Name deploy– deploys the specified component

Synopsis

```
deploy
[--help]
[ --virtualservers  virtual-servers]
[--contextroot context-root] [--force={false|true}]
[ --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}]
[ --libraries  jar-file[,jar-file]*]
[--target target]
[--type pkg-type]
[--properties(name=value)[:name=value]*]
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`.

This subcommand is supported in remote mode only.

Options

`--help`
Displays the help text for the subcommand.

`--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.

`--force`
If set to true, makes sure the component is redeployed even if the specified component has already been deployed or already exists. The default is false.

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

`--verify`
Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--name`
Name of the deployable component.

`--upload`
When set to true (the default), uploads the deployable file to the administration server. The deployable file must be accessible from the client. If the file is accessible to both server and client, set the `--upload` option to false.

`--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`, `oracle`, `derby`, `javadb`, `postgresql`, `pointbase`, and `sybase`, case-insensitive. If not specified, the value of the `database-vendor-name` attribute in `sun-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 `sun-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`
Creates tables at deployment of an application with unmapped container-managed persistence (CMP) beans. Default is the `create-tables-at-deploy` entry in the `cmp-resource` element of the `sun-ejb-jar.xml` file.

`--dropandcreatetables`
If set to true, when the component is redeployed, the tables created by the previous deployment are dropped before creating the new tables. Applies to already deployed applications with unmapped CMP beans. If not set to true, the tables are dropped if the `drop-tables-at-undeploy` entry in the `cmp-resource` element of the `sun-ejb-jar.xml` file is set to true. The new tables are created if the `create-tables-at-deploy` entry in the `cmp-resource` element of the `sun-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.

`--deploymentplan`
Deploys the deployment plan, which is a JAR containing Sun-specific descriptors. This should be passed along when deploying a pure EAR file. A pure EAR file is an EAR without Sun-specific descriptors.

`--enabled`
If set to true (default), allows users to access the application. If set to false, users will not be able to access the application. In a domain whose profile is the cluster profile or the enterprise profile, 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.

`--generatermistubs`

Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--availabilityenabled`

Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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 *instance-root*`/lib/applibs`. The libraries are made available to the application in the order specified.

`--target`

Do not specify this option. This option is retained for compatibility with other releases. If you specify this option, a syntax error does not occur. Instead, the subcommand runs successfully and the option is silently ignored.

`--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`
`--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. 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:

`default-EE6—app-name`

The default Java EE 6 name of the Java EE application (EAR file). The default Java EE 6 name is not always the same as the name attribute. According to the Java EE 6 specification, the default Java EE 6 name is the archive name minus the suffix.

`java-web-start-enabled`

Specifies whether Java Web Start access is permitted for an application client module. Default is true.

`class-name`
The fully qualified name of a lifecycle module class file. A lifecycle module class must implement the `com.sun.appserv.server.LifecycleListener` interface.

`classpath`
The classpath for a lifecycle module. Specifies where the module is located. Default is the value of `application-root` attribute of the `domain` element.

`load-order`
Determines the order in which lifecycle modules are loaded at startup. Modules with smaller integer values are loaded sooner. Values can range from 101 to the operating system's `MAXINT`. Values from 1 to 100 are reserved.

`is-failure-fatal`
Determines whether the server is shut down if a lifecycle module fails. Default is false.

`jruby-home`
Specifies the directory where the JRuby container is installed. Overrides the `jruby-home` attribute of the JRuby container. Default is *as-install*`/jruby`.

`jruby-runtime`
Specifies the initial number of JRuby runtimes to start. Must be at greater than zero, at least `jruby.runtime.min`, and `jruby.runtime.max` or less. Overrides the `jruby-runtime` attribute of `jruby-runtime-pool`. Default is 1.

`jruby-runtime-min`
Specifies the minimum number of JRuby runtimes in the pool. Must be greater than zero, `jruby.runtime` or less, and `jruby.runtime.max` or less. Overrides the `jruby-runtime-min` attribute of `jruby-runtime-pool`. Default is 1.

`jruby-runtime-max`
Specifies the maximum number of JRuby runtimes in the pool. Must be greater than zero, at least `jruby.runtime.min`, and at least `jruby.runtime`. Overrides the `jruby-runtime-max` attribute of `jruby-runtime-pool`. Default is 1.

`jruby-rackEnv`
Specifies the environment in which a JRuby application such as Rails or Merb runs. Allowed values are development, production, or test. Default is development.

`jruby-applicationtype`
Specifies the name of a supported framework or the path to a script that initializes the user's framework. Allowed values corresponding to supported frameworks are Rails, Merb, or Sinatra. Setting this property bypasses the normal, and potentially lengthy, auto-detection process and forces deployment on the specified framework. If the deployed application is not written for the specified framework, errors result. Default is computed through auto-detection.

`jruby-MTSafe`
If true, specifies that a framework being started using `jruby.applicationType` is thread-safe and therefore does not need a pool created for it. This property affects

applications started using an auto-detected user-provided startup script. If `jruby.applicationType` is set and `jruby.MTsafe` is not set or is set to false, the application starts with a pool of application instances, and each instance of the application is accessed by one thread at a time. This property only affects frameworks being launched where the thread safety cannot be automatically determined. Setting `jruby.MTsafe` to true does not cause an auto-detected Rails 2.1.x application to be launched in thread-safe mode, nor can it be used to force a thread-safe framework to start in pooled mode. Default is computed through auto-detection.

`compatibility`

Specifies the Enterprise Server release with which to be backward compatible in terms of JAR visibility requirements for application clients. The only allowed value is `v2`, which refers to GlassFish version 2 or Enterprise 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.

Operands

filepath

Path to the deployable file on the local client machine if the `upload` option is set to `true`; otherwise the absolute path to the file on the server machine.

Examples

EXAMPLE 1 Deploying an Enterprise Application

This example deploys the enterprise application packaged in the `Cart.ear` file.

```
asadmin> deploy Cart.ear
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.

```
asadmin> deploy hello.war
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.

```
asadmin> deploy  --force=true --contextroot greetings hello.war
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.

```
asadmin> deploy --createtables=true EmployeeEJB.jar
Command deploy executed successfully
```

EXAMPLE 5 Deploying a Connector Module

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

```
asadmin> deploy jdbcra.rar
Command deploy executed successfully
```

Exit Status

0	command executed successfully
1	error in executing the command

See Also

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

`asadmin`(1M)

Sun GlassFish Enterprise Server v3 Preview Application Deployment Guide