JBoss 4.0 Upgrade Guide

The JBoss Application Server (JBoss AS) 4.0 is a production-ready Java 2 Enterprise Edition (J2EE) application server. It is the first officially certified J2EE 1.4-compliant application server in the industry, providing full support for J2EE Web Services and the Service Oriented Architecture (SOA). Beyond the standard J2EE, JBoss AS also supports next generation middleware technologies, including Aspect-Oriented Programming (AOP) with JBoss AOP, Object-Relational data persistence with Hibernate, and clustering support with JGroups and JBoss Cache. To see a comprehensive list of new features in JBoss AS 4.0, please refer to the "What's new in JBoss AS 4.0" document.

Upgrading your servers to JBoss 4.0 allows you to take advantage of new features in J2EE 1.4 as well as Open Source innovations, such as JBoss AOP, Hibernate and JBoss Cache, from the JBoss community. For most users who are already deploying JBoss AS 3.2.x, the upgrading process should be smooth with little or no tweaking of your applications. In this document, we discuss common problems and pitfalls our users have encountered in their upgrading efforts.

1. Runtime Environment

JBoss AS 4.0 requires JDK 1.4 or higher to run. A known issues exists in Sun's JDK 1.4.2_05, which results in a "Real-time signal 28" error at server shutdown. The incomplete shutdown could cause problems at next start-up. You should use JDK 1.4.2_06 and later if possible. If you run JBoss AS on Solaris or Suse Linux operating systems, there are also two known issues with specific versions of those operating systems.

- JBoss hangs (or have memory problems) on Solaris 1.4.1 due to a known deadlock issue. This issue is resolved in Solaris 1.4.2.
- On Suse Linux 9.1, JBoss fails to connect to the MySQL database due to a problem with how Suse 9.1 handles "localhost". Suse 9.0 should work without any problems.

2. Class Loader and Server Configurations

The most common JBoss AS 4.0 upgrading problems arise from the user's mis-understanding of how class loaders work in JBoss AS 4.0's default server configurations. This problem is especially confusing in JBoss AS 4.0.0. The symptom of this problem is that a JBoss AS 3.2.x application throws the `NoClassFoundException` or other exceptions indicating missing or conflicting resources, when it deployed in JBoss AS 4.0.0. If you are interested in learning more on how the JBoss `ClassLoader` works, please check out this wiki item: http://www.jboss.org/wiki/Wiki.jsp?page=JBossClassLoadingUseCases.

The reason is that the `default` configuration in JBoss AS 4.0.0 uses a scoped class loader, which compartmentalizes each deployed application, in order to conform to the J2EE 1.4 certification requirements. However, in JBoss AS 3.2.x, the class loader in the `default` configuration allows different deployment packages to share classes and resources. The shared class loader has better performance and is used by most existing JBoss AS 3.2.x applications. In JBoss AS 4.0.0, the shared class loader configuration is the `standard` configuration. So, the first step to diagnose class loader problems is to deploy your application in the `deploy/standard` directory and start JBoss AS 4.0.0 in the `standard` configuration using the following command.

```
run -c standard
```

In addition to the class loader problems, there are also other side-effects associated with the name change of the default configuration in JBoss 4.0.0. For instance, the service binding manager is available in the `standard` configuration but not in the `default` configuration in JBoss AS 4.0.0.

In JBoss AS 4.0.1, we tried to reduce the confusion by eliminating the `standard` configuration and use the shared class loader again for the `default` configuration. To configure a J2EE certified configuration in JBoss 4.0.1, please refer to the "What's New in JBoss AS 4.0" document. So, your JBoss 3.2.x application should run fine in the `default` configuration of JBoss AS 4.0.1 but your JBoss AS 4.0.0 applications may not be.

3. Configure the Server

If you customize the JBoss AS's default configurations and libraries, there are also things to look out for.

- In JBoss AS 4.0, the EAR, EJB and BSH deployers are extracted from the `conf/jboss-service.xml` to the `deploy` directory as `ear-deployer.xml` , `ejb-deployer.xml` and `bsh-deployer.xml` respectively. So, watch out if customize those configuration settings.
- You are not supposed to make changes to the files in the `JBOSS_HOME/lib` directory. To add new system-wide libraries in JBoss AS 4.0, you can copy them into the new `JBOSS_HOME/lib/endorsed` directory.

4. Deployment Order

If your JBoss applications must be started in a particular order to function correctly, there is a slight chance that they might be affected by the upgrade to JBoss AS 4.0.

- In JBoss AS 4.0, a new deployer archive format `.deployer` and a new XML deployer format `*-deployer.xml` are supported. The `.deployer` suffix is equivalent to the `.sar` suffix, and the `-deployer.xml` file name suffix is equivalent to the `-service.xml` descriptor file name suffix in JBoss AS 3.2.x. However, the `.deployer` archives and `*-deployer.xml` files are sorted ahead of any other service types and are deployed first when the server starts up.
- If two packages have the same prefix and file type, their deployment order on JBoss 3.2.x is dependent on the underlying platform. In JBoss 4.0, it is no longer the case. With the introduction of the `AlphaNumericDeploymentSorter` class, JBoss AS 4.0 adds a final comparison of the file names using the `String` class' `compareToIgnoreCase()` method. To learn more about how the deployment sorter works, please check out this wiki item: http://www.jboss.org/wiki/Wiki.jsp?page=URLComparator .

5. Deploy Applications Inside of an Application

If you put a deployment in the `META-INF` directory (e.g. a `META-INF/some-service.xml` in an EJB jar file), it is not deployed in JBoss AS 4.0. This did work in JBoss AS 3.2.x, but it was not the intended behavior. The `META-INF` directory is reserved for information about the current deployment. If you want to make a subdeployment (for those archives that support simple Russian Doll packaging), place it in the root of the parent deployment. To add a -

service.xml or other non j2ee deployment to an EAR, you need to reference it from the META-INF/jboss-app.xml in a service element. To learn more on this subject, please refer to the wiki item: http://www.jboss.org/wiki/Wiki.jsp?page=JBoss4FAQ.

6. Resource Adaptors

From JBoss AS 3.2.x to JBoss AS 4.0, a rather significant change has been made to the resource adaptor descriptor in order to accommodate the needs of JCA 1.5, which is part of J2EE 1.4. While you can still deploy JBoss AS 3.2.x rar files in JBoss AS 4.0, you will need to revise your JBoss AS 3.2.x resource adaptor descriptors to migrate to JBoss AS 4.0.

To deploy a resource adaptor in JBoss is a two step process: First, take your standard rar deployment file (either as a standalone deployment or part of another deployment like an ear) and place it in the deploy directory. Then, you need a JBoss AS specific -ds.xml file in the deploy directory, which actually deploys the ConnectionFactory from the rar deployment, configuring the pool, JNDI name and other parameters. In essence, the rar archive acts as a template for the connection factory deployment in the ds.xml.

In JBoss AS 3.2.x, the adapter-display-name value in the ds.xml file must match the display-name value from the ra.xml file inside the rar archive. However, in JBoss AS 4.0, the adapter-display-name is no longer used. Instead, the rar-name and connection-definition elements are used to support multiple connection factories (i.e., connection-definitions). The following example shows how to define a connection factory from a rar file inside a ear file.

```
<tx-connection-factory>
  ...
  <rar-name>myapplication.ear#jms-ra.rar</rar-name>
  <connection-definition>
    org.jboss.resource.adapter.jms.JmsConnectionFactory
  </connection-definition>
  ...
</tx-connection-factory>
```

To learn more about resource adapters in JBoss AS, please check out this wiki item: http://www.jboss.org/wiki/Wiki.jsp?page=HowDoIDeployMyResourceAdapter.

7. Web Services

Significant changes have been made in the Web Services module from JBoss AS 3.2.x to 4.0. In JBoss 4.0, the JBoss.Net module is no longer supported and a new module called JBossWS is developed to conform to the J2EE Web Services specification. An overview of JBossWS is available from this wiki item: http://www.jboss.org/wiki/Wiki.jsp?page=JBossWS.

If you have JBoss.Net server applications that need to run in JBoss AS 4.0, it possible to replace the JBossWS module with the JBoss.Net module in located in the docs/examples directory. But you cannot have both JBossWS and JBoss.Net running at the same time.

For Web Services clients, the Axis generated stubs for JBoss.Net applications should work fine for the new JBossWS applications.

8. Hibernate

JBoss AS 4.0 bundles the Hibernate library. If your JBoss AS 3.2.x application uses Hibernate, it probably already bundles Hibernate library in its own deployment archive. You should avoid any library conflicts -- make sure that your Hibernate client and server uses the same version of Hibernate library. Otherwise, the `java.io.InvalidClassException` would be thrown.

JBoss AS 4.0 provides Hibernate integration support such as obtaining session factories via JNDI names. If you manage Hibernate sessions inside your own application, the JBoss AS EJB container does not understand how Hibernate opens and closes connections. In your server log, you might see frequent messages like "[CachedConnectionManager] Closing a connection for you". That does NOT indicate a connection leak. However, it is recommended that you use the JBoss Hibernate integration module to manage your Hibernate sessions and avoid those warnings.

A caveat for the above statement is that the JBoss Hibernate integration module is not well tested yet. There is a known conflict between the Hibernate integration code and the JBoss TreeCache code, which could result in `MappingExceptions` in some cases. If you are not using the Hibernate integration stuff, you should be able to complete remove the jars from the `lib` directory (`hibernate2`, `cglib`, and `odmg`), remove the `jboss-hibernate-service.xml` file from the `deploy` directory, and then bundle the jars inside your ear as before.

9. Clustering and Caching

Similar to the case with Hibernate, JBoss AS 4.0 provides built-in support for JBoss Cache. So, if your application still bundles its own JBoss Cache library in the EAR deployment file, it is time to remove them and use the library JARs in `JBOSS_HOME/server/config/lib` directory instead. Failing to do could result in `LinkageErrors` occasionally.

In clustered web applications, some TCP-multicasting users have noticed replication failures from `JBossCacheManager` when the load is heavy. To fix this problem, try switch to UDP-multicasting. Or, if you have to use TCP, you can work it around by setting the cache mode to `REPL_ASYNC`.

10. Other Known Issues

In this section, we lists some common known issues of JBoss AS 4.0, which have not been covered in previous sections. This is by no means a complete bug list. Most of those issues have already been fixed in JBoss 4.0.1. But if you use JBoss 4.0.0, you might encounter them. In this case, we recommend you upgrade.

- There is a thread bug in JBoss AS 4.0.0 that might cause JBoss AS to hang on multi-processor boxes. It has been fixed in JBoss AS 4.0.1.
- A bug in JBoss Cache requires users to restart the server after re-deploy a ear in order to make the changes in data sources to take effect. It is already patched.
- A bug in JBoss 4.0.0 could result in the "fail to obtain type-mapping metadata" error. It is fixed now. If you use an affected version of the server, uncomment the `datasource-mapping` element in the `standardjbosscmp-jdbc.xml` file to work around.

- The `PooledInvokerHA` included in JBoss AS 4.0 is not production ready. It has not gone through stress testing.
- In JBoss AS 4.0.0's web module, the `request.isUserInRole()` method in JSP always returns `true` . It has been fixed in JBoss 4.0.1.
- The transaction logging fails. Transaction logging is an experimental feature in JBoss AS 4.0. You should use the `NoLogTxLogger` element in the `deploy/transaction-service.xml` to disable it. For production use of logging transaction manager, we recommend you use Arjuna together with JBoss AS.