Migrating applications from Geronimo v1.0

{scrollbar}

top

Some changes have been introduced in this new relase that affects the configuration and deployment plans of existing installations. In this section we will focus on highlighting those differences, how to apply the necessary changes as well as some migration aid tools when available.

The changes from v1.0 to this new release are not limited to just deployment plans as mentione in the Summary of changes section

Using the upgrader tool

You can use the upgrader tool to migrate your geronimo plans from 1.0 to 1.1. To use the upgrader tool, at the **`<geronimo_home>\bin`** directory, execute:
`java -jar upgrade.jar`

Parameter usage:
`inputPlan outputPlan`
or
`inputPlan`
in which case the output will be in the same location as inputPlan with '.upgraded' appended

After you run the upgrader tool, there are some additional steps if your application uses database or JMS.:

1) Manually add the Database or JMS rar module as the dependency:

If the application uses resource reference in 1.1, resource references need to be resolvable in the set of ancestors of your module. In other words, your application has to depend on the rar deployment.

Example:

```
xmlsolid <sys:dependencies> <sys:dependency> <sys:groupId>Sample</ sys:groupId> <sys:artifactId>EmployeeDatasource</sys:artifactId> <sys:
version>1.1</ sys:version> <sys:type>rar</ sys:type> </sys:dependency> </sys:dependencies>
```

Without the dependency specified, you may run into the following error upon deployment of your application:

```
solid > ------------------------------------------------------ > Error: Unable to distribute XXX: Unable to resolve > resource reference 'XXX' (Could not find resource >
'XXX'. Perhaps it has not yet been configured, > or your application does not have a dependency declared for that > resource module?)
```

2) Manually convert the derby jar dependencies to system-datasource:

If your 1.0 database connection pool plan has the following as dependencies:

```
xmlsolid <dependency> <uri>org.apache.derby/derby/10.1.1.0/jar</uri> </dependency> <dependency> <uri>org.apache.derby/derbynet/10.1.1.0/jar</uri> <
/dependency>
```

The upgrader tool will convert them to:

```
xmlsolid <dep:dependency> <dep:groupId>org.apache.derby</dep:groupId> <dep:artifactId>derby</dep:artifactId> <dep:version>10.1.1.0</dep:version>
<dep:type>jar</dep:type> </dep:dependency> <dep:dependency> <dep:groupId>org.apache.derby</dep:groupId> <dep:artifactId>derbynet</dep:
artifactId> <dep:version>10.1.1.0</dep:version> <dep:type>jar</dep:type> </dep:dependency>
```

However, this loads a second copy of derby in a second classloader, and you will hit the following error when you run your application:

```
SQL Exception: Failed to start database 'XXX', see the next exception for details.
```

Instead you need a car dependency on system-datasource so you are sharing it's copy of derby.

```
xmlsolid <dep:dependency> <dep:groupId>geronimo</dep:groupId> <dep:artifactId>system-database</dep:artifactId> <dep:type>car</dep:type> </dep:
dependency>
```

Using the administration console

When you deploy an application using the administration console the portlet will automatically detect a 1.0 plan and invoke the upgrader tool described above to provide a 1.1 plan that you can save locally, make adjustments to if necessary, and then use to deploy your application. The console currently only provides this functionality for external plan files.