to the transactions that you want to act on, then use the buttons provided. You can either manually Commit or Rollback the transaction.

- If the procedure above does not work gather three Javacores, at one minute intervals, against the PID of the server hosting the MDB.

Resolving the Problem

- Messages are in a **Locked** state **with no Transaction ID** but moving slowly:

Interpretation: Messages are being consumed by the MDB but at a slow rate.

Possible Solution:

- Increase Maximum MDB Concurrency (see slide 16)
- Manually stop and restart the MDB
- If the procedures above do not work gather three Javacores against the PID of the server hosting the MDB.

Resolving the Problem

- Messages are in a **Locked** state **with no Transaction ID** but not moving:

Interpretation:

- The message has been passed to a consumer, and that consumer has not yet completed.
- The message has been passed to a consumer, and that consumer has completed but we haven't tried to remove the message yet.
- The message has been passed over the network to a client application preemptively (up to one per client with read-ahead disabled, and many more if read-ahead is enabled) - the client has not yet consumed it.

Possible Solution:

- Increase Maximum MDB Concurrency (see slide 16)
- Manually stop and restart the MDB
- If the procedures above do not work gather three Javacores against the PID of the server hosting the MDB.

Resolving the Problem

- Messages are in a **Locked** state **with Transaction ID** but moving slowly:

Interpretation:

- An application has consumed the message with a transaction, and is performing some processing that is taking a long time to complete (or it may have deadlocked).
- A transaction has become in-doubt (one of the resources managers failed during the prepare phase or between prepare and commit), and the transaction manager has not been yet able to contact the messaging engine and tell it whether to commit/rollback the transaction.

Possible Solution:

- Increase Maximum MDB Concurrency (see slide 16)
- Manually stop and restart the MDB
- If the procedures above do not work gather three Javacores against the PID of the server hosting the MDB.

Resolving the Problem

- Messages are in an **UnLocked** state **with no Transaction ID** but not moving:

Interpretation:

- No thread is listening for messages on this destination.
- The match criteria of all the consumers for this destination do not match the message.
- All consumers for this destination are pointing at a different partition of the destination in a WLM environment.
- All consumers are maxed out processing messages and can't accept new ones (this would normally happen when we have a mix of **locked** and **unlocked** messages).

Resolving the Problem

Possible Solution:

- Make sure the Activation Specification for the MDB is configured correctly
- Increase Maximum MDB Concurrency (see slide 16)
- Manually stop and restart the MDB
- If the procedures above do not work gather three Javacores against the PID of the server hosting the MDB.

Javacores

In situations where messages are not consumed from a destination (customers often describe them as being “stuck” on the queue) it is usually a good idea to gather Javacores. Javacores are a ‘snapshot’ of the Java Virtual Machine (JVM) that show what each thread in the JVM is doing. For stuck messages Javacores should be gathered against the JVM running the MDB. A sequence of three Javacores, about one minute apart, should be gathered. The detailed addendum includes links to how to gather Javacores on all of the major distributed (non-z/OS) platforms.

The addendum also includes an actual javacore example that shows an MDB that is stuck in it’s onMessage() method. A detailed explanation of how to read the Javacore is also included.

Message States

We have already reviewed the most common message states that you may encounter. A complete list of all possible messages state and what they mean is available in the detailed addendum.

Summary

This presentation is designed to help you understand and investigate why messages are stuck or accumulating on a destination. A basic understanding of how MDBs work was presented, including an overview of onMessage() and the importance of understanding what onMessage() does. This was followed by a discussion of message states and how to interpret them.

Lastly, Javacores and their importance was discussed, including an example that shows a typical hung MDB scenario and how to interpret the call stack.

It is hoped that this presentation will help you deal with stuck messages in the future!

Additional WebSphere Product Resources

- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at: http://www.ibm.com/developerworks/websphere/community/
- Learn about other upcoming webcasts, conferences and events: http://www.ibm.com/software/websphere/events_1.html
- Join the Global WebSphere User Group Community: http://www.websphere.org
- Access key product show-me demos and tutorials by visiting IBM® Education Assistant: http://www.ibm.com/software/info/education/assistant
- View a Flash replay with step-by-step instructions for using the Electronic Service Request (ESR) tool for submitting problems electronically: http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My support emails: http://www.ibm.com/software/support/einfo.html

Connect with us!

1. **Get notified on upcoming webcasts**
   Send an e-mail to wsehelp@us.ibm.com with subject line “wste subscribe” to get a list of mailing lists and to subscribe
2. **Tell us what you want to learn**
   Send us suggestions for future topics or improvements about our webcasts to wsehelp@us.ibm.com
3. **Be connected!**
   Connect with us on Facebook
   Connect with us on Twitter

Questions and Answers