Controlling Applications - EntireX Wrapper for Enterprise JavaBeans

This chapter covers the following topics:

- Environment Entries to Control EJB
- Using Security/Encryption
- Using Natural Security
- Using Compression
- Using Internationalization with Wrapper for EJB
- Tracing
- Deployment with an Application Server

Environment Entries to Control EJB

To define the behavior of the EJB

- Use the following environment entries

Entry Name	Default Value	Description
Broker	Localhost:1971	Broker ID
Server	RPC/SRV1/CALLNAT	Service name (class/server/service)
User	EjbUserName	Broker user ID
Password	null	Broker user password
Compression Level	no	Compression level (No,Yes,0-9)
Codepage	false	Use codepage
Security	false	Use EntireX security
Encryption	0	Encryption level (0,1,2)
Natural Security	false	Use Natural Security
RPC User	null	RPC user ID
RPC Password	null	RPC user password
Verbose	false	Bean log information
Trace	0	Java ACI trace level (0,1,2)
Logfile	null	Java ACI trace file

Using Security/Encryption

To use EntireX Security

- Set entry name `Security` to "true", User to the current user and Password to the current password.

With these settings the Broker checks user and password.

To use EntireX Security with encryption level broker

- Set entry name `Security` to "true", Encryption equals 1, `User` to the current user and `Password` to the current password.

With these settings the Broker checks user and password, the data of send and receive calls are encrypted from the application server to the Broker.

To use EntireX Security with encryption level target

- Set entry name `Security` to "true", Encryption equals 2, `User` to the current user and `Password` to the current password.

With these settings the Broker checks user and password, and the data of send and receive calls are encrypted from the application server to the RPC server.

Using Natural Security

To use Natural Security

- Set entry name `Natural Security` to "true", `User` to the current Natural user and `Password` to the current Natural password.

With these settings the Broker checks Natural user and Natural password.

If you want to use EntireX Security and Natural Security and the Natural Security user name is not the same as the EntireX Security user name, set entry name `RPC User` to the current Natural user. If the Natural Security password is the same as the EntireX Security password, set entry name `RPC User` to the current Natural user.

Using Compression

To use EntireX Compression

- Set entry name `Compression level` to one of the following values:

Value	Description	Data Type
9	Best Compression	string
1	Best Speed	string
Y(es) or 6	Default Compression	string
8	Deflated	string
N(o) or 0	No Compression	string

Using Internationalization with Wrapper for EJB

It is assumed that you have read the document *Internationalization with EntireX* and are familiar with the various internationalization approaches described there.

EntireX Java components use the codepage configured for the Java virtual machine (JVM) to convert the Unicode (UTF-16) representation within Java to the multibyte or single-byte encoding sent to or received from the broker by default. This codepage is also transferred as part of the locale string to tell the broker the encoding of the data if communicating with a broker version 7.2.x and above.

To change the default, see your JVM documentation. On some JVM implementations, it can be changed with the `file.encoding` property. On some UNIX implementations, it can be changed with the `LANG` environment variable.

Which encodings are valid depends on the version of your JVM. For a list of valid encodings, see *Supported Encodings* in your Java documentation. The encoding must also be a supported codepage of the broker, depending on the internationalization approach.

With the entry name `Codepage` you can

- force a locale string to be sent if communicating with Broker version 7.1.x and below. Set entry name `Codepage` to "true" for this purpose.
- not use a codepage other than the default encoding of the JVM.

Tracing

To switch on bean tracing

- Set entry name `Verbose` to "true".

Some EJB-relevant information is written to the standard output of the application server.

To switch on Java ACI tracing

- Set entry name `Trace` to "0", "1", "2" or "3".

  The output will be written to standard output.

Trace Level	Description
0	No trace output
1	Trace all Broker calls and other major actions
2	Dump the send-and-receive buffer
3	Dump the buffers sent to the Broker and received from the Broker

To write the trace information to a file

- Set the entry name `Logfile` to the corresponding file.

Deployment with an Application Server

General Information

Note:
Version numbers to third-party products are listed centrally in the section *EntireX Prerequisites*. In the examples, notation such as *nn* or *nnn* refers to the two- or three-digit version numbers.

Make sure that the application server can access the EntireX Java Runtime.

For the compilation of the generated sources and the generation of the deployment descriptors we use Ant and XDoclet.

Load, install, and verify Ant. See *<cd-root>/share/3rdparty/ant nnn.zip* or *http://ant.apache.org/*.

Load, install, and verify XDoclet. See *<cd-root>/share/3rdparty/xdoc nnn.zip* or *http://sourceforge.net/projects/xdoclet/*.

Windows users:
If your path names or file names contain blanks, you have to use the short name notation without blanks, e.g.: Use *C:\PROGRA~1\XYZ* instead of *C:\Program Files\xyz*

Please refer to the vendor for more details about the application server used.

Deployment with a WebSphere Application Server under UNIX

Note:
The following description has been verified with the delivered EJB example and WebSphere 5 on RedHat Linux Advanced Server 2.1 for x86 and might be different for other WebSphere releases.

The following steps are described in this section:

- Compiling the EJB Example
- Generating Code for Deployment
- Creating an Application Client
- Creating an Application
- EJB Deployment

Compiling the EJB Example

To compile the example

1. You can use the *build.xml* file as a template, which is provided in directory *examples/EJBWrapper*.

2. Change the assignment *antfile="example.xml"* according to your own XML file name.

3. Adapt the basedir and the following properties in the *build.xml* file:

   ```
   <project name="Example" default="all" basedir="/SAG/exx/vnnn/examples/ejb_wrapper">
      <target name="init">
        <property name="exx.classes" value="/SAG/exx/vnnn/classes/entirex.jar"/>
        <property name="ant.classes" value="/SAG/antnnn/lib/ant.jar"/>
        <property name="xdoclet.classes" value="/SAG/xdocnnn/lib/xdoclet.jar"/>
      </target>
      <target name="ibm_init" depends="init">
        <property name="java.classes" value="/opt/jdknnn_06/lib/tools.jar"/>
        <property name="j2ee.classes" value="/opt/j2sdkeennn/lib/j2ee.jar"/>
      </target>
   ```

4. Compile the bean components:

   ```
   ant -buildfile build.xml ibm_compile
   ```

5. Compile the client main program:

   ```
   ant -buildfile build.xml ibm_compile_client
   ```

Generating Code for Deployment

To generate code for deployment

1. Start the Assembly tool of WebSphere:

   ```
   /opt/WebSphere/AppServer/bin/assembly.sh
   ```

2. From the **New** tab choose EJB Module.

   In the left frame of the screen the types of beans belonging to the module are listed (Session Beans, Entity Beans, Message Driven Beans, etc.), while the right frame shows the properties.

3. Create a new deployment descriptor for the bean by selecting **Session Beans** and choosing **New**.

   The window **New Session Bean** will be displayed.

4. In the tab **General** enter the EJBName, the EJB class, and in the Remote section Home and Interface:

   ```
   EJB name:  EJBExampleBean
   EJB class: example.ejb.EJBExampleBean
   Remote
   Home:      example.interfaces.EJBExampleHome
   Interface: example.interfaces.EJBExample
   ```

5. In the tab **Advanced** choose session type Stateful.

6. In the tab **Bindings** set the JNDI Name

   ```
   JNDI name: EJBExample
   ```

7. Confirm with **OK**.
8. On the left screen choose **Add Files**.
9. Add all class files created by the *EntireX Workbench* EJB function and compile.
   - Browse and select *exx/vnnn/examples/ejb_wrapper* and Add Client.class.
   - Browse to *exx/vnnn/examples/ejb_wrapper* and choose **EJB**.
   - Select **example** in the right window and choose **Add.**

     (it is important that the selected files have a path name such as *example/ejb/file.class*).
10. Add all class files of *entirex.jar*
    - Browse to *exx/vnnn/classes* and choose *entirex.jar*.

      Select **com**, choose **Add** and **OK** (do not add the META-INF).
    - Save the archive EJBExample.jar.
11. From the **File** menu, choose **Generate Code** for deployment... and then **Generate Now**.

    A file *Deployed_EJBExample.jar* will be generated.
12. Close the bean module.

Creating an Application Client

To create an application client

1. Create a new application client. From the **File** menu, choose **New** and **Application Client**.
2. Change the Display name and complete the text field **Client** in the Main class.
3. Select **EJB References** and choose **New**.
4. In the tab **General** enter the Bean name *ejb/MyExample*.
5. Obtain the environment naming context of the application client.
6. Enter the Home and Remote interface:

   ```
   Name:   ejb/MyExample
   Home:   example.interfaces.EJBExampleHome
   Remote: example.interfaces.EJBExample
   ```

7. In the tab **Bindings** enter the JNDI name.

   ```
   JNDI name: EJBExample
   ```

8. Save the application client (*ApplicationClient.jar*).

Creating an Application

To create a new application

1. From the **File** menu, choose **New** and choose **Application**.
2. Change the Display name.
3. Select Application Clients and import *ApplicationClient.jar*.
4. Select EJB Module and import *Deployed_EJBExample.jar*.
5. Save the application as *Application.ear*.

EJB Deployment

To deploy the EJB

1. Open an Administration console in a Browser (http://localhost:9090/admin).
2. From the **File** menu, choose **Applications** and choose **Install New Application** and enter the path to your application (enter path to the file *Application.ear*).
3. Choose **Next**.
4. Specify the prefix you have already chosen for the client application (*etb/MyExample*) and choose **Do not override existing bindings**.
5. Choose **Next**.
6. Choose **Enable class reloading** and **Next**.
7. Verify the JNDI name and choose **Next**.
8. Choose EJB Module, **Next** and then **Finish**.
9. Save the configuration with **Save to Master Configuration**.
10. Choose **Enterprise Applications** and start the application.