**Name** deploy – deploys the specified component

**Synopsis**

```
deploy [--help]
[--force={false|true}]
[--virtualservers virtual_servers]
[--contextroot context_root]
[--precompilejsp={false|true}]
[--verify={false|true}]
[--name component_name]
[--upload={true|false}]
[--retrieve local_dirpath]
[--dbvendorname dbvendorname]
[--createtables={true|false}|--dropandcreatetables={true|false}]
[--uniquetablenames={true|false}]
[--deploymentplan deployment_plan]
[--enabled={true|false}]
[--generatermistubs={false|true}]
[--availabilityenabled={false|true}]
[--asyncreplication={true|false}]
[--lbenabled={true|false}]
[--keepstate={false|true}]
[--libraries jar_file[,jar_file]*]
[--target target]
[--type pkg-type]
[--properties(name=value)[:name=value]*]
[file_archive|filepath]
```

**Description** The `deploy` subcommand deploys applications to the server. Applications can be enterprise applications, web applications, Enterprise JavaBeans (EJB) modules, connector modules, and application client modules. If the component is already deployed or already exists, it is forcibly redeployed if the `--force` option is set to `true` (default is `false`).

The `--createtables` and `--dropandcreatetables` options are boolean flags and therefore can take the values of *true* or *false*. These options are only used during deployment of CMP beans that have not been mapped to a database (that is, no `sun-cmp-mappings.xml` descriptor is provided in the module's `META-INF` directory). They are ignored otherwise.

The `--createtables` and `--dropandcreatetables` options are mutually exclusive; only one should be used. If drop and/or create tables fails, the deployment does not fail; a warning message is provided in the log file.

This subcommand is supported in remote mode only.

**Options** `--help`
`-?`
Displays the help text for the subcommand.

`--force`
If set to `true`, redeploys the component even if the specified component has already been deployed or already exists. Default is `false`.

`--virtualservers`
One or more virtual server IDs. Multiple IDs are separated by commas.

`--contextroot`
Valid only if the archive is a web module. It is ignored for other archive types; defaults to filename without extension.

`--precompilejsp`
By default this option does not allow the JSP to be precompiled during deployment. Instead, JSPs are compiled during runtime. Default is `false`.

`--verify`
If set to true and the required verifier packages are installed from the Update Tool, the syntax and semantics of the deployment descriptor is verified. Default is `false`.

`--name`
Name of the deployable component.

The name can include an optional version identifier, which follows the name and is separated from the name by a colon (`:`). The version identifier must begin with a letter or number. It can contain alphanumeric characters plus underscore (`_`), dash (`-`), and period (`.`) characters. For more information about module and application versions, see the "Module and Application Versions" in *GlassFish Server Open Source Edition 3.1 Application Deployment Guide*.

`--upload`
Specifies whether the subcommand uploads the file to the DAS. In most situations, this option can be omitted.

Valid values are as follows:

`false`
The subcommand does not upload the file and attempts to access the file through the specified file name. If the DAS cannot access the file, the subcommand fails.

For example, the DAS might be running as a different user than the administration user and does not have read access to the file. In this situation, the subcommand fails if the `--upload` option is `false`.

`true`
The subcommand uploads the file to the DAS over the network connection.

The default value depends on whether the DAS is on the host where the subcommand is run or is on a remote host.

- If the DAS is on the host where the subcommand is run, the default is `false`.

- If the DAS is on a remote host, the default is `true`.

If a directory *filepath* is specified, this option is ignored.

`--retrieve`
Retrieves the client stub JAR file from the server machine to the local directory.

`--dbvendorname`
Specifies the name of the database vendor for which tables are created. Supported values include `db2`, `mssql`, `mysql`, `oracle`, `derby`, `javadb`, `postgresql`, and `sybase`. These values are case-insensitive. If not specified, the value of the `database-vendor-name` attribute in `glassfish-ejb-jar.xml` is used. If no value is specified, a connection is made to the resource specified by the `jndi-name` subelement of the `cmp-resource` element in the `glassfish-ejb-jar.xml` file, and the database vendor name is read. If the connection cannot be established, or if the value is not recognized, SQL-92 compliance is presumed.

`--createtables`
If specified as true, creates tables at deployment of an application with unmapped CMP beans. If specified as false, tables are not created. If not specified, the value of the `create-tables-at-deploy` entry in the `cmp-resource` element of the `glassfish-ejb-jar.xml` file determines whether or not tables are created. No unique constraints are created for the tables.

`--dropandcreatetables`
If specified as true when the component is redeployed, the tables created by the previous deployment are dropped before creating the new tables. Applies to deployed applications with unmapped CMP beans. Preexisting tables will not be dropped on the initial deployment of an application or on a deployment that follows an explicit undeploy. If specified as false, tables are neither dropped nor created. If not specified, the tables are dropped if the `drop-tables-at-undeploy` entry in the `cmp-resource` element of the `glassfish-ejb-jar.xml` file is set to true, and the new tables are created if the `create-tables-at-deploy` entry in the `cmp-resource` element of the `glassfish-ejb-jar.xml` file is set to true.

`--uniquetablenames`
Guarantees unique table names for all the beans and results in a hash code added to the table names. This is useful if you have an application with case-sensitive bean names. Applies to applications with unmapped CMP beans.

`--deploymentplan`
Deploys the deployment plan, which is a JAR file that contains GlassFish Server descriptors. Specify this option when deploying a pure EAR file. A pure EAR file is an EAR without GlassFish Server descriptors.

`--enabled`
Allows users to access the application. If set to `false`, users will not be able to access the application. This option enables the application on the specified target instance or cluster.

If you deploy to the target `domain`, this option is ignored, since deploying to the domain doesn't deploy to a specific instance or cluster. The default is `true`.

`--generatermistubs`
If set to `true`, static RMI-IIOP stubs are generated and put into the `client.jar`. If set to `false`, the stubs are not generated. Default is `false`.

`--availabilityenabled`
This option controls whether high-availability is enabled for web sessions and for stateful session bean (SFSB) checkpointing and potentially passivation. If set to false (default) all web session saving and SFSB checkpointing is disabled for the specified application, web application, or EJB module. If set to true, the specified application or module is enabled for high-availability. Set this option to true only if high availability is configured and enabled at higher levels, such as the server and container levels.

`--asyncreplication`
This option controls whether web session and SFSB states for which high availability is enabled are first buffered and then replicated using a separate asynchronous thread. If set to true (default), performance is improved but availability is reduced. If the instance where states are buffered but not yet replicated fails, the states are lost. If set to false, performance is reduced but availability is guaranteed. States are not buffered but immediately transmitted to other instances in the cluster.

`--lbenabled`
This option controls whether the deployed application is available for load balancing. The default is true.

`--keepstate`
This option controls whether web sessions, SFSB instances, and persistently created EJB timers are retained between redeployments.

The default is false. This option is supported only on the default server instance, named `server`. It is not supported and ignored for any other target.

Some changes to an application between redeployments prevent this feature from working properly. For example, do not change the set of instance variables in the SFSB bean class.

For web applications, this feature is applicable only if in the `glassfish-web-app.xml` file the `persistence-type` attribute of the `session-manager` element is `file`.

For stateful session bean instances, the persistence type without high availability is set in the server (the `sfsb-persistence-type` attribute) and must be set to `file`, which is the default and recommended value.

If any active web session, SFSB instance, or EJB timer fails to be preserved or restored, *none* of these will be available when the redeployment is complete. However, the redeployment continues and a warning is logged.

To preserve active state data, GlassFish Server serializes the data and saves it in memory. To restore the data, the class loader of the newly redeployed application deserializes the data that was previously saved.

`--libraries`
A comma-separated list of library JAR files. Specify the library JAR files by their relative or absolute paths. Specify relative paths relative to *domain-dir*`/lib/applibs`. The libraries are made available to the application in the order specified.

`--target`
Specifies the target to which you are deploying. Valid values are:

`server`
Deploys the component to the default server instance `server` and is the default value.

`domain`
Deploys the component to the domain. If `domain` is the target for an initial deployment, the application is deployed to the domain, but no server instances or clusters reference the application. If `domain` is the target for a redeployment (the `--force` option is set to true), and dynamic reconfiguration is enabled for the clusters or server instances that reference the application, the referencing clusters or server instances automatically get the new version of the application. If redeploying, and dynamic configuration is disabled, the referencing clusters or server instances do not get the new version of the application until the clustered or standalone server instances are restarted.

*cluster_name*
Deploys the component to every server instance in the cluster.

*instance_name*
Deploys the component to a particular stand-alone sever instance.

`--type`
The packaging archive type of the component that is being deployed. Possible values are as follows:

`osgi`
The component is packaged as an OSGi Alliance bundle.

The `--type` option is optional. If the component is packaged as a regular archive, omit this option.

`--properties` or `--property`
Optional keyword-value pairs that specify additional properties for the deployment. The available properties are determined by the implementation of the component that is being deployed or redeployed. The `--properties` option and the `--property` option are equivalent. You can use either option regardless of the number of properties that you specify.

You can specify the following properties for a deployment:

`jar-signing-alias`
Specifies the alias for the security certificate with which the application client container JAR file is signed. Java Web Start will not run code that requires elevated permissions unless it resides in a JAR file signed with a certificate that the user's system trusts. For your convenience, GlassFish Server signs the JAR file automatically using the certificate with this alias from the domain's keystore. Java Web Start then asks the user whether to trust the code and displays the GlassFish Server certificate information. To sign this JAR file with a different certificate, add the certificate to the domain keystore, then use this property. For example, you can use a certificate from a trusted authority, which avoids the Java Web Start prompt, or from your own company, which users know they can trust. Default is `s1as`, the alias for the self-signed certificate created for every domain.

`java-web-start-enabled`
Specifies whether Java Web Start access is permitted for an application client module. Default is true.

`compatibility`
Specifies the GlassFish Server release with which to be backward compatible in terms of JAR visibility requirements for applications. The only allowed value is `v2`, which refers to Sun GlassFish Enterprise Server version 2 or Sun Java System Application Server version 9.1 or 9.1.1. The Java EE 6 platform specification imposes stricter requirements than Java EE 5 did on which JAR files can be visible to various modules within an EAR file. In particular, application clients must not have access to EJB JAR files or other JAR files in the EAR file unless references use the standard Java SE mechanisms (extensions, for example) or the Java EE library-directory mechanism. Setting this property to `v2` removes these Java EE 6 restrictions.

`keepSessions={false|true}`
Superseded by the `--keepstate` option.

If the `--force` option is set to `true`, this property can by used to specify whether active sessions of the application that is being redeployed are preserved and then restored when the redeployment is complete. Applies to HTTP sessions in a web container. Default is `false`.

`false`
Active sessions of the application are *not* preserved and restored (default).

`true`
Active sessions of the application are preserved and restored.

If any active session of the application fails to be preserved or restored, *none* of the sessions will be available when the redeployment is complete. However, the redeployment continues and a warning is logged.

To preserve active sessions, GlassFish Server serializes the sessions and saves them in memory. To restore the sessions, the class loader of the newly redeployed application deserializes any sessions that were previously saved.

`preserveAppScopedResources`
If set to `true`, preserves any application-scoped resources and restores them during redeployment. Default is `false`.

Other available properties are determined by the implementation of the component that is being redeployed.

**Operands** *file_archive|filepath*
The path to the archive that contains the application that is being deployed. This path can be a relative path or an absolute path.

The archive can be in either of the following formats:

- An archive file, for example, `/export/JEE_apps/hello.war`.

  If the `--upload` option is set to `true`, this is the path to the deployable file on the local client machine. If the `--upload` option is set to `false`, this is the path to the file on the server machine.
- A directory that contains the exploded format of the deployable archive. This is the path to the directory on the server machine.

  If you specify a directory, the `--upload` option is ignored.

**Examples** **EXAMPLE 1** Deploying an Enterprise Application

This example deploys the enterprise application packaged in the `Cart.ear` file to the default server instance `server`. You can use the `--target` option to deploy to a different server instance or to a cluster.

```
asadmin> deploy Cart.ear
Application deployed successfully with name Cart.
Command deploy executed successfully
```

**EXAMPLE 2** Deploying a Web Application With the Default Context Root

This example deploys the web application in the `hello.war` file to the default server instance `server`. You can use the `--target` option to deploy to a different server instance or to a cluster.

```
asadmin> deploy hello.war
Application deployed successfully with name hello.
Command deploy executed successfully
```

**EXAMPLE 3** Forcibly Deploying a Web Application With a Specific Context Root

This example forcibly deploys the web application in the `hello.war` file. The context root of the deployed web application is `greetings`. If the application has already been deployed, it is redeployed.

**EXAMPLE 3** Forcibly Deploying a Web Application With a Specific Context Root *(Continued)*

```
asadmin> deploy  --force=true --contextroot greetings hello.war
Application deployed successfully with name hello.
Command deploy executed successfully
```

**EXAMPLE 4** Deploying an Enterprise Bean

This example deploys a component based on the EJB specification (enterprise bean) with CMP and creates the database tables used by the bean.

This example uses the `--target` option. The target in this example is an existing cluster, `cluster1`.

```
asadmin> deploy --createtables=true --target cluster1 EmployeeEJB.jar
Application deployed successfully with name EmployeeEJB.
Command deploy executed successfully
```

**EXAMPLE 5** Deploying a Connector Module

This example deploys a connector module that is packaged in an RAR file.

This example uses the `--target` option. The target in this example is an existing standalone server instance that does not belong to a cluster.

```
asadmin> deploy --target myinstance jdbcra.rar
Application deployed successfully with name jdbcra.
Command deploy executed successfully
```

## Exit Status

| | |
|---|---|
| 0 | subcommand executed successfully |
| 1 | error in executing the subcommand |

## See Also

`redeploy`(1), `list-components`(1), `undeploy`(1)

`asadmin`(1M)

*GlassFish Server Open Source Edition 3.1 Application Deployment Guide*