Geronimo GShell Commands

GShell Commands

GShell is a command-line processing environment that can be used for the execution of commands. It includes support for editing, command history, input /output redirection, and more. A number of Geronimo administrative commands have been implemented using GShell.:

Commands	Description
help or ?	Display help information
echo or print	Print arguments to STDOUT
source or .	Load a file or URL to the current shell
clear	Clear the terminal screen
set	Set a variable
unset	Unset a variable
exit or quit	Exit GShell
geronimo/start-server	Start a server
geronimo/stop-server	Stop the server
geronimo/wait-for-server	Wait for the server to start
geronimo/start-client	Start an application client
deploy/connect	Connect to a Geronimo server
deploy/login	Save the username and password for this connection
deploy/disconnect	Disconnect from a Geronimo server
deploy/deploy	Deploy a module
deploy/redeploy	Redeploy a module
deploy/undeploy	Undeploy a module
deploy/distribute	Distribute a module
deploy/start	Start a module
deploy/restart	Restart a module
deploy/stop	Stop a module
deploy/list-modules	List modules
deploy/list-targets	List targets
deploy/list-plugins	Install plug-ins into the server
deploy/install-library	Install library
deploy/install-plugin	Install a plug-in
deploy/assemble	Extract a Geronimo server from the current one
deploy/new-instance	Create a new instance
remote/rsh	Connect to a remote GShell server
remote/rsh-server	Start a remote GShell server
cluster/heartbeat	Monitor cluster heartbeat
cluster/deploy	Administer cluster
jaxws/java2ws	Generate WSDL file from class
jaxws/wsgen	Generate JAX-WS artifacts from class
jaxws/wsdl2java	Generate Java classes from WSDL
jaxws/wsimport	Generate JAX-WS artifacts from WSDL

Running GShell and Getting Help

A simple launcher script/.bat file is located in the `<Geronimo_HOME>/bin` directory, where <*Geronimo_HOME*> is the server's installation directory.

1. On linux/Unix/Solaris, execute **<Geronimo_HOME>/bin/gsh.sh**
2. On Windows, execute **<Geronimo_HOME>\bin\gsh.bat**

Note: do not launch GShell with the --secure option (supported in Geronimo 2.1.0.1 or later). It is only supported in GShell commands, but not the scripts.

The **help** (or **?**) command alone will display all GShell commands that are available in the current environment. To obtain help information on any specific command, use the **--help** (**-h**) option. Here is an example:

```
deploy/list-modules --help
```

or

```
deploy/list-modules -h
```

You can use the **exit** or **quit** command to exit GShell.

GShell Commands

Note: Windows users, use forward slash "/" instead of the traditional back slash "\" as directory seperators within GShell commands. Using back slash may cause errors in processing the command.

General Options

Here are common options that apply to most of GShell commands:

Option	Usage	Description
-u, or --user	-u <user>	It is used to provide username. Initially the user name is **system**. If you don't provide this option, you will be prompted to.
-w, or --password	-w <password>	It is used to provide password. Initially the password is **manager**.If you don't provide this option, you will be prompted to.
-s, --hostname, or --server	-s <server hostname>	This option can be used to specify the hostname. If no hostname is specified then the hostname defaults to **localhost**.
-p, or --port	-p <port>	This option can be used to specify a port to contact the host. If not specified, the default port is **1099**.
--secure	--secure	Can be used to communicate with JMX server via a secure channel.This option is only available in Geronimo 2.1.0.1 or later.

Note: For Geronimo 2.1.0.1 or above, you can work with the RMI/JMX **--secure** option in GShell commands. You may need to check out the topic Configuring secure JMX server before using this option.

Echo or print

The **echo** (**print**) command is used to print arguments to STDOUT.

Source

The **source** command takes an external file (or URL) and reads the content in line by line, executing each line. For example, if you have a file named example.gsh, with its content like this:

example.gsh

```
echo "Hello"
echo "Testing source now"
echo "Bye"
```

In GShell, use source command following this syntax:

```
source ./example.gsh
```

or

```
. ./example.gsh
```

You will get the following results:

```
Hello
Testing source now
Bye
```

Clear

The **clear** command can be used to clear the screen.
Note: This command is not available on the Windows platform.

Setting and unsetting variables

The **set** command can be used to set a variable, and follows this syntax:

```
set <variable1>=string
set <variable2>="A string separated by space"
```

variable1 is a variable containing no space or special character, so there is no need to add the quotation marks. Here is an example:

```
set username=system
set password=manager
deploy/connect -u $username -w $password

set newstring="two strings"
echo $newstring
```

The **unset** command is used to cancel your previous setting.

```
unset <variable>
```

Starting and stopping a server

The server can be started through GShell using the **geronimo/start-server** command. This command provides the following options:

```
geronimo/start-server -A <JAR> -D <name=value> -G <name=value> -H <dir> -J <flag> -P <name>
-b -j <dir> -l <file>  -m <module> -p <port> -q -t <time> -u <user> -v -w <password> --secure
```

These options are described in the following table besides the general options:

Option	Usage	Description
-A, or --javaagent	-A <JAR>	Identify the specific Java Agent with a JAR file containing its path. To disable it, set it to 'none'.
-D, or --property	-D <name=value>	Define system properties.
-G, or --gproperty	-G <name=value>	Define org.apache.geronimo properties. This option is probably used if you start two or more Geronimo instances on your server.
-H, or --home	-H <dir>	Provide a specific Geronimo home directory. This option is probably used if you start two or more Geronimo instances on your server.
-J, or --javaopt	-J <flag>	Set a Java Virtual Machine (JVM) flag.
-P, or --profile	-P <name>	Select a configuration profile.
-b, or --background	-b	If provided, the server process will run in the background.
-j, or --jvm	-j <dir>	Use a specific JVM for the server process.
-l, or --logfile	-l <file>	Capture the console output to a log file.
-m, or --module	-m <module>	Start up a specific module.
-q, or --quiet	-q	Suppress warning and informative message.

-t, or --timeout	-t <time>	Identify the timeout in seconds.
-v, or --verbose	-v	Enable verbose output, resulting in more console output than is normally present.

The server can be stopped using the **geronimo/stop-server** command. It uses the following syntax:

```
geronimo/stop-server -u <user> -w <password> -s <server hostname> -p <port> --secure
```

See general options for information about the options of this command.

Waiting for the server to start

The **geronimo/wait-for-server** command is used to verify if the server has started in the given time (in seconds). It has the following syntax:

```
geronimo/wait-for-server -u <user> -w <password> -s <server hostname> -p <port> -t <time> --secure
```

Option	Usage	Description
-t, or --timeout	-t <time>	Can be used to specify the time (in seconds) to wait while verifying the that the server has started. -1 means the command will wait infinitely

If the **-t** option is not provided, the default timeout is 60 seconds. See general options for information about the rest of options.

Starting an application client

Before starting a client, you have to deploy the application to the server. See creating deployment plans for information about deployment plan templates for application clients, and deploying modules for how to deploy your applications to the server.
The **geronimo/start-client** command has the following syntax:

```
geronimo/start-client <config-name> <args> -A <JAR> -D <name=value> -G <name=value> -H <dir> -J <flag> -P
<name>
-b -j <dir> -l <file> -t <time> -v --secure
```

where *config-name* is the configurations for your application client, and *args* are application specific arguments. The **geronimo/start-client** command can be issued with the following options:

Option	Usage	Description
-A, or --javaagent	-A <JAR>	Identify the specific Java Agent with a JAR file containing its path. To disable it, set it to 'none'.
-D, or --property	-D <name=value>	Define system properties.
-G, or --gproperty	-G <name=value>	Define org.apache.geronimo properties. This option is probably used if you start two or more Geronimo instances on your server.
-H, or --home	-H <dir>	Provide a specific Geronimo home directory. This option is probably used if you start two or more Geronimo instances on your server.
-J, or --javaopt	-J <flag>	Set a Java Virtual Machine (JVM) flag.
-P, or --profile	-P <name>	Select a configuration profile.
-b, or --background	-b	If provided, the server process will run in the background.
-j, or --jvm	-j <dir>	Use a specific JVM for the server process.
-l, or --logfile	-l <file>	Capture the console output to a log file.
-q, or --quiet	-q	Suppress warning and informative message.
-t, or --timeout	-t <time>	Identify the timeout in seconds.
-v, or --verbose	-v	Enable verbose output, resulting in more console output than is normally present.

Connecting to an already running server

GShell allows you to run a series of commands on a remote server. To do that you first need to connect to the remote server. The **deploy/connect** command can be used to connect to an instance of Geronimo that is already running.

```
deploy/connect -u <user> -w <password> -s <server hostname> -p <port> --secure
```

See general options for information about the options of this command.

The **deploy/disconnect** command can be used to disconnect from an already connected server. Since only one instance of the server can be connected at a time, no additional options are needed to specify which server to disconnect from. If you are trying to connect to a second server instance, use this command to disconnect first.

Saving the username and password for current connection

Gshell allows you to save your credential after connecting to a running server. Simply specify your username and password with **deploy/login**, and you will not be bothered with inputting your credential repeatedly. This command behaves in the same way as login command option of deploy.

```
deploy/login -u <user> -w <password> -s <server hostname> -p <port>
```

See general options for information about the options of this command.

Deploying modules

The **deploy/deploy** command can be used to deploy a module to a server that you have previously connected to as mentioned in connecting to an already running server. If no existing connection is available, the **deploy/deploy** command will first establish a connection and then execute the specific command. Once deployed, a module is identified by its module ID within Geronimo. The **deploy/deploy** command has the following syntax:

```
deploy/deploy <module> <deployment plan> -u <user> -w <password> -s <server hostname> -p <port> -t <target1;
target2> -i --secure
```

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 deployment plan for a WAR file is not in the WEB-INF directory, its location must be specified after the module in the command.

Option	Usage	Description
-i, or --inPlace	-i	Can be used to specify an in-place deployment from the directory you are actually developing the application.
-t, or --targets	-t <target1; target2>	Can be used to specify the repository targets to which the module should be deployed. You can list targets to get a list of targets available on the Geronimo server.

If the **-i** option is provided, the path to the application would need to be provided in place of the location of the module. See general options for information about the other options of this command.

Redeploying modules

The **deploy/redeploy** command is used to deploy a newer version of a module onto a server where the older module is already deployed. It functions in a similar way to **deploy/deploy** but lacks an **in-place** deployment option.
The **deploy/redeploy** command has the following syntax:

```
deploy/redeploy <module> <deployment plan> <module_id> -u <user> -w <password> -s <server hostname> -p <port> --
secure
```

If you do not specify the *module_id*, the plan supplied (or plan inside the module) will be used to determine the actual configuration that you wish to redeploy. Redeploying a plan with an existing module ID allows you to modify the configuration of a running module without intermediate undeployment.
See general options for information about the options of this command.

Undeploying modules

The **deploy/undeploy** command is used to properly remove a module from a server. Once undeployed, the module cannot be started again, unless you use the deploy command again. The module id must be provided for the module you wish to undeploy.

```
deploy/undeploy -u <user> -w <password> -s <server hostname> -p <port> <module_id> --secure
```

See general options for information about the options of this command.

Distributing modules

The **deploy/distribute** command works exactly like **deploy/deploy** except the module is not started once it has been deployed into the server and is not marked to be started each time the server starts. The command can be issued in the same way, with the same options, as **deploy/deploy**. The **-t** option can be used to specify the repository targets to which the module should be distributed. You can list targets to get a list of targets available on the Geronimo server. See general options for information about other options of this command.

```
deploy/distribute <module> <deployment plan>  -u <user> -w <password> -s <server hostname> -p <port> -i -t
<target1;target2> --secure
```

Starting modules

The **deploy/start** command starts a previously deployed module that is not running, and uses this syntax:

```
deploy/start -u <user> -w <password> -s <server hostname> -p <port> <module_id> --secure
```

See general options for information about the options of this command.

Stopping modules

The **deploy/stop** command stops a running module, and uses this syntax:

```
deploy/stop -u <user> -w <password> -s <server hostname> -p <port> <module_id> --secure
```

The command can be issued with the same options, as **deploy/start**.

Restarting modules

The **deploy/restart** command ommand restart a module that is already running, or a previously stopped module.

```
deploy/restart -u <user> -w <password> -s <server hostname> -p <port> <module_id> --secure
```

The command can be issued with the same options, as **deploy/start**.

Listing modules

The **deploy/list-modules** command lists available modules on an active server, and uses this syntax:

deploy/list-modules -u <user> -w <password> -s <server hostname> -p <port> *filterOptions* --secure

where

${renderedContent}
${renderedContent}

By default, all started or stopped modules are displayed. Any started modules are shown with a "+" sign next to them. All Webtier modules that have a externally accessible URL associated with them will also have this URL shown next to the module. Running modules are represented by their module IDs in Geronimo. See general options for information about other options of this command.

Listing targets

The **deploy/list-targets** command lists available targets on an active server, and uses this syntax:

```
deploy/list-targets -u <user> -w <password> -s <server hostname> -p <port> --secure
```

See general options for information about the options of this command.

Listing plug-ins

The **deploy/list-plugins** command lists available configurations on an active server, and uses this syntax:

```
deploy/list-plugins -u <user> -w <password> -s <server hostname> -p <port> -r <repository> -rr -rl --secure
```

The options are explainied in the following table:

Option	Usage	Description
-rr, or --refresh-repository	-rr	Refresh the repository.
-rl, or --refresh-list	-rl	Refresh the plug-in list.
-r, or --repository	-r <repository>	Can be used to provide the repository URL.

This command lists server plugins suitable for installation on your configured server, and will allow you to select them to be downloaded and installed. See general options for information about other options of this command.

Installing libraries

The **deploy/install-library** command can be used to install a library, and uses this syntax:

```
deploy/install-library <libraryFile> -g <groupId> -u <user> -w <password> -s <server hostname> -p <port> --
secure
```

where *libraryFile* specifies the library file, usually a JAR. If the file name is not in a Maven recognizable format, you have to rename it following this format:

```
<artifactId>-<version>.<type>
```

The options are explainied in the following table:

Option	Usage	Description
-g, or --groupId	-g <groupId>	Can be used to specify the group ID of the library.

See general options for information about other options of this command.

If successfully installed, the library will be found in `<geronimo_home>/repository`, where *<geronimo_home>* is the server's installation directory.

Installing a plug-in

The **deploy/install-plugin** command can be used to install a plug-in (must be a CAR file) on the active server, and uses this syntax:

```
deploy/install-plugin <plugin> -u <user> -w <password> -s <server hostname> -p <port> --secure
```

where *plugin* specifies the plug-in to be installed. See general options for information about the options of this command.

Assembling

The **deploy/assemble** command can be used to extract a customer Geronimo server from the current one.

```
deploy/assemble -a <artifact> -f <format> -g <groupId> -l -p <port> -s <server hostname> -t <path> -u <user> -w
<password>
-s <server hostname> -p <port> --secure
```

The options are explained in the following table:

Option	Usage	Description
-a, or --artifact	-a <artifact>	Can be used to provide the server artifact name.
-f, or --format	-f <format>	Can be used to specify if the assembly is in .zip or tar.gz format.
-g, or --groupId	-g <groupId>	Can be used to specify the group ID of the library.
-t, or --path	-t <path>	Can be used to provide the assembly location, where your specific plug-ins are stored. The default location is `<geronimo_home>/var/temp/`

-l, or --list	-l	Can be used to refresh the plug-in list.

See general options for information about other options of this command.

Your successfully assembled server will be found in `<geronimo_home>/var/temp`, where *<geronimo_home>* is the server's installation directory.

Creating a new instance

The **deploy/new-instance** command can be used to creating a new server instance from the current one, and uses this syntax:

```
deploy/new-instance <SERVER_NAME> -u <user> -w <password> -s <server hostname> -p <port> --secure
```

where *SERVER_NAME* is the name of the new instance. Look into Running Multiple Geronimo Instances for more information about how to initiate the new instance.

Connecting to a remote Gshell

This **remote/rsh** can be used to execute gshell commands on a remote Gshell server.

```
remote/rsh tcp://<REMOTEIP>:<PORT> <GShellCommand>
```

Where *REMOTEIP* is the IP address on which the remote GShell server is running, *PORT* is the listening port on the remote Gshell server and *GShellCommand* is the command which can be executed in any GShell environment.

Starting a remote GShell

The **remote/rsh-server** command can be used to start a remote GShell session for listening on a port to accept requests from foreign address.

```
remote/rsh-server tcp://<LOCALIP>:<PORT>
```

where *LOCALIP* is the IP address of the GShell server, *PORT* is the listening port that you can specify any unoccupied port number.

Monitoring cluster heartbeat

The **cluster/heartbeat** command can be used to monitor cluster heartbeat when you enabled plugin based Farming.

```
cluster/heartbeat -f <Regex>
```

where *Regex* is the regular expression to filter heartbeat data displayed. You can use **--filter** other than **-f** option.

Administering cluster

The **cluster/deploy** command can be used to deploy/undeploy a plugin or a list of plugins on a specified cluster. This command is only workable when you enabled Plugin based Farming and performed the command on a controller node.

```
cluster/deploy ACTION -c <clustername> -l <pluginlistname> -a <pluginartifactID>
```

The options are explained in the following table:

Option	Usage	Description
ACTION	add/remove	Action (add/remove) to perform
-c, or --cluster)	-c <clustername>	Can be used to specify name of the cluster to perform action on
-l, or --pluginlist	-l <pluginlist>	Can be used to specify name of the plugin List to perform action on
-a, or --pluginartifact	-a <pluginartifactID>	Can be used to specify name of the plugin to perform action on

See general options for information about other options of this command.

Generating WSDL file from class

The **jaxws/java2ws** command can be used to generate a WSDL file, wrapper bean, server side code and client side code from a Web service endpoint's implementation (SEI) class and associated types classes. It uses this syntax:

```
jaxws/java2ws -databinding <jaxb or aegis> -frontend <jaxws or simple> -wsdl -wrapperbean -client -server -ant -
wrapperbean -o
<output-file> -d <resource-directory> -s <source-directory> -classdir <compile-classes-directoty> -cp <class-
path> -soap12 -t
<target-namespace> -beans <pathname of the bean definition file> -servicename <service-name> -portname <port-
name> -address
<address> -createxsdimports -h -v -verbose -quiet <classname>
```

These options are described in the following table:

Option	Usage	Description
-help or -h	-help or -h	Can be used to obtain the help information.
--databinding	-databinding <jaxb or aegis>	Can be used to specify the data binding (jaxb or aegis). By default it is jaxb for jaxws frontend, and aegis for simple frontend.
-frontend	-frontend <jaxws or simple>	Can be used tp specify the frontend. Jaxws and the simple frontend are supported.
-wsdl	-wsdl	Can be used to generate the WSDL file.
-client	-client	Can be used to generate client side code.
-server	-server	Can be used to generate server side code.
-ant	-ant	Can be used to generate an Ant `build.xml` script.
-wrapperbean	-wrapperbean	Can be used to generate the wrapper and fault bean.
-o	-o <output-file>	Can be used to specify the name of the generated WSDL file.
-d	-d <resource-directory>	Can be used to specify the directory in which the resource files are located. The wsdl file will be placed into this directory by default.
-s	-s <source-directory>	Can be used to specify the directory in which the generated source files (wrapper bean ,fault bean ,client side or server side code) are located.
-classdir	-classdir <compile-classes-directoty>	Can be used to specify the directory in which the generated sources are compiled into. If not specified, the files are not compiled.
-cp	-cp <class-path>	Can be used to specify the SEI and types class search path of directories and zip/jar files.
-soap12	-soap12	Can be used to indicate that the generated WSDL is to include a SOAP 1.2 binding.
-t	-t <target-namespace>	Can be used to specify the target namespace in the generated WSDL file.
-beans	-beans <pathname of the bean definition file>	Can be used to specify the path and name of the generated bean definition file.
-servicename	-servicename <service-name>	Can be used to specify the value of the generated service element's name attribute.
-portname	-portname <port-name>	Can be used to specify the port name to use in the generated WSDL.
-address	-address <address>	Can be used to specify the port address.
-createxsdimports	-createxsdimports	Can be used to output schemas to separate files and load them by imports instead of inlining them into the WSDL.
-v	-v	Can be used to obtain the version number.
-verbose	-verbose	Can be used to display comments during the code generation process.
-quiet	-quiet	Can be used to suppress comments during the code generation process.
<classname>	<classname>	Can be used to specify the name of the SEI class.

Generating JAX-WS artifacts from class

The **jaxws/wsgen** command can be used to generate necessary portable artifacts for JAX-WS applications from Java classes. Unlike **jaxws/java2ws**, this command generates a WSDL file only when requested. It uses this syntax:

```
jaxws/wsgen -classpath <path> -cp <path> -d <directory> -extension -help -keep -r <directory> -verbose -version
-wsdl[:protocol]
 -servicename <name> -portname <name>
```

These options are described in the following table:

Option	Usage	Description
-classpath or -cp	-classpath <path> or -cp <path>	Can be used to specify the location of the service implementation class.
-d	-d <directory>	Can be used to specify the directory in which the generated output files will be placed.
-extension	-extension	Can be used to allow custom extensions for functionality not specified by the JAX-WS specification. Use of the extensions can result in applications that are not portable or do not interoperate with other implementations. Here is a list of the extentions available: • **-XadditionalHearders** • **-Xauthfile** • **-Xbebug** • **-Xno-address-databindling** • **-Xnocompile**
-help	-help	Can be used to obtain the help information.
-keep	-keep	Can be used to keep the generated source files.
-r	-r <directory>	Can be used to specify the directory in which generated WSDL file is placed. This parameter is only used in conjunction with the -wsdl parameter.
-verbose	-verbose	Can be used to output messages about what the compiler is doing.
-version	-version	Can be used to obtain the version number. If you specify this option, only the version information will be output and normal command processing will not occur.
-wsdl	-wsd [:protocol]	Can be used to direct wsgen to generate a WSDL file and is typically used by a developer to review a WSDL file before the endpoint is deployed. By default, wsgen does not generate a WSDL file. The protocol can be used to specify the protocol used in the wsdl:binding, and is optional. Valid values for protocol are soap 1.1 and Xsoap 1.2. The default value is soap 1.1. The Xsoap 1.2 value can only be used in conjunction with the `-extension` option.
-servicename	-servicename <name>	Can be used to specify a wsdl:service name to be generated in the WSDL file. This parameter is only used in conjunction with the -wsdl option.

Generating Java classes from WSDL

The **jaxws/wsdl2java** command can be used to create Java SEI classes from WSDL, and uses this syntax:

```
jaxws/wsdl2java -fe <front-end-name>* -db <data-binding-name>* -wv <wsdl-version> -p <[wsdl-namespace = ]
package-name>* -sn
<service-name> -b <bindling-file-name> -catalog <catalog-file-name> -d <output-directory> -compile -classdir
<compile-classes-directory> -impl -server -client
-all -autoNameResolution -defaultValues<=class-name-for-DefaultValueProvider> -ant -nexclude <schema-namespace
[=java=package-
name]>* -exsh <<true,false>> -dns <<true,false>> -dex <<true,false>> -validate -keep -wsdllocation
<wsdlLocation> -xjc
<xjc-arguments> -noAddressBinding -h -v -verbose -quiet <wsdlurl>
```

These options are described in the following table:

Option	Usage	Description
-help or -h	-help or -h	Can be used to obtain the help information.
-fe	-fe <frontend-name>	Can be used to specify the frontend. By default it is JAXWS frontend. Currently only JAXWS frontend is supported.
-db	-db <databinding-name>	Can be used to specify the data binding. By default it is jaxb. Currently only JAXB databinding is supported.
-wv	-wv <wsdl-version>	Can be used to specify the wsdl version. By default it is WSDL 1.1. Currently only WSDL 1.1 version is supported.
-p	-p <[wsdl-namespace =]package-name>*	Can be used to specify zero or more package names for the generated code.
-sn	-sn <service-name>	Can be used to specify the WSDL service name for the generated code.
-b	-b <bindling-file-name>	Can be used to specify zero or more JAXWS or JAXB binding files. You can use spaces to separate more than one entry.

-catalog	-catalog <catalog-file-name>	Can be used to specify catalog file that maps the imported wsdl/schema.
-d	-d <output-directory>	Can be used to specify the directory into which the generated code files are written.
-compile	-compile	Can be used to compile generated Java files.
-classdir	-classdir <compile-classes-directory>	Can be used to specify the directory into which the compiled class files are written.
-impl	-impl	Can be used to generate starting point code for an implementation object.
-client	-client	Can be used to generate starting point code for a client mainline.
-server	-server	Can be used to generate starting point code for a server mainline.
-all	-all	Can be used to generate all starting point code: types, service proxy, service interface, server mainline, client mainline, implementation object, and an Ant `build.xml` file.
-autoNameResolution	-autoNameResolution	Can be used to automatically resolve naming conflicts without binding customizations.
-defaultValues	-defaultValues=[DefaultValueProvider impl]	Can be used to generate default values for the impl and client. You can also provide a custom default value provider. The default provider is RandomValueProvider.
-ant	-ant	Can be used to generates the Ant `build.xml` file.
-nexclude	-nexclude <schema-namespace [=java=package-name]>*	Can be used to ignore the specified WSDL schema namespace when generating code. This option can be specified multiple times. Java package name used by types described in the excluded namespace(s) can also be specified. The java package name is optional.
-exsh	-exsh <<true,false>>	Can be used to enable or disable processing of implicit SOAP headers (SOAP headers defined in the wsdl:binding but not wsdl: portType section.) By default it is false.
-dns	-dns <<true,false>>	Can be used to enable or disable the loading of the default namespace package name mapping. Default is true and http://www.w3.org/2005/08/addressing=org.apache.cxf.ws.addressing namespace package mapping will be enabled.
-dex	-dex <<true,false>>	Can be used to enable or disable the loading of the default excludes namespace mapping. Default is true.
-validate	-validate	Can be used to enable validating the WSDL before generating the code.
-keep	-keep	Can be used to indicate that the code generator will not overwrite any preexisting files. You will be responsible for resolving any resulting compilation issues.
-wsdlLocation	-wsdllocation <wsdlLocation>	Can be used to specify the value of the @WebServiceClient annotation's wsdlLocation property.
-xjc	-xjc <xjc-arguments>	Can be used to specify a comma separated list of arguments that are passed directly to the XJC processor when using the JAXB databinding. A list of available XJC plugins can be obtained using `-xjc-X`.
-noAddressBinding	-noAddressBinding	Can be used to direct the code generator to generate the older CXF proprietary WS-Addressing types instead of the JAX-WS 2.1 compliant WS-Addressing types.
-v	-v	Can be used to obtain the version number.
-verbose	-verbose	Can be used to display comments during the code generation process.
-queit	-quiet	Can be used to suppress comments during the code generation process.
<wsdlurl>	<wsdlurl>	Can be used to specify the path and name of the WSDL file in generating the code.

Generating JAX-WS artifacts from WSDL

The **jaxws/wsimport** command can be used to generates the required portable artifacts for JAX-WS Web service applications from an existing WSDL file. It uses this syntax:

```
jaxws/wsimport -b <path> -B <jaxbOption> -catalog <file> -d <directory> -extension -help -httpproxy:<host>:
<port> -keep -p <pkg>
-quiet -s <directory> -target <version> -verbose -version -wsdllocation <location>
```

These options are described in the following table:

Option	Usage	Description
-b	<path>	Can be used to specify the external JAX-WS or JAXB binding files. You can specify multiple JAX-WS and JAXB binding files by using the -b option; however, each file must be specified with its own -b option.

-B	-B <jaxbOption>	Can be passed to JAXB schema complier.
-catalog	-catalog <file>	Can be used to specify the catalog file that resolves external entity references. It supports TR9401, XCatalog, and OASIS XML Catalog formats.
-d	<directory>	Can be used to specify the directory in which the generated output files are placed.
-extension	-extension	Can be used to allow custom extensions for functionality that are not specified by the JAX-WS specification. The use of custom extensions can result in applications that are not portable or do not interoperate with other implementations.
-help	-help	Can be used to obtain the help information.
-httpproxy	-httpproxy: <host>: <port>	Can be used to specify an HTTP proxy. The default port value is 8080.
-keep	-keep	Can be used to keep the generated source files.
-p	-p <pkg>	Can be used to specify a target package with this command-line option and overrides any WSDL file and schema binding customization for the package name and the default package name algorithm defined in the JAX-WS specification.
-s	-s <directory>	Can be used to specify the directory in which the generated source files are placed.
-target	-target <version>	Can be used to specify the version of JAXWS specification in generating the code.
-verbose	-verbose	Can be used to output messages about what the compiler is doing.
-version	-version	Can be used to obtain the version information. If you specify this option, only the version information is included in the output and normal command processing does not occur.
-wsdlLocation	-wsdllocation <location>	Can be used to specify the @WebServiceClient.wsdlLocation value.