How to migrate your application from JBoss v2.4 to JOnAS v3.0

Emmanuel Cecchet
INRIA

emmanuel.cecchet@inrialpes.fr

Julie Marguerite
INRIA

julie.marguerite@inrialpes.fr

François Exertier
BULL

francois.exertier@bull.net

How to migrate your application from JBoss v2.4 to JOnAS v3.0

by Emmanuel Cecchet, Julie Marguerite, and François Exertier

Guide for JOnAS v3.0 and JBoss v2.4 Edition

Table of Contents

Chapter 1. Introduction

In this document we explain how to migrate your application from JBoss v2.4 to JOnAS v3.0. Note that this guide is also valid for JOnAS versions from JOnAS 2.4 to JOnAS 3.0, and that EJB 2.0 stuff is not addressed. We assume that JOnAS is already installed. If you need to install JOnAS check the documentation (http://www.objectweb.org/jonas/doc/index.html).

Note that you can find more detailed information about JBoss and JOnAS in the documentation provided with each software. However, we will try to give all the necessary information so that you don't have to go deeply in all those documentations.

Basically, you won't have to modify the beans implementation but you will have to generate the JOnAS specific deployment descriptor and to generate container classes using a tool called GenIC.

We start by presenting how to migrate your deployment descriptors from JBoss to JOnAS. Then, we explain how to generate container classes for JOnAS and how to lookup UserTransaction from the client side. You can find the complete deployment descriptors and the code of the beans we use in the examples in the appendices at the end of this document.

Note: This document does not cover message driven beans.

Chapter 2. Migrating your deployment descriptors

2.1. Introduction

This section explains the differences in the deployment descriptors of JBoss and JOnAS. As an example, we use a stateless session bean named **SB_BrowseRegions** and an entity bean called **Region**.The underlying database in called **rubis**. SB_BrowseRegions is used to retrieve the list of regions from the database.

In both JBoss and JOnAS, the deployment descriptors must be in the `META-INF` directory.

2.2. The ejb-jar.xml file

This file is valid for both JOnAS and JBoss. It is used by the EJB server to identify which classes are the home and remote interfaces and the implementation of each bean. It is also used to describe the interactions between the different beans. Note that there is a slight difference between JOnAS and JBoss regarding inter-EJB references (see Section 2.5).

2.3. Migrating JBoss specific descriptors

You have to integrate both jboss.xml and jaws.xml in the JOnAS specific deployment descriptor named `jonas-ejb-jar.xml`. This file contains, for each bean, the JNDI name of the home object, EJB references, datasource, JMS administered objects and the information about the mapping of the bean to the database. Note that the JNDI configuration used in JOnAS does not support hierarchical namespace so if you used hierarchical JNDI names with jboss.xml you have to change them.

2.3.1. Migrating jboss.xml

The `jboss.xml` contains the JNDI name of the beans and of the datasource. You will have to move this to the `jonas-ejb-jar.xml` file.

2.3.1.1. Session Bean

This is the `jboss.xml` file for SB_BrowseRegion:

```
<jboss>
    <session>
      <ejb-name>SB_BrowseRegions</ejb-name>
      <jndi-name>SB_BrowseRegionsHome</jndi-name>
      <resource-ref>
        <res-ref-name>jdbc/rubis</res-ref-name>
        <resource-name>rubis</resource-name>
      </resource-ref>
    </session>
</jboss>
```

The equivalent `jonas-ejb-jar.xml` file looks like:

```
<jonas-ejb-jar>
    <jonas-session>
        <ejb-name>SB_BrowseRegions</ejb-name>
        <jndi-name>SB_BrowseRegionsHome</jndi-name>
        <jonas-resource>
            <res-ref-name>jdbc/rubis</res-ref-name>
            <jndi-name>mysql</jndi-name>
        </jonas-resource>
    </jonas-session>
</jonas-ejb-jar>
```

2.3.1.2. Entity Bean

This is the `jboss.xml` file for Region:

```
<jboss>
    <entity>
      <ejb-name>Region</ejb-name>
      <jndi-name>RegionHome</jndi-name>
      <resource-ref>
        <res-ref-name>jdbc/rubis</res-ref-name>
        <resource-name>rubis</resource-name>
      </resource-ref>
    </entity>
</jboss>
```

Here is the equivalent `jonas-ejb-jar.xml` file:

```
<jonas-ejb-jar>
    <jonas-entity>
        <ejb-name>Region</ejb-name>
        <jndi-name>RegionHome</jndi-name>
        <jonas-resource>
            <res-ref-name>jdbc/rubis</res-ref-name>
            <jndi-name>rubis</jndi-name>
        </jonas-resource>
    </jonas-entity>
</jonas-ejb-jar>
```

2.3.2. Migrating jaws.xml

The `jaws.xml` file contains the mapping of the beans to the database tables and the finders declaration. You will have to move this to the `jonas-ejb-jar.xml` file.

This is the `jaws.xml` file for Region:

```
<jaws>
   <entity>
      <ejb-name>Region</ejb-name>
      <table-name>regions</table-name>
      <create-table>false</create-table>
      <cmp-field>
         <field-name>id</field-name>
         <column-name>id</column-name>
      </cmp-field>
      <cmp-field>
         <field-name>name</field-name>
         <column-name>name</column-name>
      </cmp-field>
   </entity>
</jaws>
```

Here is the equivalent `jonas-ejb-jar.xml` file:

```
<jonas-ejb-jar>
    <jonas-entity>
```

```
        <jdbc-mapping>
            <jndi-name>mysql</jndi-name>
            <jdbc-table-name>regions</jdbc-table-name>
            <cmp-field-jdbc-mapping>
                <field-name>id</field-name>
                <jdbc-field-name>id</jdbc-field-name>
            </cmp-field-jdbc-mapping>
            <cmp-field-jdbc-mapping>
                <field-name>name</field-name>
                <jdbc-field-name>name</jdbc-field-name>
            </cmp-field-jdbc-mapping>
        </jdbc-mapping>
    </jonas-entity>
</jonas-ejb-jar>
```

2.4. Declaring finders

JBoss automatically generates the following finders if they are declared in the Home interface:

- findAll()
- findByPrimaryKey(primaryKey)
- findByFieldName(fieldValue) where FieldName is the name of a field in your bean.

JOnAS does not automatically generate these finders so you will have to declare them in `jonas-ejb-jar.xml`. Here is an example of the declaration of these finders as they should be in JOnAS. The bean Region has a field id that matches its primary key and a field name which is the name of the region.

```
<jonas-ejb-jar>
 <jonas-entity>
        <ejb-name>Region</ejb-name>
        <jdbc-mapping>
            <finder-method-jdbc-mapping>
                <jonas-method>
                    <method-name>findByPrimaryKey</method-name>
                </jonas-method>
                <jdbc-where-clause>where id=?</jdbc-where-clause>
            </finder-method-jdbc-mapping>

            <finder-method-jdbc-mapping>
                <jonas-method>
                    <method-name>findByName</method-name>
                </jonas-method>
```

```
                <jdbc-where-clause>where name=?</jdbc-where-clause>
            </finder-method-jdbc-mapping>

            <finder-method-jdbc-mapping>
                <jonas-method>
                    <method-name>findAll</method-name>
                </jonas-method>
                <jdbc-where-clause></jdbc-where-clause>
            </finder-method-jdbc-mapping>
        </jdbc-mapping>
 </jonas-entity>
</jonas-ejb-jar>
```

With JBoss you have to declare customized finders in `jaws.xml`. You will also have to move these finders in the jonas-ejb-jar.xml file. Note that there is a slight syntax difference between JBoss and JOnAS regarding unknown values in queries. You don't have to number the parameters with JOnAS. Example of a customized finder declaration in JBoss:

```
<jaws>
  <finder>
       <name>findUserCurrentSellings</name>
       <query>
         items.seller={0} AND items.end_date>=NOW()
       </query>
       <order></order>
  </finder>
</jaws>
```

This finder should be written this way with JOnAS:

```
<jonas-ejb-jar>
 <finder-method-jdbc-mapping>
     <jonas-method>
         <method-name>findUserCurrentSellings</method-name>
     </jonas-method>
     <jdbc-where-clause>
       WHERE items.seller=? AND items.end_date>=NOW()
     </jdbc-where-clause>
 </finder-method-jdbc-mapping>
</jonas-ejb-jar>
```

2.5. Declaring EJB references

More information about EJB references in JOnAS, including local references from EJB 2.0, may be found in the JOnAS documentation (http://www.objectweb.org/jonas/current/doc/PG_Environment.html#EJBRef).

2.5.1. References between beans in the same jar

This section explains how to declare references between beans that are packaged in the same jar. As only the standard deployment descriptor `ejb-jar.xml` is concerned, there is no difference between JOnAS and JBoss

With both JBoss and JOnAS you have to declare an ejb-ref element in the `ejb-jar.xml`. and you must also declare an ejb-link attribute in the ejb-ref element of the bean. There is nothing to add in the JBoss or JOnAS specific descriptors. In the following example the session bean SB_BrowseRegions calls methods on the Region entity bean. So for SB_BrowseRegions you need an ejb-ref element in the `ejb-jar.xml` file that references the bean Region:

Common `ejb-jar.xml` file for SB_BrowseRegions:

```
<ejb-jar>
    <enterprise-beans>
       <session>
         <description>Deployment descriptor for SB_BrowseRegions Bean</description>
            <ejb-ref>
               <description>This is the reference to the region bean</description>
               <ejb-ref-name>ejb/Region</ejb-ref-name>
               <ejb-ref-type>Entity</ejb-ref-type>
               <ejb-link>Region</ejb-link>
               <home>edu.rice.rubis.beans.RegionHome</home>
               <remote>edu.rice.rubis.beans.Region</remote>
            </ejb-ref>
      <session>
    <enterprise-beans>
<ejb-jar>
```

2.5.2. References between beans in different jars but in the same ear

This section explains how to declare references between beans that are packaged in different jars, but that are packaged in the same J2EE application (ear). With JBoss 2.4 and JOnAS versions before 2.6, you should proceed as if the beans were in separate jars and in separate ears, see next section. With JOnAS (from version 2.6), you proceed in the standard way, i.e. only the standard deployment descriptor is

concerned. So for migrating from JBoss to JOnAS, just suppress the corresponding part of the `jboss.xml` file and add the ejb-link element to the `ejb-jar.xml`.

With JOnAS you have to declare an ejb-ref element in the `ejb-jar.xml`. and you must also declare an ejb-link attribute in the ejb-ref element of the bean. There is nothing to add in the JOnAS specific descriptors. The value of the ejb-link element should be the name of the target bean, prefixed by the name of the containing ejb-jar file and '#' (e.g. "My_EJBs.jar#bean1"). In the following example the session bean SB_BrowseRegions calls methods on the Region entity bean. So for SB_BrowseRegions you need an ejb-ref element in the `ejb-jar.xml` file that references the bean Region. If the Region bean is in a separate `Region.jar` jar file, the reference will be as below:

`ejb-jar.xml` file for SB_BrowseRegions:

```
<ejb-jar>
    <enterprise-beans>
       <session>
         <description>Deployment descriptor for SB_BrowseRegions Bean</description>
            <ejb-ref>
               <description>This is the reference to the region bean</description>
               <ejb-ref-name>ejb/Region</ejb-ref-name>
               <ejb-ref-type>Entity</ejb-ref-type>
               <ejb-link>Region.jar#Region</ejb-link>
               <home>edu.rice.rubis.beans.RegionHome</home>
               <remote>edu.rice.rubis.beans.Region</remote>
            </ejb-ref>
      <session>
    <enterprise-beans>
<ejb-jar>
```

2.5.3. References between beans in different jars and different ears

This section explains how to declare references between beans that are packaged in different jars and in different ears.

With both JBoss and JOnAS you have to declare an ejb-ref element in the `ejb-jar.xml` but unlike when beans are in the same jar, you don't have to declare an ejb-link attribute. With JBoss you have to declare an ejb-ref element in `jboss.xml` and provide the full JNDI name of the bean. With JOnAS you have to declare a jonas-ejb-ref element in `jonas-ejb-jar.xml` that contains the JNDI name of the referenced bean home interface.

In the following example the session bean SB_BrowseRegions calls methods on the Region entity bean but the beans are packaged in different jars. So for SB_BrowseRegions you need an ejb-ref element in

the `ejb-jar.xml` file and a jonas-ejb-ref element in the `jonas-ejb-jar.xml` file that references the bean Region instead of an ejb-ref elment in the the `jboss.xml`:

Common `ejb-jar.xml` file for SB_BrowseRegions:

```
<ejb-jar>
    <enterprise-beans>
       <session>
         <description>Deployment descriptor for SB_BrowseRegions Bean</description>
            <ejb-ref>
               <description>This is the reference to the region bean</description>
               <ejb-ref-name>ejb/Region</ejb-ref-name>
               <ejb-ref-type>Entity</ejb-ref-type>
               <home>edu.rice.rubis.beans.RegionHome</home>
               <remote>edu.rice.rubis.beans.Region</remote>
            </ejb-ref>
      <session>
    <enterprise-beans>
<ejb-jar>
```

Example of `jboss.xml` file for SB_BrowseRegions:

```
<jboss>
    <session>
        <ejb-name>SB_BrowseRegions</ejb-name>
        <ejb-ref>
            <ejb-ref-name>ejb/Region</ejb-ref-name>
            <jndi-name>protocol://serverName/directory/RegionHome</jndi-name>
        </ejb-ref>
    </session>
</jboss>
```

Here is the equivalent `jonas-ejb-jar.xml` file for SB_BrowseRegions:

```
<jonas-ejb-jar>
    <jonas-session>
        <ejb-name>SB_BrowseRegions</ejb-name>
        <jonas-ejb-ref>
            <ejb-ref-name>ejb/Region</ejb-ref-name>
            <jndi-name>RegionHome</jndi-name>
        </jonas-ejb-ref>
    </jonas-session>
</jonas-ejb-jar>
```

Chapter 3. Generating the container classes

JOnAS uses a specific compiler to statically generate stubs and skeletons for the remote objects by way of the rmi compiler or jeremie compiler. This allows faster loading and execution of beans compared to the dynamic approach used in JBoss. Once you have the deployment descriptors, this is a straight forward process that can be performed using GenIC, a tool provided in the JOnAS distribution. The corresponding ANT task for JOnAS may also be used.

Here is how to use GenIC:

```
GenIC [Options] InputFile
```

InputFile is either a jar file or a xml deployment descriptor. Example :

```
GenIC myapp.jar
```

```
GenIC ejb-jar.xml
```

A complete description of the GenIC command and its options can be found at JOnAS tools documentation (http://www.objectweb.org/jonas/current/doc/Tools.html#genic). The JOnAS ANT task is also described in the JOnAS documentation (http://www.objectweb.org/jonas/current/doc/ant-ejbjar.html).

The GenIC utility :

- firstly, generates the sources of the container classes for all the beans defined in the deployment descriptor,
- secondly, compiles these classes by way of the java compiler,
- thirdly, generates stubs and skeletons for those remote objects by way of the rmi compiler, and
- lastly, if the InputFile is an ejb-jar file, adds the generated classes in this ejb-jar file.

Chapter 4. User transactions from the client side

Unlike JBoss, JOnAS supports full class names. With JOnAS, you have to add the prefix "javax.transaction." or "java:comp/" when calling a UserTransaction from a Java servlet. You have to add the prefix "javax.transaction." when calling a UserTransaction from a pure Java client.

Calls in JBoss look like:

```
UserTransaction utx;
utx = (javax.transaction.UserTransaction)
      initialContext.lookup("UserTransaction");
```

Calls from a servlet with JOnAS look like:

```
UserTransaction utx;
utx = (javax.transaction.UserTransaction)
      initialContext.lookup("javax.transaction.UserTransaction");
```

or

```
UserTransaction utx;
utx = (javax.transaction.UserTransaction)
      initialContext.lookup("java:comp/UserTransaction");
```

Calls from a pure Java client with JOnAS look like:

```
UserTransaction utx;
utx = (javax.transaction.UserTransaction)
      initialContext.lookup("javax.transaction.UserTransaction");
```

Chapter 5. Further reading

You can find additional resources about JOnAS on Objectweb web site (http://www.objectweb.org).

5.1. JOnAS documentation

JOnAS documentation is available at: http://www.objectweb.org/jonas/doc/index.html

5.2. JOnAS mailing list

You can post messages on JOnAS users mailing list: jonas@objectweb.org

Appendix A. ejb-jar.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN"
                     "http://java.sun.com/j2ee/dtds/ejb-jar_2_0.dtd">
<ejb-jar>
    <enterprise-beans>
       <session>
         <description>Deployment descriptor for Rubis SB_BrowseRegions Bean</description>
            <display-name>RUBiS SB_BrowseRegions Bean</display-name>
            <ejb-name>SB_BrowseRegions</ejb-name>
            <home>edu.rice.rubis.beans.SB_BrowseRegionsHome</home>
            <remote>edu.rice.rubis.beans.SB_BrowseRegions</remote>
            <ejb-class>edu.rice.rubis.beans.SB_BrowseRegionsBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Bean</transaction-type>
            <ejb-ref>
                <description>This is the reference to the region bean</description>
                <ejb-ref-name>ejb/Region</ejb-ref-name>
                <ejb-ref-type>Entity</ejb-ref-type>
                <ejb-link>Region</ejb-link>
                <home>edu.rice.rubis.beans.RegionHome</home>
                <remote>edu.rice.rubis.beans.Region</remote>
            </ejb-ref>
            <resource-ref>
                <res-ref-name>jdbc/rubis</res-ref-name>
                <res-type>javax.sql.DataSource</res-type>
                <res-auth>Container</res-auth>
            </resource-ref>
         </session>

        <entity>
            <description>Region Bean deployment descriptor</description>
            <display-name>Region</display-name>
            <ejb-name>Region</ejb-name>
            <home>edu.rice.rubis.beans.RegionHome</home>
            <remote>edu.rice.rubis.beans.Region</remote>
            <ejb-class>edu.rice.rubis.beans.RegionBean</ejb-class>
            <persistence-type>Container</persistence-type>
            <prim-key-class>edu.rice.rubis.beans.RegionPK</prim-key-class>
            <reentrant>False</reentrant>
            <cmp-field>
                <description>Region name</description>
                <field-name>name</field-name>
            </cmp-field>
            <cmp-field>
                <description>Region id</description>
                <field-name>id</field-name>
            </cmp-field>
            <resource-ref>
```

```
                <res-ref-name>jdbc/rubis</res-ref-name>
                <res-type>javax.sql.DataSource</res-type>
                <res-auth>Container</res-auth>
            </resource-ref>
        </entity>
    </enterprise-beans>

    <assembly-descriptor>
        <container-transaction>
            <method>
                <ejb-name>Region</ejb-name>
                <method-name>*</method-name>
            </method>
            <trans-attribute>Required</trans-attribute>
        </container-transaction>
    </assembly-descriptor>
</ejb-jar>
```

Appendix B. jboss.xml

```
<?xml version='1.0' encoding='UTF-8' ?>
<jboss>
  <enterprise-beans>
    <entity>
      <ejb-name>Region</ejb-name>
      <jndi-name>RegionHome</jndi-name>
      <resource-ref>
        <res-ref-name>jdbc/rubis</res-ref-name>
        <resource-name>rubis</resource-name>
      </resource-ref>
    </entity>

    <session>
      <ejb-name>SB_BrowseRegions</ejb-name>
      <jndi-name>SB_BrowseRegionsHome</jndi-name>
      <resource-ref>
        <res-ref-name>jdbc/rubis</res-ref-name>
        <resource-name>rubis</resource-name>
      </resource-ref>
    </session>

  </enterprise-beans>

  <resource-managers>

    <resource-manager>
      <res-name>rubis</res-name>
      <res-jndi-name>java:/rubis</res-jndi-name>
    </resource-manager>

  </resource-managers>
</jboss>
```

Appendix C. jaws.xml

```
<?xml version="1.0"?>
<jaws>
    <enterprise-beans>
        <entity>
           <ejb-name>Region</ejb-name>
           <table-name>regions</table-name>
           <create-table>false</create-table>
           <cmp-field>
               <field-name>id</field-name>
               <column-name>id</column-name>
           </cmp-field>
           <cmp-field>
               <field-name>name</field-name>
               <column-name>name</column-name>
           </cmp-field>
        </entity>
    </enterprise-beans>
</jaws>
```

Appendix D. jonas-ejb-jar.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE jonas-ejb-jar PUBLIC "-//ObjectWeb//DTD JOnAS 2.4//EN"
                            "http://www.objectweb.org/jonas/dtds/jonas-ejb-jar_2_4.dtd">
<jonas-ejb-jar>
    <jonas-session>
        <ejb-name>SB_BrowseRegions</ejb-name>
        <jndi-name>SB_BrowseRegionsHome</jndi-name>
        <jonas-resource>
            <res-ref-name>jdbc/rubis</res-ref-name>
            <jndi-name>rubis</jndi-name>
        </jonas-resource>
    </jonas-session>

    <jonas-entity>
        <ejb-name>Region</ejb-name>
        <jndi-name>RegionHome</jndi-name>
        <jonas-resource>
            <res-ref-name>jdbc/rubis</res-ref-name>
            <jndi-name>mysql</jndi-name>
        </jonas-resource>

        <jdbc-mapping>
            <jndi-name>mysql</jndi-name>
            <jdbc-table-name>regions</jdbc-table-name>
            <cmp-field-jdbc-mapping>
                <field-name>id</field-name>
                <jdbc-field-name>id</jdbc-field-name>
            </cmp-field-jdbc-mapping>
            <cmp-field-jdbc-mapping>
                <field-name>name</field-name>
                <jdbc-field-name>name</jdbc-field-name>
            </cmp-field-jdbc-mapping>

            <finder-method-jdbc-mapping>
                <jonas-method>
                    <method-name>findByPrimaryKey</method-name>
                </jonas-method>
                <jdbc-where-clause>where id=?</jdbc-where-clause>
            </finder-method-jdbc-mapping>

            <finder-method-jdbc-mapping>
                <jonas-method>
                    <method-name>findByName</method-name>
                </jonas-method>
                <jdbc-where-clause>where name=?</jdbc-where-clause>
            </finder-method-jdbc-mapping>

            <finder-method-jdbc-mapping>
```

```
                <jonas-method>
                    <method-name>findAll</method-name>
                </jonas-method>
                <jdbc-where-clause></jdbc-where-clause>
            </finder-method-jdbc-mapping>
        </jdbc-mapping>

    </jonas-entity>
</jonas-ejb-jar>
```

Appendix E. Home interface: RegionHome

```
import java.rmi.RemoteException;
import java.util.Collection;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;
import javax.ejb.FinderException;
import javax.ejb.RemoveException;

/** This is the Home interface of the Region Bean */

public interface RegionHome extends EJBHome {

  /**
   * This method is used to create a new Region Bean. Note that the region
   * id is automatically generated by the database (AUTO_INCREMENT) on the
   * primary key.
   *
   * @param name Region name
   *
   * @return pk primary key set to null
   */
  public Region create(String name) throws CreateException, RemoteException, RemoveExceptio


  /**
   * This method is used to retrieve a Region Bean from its primary key,
   * that is to say its id.
   *
   * @param id Region id (primary key)
   *
   * @return the Region if found else null
   */
  public Region findByPrimaryKey(RegionPK id) throws FinderException, RemoteException;


  /**
   * This method is used to retrieve a Region Bean from its name.
   *
   * @param name Region name
   *
   * @return the Region if found else null
   */
  public Region findByName(String name) throws FinderException, RemoteException;


  /**
   * This method is used to retrieve all categories from the database!
   *
   * @return List of all categories (eventually empty)
```

```
   */
  public Collection findAll() throws RemoteException, FinderException;
}
```

Appendix F. Bean implementation: RegionBean

```
import java.rmi.*;
import javax.ejb.*;
import javax.rmi.PortableRemoteObject;
import javax.naming.InitialContext;

/**
 * RegionBean is an entity bean with "container managed persistence".
 * The state of an instance is stored into a relational database.
 * The following table should exist:
 * <pre>
 * CREATE TABLE regions (
 *    id   INTEGER UNSIGNED NOT NULL UNIQUE,
 *    name VARCHAR(20),
 *    PRIMARY KEY(id)
 * );
 * </pre>
 * @version 1.0
 */

public class RegionBean implements EntityBean
{

  /* Class member variables */

  public Integer id;
  public String  name;

[...]
}
```