

# Deployment Plans-Level-1

{scrollbar}

top

## First step on Deployment plans for Geronimo v1.1

- [Introduction](#)
- [#Apache Geronimo v1.1 Deployment Plans](#)
- [#Work out Deployment Plans with Samples](#)  
(I will include the completed samples here)
  - Deploying and Undeploying Simple HelloWorld (war module,web.xml,geronimo-web.xml)
  - Deploying and Undeploying HelloWorld J2EE Samples(JAR containing EJB ,ejb-jar.xml,openejb.xml)
  - Deploying and Undeploying JCA sample with Geronimo
- [#Discussion](#)
- [#Conclusion](#)

## Introductionintro

Every service,application or resource in Geronimo is configured with an xml deployment plan.Deployment plans are the Geronimo version of the J2EE deployment descriptors. They are still XML files based on XML schemas and containing the configuration details for a specific application module. Via the deployment plans you can not only deploy application modules but also other configurations such as a security realm etc..This is the first step in Geronimo v1.1 deployment plans which with cover a simple sample.

The next step of deployment plan will be ready for users soon, which will provide you with complex and advance samples.This first step will help beginners of Apache Geronimo v1.1 get an insight.

## Apache Geronimo v1.1 Deployment Plans

The following table illustrates the deployment descriptors name and file location for both standard J2EE and Apache Geronimo specific.

File	Standard Deployment Descriptors in the J2EE specification	Apache Geronimo specific Deployment plan
Web Application Archive (WAR)	web.xml under the WEB-INF directory	geronimo-web.xml
Enterprise Web application archive (EAR)	application.xml	geronimo-application.xml
J2EE Connector resources archive (RAR)	ra.xml	geronimo-ra.xml
J2EE client application archive (JAR)	client.xml	geronimo-application-client.xml
JAR containing EJBs	ejb-jar.xml under the META-INF directory	openejb-jar.xml

## Work Out Deployment Plans with Samples

### Assumptions

It is assumed that you have installed the either Tomcat or the Jetty version of Geronimo v1.1 successfully and it is working. Please find the further details to get the software and install it.(need to direct the link to Installation Guide and the User Guide)

### Deployment plan for WAR Module

#### Deploying and Undeploying Helloworld sample.

This section will cover understanding how the deployment plan works for different application modules.

First let's have look at the simple "HelloWorld"example which creates a Web Application Archive (WAR) under the WEB-INF directory and has a geronimo-web.xml as the Apache Geronimo specific deployment Plan.

Here are the steps to understand the deployment plan for the "HelloWorld" Example

- 1.Create a folder in called <app\_home> in your working directory
- 2.Open a new text file and save it as "HelloWorld.jsp" in side the app\_home directory
- 3.Copy and paste the following code with the "HelloWorld.jsp" in it.

```
solidHelloWorld.jsp <%@ page language="java" contentType="text/html; charset=ISO-8859-1" pageEncoding="ISO-8859-1"%> <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"> <html> <head> <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"> <title>HelloWorld!</title> </head> <body bgcolor="#707DB8"> <h1> <font face="courier" color="white"> Hello world from GERONIMO V1.1! </font> </h1> <font face="courier" color="white"> ${datetime}</font> </html>
```

4. Create a new folder called **WEB-INF** inside the `app_home` directory.

5. Open a new text file and save it as "geronimo-web.xml" and this is the Apache Geronimo v1.1 deployment plan for this sample module.

6. Copy and paste the following xml code in that file and save it inside the `<app_home\WEB-INF>` directory.

```
solidgeronimo-web.xml <?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1"> <dep:environment
xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.1"> <dep:moduleId> <dep:groupId>geronimo</dep:groupId> <dep:artifactId>HelloWorld<
/dep:artifactId> <dep:version>1.1</dep:version> <dep:type>war</dep:type> </dep:moduleId> </dep:environment> <context-root>/hello</context-root> <
/web-app>
```

7. Open another new text file save it in `<app_home\WEB-INF>` directory and name it as "web.xml"

8. Copy and paste the following xml code in it and save.

```
solidweb.xml <?xml version="1.0" encoding="UTF-8"?> <web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org
/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"> <welcome-file-list>
<welcome-file>HelloWorld.jsp</welcome-file> </welcome-file-list> </web-app>
```

9. The next step is packaging the application. From a command line window change the directory to `<app_home>` and run the following command to package the HelloWorld application into a single .war file.

This will create a HelloWorld.war file in the `<app_home>` directory. The next step is to deploy this application.

10. User can use the preferred deployment approach either command line or the Geronimo Web Console. Here it describes how to use the command line.

11. Change directory to `<geronimo_home>/bin` and run the following command:

12. For further instructions about deployment process please refer the Quick Start or Sample Applications working. The sample application code can be downloaded here.

[HelloWorld web Sample](#)

Access Hello World web sample

<http://localhost:8080/hello>

## Deploying JAR containing Simple EJB application

This section will cover Geronimo v1.1 deployment plan for the simple helloworld ejb.

Here is the folder structure for the above simple application. Sample Application is available to download here

[HelloWorld ejb Sample](#)

```
java HelloWorld | _ejbModule | | _org.geronimo.ejbsample | | | _HelloBean | | | _HelloHome | | | _HelloObject | | | | _org.geronimo.ejbsample.client |
| _HelloWorld | _META-INF | | _ejb-jar.xml | | _openejb-jar.xml | | _dist | | _build | | _build.xml
```

Here is the ejb-jar.xml for the above sample

```
solidejb-jar.xml <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise JavaBeans 2.0//EN"
"http://java.sun.com/dtd/ejb-jar_2_0.dtd"> <ejb-jar> <enterprise-beans> <session> <ejb-name>HelloBean</ejb-name> <home>org.geronimo.ejbsample.
HelloHome</home> <remote>org.geronimo.ejbsample.HelloObject</remote> <ejb-class>org.geronimo.ejbsample.HelloBean</ejb-class> <session-
type>Stateless</session-type> <transaction-type>Container</transaction-type> </session> </enterprise-beans> <assembly-descriptor> <container-
transaction> <method> <ejb-name>HelloBean</ejb-name> <method-name>*</method-name> </method> <trans-attribute>Required</trans-attribute> <
/container-transaction> </assembly-descriptor> </ejb-jar>
```

Geronimo v1.1 deployment plan for the above sample

```
solidopenejb.jar <openejb-jar xmlns="http://www.openejb.org/xml/ns/openejb-jar-2.1"> <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns
/deployment-1.1"> <dep:moduleId> <dep:groupId>org.geronimo.</dep:groupId> <dep:artifactId>ejbsample</dep:artifactId> <dep:version>1.0</dep:
version> <dep:type>car</dep:type> </dep:moduleId> <dep:dependencies> <dep:hidden-classes> <dep:non-overridable-classes> </dep:environment>
<enterprise-beans> <session> <ejb-name>HelloBean</ejb-name> <jndi-name>org.geronimo.ejbsample.HomeBean</jndi-name> </session> </enterprise-
beans> </openejb-jar>
```

### Steps to deploying the sample

There are two ways in which the sample can be deployed :-

1. The sample can be built from scratch using the build.xml and editing the `geronimo_home` as your directory. Obtain the helloworld-ejbs.jar from the **dist** directory and use the Geronimo web console. Browse for the helloworld-ejbs.jar file and click on the **Install** button.

2. Use the following command line code to deploy the application.

### Running the Client to test the HelloWorld EJB service

`org.geronimo.ejbsample.client.HelloWorld.java` is the client code for the above application. Set the necessary class path to run client in your environment.

You will see the "Hello world" print on your command line

\*Image needed to be upload.

## Deploying the EAR Application On Geronimo v1.1

This sample application gives you an overview of basic steps to deploy an EAR application in Geronimo v1.1.

Though this is a simple application it helps new users to understand the basics of Geronimo v1.1 specific deployment plan for an EAR and move on to work with complex applications. In this sample ,Servlets are used in the back end while JSP is used in the front end.

The following diagram depicts the folder structure

```
java HelloWorldEar | _src | | _HelloWorld.java | _web | | _index.jsp | _build | | _src | | | _HelloWorld.class | | _ear | | | _META_INF | | | | _application.xml | | |
| _geronimo-application.xml | | | _helloworld.war | | | | _war | | | _WEB-INF | | | | _classes | | | | _web.xml | | | | _geronimo-web.xml | | | _index.jsp | | |
| _deploymentdescriptors | | _application.xml | | _geronimo-application.xml | | _web.xml | | _geronimo-web.xml | | _build.xml | | _helloworld.ear
```

The HelloWorldEar application's content can be described as follows:-

- src-HelloWorld servlet
  - web-JSP
  - build -where all the build outputs go and it contains sub folders such as; src ,war,ear,deploymentdescriptors.
  - build.xml-The build file is also attached in the source bundle. In order to run this build file,change the class paths and <geronimo\_home> as relevant to your environment.
  - helloworld.ear - This is the final distribution of the application and it will be created at the root folder as shown.
- Source code can be downloaded from [here](#)

### HelloWorldEar EAR Sample

Here is the application.xml for the above application

```
solidapplication.xml <?xml version="1.0" encoding="ISO-8859-1"?> <application> <display-name>HelloWorldEar</display-name> <module> <web> <web-
uri>helloworld.war</web-uri> <context-root>/hello</context-root> </web> </module> </application>
```

Here is the Geronimo specific deployment plan geronimo-application.xml

```
solidgeronimo-application.xml <application application-name="HelloWorldEar" xmlns="http://geronimo.apache.org/xml/ns/j2ee/application-1.1" xmlns:sec="
http://geronimo.apache.org/xml/ns/security-1.1" xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1"> <sys:environment> <sys:moduleId> <sys:
groupId>default</sys:groupId> <sys:artifactId>HelloWorldEar</sys:artifactId> <sys:version>1-default</sys:version> <sys:type>car</sys:
type> </sys:moduleId> <sys:dependencies/> <sys:hidden-classes/> <sys:non-overridable-classes/> </sys:environment> </application>
```

Here is the web.xml

```
solidweb.xml <?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1"> <dep:environment xmlns:
dep="http://geronimo.apache.org/xml/ns/deployment-1.1"> <dep:moduleId> <dep:groupId>geronimo</dep:groupId> <dep:artifactId>HelloWorldEar</
dep:artifactId> <dep:version>1.1</dep:version> <dep:type>war</dep:type> </dep:moduleId> </dep:environment> <context-root>/hello</context-root> <
/web-app>
```

Here is the geronimo-web.xml for the above WAR module.

```
solid <?xml version="1.0" encoding="UTF-8"?> <web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1"> <dep:environment xmlns:dep="
http://geronimo.apache.org/xml/ns/deployment-1.1"> <dep:moduleId> <dep:groupId>geronimo</dep:groupId> <dep:artifactId>HelloWorldEar</dep:
artifactId> <dep:version>1.1</dep:version> <dep:type>war</dep:type> </dep:moduleId> </dep:environment> <context-root>/hello</context-root> </web-
app>
```

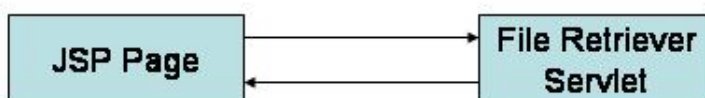
Direct deployment can be done using the Geronimo v1.1 Web console or else application can be deployed using the following command.

## Deploying a J2EE Connector resources archive (RAR) with Geronimo v1.1

This section will describe the deployment plan for RAR application with Geronimo v1.1.The base of this sample has been borrowed and modified from User Guide.(Migration to Apache Geronimo-JBoss to Geronimo JCA migration.html)This sample contains a File Retriever JCA sample application it consists of two modules that, (for the purpose of this sample)for this sample purposes, are deployed separately and not as a single EAR archive. The first module is a simple file system resource adapter that conforms to the JCA architecture without any server-specific extensions. The adapter provides two functions:

- Lists names of files and directories located in a specific repository being a directory in the underlying file system.
- Retrieves the content of one of the files.

The second module is a Web application that uses the adapter and provides online access to the files.



The user views a list of directories and files located in the repository and clicks a file to download it from the server. By default the resource adapter is configured so that the root directory of an application server is used as the repository.

Folder structure for the JCA sample is shown below.

jca | \_src | \_meta | \_geronimo | \_geronimo-ra.xml | \_ra.xml | \_jca-plan.xml | \_web | \_WEB-INF | \_geronimo-web.xml | \_web.xml | \_jca.rar | \_jca.war

#### J2EE RAR Deployment Plan

```
solidra.xml <connector xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/connector_1_5.xsd" version="1.5" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <display-name>FileRetriever</display-name> <vendor-name>IBM</vendor-name> <eis-type>File system</eis-type> <resourceadapter-version>1.0</resourceadapter-version> <resourceadapter> <outbound-resourceadapter> <connection-definition> <managedconnectionfactory-class>com.ibm.j2g.jca.connector.impl.FileRetrieverManagedConnectionFactory </managedconnectionfactory-class> <config-property> <description>Path to the directory being the file repository</description> <config-property-name>RepositoryPath</config-property-name> <config-property-type>java.lang.String</config-property-type> </config-property> <connectionfactory-interface>com.ibm.j2g.jca.connector.FileRetrieverConnectionFactory </connectionfactory-interface> <connectionfactory-impl-class>com.ibm.j2g.jca.connector.impl.FileRetrieverConnectionFactoryImpl </connectionfactory-impl-class> <connection-interface>com.ibm.j2g.jca.connector.FileRetrieverConnection</connection-interface> <connection-impl-class>com.ibm.j2g.jca.connector.impl.FileRetrieverConnectionImpl </connection-impl-class> </connection-definition> <transaction-support>NoTransaction</transaction-support> <reauthentication-support>false</reauthentication-support> </outbound-resourceadapter> </resourceadapter> </connector>
```

#### J2EE RAR Geronimo v1.1 deployment plan

```
solidgeronimo-ra.xml <connector version="1.5" xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.1"> <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.1"> <dep:moduleid> <dep:groupid>com</dep:groupid> <dep:artifactid>ibm</dep:artifactid> <dep:version>j2g</dep:version> <dep:type>jca.rar</dep:type> </dep:moduleid> <dep:dependencies/> <dep:hidden-classes/> <dep:non-overrideable-classes/> </dep:environment> <resourceadapter> <outbound-resourceadapter> <connection-definition> <connectionfactory-interface>com.ibm.j2g.jca.connector.FileRetrieverConnectionFactory </connectionfactory-interface> <connectiondefinition-instance> <name>FileRetriever</name> <!--The following path refers to the Geronimo home directory--> <config-property-setting name="RepositoryPath">../</config-property-setting> <connectionmanager> <no-transaction/> <no-pool/> </connectionmanager> </connectiondefinition-instance> </connection-definition> </outbound-resourceadapter> </resourceadapter> </connector>
```

To deploy the adapter on Geronimo you also need a deployment plan specific to this server. Such a plan can be either named `geronimo-ra.xml` and be placed into the `META-INF` folder of the corresponding RAR archive, or can have any name and stay outside the archive. In the latter case, the path to the plan should be specified during the deployment of the resource adapter.

The deployment plan used by the sample application contains the repository path configuration parameter that may be updated before the deployment. Because of this, the `jca-plan.xml` file is not included in the RAR archive but it is placed into the `<jca_home>/meta/geronimo` directory instead. The content of this file is listed in the following example.

```
solid <!--This is Geronimo-specific descriptor--> <connector version="1.5" xmlns="http://geronimo.apache.org/xml/ns/j2ee/connector-1.1"> <dep:environment xmlns:dep="http://geronimo.apache.org/xml/ns/deployment-1.1"> <dep:moduleid> <dep:groupid>com</dep:groupid> <dep:artifactid>ibm</dep:artifactid> <dep:version>j2g</dep:version> <dep:type>jca.rar</dep:type> </dep:moduleid> <dep:dependencies> <dep:dependency> <dep:groupid>geronimo</dep:groupid> <dep:artifactid>geronimo-connector</dep:artifactid> <!-- <dep:type>car</dep:type>--> </dep:dependency> </dep:dependencies> <dep:hidden-classes/> <dep:non-overrideable-classes/> </dep:environment> <resourceadapter> <outbound-resourceadapter> <connection-definition> <connectionfactory-interface>com.ibm.j2g.jca.connector.FileRetrieverConnectionFactory </connectionfactory-interface> <connectiondefinition-instance> <name>FileRetriever</name> <config-property-setting name="RepositoryPath">../</config-property-setting> <connectionmanager> <no-transaction/> <no-pool/> </connectionmanager> </connectiondefinition-instance> </connection-definition> </outbound-resourceadapter> </resourceadapter> </connector>
```

In this deployment plan the attribute `<dep:moduleid>` specifies the unique name `com/ibm/j2g/jca.rar` used to identify the RAR module in Geronimo and that this module is deployed on the server as a separate component.

The descriptor also:

- Defines the `FileRetriever` name which looks up the resource adapter factory.
- Specifies that neither transactions nor connection pools are supported.
- Provides the value of the repository path configuration parameter.

### Building the sample Application.

In order to build the modules of the File Retriever JCA application you need to update the `build.properties` file and property such as `<geronimo_home>` matches your environment.

Next step is the deploying the RAR application with Geronimo.

As stated in the previous examples user will have two choices of deploying an application with Geronimo either Geronimo web console or command line. Here the command line option is described following.

From a command line, change directory to `<jca_home>` and type the following commands:

Deploying `jca-plan.xml` and RAR

Once the JCA application is deployed, open a Web browser and access the following URL:

<http://localhost:8080/jca>

## Deploying an J2EE client application archive (JAR) with Geronimo v1.1

## Conclusion