

JBoss World 2006
LAS VEGAS

A Step-by-Step Guide To
Developing, Configuring and
Deploying a JCA 1.5 Connector to
the JBoss Application Server

Matt Cannata and Greg Rochon
Michigan Millers Mutual Insurance Company

© JBoss Inc. 2006

Agenda

- Why use a JCA 1.5 Connector?
- Creating the ResourceAdapter
 - ✓ ResourceAdapter Implementation Details
 - ✓ Using the WorkManager
- Connecting to the ResourceAdapter
 - ✓ Required JCA Implementations
 - ✓ JCA Implementation Details
 - ✓ Required CCI Implementations
 - ✓ CCI Implementation Details
 - ✓ Putting The Connector Together
- Calling Your Connector
 - ✓ Writing the Service Descriptor
 - ✓ RAR and EAR Structure
 - ✓ Deploying Your Connector
 - ✓ Calling Your Connector
- Conclusions

JBoss World
LAS VEGAS

2

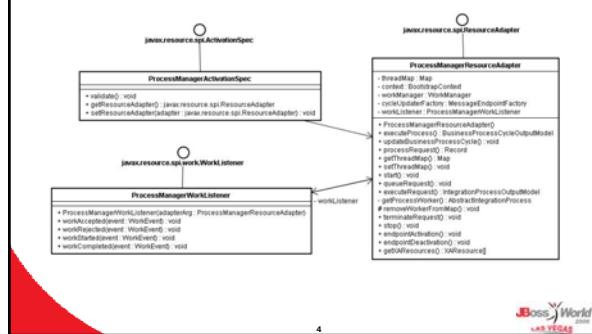
Why Use a JCA 1.5 Connector?

- Multi-Threading
 - ✓ Challenges of using multi-threading in a Java EE environment
 - ✓ Pros and Cons of using JCA over other methods
- Legacy System Connection Management
- Transaction Management
- Security

JBoss World
LAS VEGAS

3

Creating the ResourceAdapter



4

ResourceAdapter Implementation Details

- Implement `javax.resource.spi.ResourceAdapter`:
 1. `void endpointActivation(MessageEndpointFactory endpointFactory, ActivationSpec spec)`
 2. `void endpointDeactivation(MessageEndpointFactory endpointFactory, ActivationSpec spec)`
 3. `XAResource[] getXAResources(ActivationSpec[] specs)`
 4. `void start(BootstrapContext ctx)`
 5. `void stop()`
- Get the WorkManager from the BootstrapContext:


```
public void start(BootstrapContext ctxArg)
    throws ResourceAdapterInternalException {
    ctx = ctxArg;
    workManager = ctx.getWorkManager();
}
```

JBoss World
LAS VEGAS

5

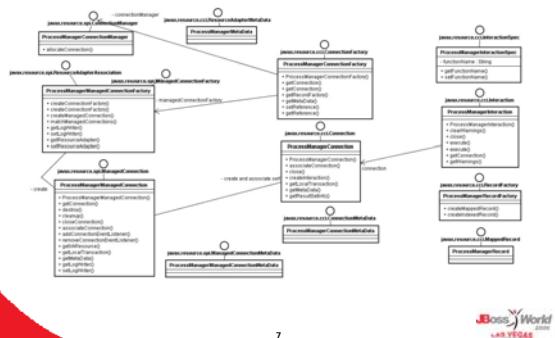
Using The WorkManager

- Running threads using the WorkManager
 - ✓ `doWork()`
 - ✓ `scheduleWork()`
 - ✓ `startWork()`
- Assigning a WorkListener and ExecutionContext
- The WorkListener Interface
 1. `void workAccepted(WorkEvent e)`
 2. `void workCompleted(WorkEvent e)`
 3. `void workRejected(WorkEvent e)`
 4. `void workStarted(WorkEvent e)`
- The Work Interface
 - ✓ `release()` method

JBoss World
LAS VEGAS

6

Connecting to the ResourceAdapter



7

Required JCA Implementations

Component	Interfaces	Purpose
ManagedConnection	<code>javax.resource.spi.ManagedConnection</code>	Represents actual connection to adapter, provide hooks for connection management.
ManagedConnectionFactory	<code>javax.resource.spi.ManagedConnectionFactory</code> , <code>javax.resource.spi.ResourceAdapterAssociation</code>	Provides interface to create managed connections, associates ResourceAdapter with connections.
ConnectionManager	<code>javax.resource.spi.ConnectionManager</code>	Provides connection management support in non-managed environments.
ManagedConnectionMetaData	<code>javax.resource.spi.ManagedConnectionMetaData</code>	Provides information about the managed connections.

8

JCA Implementation Details

- **ManagedConnection**
 - ✓ Keep reference to ResourceAdapter, initialize in constructor.
 - ✓ Maintain a list of connections, and connection event listeners as connection state.
- **ManagedConnectionFactory**
 - ✓ The ApplicationServer will call the `setResourceAdapter()` method when it creates the factory. Keep a reference to the ResourceAdapter.
 - ✓ Pass ResourceAdapter reference to ManagedConnections in the `createManagedConnection()` method.
- **ConnectionManager**
 - ✓ For un-secured resource adapters, the ConnectionManager Interface can be implemented with the following:

```
public Object allocateConnection(ManagedConnectionFactory managedConnectionFactory,
    ConnectionRequestInfo requestInfo) throws ResourceException {
    ManagedConnection mc = managedConnectionFactory.createManagedConnection(null,
        requestInfo);
    return mc.getConnection(null, requestInfo);
}
```

9

Required CCI Implementations

Component	Interfaces	Purpose
Connection	<code>javax.resource.cci.Connection</code>	Provides interface to the ManagedConnection
ConnectionFactory	<code>javax.resource.cci.ConnectionFactory</code>	Provides interface to the ManagedConnectionFactory
Interaction	<code>javax.resource.cci.Interaction</code>	Provides interface to execute ResourceAdapter methods.
InteractionSpec	<code>javax.resource.cci.InteractionSpec</code> , <code>java.io.Serializable</code>	Holds properties used when executing ResourceAdapter methods.
Record	Any sub-interface of <code>javax.resource.cci.Record</code>	Holds input to and output of ResourceAdapter methods
RecordFactory	<code>javax.resource.cci.RecordFactory</code>	Creates records for interactions.

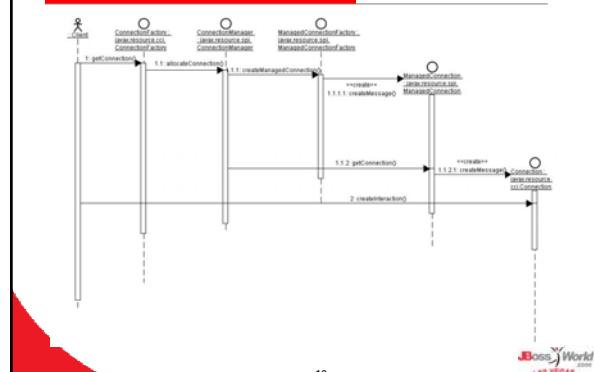
10

CCI Implementation Details

- **Connection**
 - ✓ Holds references to the ResourceAdapter and the ManagedConnection.
 - ✓ Passes the ResourceAdapter to new Interactions in the `createInteraction()` method.
- **ConnectionFactory**
 - ✓ Holds references to the ManagedConnectionFactory and ConnectionManager.
 - ✓ Calls the ConnectionManager to allocate connections.
- **Interaction**
 - ✓ Pulls arguments from Record and InteractionSpec to pass to the ResourceAdapter
 - ✓ Manages exceptions encountered during argument parsing, and request execution.
 - ✓ Returns any ResourceAdapter output to the client.

11

Putting The Connector Together



12

Writing the Service Descriptor

- Build the ra.xml file using the schema at:
http://java.sun.com/xml/ns/j2ee/connector_1_5.xsd
- Setup your ResourceAdapter class:
`<resourceadapter-class><your-resource-adapter-class></resourceadapter-class>`
- Configure the Outbound Connections:
`<outbound-resourceadapter>
 <connection-definition>
 <managed-connectionfactory-class>
 <your-managed-connection-factory-class>
 </managed-connectionfactory-class>
 <connectionfactory-interface>
 javax.resource.cci.ConnectionFactory
 </connectionfactory-interface>
 <connectionfactory-impl-class>
 <your-connection-factory-class>
 </connectionfactory-impl-class>
 <connection-interface>
 javax.resource.cci.Connection
 </connection-interface>
 <connection-impl-class>
 <your-connection-class>
 </connection-impl-class>
 </connection-definition>`

13

JBossWorld
LAS VEGAS

Writing the Service Descriptor (Cont.)

- Configure The Outbound Connections (Cont.):
`<transaction-support>NoTransaction</transaction-support>
<authentication-mechanism>
 <authentication-mechanism-type></authentication-mechanism-type>
<type>
 <credential-interface>
 javax.resource.spi.security.PasswordCredential
 </credential-interface>
 <authentication-mechanism>
 <reauthentication-support>false</reauthentication-support>
 </authentication-mechanism>
</outbound-resourceadapter>`
- Configure The Inbound Connections:
`<inbound-resourceadapter>
 <messageadapter>
 <messagelistener>
 <messagelistener-type><your-listener-interface>
 </messagelistener-type>
 <activationspec>
 <activationspec-class><your-activation-spec>
 <activationspec-class><your-activation-spec>
 </activationspec>
 </messagelistener>
 </inbound-resourceadapter>`

14

JBossWorld
LAS VEGAS

RAR and EAR Structure

- The RAR file only needs a manifest and your ra.xml file:
`meta-inf/Manifest.mf
meta-inf/ra.xml`
- Place the RAR file into the root of your EAR
- Add the module to your application.xml:

```
<module>  
  <connector><your-rar-file-name></connector>  
</module>
```

15

JBossWorld
LAS VEGAS

Deploying Your Connector

- Deploy your EAR, containing your RAR Archive
- In your deploy directory, create a MBeans descriptor file with the following information:
`<?xml version="1.0" encoding="UTF-8"?>
<connection-factories>
 <no-tx-connection-factory>
 <jndi-name><your-jndi-name></jndi-name>
 <rar-name><your-ear-name>#<your-rar-name></rar-name>
 <connection-definition>
 javax.resource.cci.ConnectionFactory
 </connection-definition>
 </no-tx-connection-factory>
</connection-factories>`

16

JBossWorld
LAS VEGAS

Calling Your Connector

- Lookup the ConnectionFactory:
`InitialContext initial = new InitialContext();
javax.resource.cci.ConnectionFactory factory =
 (ConnectionFactory)initial.lookup("java:<your-jndi-name>");`
- Get a Connection:
`javax.resource.cci.Connection connection = factory.getConnection();`
- Create an Interaction:
`javax.resource.cci.Interaction act = connection.createInteraction();`
- Create InputRecord:
`inputRecord =
 factory.getRecordFactory().createMappedRecord("<your-request-name>");`
- Execute Interaction:
`act.execute(spec, inputRecord);`

17

JBossWorld
LAS VEGAS

Conclusions

- JCA 1.5 Resource Adapters provide specification friendly thread management in the enterprise environment.
- ResourceAdapters can accept generalized messages from applications using CCI.
- ResourceAdapters can send messages to applications using Message Driven Beans.
- Working with the JCA 1.5 connector architecture is complicated.

18

JBossWorld
LAS VEGAS