Overview

IBM's WebSphere Optimized Local Adapters come shipped with a set of samples to help illustrate the usage of WOLA.

One of the samples is an EJB that serves as the target for inbound[1] communications over WOLA. That sample -- `OLASample2.ear` -- is provided as an EJB 2.1 specification level EJB.

The question comes up: "May the target be EJB 3.0[2]?" The answer to that is "Yes."

The next question that comes up: "Is there a sample?" The answer to that is "Yes."

This document supplies an illustration of the EJB 3.0 sample[3]. Two EAR files -- one without a bindings XML file, the other with a bindings XML file -- accompany this document at the WP101490 Techdoc[4] at `ibm.com/support/techdocs`.

Overview

The sample application is really pretty simple. It is *not* an echo program; it simply takes whatever you send it, issues a `System.out.println()`, then returns null. But it does serve to illustrate how an EJB 3.0 program would be implemented to use WOLA.

The source is provided inline here, with commentary provided in separate text boxes:

```
/**
 * The following code is sample code created by IBM Corporation.
 * This sample code is not part of any standard IBM product and
 * is provided to you solely for the purpose of assisting you in
 * the development of your applications.  The code is provided
 * 'as is', without warranty or condition of any kind.  IBM shall
 * not be liable for any damages arising out of your use of the
 * sample code, even if IBM has been advised of the possibility
 * of such damages.
 */
package com.ibm;

import javax.ejb.Stateless;
import javax.ejb.RemoteHome;

/**
 * Session Bean implementation class OLAEJB3TestBean
 */
@Stateless
```

When an EJB serves as the target for inbound calls over WOLA, it must be defined as stateless.

```
@RemoteHome(com.ibm.websphere.ola.ExecuteHome.class)
```

The `@RemoteHome` annotation must be defined and set to `com.ibm.websphere.ola.ExecuteHome`.

That class (and other WOLA support classes) are included in the file `ola_apis.jar`, which is found at the following locations:

- V7.0.0.4 or later V7 levels: `/<install_root>/mso/OLA/lib/ola_apis.jar`
- V8.0.0.0 or later: `/<install_root>/util/zos/OLASamples/lib/ola_apis.jar`

[1] The term "inbound" is from the perspective of WAS z/OS. The communication initiates from the outside address space and comes inbound to WAS z/OS. In that model, the target of the communication must be a stateless EJB.

[2] Support for EJB 3.0 was first provided with WAS V6.1 (all platforms). EJB 3.1 came into support in V8.

[3] Credit to Jim Mulvey and Tim Kaczynski of WAS z/OS development, and Lee-Win Tai of the IBM ATS organization.

[4] URL: `http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101490`

```
public class OLAEJB3TestBean
```

Notice that the EJB class does not "implement" anything -- this is because the remote interface cannot be overridden for OLA. It is `com.ibm.websphere.ola.Execute`, and the EJB3 infers this from the `@RemoteHome` annotation.

Both the `ExecuteHome` and `Execute` classes are included in the file `ola_apis.jar`, which is found at the following locations:

- V7.0.0.4 or later V7 levels: `/<install_root>/mso/OLA/lib/ola_apis.jar`
- V8.0.0.0 or later: `/<install_root>/util/zos/OLASamples/lib/ola_apis.jar`

```
{
    /**
     * Default constructor.
     */
    public OLAEJB3TestBean()
    {
    }

    /**
     * Execute method
     */
```

This is the only method called for WOLA. When your COBOL[5] application uses WOLA to call this EJB, this is the method that gets control. The input parameters are passed in `byte[] inbytes`, and your return parameters are returned as `byte[]`.

In this sample nothing is done with input parameters, and nothing is returned. It simply prints an output line to indicate the method had been successfully invoked.

```
    public byte[] execute(byte[] inbytes)
    {
      System.out.println(" !!TDK inside EJB3 execute method");
      return null;
    }
}
```

IBM Rational Application Developer

IBM Rational Application Developer first supported the Java EE 5 specification (along with EJB 3.0) in the Version 7.5 release. Other IDE platforms can be used to develop EJB 3.0.

Two other points related to the use of the tooling:

- When you create an EJB, make sure it is EJB 3.x and *not* EJB 1.x - 2.x
- Your EJB project/module should be set to 3.0 or higher.

EJB JNDI Name

The JNDI name on the remote interface is also the WOLA "service" name. It is what the outside COBOL (or other language) program uses on the `BBOA1INV`[6] API `requestservicename` parameter. When the call comes over WOLA, the last leg of the journey is a lookup of the target EJB using the JNDI name specified as the "service" name.

The JNDI name may be any valid string of characters. The point here is that whatever value the outside program expects with its coding of the service name should be matched as the value for the remote interface JNDI name.

[5] WOLA supports COBOL, C/C++, High Level Assembler and PL/I for its native language APIs.

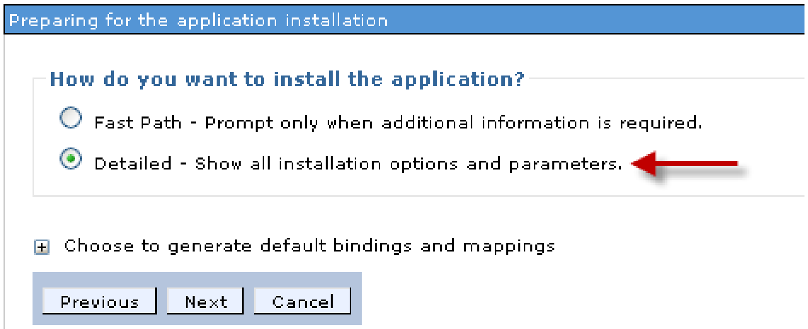
[6] Or `BBOA1SRQ`. For more on the native APIs, see the InfoCenter and search on the string `olaapis`, or see the "Native API Primer" at the WP101490 Techdoc at `ibm.com/support/techdocs`.

There are two ways[7] you can set this JNDI name:

1. Specify it in the Administrative Console at time of application deployment, or
2. Specify it in a bindings file that's included with the application

Specify JNDI in Admin Console

The supplied `OLAEJB3.ear` file has no bindings and no specification of the JNDI. You may set the value in the Admin Console by checking the "Detailed" radio button at the start of the installation:



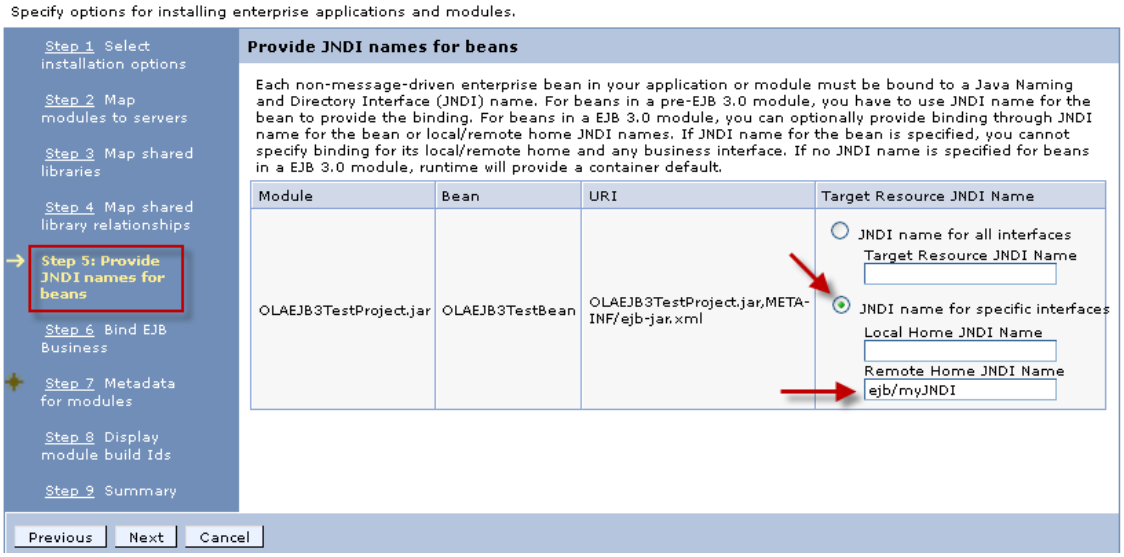
Preparing for the application installation

How do you want to install the application?

○ Fast Path - Prompt only when additional information is required.

● Detailed - Show all installation options and parameters.

And then specifying a value[8] on Step 5 of the installation:



Specify options for installing enterprise applications and modules.

Step 1 Select installation options
Step 2 Map modules to servers
Step 3 Map shared libraries
Step 4 Map shared library relationships
→ Step 5: Provide JNDI names for beans
Step 6 Bind EJB Business
Step 7 Metadata for modules
Step 8 Display module build Ids
Step 9 Summary

Provide JNDI names for beans

Each non-message-driven enterprise bean in your application or module must be bound to a Java Naming and Directory Interface (JNDI) name. For beans in a pre-EJB 3.0 module, you have to use JNDI name for the bean to provide the binding. For beans in a EJB 3.0 module, you can optionally provide binding through JNDI name for the bean or local/remote home JNDI names. If JNDI name for the bean is specified, you cannot specify binding for its local/remote home and any business interface. If no JNDI name is specified for beans in a EJB 3.0 module, runtime will provide a container default.

Module	Bean	URI	Target Resource JNDI Name
OLAEJB3TestProject.jar	OLAEJB3TestBean	OLAEJB3TestProject.jar,META-INF/ejb-jar.xml	○ JNDI name for all interfaces Target Resource JNDI Name ● JNDI name for specific interfaces Local Home JNDI Name Remote Home JNDI Name ejb/myJNDI

When you save, synchronize and start the application, you will see the following in the `SYSPRINT` of the servant region:

```
BBOO0222I: CNTR0167I: The server is binding the  com.ibm.websphere.ola.ExecuteHome
interface of the OLAEJB3TestBean enterprise bean in the OLAEJB3TestProject.jar module of
the OLAEJB3 application.  The binding location is: ejb/myJNDI
```

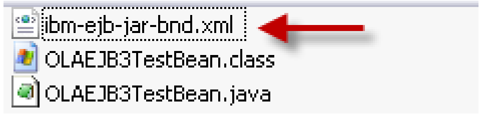
That's the service name the outside program should use.

Specify JNDI in a bindings file

The supplied `OLAEJB3_binding.ear` file has a bindings file as part of the embedded JAR file. If you open that JAR file you will see:

[7] You could also leave the value unspecified and allow WAS to auto-generate it. But that means checking in SYSPRINT what value is generated. And the value is very long and cumbersome. Easier to specify one of your own choosing.

[8] The value you provide may be whatever you wish with no embedded spaces. Common convention is for it to start with `ejb/` as shown here.



ibm-ejb-jar-bnd.xml
OLAEJB3TestBean.class
OLAEJB3TestBean.java

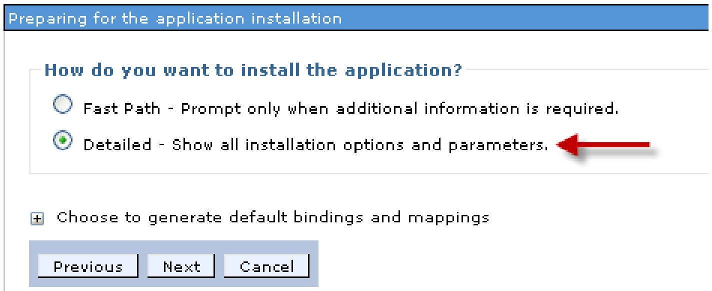
The XML file supplied in the sample EAR file has the following line:

```
<session name="OLAEJB3TestBean"
      remote-home-binding-name="ejb/com/ibm/websphere/MyCustomJNDIName"/>
```

That specifies the JNDI name that will be bound to the EJB when it is installed. You may set the JNDI value to whatever value you wish.

However, there is a necessary step in the Admin Console to make that value take effect:

- First, as before, select "Detailed" at the start of the installation:



Preparing for the application installation

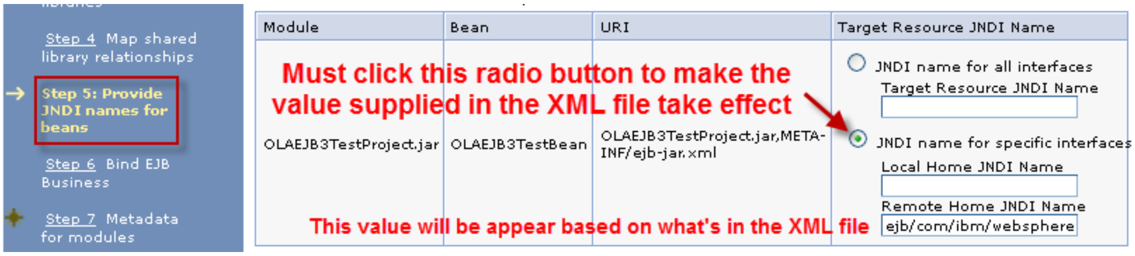
How do you want to install the application?

○ Fast Path - Prompt only when additional information is required.

● Detailed - Show all installation options and parameters.

⊞ Choose to generate default bindings and mappings

- Then on Step 5 of the installation, select the radio button



Step 4 Map shared library relationships

→ Step 5: Provide JNDI names for beans

Step 6 Bind EJB Business

Step 7 Metadata for modules

Module	Bean	URI	Target Resource JNDI Name
OLAEJB3TestProject.jar	OLAEJB3TestBean	OLAEJB3TestProject.jar,META-INF/ejb-jar.xml	○ JNDI name for all interfaces Target Resource JNDI Name ● JNDI name for specific interfaces Local Home JNDI Name Remote Home JNDI Name ejb/com/ibm/websphere

Must click this radio button to make the value supplied in the XML file take effect

This value will be appear based on what's in the XML file

Then when you finish the installation and start the application, you should see the following in the `SYSPRINT` of the servant region:

```
BBOO0222I: CNTR0167I: The server is binding the com.ibm.websphere.ola.ExecuteHome interface of
the OLAEJB3TestBean enterprise bean in the OLAEJB3TestProject.jar module of the OLAEJB3_binding
application.  The binding location is: ejb/com/ibm/websphere/MyCustomJNDIName
```

That's the service name the outside program should use.

Note: The InfoCenter has a detailed article on the format and usage of that bindings file. If interested, you may access that article with the following URL:

`http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp?topic=/com.ibm.websphere.zseries.doc/ae/cejb_bindingsejbfp.html`

Or, go to the InfoCenter and simply search on the unique string `cejb_bindingsejbfp`

Or, if viewing this as a PDF, using this link to access the page directly.

End of Document