



Author: Jon Barnett
Date: 23rd October 2003

XDoclet 1.2b4 and JBoss changes

This technote is a quick guide to adding new JBoss tags to XDoclet. While this may not be the most elegant and correct way of doing things, it does get you up and running rapidly. The guide should allow you to work out what you will be required to do in order to satisfy your own requirements. I hope it will save you the time I spent researching the linkages and operation, particularly if you do not want to know all the details of XDoclet but just want to get your application built.

The EJB problem

In my particular instance, I wanted to create a reference to the JNDI name of an EJB local interface that was not in the same application as the calling EJB. JBoss 3.2.2 does support this mode of deployment but Xdoclet 1.2b4 did not provide the tags to generate this. I spent a great deal of time looking for either XDoclet support for generating the necessary tags or looking for some other means of creating the linkage. Since I was modifying my build process to use XDoclet, this became a major stumbling block to adopting XDoclet.

XDoclet supports the following definition in the Bean code:

```
* @ejb.ejb-external-ref
*     ref-name="ejb/amity/flexcorp/EventLogger"
*     type="Session"
*     view-type="local"
*     home="com.amity.flexcorp.eventlogger.EventLoggerLocalHome"
*     business="com.amity.flexcorp.eventlogger.EventLoggerLocal"
```

This generates an acceptable `ejb-jar.xml` section.

```
<ejb-local-ref >
  <ejb-ref-name>ejb/amity/flexcorp/EventLogger</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <local-home>
    com.amity.flexcorp.eventlogger.EventLoggerLocalHome
  </local-home>
  <local>com.amity.flexcorp.eventlogger.EventLoggerLocal</local>
</ejb-local-ref>
```

However, JBoss 3.2.2 requires either a link element for the local reference in `ejb-jar.xml` or it requires an `ejb-local-ref` definition in `jboss.xml`. Since the link element only works for other EJB components that are packaged in the same application, this was not suitable. I needed the `ejb-local-ref` element.

Adding new descriptor elements

The XDoclet 1.2b4 processing information for JBoss is contained in the file XDOCLET_HOME/lib/xdoclet-jboss-module-1.2b4.jar. Since we are not adding any different processing tasks we do not need to do any programming. We are just adding new acceptable elements for the JBoss specific deployment descriptors. The first thing to do is to unpack the JBoss module.

Create a directory for your modification and unpack in that directory. For example:

```
mkdir xdoclet-jboss
cd xdoclet-jboss
jar xvf XDOCLET_HOME/lib/xdoclet-jboss-module-1.2b4.jar
```

The module is well structured and the majority of the processing information relevant to the deployment of EJB components is contained in the sub-directory xdoclet-jboss/xdoclet/modules/jboss/ejb/resources. You will note that the necessary JBoss DTD files for validating the EJB descriptors are also packaged in this area of the module. This sub-directory is where will devote most of our time so change to this sub-directory.

For this particular exercise, we are interested in the processing related to the jboss.xml file. The template used to convert the directives into descriptor tags is called jboss_xml.xdt.

We want to borrow from existing processing rules as this makes things less error prone and allows us to understand the manner in which XDoclet accomplishes its task. Reading from the JBoss 3.2.2 DTD contained in JBOSS_HOME/docs/dtd/jboss_3_2.dtd you will find that the required format for the ejb-local-ref tag is:

```
<ejb-local-ref>
  <ejb-ref-name>ref/name</ejb-ref-name>
  <local-jndi-name>some/jndi/name</local-jndi-name>
</ejb-local-ref>
```

There can be multiple tags in jboss.xml. The nested tag elements are mandatory. Checking with the JBoss 3.0 and JBoss 2.4 DTD definitions, you will find that the tag is not supported by these versions.

Looking at the jboss tag reference at the [XDoclet website](#), you will find that a tag with similar properties is the ejb-ref-jndi tag. You can locate the code that implements this processing in jboss_xml.xdt. It looks like this.

```
<XDtClass:forAllClassTags tagName="jboss:ejb-ref-jndi">
  <ejb-ref>
    <ejb-ref-name>ejb/<XDtClass:classTagValue tagName="jboss:ejb-ref-jndi" paramName="ref-name" /></ejb-ref-name>
    <jndi-name><XDtClass:classTagValue tagName="jboss:ejb-ref-jndi" paramName="jndi-name" /></jndi-name>
  </ejb-ref>
</XDtClass:forAllClassTags>
```

We also look for any examples we can find for limiting construction of the descriptors to particular versions of JBoss. Due to experience with JBoss we

know that the method-attributes implementation is different between 3.2.x and earlier versions.

```
<XDtConfig:ifConfigParamGreaterOrEquals paramName="Version" value="3.2">
  <method-attributes>
    ...
  </method-attributes>
</XDtConfig:ifConfigParamGreaterOrEquals>
```

So our complete snippet for generating the `ejb-local-ref` tag is:

```
<XDtConfig:ifConfigParamGreaterOrEquals paramName="Version" value="3.2">
  <XDtClass:forAllClassTags tagName="jboss:ejb-local-ref">
    <ejb-local-ref>
      <ejb-ref-name>ejb/<XDtClass:classTagValue tagName="jboss:ejb-local-ref"
      paramName="ref-name" /></ejb-ref-name>
      <local-jndi-name><XDtClass:classTagValue tagName="jboss:ejb-local-ref"
      paramName="jndi-name" /></local-jndi-name>
    </ejb-local-ref>
  </XDtClass:forAllClassTags>
</XDtConfig:ifConfigParamGreaterOrEquals>
```

I placed this in `jboss_xml.xdt`, just after the section defining `ejb-ref`. Please note that the `ejb-ref-name` construction for our addition prepends `ejb/` to the `ref-name` you provide. I left this in order to remain consistent with the JBoss JNDI name processing implemented by the rest of the XDoclet definition.

For completeness, I also added this snippet to `jboss-bean-body.xdt`, after the `ejb-ref` section in there, although I do not believe this has any impact that I could discern.

Finally, return to the `META-INF` sub-directory just below the root where you unpacked the module. We need to edit the `xtags.xml` file so that we generate some error message and help text for our new tag.

```

<tag>
  <level>class</level>
  <name>jboss.ejb-local-ref</name>
  <usage-description>
    Sets the JNDI name of a locally referenced bean that is
    not contained in the same application.
  </usage-description>
  <unique>>false</unique>
  <condition type="class"/>
  <parameter type="text">
    <name>ref-name</name>
    <usage-description>
      The name by which the referenced bean will be referred.
      For example, to refer to the bean Customer as
      java:comp/env/ejb/Customer name should be
      ejb/Customer. Defaults to ejb/[ejb-name], where
      [ejb-name] is the named of the referenced bean
      (Customer) prefixed by "ejb/".
    </usage-description>
    <mandatory>>true</mandatory>
  </parameter>
  <parameter type="text">
    <name>jndi-name</name>
    <usage-description>
      The JNDI name of the bean.
    </usage-description>
    <mandatory>>true</mandatory>
  </parameter>
</tag>

```

Normally, this would be all that is required. You can package the module and replace the old module. You would do this by returning to the root directory where you first unpacked the module. You should be able to list the META-INF and xdoclet sub-directories. You can then perform the following:

```

jar cvzf xdoclet-jboss-module-1.2b4.jar META-INF xdoclet
cp xdoclet-jboss-module-1.2b4.jar XDOCLET_HOME/lib

```

I did this and tried generating the deployment descriptors. During the ejbdoclet task, the following error appeared:

```

org.xml.sax.SAXParseException: Element type "ejb-local-ref" must be
declared.

```

This is an indication that the JBoss DTDs did not support the ejb-local-ref element. I checked the DTDs packaged with the module and found they were

older than the ones supplied with JBoss 3.2.2 and the XDoclet jboss_3_2.dtd did not have the `ejb-local-ref` element declared. I replaced all the relevant DTDs with those supplied with JBoss 3.2.2, repacked and copied the updated module back to the XDoclet lib directory. Running the `ejbdoclet` task again worked without any problems and the deployed EJB with the generated deployment descriptors was operational.

Final EJB results

In order to provide some information on how the completed changes operated, I present the XDoclet Bean definitions and the resultant output.

Bean code fragment:

```
* @ejb.ejb-external-ref
*   ref-name="ejb/amity/flexcorp/EventLogger"
*   type="Session"
*   view-type="local"
*   home="com.amity.flexcorp.eventlogger.EventLoggerLocalHome"
*   business="com.amity.flexcorp.eventlogger.EventLoggerLocal"
*
* @jboss.ejb-local-ref
*   ref-name="amity/flexcorp/EventLogger"
*   jndi-name="ejb/amity/flexcorp/local/EventLogger"
```

`ejb-jar.xml`:

```
<ejb-local-ref >
  <ejb-ref-name>ejb/amity/flexcorp/EventLogger</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <local-home>
    com.amity.flexcorp.eventlogger.EventLoggerLocalHome
  </local-home>
  <local>com.amity.flexcorp.eventlogger.EventLoggerLocal</local>
</ejb-local-ref>
```

`jboss.xml`:

```
<ejb-local-ref>
  <ejb-ref-name>ejb/amity/flexcorp/EventLogger</ejb-ref-name>
  <local-jndi-name>
    ejb/amity/flexcorp/local/EventLogger
  </local-jndi-name>
</ejb-local-ref>
```

Note how the `ejb-ref-name` matches between `ejb-jar.xml` and `jboss.xml` as these link the `java:comp/env/ejb/amity/flexcorp/EventLogger` you use as your Bean code reference to the JNDI name `ejb/amity/flexcorp/local/EventLogger` to which the actual EJB local home is actually bound. Checking the `jmx-console` JNDIview list should confirm correct linking.

```

+-ejb (class: org.jnp.interfaces.NamingContext)
| +-amity (class: org.jnp.interfaces.NamingContext)
| | +-flexcorp (class: org.jnp.interfaces.NamingContext)
| | | +-EventLogger[link -> ejb/amity/flexcorp/local/EventLogger] (class:
javax.naming.LinkRef)

```

Should the linking be incorrect, you will link -> null instead.

The web application problem

XDoclet 1.2b4 as we experienced with EJBs, does not support the local references introduced in JBoss 3.2.x. This extends to generating jboss-web.xml for your web applications. However, this change requires a bit more work as you need to recompile the module from the source as well. This is necessary because the web processing is simpler and doesn't cater for the new tag we are going to add. The module doesn't support the latest version of the jboss-web DTD either.

Readying the source

You will need to copy the changed files for the EJB deployment descriptors as described earlier into the relevant places in the source tree in order to retain these changes in the module we are going to build. Obtain the source code from the XDoclet site, <http://xdoclet.sourceforge.net>. Unpack your source if necessary. The CVS checkout provides an unpacked copy.

Copy your EJB related changes to the correct subdirectories under XDOCLET_HOME/xdoclet/modules/jboss/src/xdoclet/modules/jboss.

Modifying the source

In order to make the changes:

```
cd XDOCLET_HOME/xdoclet/modules/jboss/src/xdoclet/modules/jboss/web
```

We need to modify JbossWebXmlSubTask.java for the new DTD and we need to add another tag handler in JbossWebTagsHandler.java.

Add the following definitions to JbossWebXmlSubTask.java.

```

private final static String JBOSS_WEB_PUBLICID_3_2 = "-//JBoss//DTD Web
Application 2.3//EN";

private final static String JBOSS_WEB_SYSTEMID_3_2 =
"http://www.jboss.org/j2ee/dtd/jboss-web_3_2.dtd";

private final static String JBOSS_WEB_DTD_FILE_NAME_3_2 = "resources/jboss-
web_3_2.dtd";

```

You may want to group these appropriately with the existing definitions.

This defines the jboss-web DTD definition for the DOCTYPE header. We also want to modify the execute method to include the new DTD definition and also make it the default target.

Change the following:

```
protected String version = JbossSubTask.JBossVersionTypes.VERSION_3_0;
```

To:

```
protected String version = JbossSubTask.JBossVersionTypes.VERSION_3_2;
```

Rewrite execute() to:

```
public void execute() throws XDocletException
{
    if (getVersion().equals(JBossSubTask.JBossVersionTypes.VERSION_2_4)) {
        setPublicId(JBOSS_WEB_PUBLICID_2_4);
        setSystemId(JBOSS_WEB_SYSTEMID_2_4);
        setDtdURL(getClass().getResource(JBOSS_WEB_DTD_FILE_NAME_2_4));
    }
    else if (getVersion().equals(JBossSubTask.JBossVersionTypes.VERSION_2_4))
    {
        setPublicId(JBOSS_WEB_PUBLICID_3_0);
        setSystemId(JBOSS_WEB_SYSTEMID_3_0);
        setDtdURL(getClass().getResource(JBOSS_WEB_DTD_FILE_NAME_3_0));
    }
    else {
        setPublicId(JBOSS_WEB_PUBLICID_3_2);
        setSystemId(JBOSS_WEB_SYSTEMID_3_2);
        setDtdURL(getClass().getResource(JBOSS_WEB_DTD_FILE_NAME_3_2));
    }
    super.execute();
}
```

In order to cater for the new tag handling, add this fragment to JbossWebTagsHandler.java.

```
/**
 * Iterates over all \@jboss.ejb-local-ref tags.
 *
 * @param template      The body of the block tag
 * @throws XDocletException if something goes wrong
 */
public void forAllLocalRefs(String template) throws XDocletException
{
    ClassTagsHandler.forAllDistinctClassTags(getEngine(), template,
"jboss.ejb-local-ref", "ref-name");
}
```

This definition matches the EJB tag we added earlier so we keep some semblance of tag naming standards. Copy the new jboss-web DTDs from your JBoss distribution to the appropriate directories under XDOCLET_HOME/xdoclet/modules/jboss/src/xdoclet/modules/jboss. Although in theory you will only need to add the missing 3.2 DTD called jboss-web_3_2.dtd, I also replaced the existing files in the JBoss XDoclet module with the files from the JBoss 3.2.2 distribution.

Finally, modify the processing template resources/jboss_web_xml.xdt for the new local EJB reference tag.

```

<!-- EJB Local References -->
<XdtConfig:ifConfigParamGreaterOrEquals paramName="version" value="3.2">
<XdtJBossWeb:forAllLocalRefs tagName="jboss:ejb-local-ref">
  <ejb-local-ref>
    <ejb-ref-name>ejb/<XdtClass:classTagValue tagName="jboss:ejb-local-
ref" paramName="ref-name" /></ejb-ref-name>
    <local-jndi-name><XdtClass:classTagValue tagName="jboss:ejb-local-ref"
paramName="jndi-name" /></local-jndi-name>
  </ejb-local-ref>
</XdtJBossWeb:forAllLocalRefs>
</XdtConfig:ifConfigParamGreaterOrEquals>

```

Note that this is virtually the same as the fragment we created for the EJB tag processing earlier.

Final web application results

In your servlet you can now add your XDoclet tags. As an example, I had these:

```

* @web.ejb-local-ref
*   name="ejb/amity/catalogue/Products"
*   type="Session"
*   home="com.amity.catalogue.products.ProductsLocalHome"
*   local="com.amity.catalogue.products.ProductsLocal"
*
* @jboss.ejb-local-ref
*   ref-name="amity/catalogue/Products"
*   jndi-name="ejb/amity/catalogue/local/Products"

```

In `jboss-web.xml`, the following is generated from the webDoclet task:

```

<!-- EJB Local References -->
  <ejb-local-ref>
    <ejb-ref-name>ejb/amity/catalogue/Products</ejb-ref-name>
    <local-jndi-name>ejb/amity/catalogue/local/Products</local-jndi-name>
  </ejb-local-ref>

```

The corresponding output is generated for the `web.xml` descriptor.

```

<ejb-local-ref>
  <ejb-ref-name>ejb/amity/catalogue/Products</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <local-home>com.amity.catalogue.products.ProductsLocalHome</local-home>
  <local>com.amity.catalogue.products.ProductsLocal</local>
</ejb-local-ref>

```

The application containing these fragments deploys correctly in JBoss 3.2.2.

I've provided a copy of the modified 1.2b4 module at <http://www.amitysolutions.com.au/downloads/xdoclet-jboss-module-1.2b4.jar> in case the modifications are too complex to follow.

Final notes

The changes prescribed in this technical note will only generate `ejb-local-ref` tags for version 3.2 of the relevant JBoss DTDs, as compliant with the DTD support for the element. There are many configurations of XDoclet, particularly plug-ins that may default to an earlier JBoss version for the generation of tags. Refer to the documentation supplied with your plug-in on the configuration for generating the correct versions of the descriptors. You can verify the generation by checking the DOCTYPE. For example, `jboss.xml` should contain the following for `ejb-local-ref` support:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jboss PUBLIC "-//JBoss//DTD JBOSS 3.2//EN"
"http://www.jboss.org/j2ee/dtd/jboss_3_2.dtd">
```

As one reader noted, your Lombok configuration may override the programmatic default we have added in the code by supplying a specific version in `ejbgenerate.xml`. The explicit version 3.0 setting generated a JBoss 3.0 compliant deployment descriptor, and this avoided producing the necessary `ejb-local-ref` tags. This is expected as the changes we have implemented test specifically for version 3.2 generation settings. Please be aware of the DTD version dependency.