



IBM Software Group

IBM WebSphere® Application Server V7.0 Feature Pack for Service Component Architecture V1.0.1

JMS binding transactions



@business on demand.

© 2009 IBM Corporation
Updated November 9, 2009

This presentation will discuss JMS binding transactions.

JMS binding quality of service overview

- JMS binding for services and references can be configured to take advantage of transactions
- SCA JMS binding supports transacting message delivery with the global transaction of a component
- SCA transaction policies are specified as intents that represent quality of service behavior offered by the JMS binding on an SCA service or reference
 - ▶ Note: SCA JMS binding does not propagate transaction context



You can configure the Service Component Architecture (SCA) Java™ Message Service (JMS) binding for services and references to take advantage of transaction quality of service behaviors.

The SCA transaction policies are specified as intents that represent quality of service behavior offered by the JMS binding on an SCA service or reference. However, note that the SCA JMS binding does not propagate transaction context; therefore, the client and service cannot participate in the same global transaction.

JMS binding and transactions

- JMS binding supports transacting message delivery with the global transaction of a component
 - ▶ JMS binding does not propagate transaction context
- References can use the `transactedOneWay` intent to process one-way requests
- Services can use the `transactedOneWay` intent to transact one-way requests only
- Services can use the `exactlyOnce` intent to transact both one-way and request-response message patterns



The SCA JMS binding supports transacting message delivery with the global transaction of a component as mentioned in the previous slide.

SCA references can use the `transactedOneWay` intent to transact one-way requests. Services can use the `transactedOneWay` intent to transact one-way requests only or the `exactlyOnce` intent to transact both one-way and request-response message patterns.

When the `transactedOneWay` intent is used on an SCA reference, a one-way request on the reference is not sent until the global transaction of the client is committed. If the global transaction of the client is rolled back, the request is not sent.

When the `transactedOneWay` intent is used on an SCA service, a one-way request is received from the JMS binding as part of the global transaction of the client. When the `exactlyOnce` intent is used on a service, both one-way and request-response message patterns are received from the JMS binding as part of the global transaction of the client. The receipt of the message and the sending of the response for request-response messaging, is not committed until the service transaction commits. If the service transaction is rolled back, the message is again made available for delivery or the message is sent to an exception destination that is based upon the configuration of the bus destination.

The SCA runtime environment typically performs a rollback of a global transaction only if the component produces an unchecked exception error. An unchecked exception error is a subclass of `java.lang.RuntimeException` or `java.lang.Error` class. A checked exception does not force a rollback. The component can force a rollback by using the `UOWSynchronizationRegistry` interface.

Configuring transactions for the JMS binding

1. Specify the `transactedOneWay` or `exactlyOnce` intents on your SCA service or reference
2. On the administrative console, configure the bus destination to handle failed messages
 - ▶ **Service integration > Buses > *bus_name* > Destinations > *destination_queue_name* or *destination_topic_space_name***



To configure transactions for the JMS binding, first, specify the `transactedOneWay` or `exactlyOnce` intents on your SCA service or reference enable transacting message delivery with the global transaction of your component. See the component example on the next slide.

Secondly, using the administrative console, configure the bus destination to handle failed messages, **Service integration > Buses > *bus_name* > Destinations > *destination_queue_name* or *destination_topic_space_name***. Reference the next slide to see the administrative console window

Destination topic space for failed messages

Integrated Solutions Console Welcome wsdemo

View: All tasks

- Welcome
- Guided Activities
- Servers
- Applications
- Services
- Resources
- Security
- Environment
- System administration
- Users and Groups
- Monitoring and Tuning
- Troubleshooting
- Service integration
 - Buses
 - Web services
 - WS-Notification
- UDDI

Cell=wsbeta151Node01Cell, Profile=AppSrv01

Buses

Buses > jmsTestbus > Destinations

A bus destination is defined on a service integration bus, and is hosted by one or more location the destination as producers, consumers, or both to exchange messages.

Preferences

New Delete Mediate Unmediate

Select Identifier Bus Type

Select	Identifier	Bus	Type
<input type="checkbox"/>	Default.Topic.Space	jmsTestbus	Topic space
Total 1			

Destination topic space name

JMS binding transactions © 2009 IBM Corporation

On the administrative console, to configure the bus destination to handle failed messages, click

Service integration > Buses > *bus_name* > Destinations > *destination_queue_name* or *destination_topic_space_name*.

Note that you can click the “New” button to create a new queue. For more information on JMS and administrative console, reference the JMS administration slides.

Configure failed messages

Buses > jmsTestbus > Destinations > Default.Topic.Space
A topic space is a location for publish/subscribe messages.

Configuration

General Properties	Message points
Identifier Default.Topic.Space	<input type="checkbox"/> Publication points
UUID 6A93A4569E29DAB1E8F5E6DB	<input type="checkbox"/> Mediation points
Type Topic space	Additional Properties
Description	<input type="checkbox"/> Context properties
Mediation	Related Items
Quality of Service	<input type="checkbox"/> Application resources topology
<input checked="" type="checkbox"/> Enable producers to override default reliability	<input type="checkbox"/> Audit Service
Default reliability Assured persistent	
Maximum reliability Assured persistent	
Default priority 0	
Exception destination	
<input checked="" type="radio"/> None <input type="checkbox"/> Override messaging engine blocked retry timeout default	
Blocked retry timeout in milliseconds -1	
<input type="radio"/> System	
<input type="radio"/> Specify \$DEFAULT_EXCEPTION_DESTINATION	
Maximum failed deliveries per message 5	
<input checked="" type="checkbox"/> Send allowed	
<input checked="" type="checkbox"/> Receive allowed	



When you click the destination queue or topic, this configuration will come up. It's from this window you configure how failed messages for the transactions are handled.

TransactedOneWay and exactlyOnce intents

```
<component name="TransactionalComponent">
  <implementation.java class="example.TransactedImpl"
    requires="managedTransaction.global"/>
    <service name="DataUpdate" requires="exactlyOnce">
      <binding.jms>
        <destination name="jms/DataUpdate_Request"
          type="queue"/>
        <activationSpec name="jms/SCA_JMS_AS"/>
      </binding.jms>
    </service>
    <reference name="loggingService" requires="transactedOneWay">
      <binding.jms>
        <connectionFactory name="jms/SCA_JMS_CF"/>
        <destination name="jms/SCA_JMS"/>
      </binding.jms>
    </reference>
</component>
```

This example shows the use of the `transactedOneWay` and `exactlyOnce` intents. In this example, the component `TransactionalComponent` receives one-way and request-response messages from the `DataUpdate` service and subsequently sends one-way requests to the `loggingService` reference transactionally. If the component transaction rolls back, the active request is queued again and any requests to the reference are not sent. As mentioned earlier, note that the schema for `binding.jms` requires the `destination` element to appear before the `connectionFactory` element for the service

Summary

- The SCA transaction policies are specified as intents that represent quality of service behavior offered by the JMS binding on an SCA service or reference
- SCA JMS binding supports transacting message delivery with the global transaction of a component



The SCA transaction policies are specified as intents that represent quality of service behavior offered by the JMS binding on an SCA service or reference. SCA JMS binding supports transacting message delivery with the global transaction of a component.

References

- SCA JMS Binding V1.0.0

http://www.osoa.org/download/attachments/35/SCA_JMSBinding_V100.pdf?version=2

- IBM Education Assistant

http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/index.jsp?topic=/com.ibm.iea.wasfpsca/plugin_coverpage.html

<http://publib.boulder.ibm.com/infocenter/ieduasst/v1r1m0/topic/com.ibm.iea.wasfpsca/wasfpsca/1.0/Bindings.html?dmuid=20081216225737946040>

- SCA white papers

http://www.ibm.com/developerworks/websphere/library/techarticles/0812_beck/0812_beck.html



Here are some useful references

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WASv7SCA101_JMSbinding_transactions.ppt

This module is also available in PDF format at: ..\\WASv7SCA101_JMSbinding_transactions.pdf



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

IBM, the IBM logo, ibm.com, and the following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of other IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>

Java, and all Java-based trademarks and logos 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.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. 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 in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2009. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.

