Incorporating JBI Components into Java CAPS

Beta

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820–3372
05/24/2008

080828@20795

Contents

◆ ◆ ◆ CHAPTER 1

Reusing Pre-JBI Java CAPS Components in a JBI Framework

The following sections provide instructions on how to reuse repository-based ("classic") CAPS components in a JBI framework via the JBI Bridge, and how to use JBI components in the repository-based CAPS framework. If you have any questions or problems, see the Java CAPS web site at `http://goldstar.stc.com/support`.

This chapter covers the following topics:

Organization of This Work

This work is divided into the following sections:

Concepts: JBI Container and pre-JBI Java CAPS

JBI components can intercommunicate with pre-JBI Java CAPS components that are exposed as web services. The Java EE Service Engine contains all pre-JBI CAPS components and presents them to the JBI container as WSDL. JBI can thus make use of existing or new pre-JBI components that contain complex business logic and mappings, and the pre-JBI projects can call services provided by JBI components.

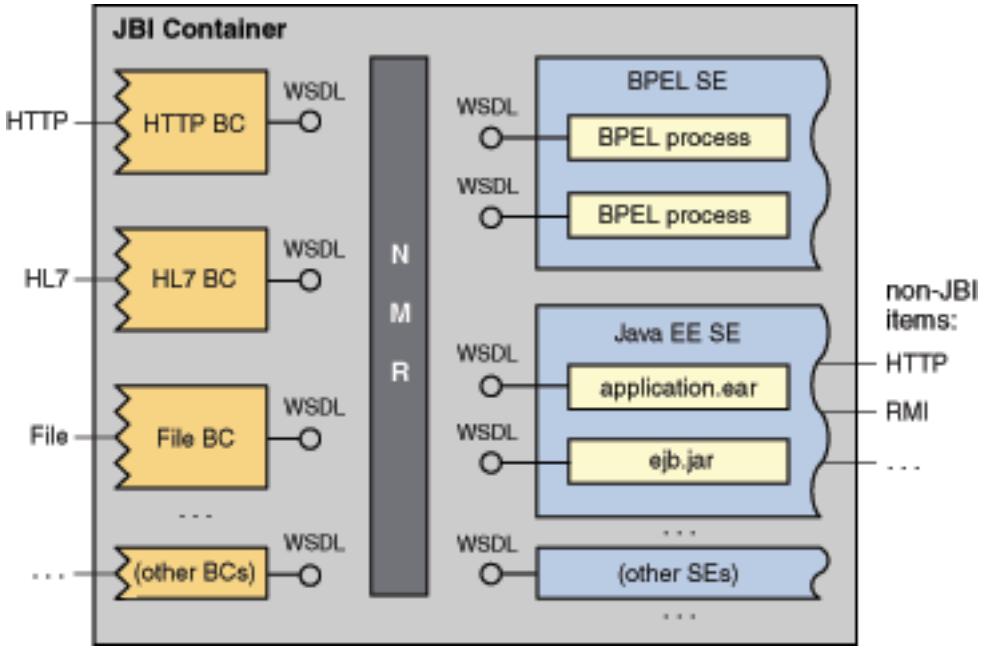

FIGURE 1–1 Intercommunication With JBI Mediated by Java EE Service Engine

For the purposes of this work, the term "pre-JBI" is used to encompass both repository-based components, such as OTDs, JCDs, and eWay adapters, as well as Java EE objects, such as JCA adapters and web service-based EJBs.

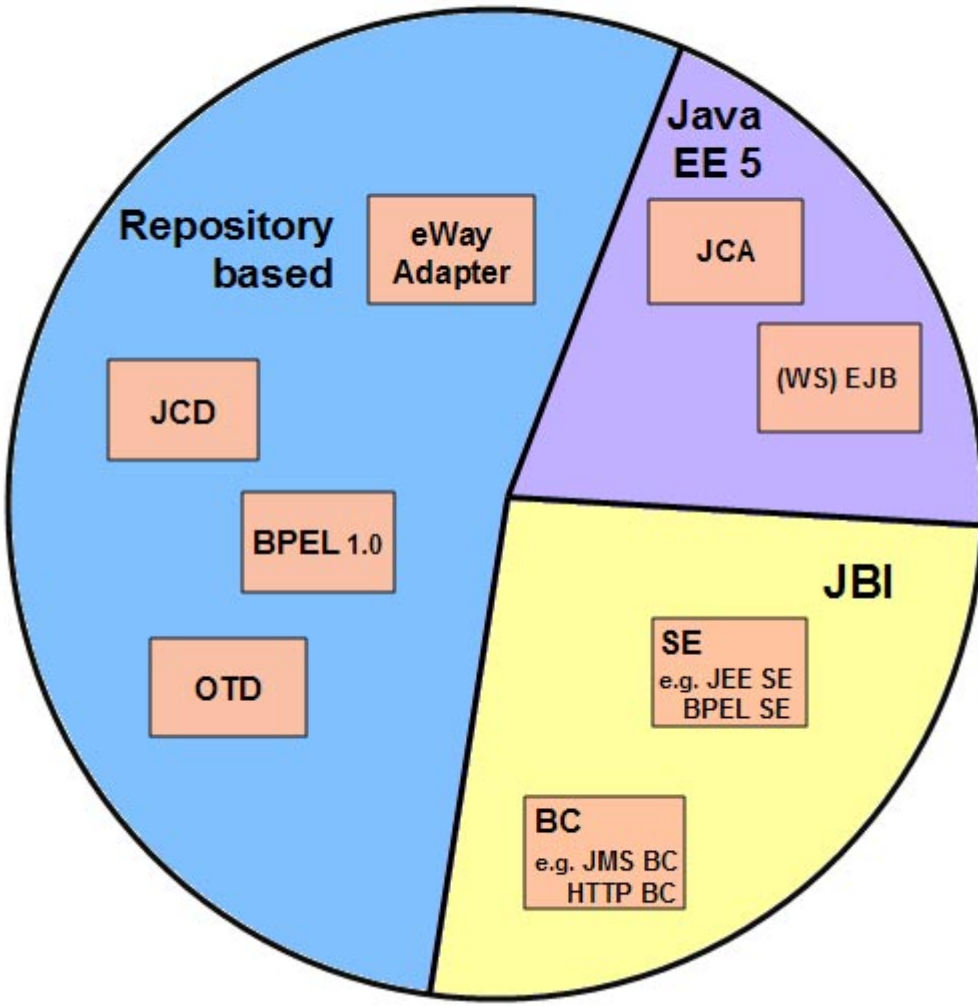

FIGURE 1–2 Categories of JBI Components and Pre-JBI Components

The JBI container can only use repository-based modules after they have been JBI-enabled. For example, a repository-based CAPS project is equipped with a "JBI hook" by adding a JBI Bridge component to a connectivity map and building a new deployment profile, creating a JBI-enabled enterprise archive (EAR) file. This EAR file is then used by the Java EE Service Engine to present WSDL to the NMR and thus to components in the JBI container.

If you have used pre-JBI technology to create EJBs or servlets, you can use the Java EE SE to expose their business logic as web services, defined in WSDL. Therefore, when they are plugged into the NMR, they intercommunicate transparently with other JBI components.

In the other direction, an existing pre-JBI CAPS project can easily be extended to use any functionality provided by JBI components, because all JBI components, by definition, are described by WSDLs that expose the web services they provide. In repository-based projects, you use the WSDL OTD to communicate with the WSDLs exposed by JBI components.

Bidirectional Reuse Scenario A: Creating a JBI-Enabled EAR File

This scenario takes you through the steps for usng a repository-based Java CAPS project and using it in a JBI project.

Why would I want to learn about this?

If you have pre-existing repository-based CAPS projects, or if you want to develop components in a repository-based environment, you can use these steps to extend "classic" CAPS into the JBI environment.

What's the big picture?

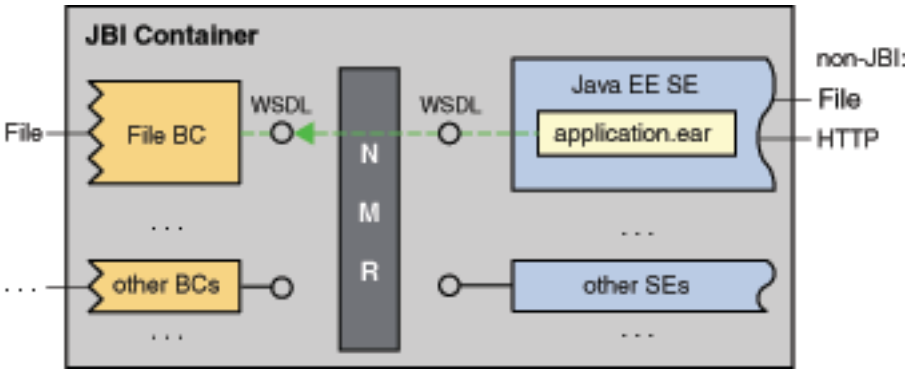

FIGURE 1–3 Using the Java EE Service Engine to Communicate with Application EAR Files

Overview of Goals

This scenario takes you through the following goals:

1. Extending a repository-based project with a JBI Bridge

   In this simple but essential goal, you add a "JBI hook" to make a non-JBI CAPS project into a JBI-enabled one.

2. Incorporating a JBI-enabled EAR file as a JBI module in a composite application project

   In this goal, you take hold of the "hook" created in the previous goal.

3. Building and deploying the Composite Application project

   In this goal, you verify that the repository-based project is sending and receiving messages through the JBI bridge.

[...]

Extending a Repository-Based Project with a JBI Bridge

Before you begin: Ensure that a valid repository-based project exists, including at least one environment, at least one Java Collaboration Definition (JCD), and at least one connectivity map.

1. Open the repository-based CAPS project.
2. For each of its environments: Add a new external system of type JBIBRIDGE.
3. For each of its connectivity maps: Drag a new instance of a JBIBRIDGE component onto the map, connect it to the input of the business logic, and configure the connection appropriately.
4. Create new deployment profiles that reference the appropriate connectivity maps and environments.
5. For each newly created deployment profile, automap its components and then build.

Result: You have created an EAR file that contains a JBI hook. Its services are now available for use within JBI projects.

Incorporating a JBI-Enabled EAR File As a JBI Module

Before you begin: If you do not already have a Composite Application project containing a BPEL module, create one. For example: 1. File New Project; category "SOA" and project type "BPEL Module"; name it prjBpel1. 2. File New Project; category "SOA" and project type "Composite Application"; name it prjCompApp1. 3. Right-click prjCompApp1 and choose Add JBI Module; then, in the Select Project dialog box, select prjBpel1, click its JAR file, and click Add project JAR Files, adding it under the JBI Modules folder.

To Incorporate a JBI-Enabled EAR File As a JBI Module

1. Right-click the Composite Application project.
2. In the Add CAPS Module wizard, step “Create CAPS Ear Link”, do the following:
   a. Supply the name of the CAPS repository.
   b. Browse to and select the deployment project.
   c. Choose an option specifying when the EAR is to be imported:
      - To import immediately, choose On add (now).
      - To import later, when the Composite Application project is first cleaned, choose On clean.
      - To postpone importation until the Composite Application project is first built, choose On build.
   d. Select an application server from the list.
3. When you have supplied all values for the wizard, click Finish.

Building and Deploying the Composite Application Project

▼ %TaskTitle%

1 ...

2 ...

Bidirectional Reuse Scenario B: Using an EJB Locally or Remotely

This scenario takes you through the steps for using a web service Enterprise Java Bean (EJB) in JBI.

Why would I want to learn about this?

If you have used EJBs or servlets to expose your business logic as a web service, the web service can be plugged directly into the NMR. This allows other applications to call into an EJB, and allows an EJB to call into another web service through the NMR.

What's the big picture?

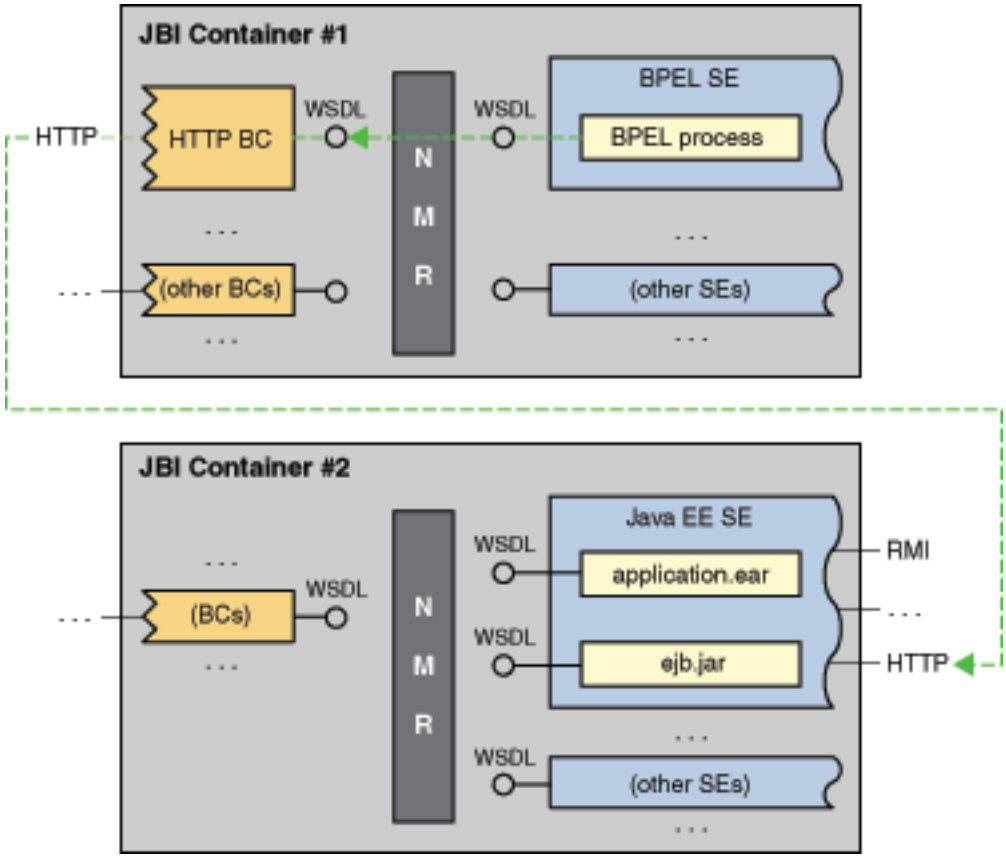

FIGURE 1–4 Calling an EJB Remotely: Multiple Containers

Overview of Goals

This scenario takes you through the following goals:

1. Creating a repository-based CAPS project with OTDs

   This step is optional if you already have a repository-based CAPS project whose OTDs you want to use in an EJB project.

2. Adding connector resources to the GlassFish server

   Because this scenario uses JMS for communication, you will add resources for two JMS queues.

3. Setting up an EJB Module project that uses OTDs and MDBs

   You will use the OTD Importer and create a message-driven bean (MDB).

4. Adding and customizing code snips from the EJB Palette

You will use drag-and-drop from the palette and the JMS JCA Wizard to seed your code with skeleton constructs. The actual implementation can vary.

[...]

Creating a Repository-Based CAPS Project with OTDs

%%%

▼ To Verb an Object

Before You Begin Do the needful.

1 **%%%**

2 **%%%**

▼ To Verb an Object

Before You Begin Do the needful.

1 **%%%**

2 **%%%**

Adding Connector Resources to the GlassFish Server

%%%

▼ To Verb an Object

Before You Begin Do the needful.

1 **%%%**

2 **%%%**

▼ To Verb an Object

Before You Begin Do the needful.

1 **%%%**

2 **%%%**

Setting Up an EJB Module Project that Uses OTDs and MDBs

%%%

▼ To Verb an Object

Before You Begin Do the needful.

1 **%%%**

2 **%%%**

Adding and customizing code snips from the EJB Palette

%%%

▼ To Verb an Object

Before You Begin Do the needful.

1 **%%%**

2 **%%%**

Bidirectional Reuse Scenario C: Using JBI Functionality in a pre-JBI CAPS Project

This scenario takes you through the steps for using JBI functionality, exposed through its WSDLs, in a repository-based CAPS project, using the WSDL OTD to communicate.

Why would I want to learn about this?

If you have JBI-compliant components whose services you want to use in repository-based CAPS projects, or if you want to use BPEL 2.0 in "classic CAPS", you can use these steps to invoke WSDL-mediated web services from JBI in a repository-based environment.

Overview of Goals

This scenario takes you through the following goals:

1. Creating a BPEL 2.0 project and incorporating it as a JBI module in a composite application project
2. Building and deploying the Composite Application project
3. Creating a repository-based CAPS project
4. Accessing the WSDL-mediated JBI web services
5. Building and deploying the repository-based CAPS project

Creating a BPEL 2.0 Project

%%%

▼ To Verb an Object

Before You Begin Do the needful.

1 **%%%**

2 **%%%**

Building and Deploying the Composite Application Project

%%%

▼ To Verb an Object

Before You Begin Do the needful.

1 **%%%**

2 **%%%**

Creating a Repository-Based CAPS project

%%%

▼ To Verb an Object

Before You Begin Do the needful.

1 %%%

2 %%%

Accessing WSDL-Mediated JBI Web Services

%%%

▼ To Verb an Object

Before You Begin Do the needful.

1 %%%

2 %%%

Building and Deploying the Repository-Based CAPS Project

%%%

▼ To Verb an Object

Before You Begin Do the needful.

1 %%%

2 %%%