Deployer tool

{scrollbar}

top

The deployer application is a Java application that manages J2EE artifacts and GBean components in the Geronimo server. If Geronimo is running, it will connect to the server and perform its action through the server's deployment service. If it cannot find a running server, it will throw an error stating it could not connect to the server or the server is unavailable.

The deployment tool can be started by using the **java -jar** to invoke the main class in <geronimo_home>/bin/deployer.jar.

Typically, the deployment tool is started by just using the **deploy** script, but you can also run the application by starting a Java virtual machine using the following syntax:

```
java -jar deployer.jar <general_options> <command> <command_options>
```

where **<general_options>** specify common options that apply to all commands and control how the application behaves, **<command>** is a command name that specifies the action to be performed, and **<command_options>** are options unique to the command specified.

General options

This section lists all the available general options for the Geronimo deployer tool.

- **--uri** <identifier>
  Where <identifier> is a Universal Resource Identifier (URI) that specifies how the deployer is to contact the server. If this flag is not specified, the deployer will attempt to contact the server using the standard port on localhost. The identifier must have the following form:
  deployer:geronimo:jmx:rmi:///jndi/rmi:[//host[:port]]/JMXConnector
  where <host> is replaced with the host name or TCP/IP address of the system where the server is running and <port> is replaced with the port number where the server is listening. If unspecified, localhost and the default port will be used.

- **--host** <host>
  Where <host> is the host name of the server you are trying to deploy that application or resource. This option allows you to deploy resources and applications to a remote server. This parameter is optional and defaults to localhost.

- **--port** <port>
  Where <port> is the port of the remote server you are trying to deploy that application or resource. This parameter is optional and defaults to port 1099.

- **--driver** <driver_path>
  Where <driver_path> is the path to the driver JAR if you want to use this tool with a server other than Geronimo. Currently, manifest Class-Path entries in that JAR are ignored.

- **--user** <username>
  Where <username> is a user name authorized to be an administrator on the server. If the command requires authorization, you must use this option.

- **--password** <password>
  Where <password> is a the password required to authenticate the user name. If this flag is not specified, the deployer will attempt to perform the command with no password, but if that fails, it will prompt you to enter a password.

- **--syserr** <select>
  Where <select> can be either true or false. If this flag is unspecified. false is assumed. Specify true when you want errors to be logged to the syserr device.

- **--verbose** <select>
  Where <select> can be either true or false. If this flag is unspecified. false is assumed. Specify true when you need more messages to determine the cause of an error.

Back to top

Commands

The available commands for the Geronimo deployer tool are listed below:

- Common commands
  - #deploy
  - #login
  - #redeploy
  - #start
  - #stop
  - #undeploy
- Other commands
  - #distribute
  - #list-modules
  - #list-targets

- Geronimo Plugins
    - #install-plugin
    - #search-plugins

Additionally, you can type **help** for further details on a given command, the syntax is as follows:

```
java -jar deployer.jar help <commands>
```

Back to top

Deploy deploy

Use the **deploy** command to add and start a new module. The deploy command has the following syntax:

```
java -jar deployer.jar <general_options> deploy <module> <deployment_plan>
```

The <module> specifies the application file name and location. The <deployment_plan> specifies the file name and location of the XML with the deployment plan. Sometimes the application module already has included in the package a deployment plan or the application is so simple that does not require any deployment plan, in these cases this parameter can be omited.

A module file can be one of the following:

- J2EE Enterprise Application Archive (EAR) file
- J2EE Web Application Archive (WAR) file
- J2EE Enterprise JavaBean Archive (JAR) file
- J2EE Java Resource Archive (RAR) file

If the server is not currently running at the time of deploying the application, the module will be marked to start next time the server is started.

The most common <general_options> would be --user and --password. The **--inPlace** option allows you point to and deploy an application directly from a directory external to Geornimo without the need for even packaging the application. In other words, you can have an application **running** in Geronimo but that application may be anywhere else on the file system.

To use this option you should type:

```
java -jar deployer.jar <general_options> deploy --inPlace <app_home>
```

Where <app_home> indicates the home directory where you have your application (exploded).

You can also deploy applications if Geronimo is not running by using the **--offline** option, the syntax for this command would be:

```
java -jar deployer.jar <general_options> --offline deploy <module>
```

Off course, you can also combine --offline and --inPlace

```
java -jar deployer.jar <general_options> --offline deploy --inPlace <app_home>
```

Back to top

Login login

Use the **login** command to save the username and password for the current connection to the file `.geronimo-deployer` in the current user's home directory. Future connections to the same server will try to use this saved authentication information instead of prompting where possible.

This information will be saved separately per connection URL, so you can specify --url or --host and/or --port on the command line to save a login to a different server.

The **login** command has the following syntax:

```
java -jar deployer.jar --user <user_name> --password <password> login
```

So, next time you run a different command that originally required user name and password, you can run the command directly, for example:

```
deploy list-modules
```

Even when the login information is not saved in clear text, it is not secure either. If you want to save the authentication securely, you should change the `.geronimo-deployer` file in your home directory so that nobody else can read or write it.

Back to top

Redeploy redeploy

Use the **redeploy** command to stop, replace and restart a module that has been deployed before. The redeploy command has the following syntax:

```
java -jar deployer.jar <general_options> redeploy <module> <deployment_plan>
```

Just like the deploy command, the redeploy command accepts the following modules file types:

- J2EE Enterprise Application Archive (EAR) file
- J2EE Web Application Archive (WAR) file
- J2EE Enterprise JavaBean Archive (JAR) file
- J2EE Java Resource Archive (RAR) file

Typically, both a module and a plan are specified. If the module contains a plan or if a default plan can be used, the plan can be omitted. However, if a plan is specified in this case, it overrides the other plans. If the plan references a server component already deployed in the server's environment, the module is omitted.

Back to top

Start start

Use the **start** command to start a previously deployed module. The start command has the following syntax:

```
java -jar deployer.jar <general_options> start <moduleIDs>
```

Where <moduleIDs> is a list of one or more modules (configID) separated by blank space. The module identification (or ConfigID) is defined at deployment time in the respective deployment plan for each module previously deployed.

Back to top

Stop stop

Use the **stop** command to stop a running module. The stop command has the following syntax:

```
java -jar deployer.jar <general_options> stop <moduleIDs>
```

Where <moduleIDs> is a list of one or more modules (configID) separated by blank space. The module identification (or ConfigID) is defined at deployment time in the respective deployment plan for each module previously deployed.

Back to top

Undeploy undeploy

Use the **undeploy** command to stop and remove a module (running or not) and its deployment information from the server. The undeploy command has the following syntax:

```
java -jar deployer.jar <general_options> undeploy <moduleIDs>
```

Where <moduleIDs> is a list of one or more modules (configID) separated by blank space. The module identification (or ConfigID) is defined at deployment time in the respective deployment plan for each module previously deployed.

This command has the same ability as with **deploy** to uninstall applications when the server is not running, this command has the following syntax:

```
java -jar deployer.jar <general_options> --offline undeploy <moduleID>
```

Back to top

Distribute distribute

Use the **distribute** command to add a new module to the server. This command does not start the module nor mark it to be started in the future. The distribute command has the following syntax:

```
java -jar deployer.jar <general_options> distribute <module> <deployment_plan>
```

Just like with the deploy command, <module> specifies the application file name and location. The <deployment_plan> specifies the file name and location of the XML with the deployment plan. Sometimes the application module already has included in the package a deployment plan or the application is so simple that does not require any deployment plan, in these cases this parameter can be omited.

A module file can be one of the following:

- J2EE Enterprise Application Archive (EAR) file
- J2EE Web Application Archive (WAR) file
- J2EE Enterprise JavaBean Archive (JAR) file
- J2EE Java Resource Archive (RAR) file

Back to top

List-modules list-modules

Use the **list-modules** command to list all available modules on the server, note that for running this command the server must be running. The list-modules command has the following syntax:

```
java -jar deployer.jar <general_options> list-modules [--all|--started|--stopped]
```

- --all : is used by default when no other option is specified. It will list all the available modules.
- --started : this option will list only the modules that are running.
- --stopped : this option will list only the modules that are not running.

Back to top

List-targets list-targets

Use the **list-targets** command to lists the targets known to the server you have connected to. The list-targets command has the following syntax:

```
java -jar deployer.jar <general_options> list-targets
```

In the case of Geronimo, each configuration store is a separate target. Geronimo does not yet support clusters as targets.

Back to top

Install-plugin install-plugin

Use the **install-plugin** command to install a Geronimo plugin previously exported from a Geronimo server or downloaded from a repository. A Geronimo plugin can be an application, a configuration such data sources and drivers or a combination. The install-plugin command has the following syntax:

```
java -jar deployer.jar install-plugin <plugin_file>
```

Back to top

Search-plugins search-plugins

Use the **search-plugins** command to list all the Geronimo plugins available in a Maven repository. The search-plugins command has the following syntax:

```
java -jar deployer.jar search-plugins <maven_repository_URL>
```

Back to top