app E.fm Page 561 Wednesday, July 10, 2002 8:08 AM

APPENDIX E

Ant task reference

Ant's distribution ships with extensive documentation in HTML format, including details for every task. Why then, do we include a task reference in this book? We felt readers would benefit from a quick reference they could consult while reading the book, or flip through while away from their computers, or as a quick reference to the valid attributes and elements of a task.

We didn't want to rewrite the documentation, as it would have taken a lot of effort for little real benefit. At the same time, we didn't want to include the current documentation as it is too detailed for a quick reference, yet not guaranteed to be accurate. Because Ant's current task documentation is decoupled from the source of the tasks, it is out of sync with the tasks in some places.

To address these needs, we created an automated process to analyze Ant's source code and extract task details directly. This process used XDoclet's custom capabilities, allowing generation of an XML file for each task. These XML files contain information such as task description and all supported attribute and element names, types, and descriptions.[1] We expect the future releases of Ant to adopt this process, improving the online documentation accordingly.

[1] The XDoclet work included a custom subtask, tag handler, and template. The process was simple from that point forward. The XML files were merged using `<concat>` and transformed into HTML using `<xslt>`. The code for this autogeneration exists within Ant's source code repository.

app E.fm Page 562 Wednesday, July 10, 2002 8:08 AM

Here then is a quick guide to all the tasks, automatically generated from the source itself. It lists all tasks in Ant 1.5 and their nested elements and attributes. We have omitted any examples, or any details, on the nested elements. For these we refer you to the online documentation, and, of course, the many chapters in our book.

E.1 REFERENCE CONVENTIONS

Tasks are listed in the following format:

<task-name> Brief description of task.

`attribute`	Attribute description. [Attribute type]
`<subelement>`	Subelement description. [Element type]

Attributes are listed first, alphabetically, followed by subelements, which are also listed alphabetically. Subelements have angle brackets (`< >`) around them.

All attributes have a type provided. Element types are only provided for common datatypes. Consult Ant's documentation for specific information on elements when a type is not noted. Boolean attributes are considered `true` when their values equal `on`, `yes`, or `true`. Any other value is considered `false`. Path attributes accept platform-independent paths, using either colon (`:`) or semi-colon (`;`) as separators and either forward (`/`) or back slashes (\) as directory separators.

Several tasks are based on a parent *MatchingTask*. MatchingTasks support a number of additional attributes and subelements, and are denoted by an asterisk (*).

E.2 COMMON TASK ATTRIBUTES

All tasks support three common attributes.

`id`	A unique task instance identifier, which can, for example, be used to refer to the task from `<script>`. [String]
`taskname`	An alias for the task; useful for logging purposes, as this name is provided instead. [String]
`description`	A field useful for commenting purposes, although it is not used or displayed by Ant. [String]

E.2.1 * MatchingTask

Tasks denoted with the asterisk (*) also support the follow attributes and subelements. These tasks operate on an implicit fileset, and typically have an attribute representing the base directory of the fileset.

`casesensitive`	Case sensitivity of the file system. [Boolean]
`defaultexcludes`	If false, turns off the default exclusions. [Boolean]
`excludes`	Comma- or space-separated list of patterns of files that must be excluded. [String]
`excludesfile`	The name of a file; each line of this file is taken to be an exclude pattern. [File]
`followsymlinks`	Indicates whether symbolic links should be followed. [Boolean]
`includes`	Comma- or space-separated list of patterns of files to include. [String]
`includesfile`	The name of a file; each line of this file is taken to be an include pattern. [File]
`<and>`	Selects files that are selected by all of the selectors it contains. [Selector]
`<contains>`	Limits the files selected to only those that contain a specific string. [Selector]
`<custom>`	Adds a custom selector. [Selector]
`<date>`	Selects files based on last modification timestamp. [Selector]
`<depend>`	Selects files whose last modified date is later than another file. [Selector]
`<depth>`	Selects files based on how many directory levels deep they are in relation to the base directory. [Selector]
`<exclude>`	Adds a single pattern to the excludes list.
`<excludesfile>`	Adds patterns contained in a file to the excludes list.
`<filename>`	Functions similarly to the `<include>` and `<exclude>` elements. [Selector]
`<include>`	Adds a single pattern to the includes list.
`<includesfile>`	Adds patterns contained in a file to the includes list.
`<majority>`	Selects files provided that a majority of the contained selectors also select it. [Selector]
`<none>`	Selects files that are not selected by any of the selectors it contains. [Selector]
`<not>`	Reverses the meaning of the single selector it contains. [Selector]
`<or>`	Selects files that are selected by any one of the elements it contains. [Selector]
`<patternset>`	Adds a patternset. [Patternset]
`<present>`	Selects files that have an equivalent file in another directory tree. [Selector]
`<selector>`	Adds selector through a reference. [Selector]
`<size>`	Limits files selected by size. [Selector]

app E.fm Page 564 Wednesday, July 10, 2002 8:08 AM

E.3 ANT'S TASKS

<ant> Builds a subproject.

`antfile`	The build file to use. [String]
`dir`	The directory to use as a base directory for the new Ant project. [File]
`inheritall`	If true, pass all properties to the new Ant project; default true. [Boolean]
`inheritrefs`	If true, pass all references to the new Ant project; default false. [Boolean]
`output`	File name to write the output to. [String]
`target`	The target of the new Ant project to execute. [String]
`<property>`	Property to pass to the new project. [See `<property>`]
`<reference>`	Reference element identifying a data type to carry over to the new project.

<antcall> Calls another target in the same project.

`inheritall`	If true, pass all properties to the new Ant project; default true. [Boolean]
`inheritrefs`	If true, pass all references to the new Ant project; default false. [Boolean]
`target`	Target to execute, required. [String]
`<param>`	Property to pass to the invoked target. [See <property>]
`<reference>`	Reference element identifying a data type to carry over to the invoked target.

<antlr> Invokes the ANTLR Translator generator on a grammar file.

`debug`	Enables ParseView debugging. [Boolean]
`diagnostic`	Flag to emit diagnostic text. [Boolean]
`dir`	The working directory of the process. [File]
`glib`	Sets an optional super grammar file. [String]
`html`	If true, emits HTML. [Boolean]
`outputdirectory`	The directory to write the generated files to. [File]
`target`	The grammar file to process. [File]
`trace`	If true, enables all tracing. [Boolean]
`tracelexer`	If true, enables lexer tracing. [Boolean]
`traceparser`	If true, enables parser tracing. [Boolean]
`tracetreewalker`	Flag to allow the user to enable tree-walker tracing. [Boolean]
`<classpath>`	Adds a classpath to be set because a directory might be given for ANTLR debug. [Path]
`<jvmarg>`	Adds a new JVM argument.

<*antstructure*> Creates a partial DTD for Ant from the currently known tasks.

`output`	The output file. [File]

<*apply*> Executes a given command, supplying a set of files as arguments.

`append`	Sets whether output should be appended or an existing file overwritten. [Boolean]
`dest`	The directory where target files are to be placed. [File]
`dir`	The working directory of the process. [File]
`executable`	The command to execute. [String]
`failifexecutionfails`	Stop the build if program cannot be started. [Boolean]
`failonerror`	Fail if the command exits with a non-zero return code. [Boolean]
`newenvironment`	Do not propagate old environment when new environment variables are specified. [Boolean]
`os`	List of operating systems on which the command may be executed. [String]
`output`	File the output of the process is redirected to. [File]
`outputproperty`	Property name whose value should be set to the output of the process. [String]
`parallel`	If true, run the command only once, appending all files as arguments. [Boolean]
`relative`	Sets whether the file names should be passed on the command line as absolute or relative pathnames. [Boolean]
`resultproperty`	The name of a property in which the return code of the command should be stored. [String]
`skipemptyfilesets`	If no source files have been found or are newer than their corresponding target files, do not run the command. [Boolean]
`type`	Type of file to operate on. [`file`, `dir`, `both`]
`vmlauncher`	If true, launch new process with VM, otherwise use the OS's shell. [Boolean]
`<arg>`	Adds a command-line argument.
`<env>`	Adds an environment variable to the launched process.
`<fileset>`	Source files to operate upon. [Fileset]
`<mapper>`	Adds mapping of source files to target files. [Mapper]
`<srcfile>`	Marker that indicates where the name of the source file should be put on the command line.
`<targetfile>`	Marker that indicates where the name of the target file should be put on the command line.

<*available*> Sets the given property if the requested resource is available at run time.

`classname`	Classname of a class which must be available to set the given property. [String]
`classpath`	Classpath to be used when searching for classes and resources. [Path]
`classpathref`	Classpath by reference. [Reference]

`file`	File which must be present in the file system to set the given property. [File]
`filepath`	Path to use when looking for a file. [Path]
`ignoresystemclasses`	Sets whether the search for classes should ignore the run-time classes and just use the given classpath. [Boolean]
`property`	Name of the property that will be set if the particular resource is available. [String]
`resource`	Name of a Java resource which is required to set the property. [String]
`type`	Sets what type of file is required. [`file`, `dir`]
`value`	Value given to the property if the desired resource is available. [String]
`<classpath>`	Classpath to be used when searching for classes and resources. [Path]
`<filepath>`	Path to search for file resources. [Path]

<*basename*>

Sets a property to the base name of a specified file, optionally minus a suffix.

`file`	File or directory to get base name from. [File]
`property`	Property to set base name to. [String]
`suffix`	Optional suffix to remove from base name. [String]

<*blgenclient*>

Generates a Borland Application Server 4.5 client JAR using as input the EJB JAR file.

`classpath`	Path to use for classpath. [Path]
`classpathref`	Reference to existing path, to use as a classpath. [Reference]
`clientjar`	Client JAR file name. [File]
`debug`	If true, turn on the debug mode for each of the Borland tools launched. [Boolean]
`ejbjar`	EJB JAR file. [File]
`mode`	Command launching mode: java or fork. [String]
`version`	No description. [Integer]
`<classpath>`	Adds path to the classpath. [Path]

<*buildnumber*>

Reads, increments, and writes a build number in a file.

`file`	The file in which the build number is stored. [File]

<*bunzip2*>

Expands a file that has been compressed with the BZIP2 algorithm.

`dest`	The destination file or directory; optional. [File]
`src`	The file to expand; required. [File]

<*bzip2*> Compresses a file with the BZIP2 algorithm.

`src`	The file to compress; required. [File]
`zipfile`	The required destination file. [File]

<*cab*> * Creates a CAB archive.

`basedir`	Base directory to look in for files to CAB. [File]
`cabfile`	The name/location of where to create the .cab file. [File]
`compress`	If true, compress the files; otherwise only store them. [Boolean]
`options`	Sets additional `cabarc` options that are not supported directly. [String]
`verbose`	If true, display `cabarc` output. [Boolean]
`<fileset>`	Adds a set of files to archive. [Fileset]

<*cccheckin*> Checks in files with ClearCase.

`cleartooldir`	Directory where the `cleartool` executable is located. [String]
`comment`	Comment string. [String]
`commentfile`	Specifies a file containing a comment. [String]
`identical`	If true, allows the file to be checked in even if it is identical to the original. [Boolean]
`keepcopy`	If true, keeps a copy of the file with a .keep extension. [Boolean]
`nowarn`	If true, suppresses warning messages. [Boolean]
`preservetime`	If true, preserves the modification time. [Boolean]
`viewpath`	Path to the item in a ClearCase view to operate on. [String]

<*cccheckout*> Checks out files in ClearCase.

`branch`	Specifies a branch to check out the file to. [String]
`cleartooldir`	Directory where the `cleartool` executable is located. [String]
`comment`	Comment string. [String]
`commentfile`	Specifies a file containing a comment. [String]
`nodata`	If true, checks out the file but does not create an editable file containing its data. [Boolean]
`nowarn`	If true, suppresses warning messages. [Boolean]
`out`	Creates a writable file under a different file name. [String]
`reserved`	If true, checks out the file as reserved. [Boolean]
`version`	If true, allows `checkout` of a version other than the latest. [Boolean]
`viewpath`	Path to the item in a ClearCase view to operate on. [String]

<*ccmcheckin*> Performs a Continuus checkin command.

`ccmdir`	Directory where the `ccm` executable is located. [String]
`comment`	Specifies a comment. [String]
`file`	Path to the file that the command will operate on. [File]
`task`	Specifies the task number used to check in the file (may use default). [String]

<*ccmcheckintask*> Performs a Continuus checkin default task command.

`ccmdir`	Directory where the `ccm` executable is located. [String]
`comment`	Specifies a comment. [String]
`file`	Path to the file that the command will operate on. [File]
`task`	Specifies the task number used to check in the file (may use default). [String]

<*ccmcheckout*> Performs a Continuus checkout command.

`ccmdir`	Directory where the `ccm` executable is located. [String]
`comment`	Specifies a comment. [String]
`file`	Path to the file that the command will operate on. [File]
`task`	Specifies the task number used to check out the file (may use default). [String]

<*ccmcreatetask*> Creates new Continuus <ccm> task and sets it as the default.

`ccmdir`	Directory where the `ccm` executable is located. [String]
`comment`	Specifies a comment. [String]
`platform`	Specifies the target platform. [String]
`release`	Specifies the `ccm` release. [String]
`resolver`	Specifies the resolver. [String]
`subsystem`	Specifies the subsystem. [String]
`task`	Specifies the task number used (may use default). [String]

<*ccmreconfigure*> Reconfigures a Continuus project, optionally recursively.

`ccmdir`	Directory where the `ccm` executable is located. [String]
`ccmproject`	`ccm` project on which the operation is applied. [String]
`recurse`	If true, recurse on subproject (default false). [Boolean]
`verbose`	If true, do a verbose reconfigure operation (default false). [Boolean]

<*ccuncheckout*> Performs a ClearCase Uncheckout command.

`cleartooldir`	Directory where the `cleartool` executable is located. [String]
`keepcopy`	If true, keep a copy of the file with a .keep extension. [Boolean]
`viewpath`	Path to the item in a ClearCase view to operate on. [String]

<ccupdate> Performs a ClearCase Update command.

`cleartooldir`	Directory where the `cleartool` executable is located. [String]
`currenttime`	If true, modification time should be written as the current time. [Boolean]
`graphical`	If true, displays a graphical dialog during the update. [Boolean]
`log`	Log file where `cleartool` records the status of the command. [String]
`overwrite`	If true, overwrites hijacked files. [Boolean]
`preservetime`	If true, modification time should be preserved from the VOB time. [Boolean]
`rename`	If true, hijacked files are renamed with a .keep extension. [Boolean]
`viewpath`	Path to the item in a ClearCase view to operate on. [String]

<checksum> * Creates or verifies file checksums.

`algorithm`	Specifies the algorithm to be used to compute the checksum. [String]
`file`	File for which the checksum is to be calculated. [File]
`fileext`	File extension that is be to used to create or identify destination file. [String]
`forceoverwrite`	Indicates whether to overwrite existing file, irrespective of whether it is newer than the source file. [Boolean]
`property`	Property to hold the generated checksum. [String]
`provider`	MessageDigest algorithm provider to be used to calculate the checksum. [String]
`readbuffersize`	The size of the read buffer to use. [Integer]
`verifyproperty`	Verify property. [String]
`<fileset>`	Files to generate checksums for. [Fileset]

<chmod> Chmod equivalent for Unix-like environments. Some of the attributes are an artifact of the task's implementation as a subclass of `<exec>`.

`append`	Indicates whether output should be appended to or overwrite an existing file. [Boolean]
`defaultexcludes`	Sets whether default exclusions should be used. [Boolean]
`dest`	The directory where target files are to be placed. [File]
`dir`	The directory that holds the files whose permissions must be changed. [File]
`excludes`	Set of exclude patterns. [String]
`executable`	The command to execute. [String]
`failifexecutionfails`	Stops the build if program cannot be started. [Boolean]
`failonerror`	Fail if the command exits with a nonzero return code. [Boolean]
`file`	The file or single directory for which the permissions must be changed. [File]
`includes`	Set of include patterns. [String]

`newenvironment`	Do not propagate old environment when new environment variables are specified. [Boolean]
`os`	List of operating systems on which the command may be executed. [String]
`output`	File the output of the process is redirected to. [File]
`outputproperty`	Property name whose value should be set to the output of the process. [String]
`parallel`	If true, runs the command only once, appending all files as arguments. [Boolean]
`perm`	The new permissions. [String]
`relative`	Indicates whether the file names should be passed on the command line as absolute or relative pathnames. [Boolean]
`resultproperty`	The name of a property in which the return code of the command should be stored. [String]
`skipemptyfilesets`	If no source files have been found or are newer than their corresponding target files, do not run the command. [Boolean]
`type`	Type of file to operate on [`file`, `dir`, `both`].
`vmlauncher`	If true, launches new process with VM, otherwise uses the OS's shell. [Boolean]
`<arg>`	Adds a command-line argument.
`<env>`	Adds an environment variable to the launched process.
`<exclude>`	Adds a name entry on the exclude list.
`<fileset>`	Source files to operate upon. [Fileset]
`<include>`	Adds a name entry on the include list.
`<mapper>`	Adds mapping of source files to target files. [Mapper]
`<patternset>`	Adds a set of patterns. [Patternset]
`<srcfile>`	Indicates where the name of the source file should be put on the command line.
`<targetfile>`	Indicates where the name of the target file should be put on the command line.

<concat>

Concatenates a series of files into a single file. This task supports nested text, which is appended to the end if specified.

`append`	Behavior when the destination file exists. [Boolean]
`destfile`	Destination file, or uses the console if not specified. [File]
`encoding`	Encoding for the input files, used when displaying the data via the console. [String]
`<filelist>`	List of files to concatenate. [Filelist]
`<fileset>`	Set of files to concatenate. [Fileset]

<condition>

Task to conditionally set a property.

`property`	The name of the property to set. [String]
`value`	The value for the property to set, if condition evaluates to true. [String]

`<and>`	True if all nested conditions evaluate to true.
`<available>`	Identical to the `<available>` task.
`<checksum>`	Identical to the `<checksum>` task.
`<contains>`	Tests whether one string contains another.
`<equals>`	Tests whether two strings are equal.
`<filesmatch>`	Tests that two files match, byte for byte.
`<http>`	Checks for a valid response from a web server of a specified URL.
`<isfalse>`	Tests whether a string value is not `<istrue>`.
`<isset>`	Tests whether a property has been set.
`<istrue>`	Tests whether a string evaluates to `true`, `on`, or `yes`.
`<not>`	Negates results of single nested condition.
`<or>`	True if one nested condition is `true`.
`<os>`	Tests whether the current operating system is of a given type.
`<socket>`	Checks for the existence of a TCP/IP listener at the specified host and port.
`<uptodate>`	Identical to the `<uptodate>` task.

<copy> Copies a file or directory to a new file or directory.

`encoding`	Character encoding. [String]
`failonerror`	If false, notes errors to the output but keeps going. [Boolean]
`file`	Single source file to copy. [File]
`filtering`	If true, enables filtering. [Boolean]
`flatten`	When copying directory trees, the files can be flattened into a single directory. [Boolean]
`includeemptydirs`	Used to copy empty directories. [Boolean]
`overwrite`	Overwrites any existing destination file(s). [Boolean]
`preservelastmodified`	Gives the copied files the same last modified time as the original files. [Boolean]
`todir`	Destination directory. [File]
`tofile`	Destination file. [File]
`verbose`	Used to force listing of all names of copied files. [Boolean]
`<fileset>`	Adds a set of files to copy. [Fileset]
`<filterchain>`	Adds a FilterChain. [FilterChain]
`<filterset>`	Adds a filterset. [Filterset]
`<mapper>`	Defines the mapper to map source to destination files. [Mapper]

<csc> * Compiles C# source into executables or modules.

`additionalmodules`	Semicolon-separated list of modules to refer to. [String]
`debug`	Debug flag on or off. [Boolean]
`definitions`	Semicolon-separated list of defined constants. [String]
`destdir`	Destination directory of files to be compiled. [File]

`destfile`	Name of exe/library to create. [File]
`docfile`	File for generated XML documentation. [File]
`extraoptions`	Any extra options that are not explicitly supported by this task. [String]
`failonerror`	If true, fails on compilation errors. [Boolean]
`filealign`	File alignment. [Integer]
`fullpaths`	If true, prints the full path of files on errors. [Boolean]
`includedefault-references`	If true, automatically includes the common .NET assemblies, and tells the compiler to link in mscore.dll. [Boolean]
`incremental`	Incremental compilation flag on or off. [Boolean]
`mainclass`	Name of main class for executables. [String]
`noconfig`	Do not read in the compiler settings files csc.rsp. [Boolean]
`optimize`	If true, enables optimization flag. [Boolean]
`outputfile`	Output file. [File]
`referencefiles`	Path of references to include. [Path]
`references`	Semicolon-separated list of DLLs to refer to. [String]
`srcdir`	Source directory of the files to be compiled. [File]
`targettype`	Type of target. [String]
`unsafe`	If true, enables the `unsafe` keyword. [Boolean]
`utf8output`	If true, requires all compiler output to be in UTF8 format. [Boolean]
`warnlevel`	Level of warning currently between 1 and 4 with 4 being the strictest. [Integer]
`win32icon`	File name of icon to include. [File]
`win32res`	File name of a Win32 resource (.RES) file to include. [File]

<*cvs*> Performs operations on a CVS repository.

`append`	Indicates whether to append output when redirecting to a file. [Boolean]
`command`	The CVS command to execute. [String]
`compression`	If true, this is the same as `compressionlevel="3"`. [Boolean]
`compressionlevel`	If set to a value 1–9 it adds `-zN` to the `cvs` command line, else it disables compression. [Integer]
`cvsroot`	The `CVSROOT` variable. [String]
`cvsrsh`	The `CVS_RSH` variable. [String]
`date`	Use the most recent revision, no later than the given date. [String]
`dest`	The directory where the checked-out files should be placed. [File]
`error`	The file to direct standard error from the command. [File]
`failonerror`	Stop the build process if the command exits with a return code other than 0. [Boolean]
`noexec`	If true, report only and do not change any files. [Boolean]
`output`	The file to direct standard output from the command. [File]
`package`	The package/module to operate upon. [String]

`passfile`	Password file to read passwords from. [File]
`port`	Port used by CVS to communicate with the server. [Integer]
`quiet`	If true, suppress informational messages. [Boolean]
`tag`	The tag of the package/module to operate upon. [String]
`<commandline>`	Adds direct command line to execute.

<cvschangelog> Examines the output of CVS log data and groups related changes together.

`daysinpast`	Number of days worth of log entries to process. [Integer]
`destfile`	Output file for the log. [File]
`dir`	Base directory for CVS. [File]
`end`	Date at which the changelog should stop. [Date]
`start`	Date at which the changelog should start. [Date]
`usersfile`	Lookup list of user names & addresses. [File]
`<fileset>`	Adds a set of files about which cvs logs will be generated. [Fileset]
`<user>`	Adds a user to list changelog knows about.

<cvspass> Adds a new entry to a CVS password file.

`cvsroot`	The CVS repository to add an entry for. [String]
`passfile`	Password file to add the entry to. [File]
`password`	Password to be added to the password file. [String]

<cvstagdiff> Examines the output of cvs diff between two tags.

`compression`	If true, this is the same as `compressionlevel="3"`. [Boolean]
`compressionlevel`	If set to a value 1–9, it adds `-zN` to the `cvs` command line, else it disables compression. [Integer]
`cvsroot`	The `CVSROOT` variable. [String]
`cvsrsh`	The `CVS_RSH` variable. [String]
`destfile`	Output file for the diff. [File]
`enddate`	End date. [String]
`endtag`	End tag. [String]
`failonerror`	Stop the build process if the command exits with a return code other than 0. [Boolean]
`package`	The package/module to analyze. [String]
`passfile`	Password file to read passwords from. [File]
`port`	Port used by CVS to communicate with the server. [Integer]
`quiet`	If true, suppress informational messages. [Boolean]
`startdate`	Start date. [String]
`starttag`	Start tag. [String]

<*ddcreator*> *

Builds a serialized deployment descriptor given a text file description of the descriptor in the format supported by WebLogic.

`classpath`	Classpath to be used for this compilation. [String]
`descriptors`	Directory from where the text descriptions of the deployment descriptors are to be read. [String]
`dest`	Directory into which the serialized deployment descriptors are written. [String]

<*delete*> *

Deletes a file or directory, or set of files defined by a fileset.

`defaultexcludes`	Sets whether default exclusions should be used. [Boolean]
`dir`	Directory from which files are to be deleted. [File]
`excludes`	Set of exclude patterns. [String]
`excludesfile`	Name of the file containing the excludes patterns. [File]
`failonerror`	If false, notes errors but continues. [Boolean]
`file`	Name of a single file to be removed. [File]
`includeemptydirs`	If true, deletes empty directories. [Boolean]
`includes`	Set of include patterns. [String]
`includesfile`	Name of the file containing the include patterns. [File]
`quiet`	If true, and the file does not exist, does not display a diagnostic message or modify the exit status to reflect an error. [Boolean]
`verbose`	If true, lists all names of deleted files. [Boolean]
`<exclude>`	Adds a name entry on the exclude list.
`<excludesfile>`	Adds a name entry on the exclude files list.
`<fileset>`	Adds a set of files to be deleted. [Fileset]
`<include>`	Adds a name entry on the include list.
`<includesfile>`	Adds a name entry on the include files list.
`<patternset>`	Adds a set of patterns. [Patternset]

<*depend*> *

Generates a dependency file for a given set of classes.

`cache`	Dependency cache file. [File]
`classpath`	Classpath to be used for this dependency check. [Path]
`classpathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`closure`	If true, transitive dependencies are followed until the closure of the dependency set if reached. [Boolean]
`destdir`	Destination directory where the compiled Java files exist. [Path]
`dump`	If true, the dependency information will be written to the debug level log. [Boolean]
`srcdir`	Directories path to find the Java source files. [Path]
`<classpath>`	Adds a classpath to be used for this dependency check. [Path]

<dependset> * Examines and removes out-of-date target files.

`<srcfilelist>`	Adds a list of source files. [Filelist]
`<srcfileset>`	Adds a set of source files. [Fileset]
`<targetfilelist>`	Adds a list of target files. [Filelist]
`<targetfileset>`	Adds a set of target files. [Fileset]

<dirname> Determines the directory name of the specified file.

`file`	Path to take the dirname of. [File]
`property`	The name of the property to set. [String]

<ear> * Creates an EAR archive.

`appxml`	File to incorporate as application.xml. [File]
`basedir`	Directory from which to archive files; optional. [File]
`compress`	Indicates whether to compress the files or only store them; optional, default=true;. [Boolean]
`destfile`	The file to create; required. [File]
`duplicate`	Sets behavior for when a duplicate file is about to be added. [`add`, `preserve`, `fail`]
`encoding`	Encoding to use for file names, defaults to the platform's default encoding. [String]
`filesonly`	If true, emulates Sun's JAR utility by not adding parent directories; optional, defaults to false. [Boolean]
`index`	Indicates whether to create an index list for classes. [Boolean]
`manifest`	The manifest file to use. [File]
`update`	If true, updates an existing file, otherwise overwrites any existing one; optional, defaults to false. [Boolean]
`whenempty`	Behavior of the task when no files match. [`fail`, `skip`, `create`]
`<archives>`	Adds `zipfileset`. [ZipFileset]
`<fileset>`	Adds a set of files. [Fileset]
`<manifest>`	Allows the manifest for the archive file to be provided inline in the build file rather than in an external file.
`<metainf>`	Adds a `zipfileset` to include in the META-INF directory. [ZipFileset]
`<zipfileset>`	Adds a set of files that can be read from an archive and given a prefix/fullpath. [ZipFileset]
`<zipgroupfileset>`	Adds a group of Zip files. [Fileset]

<echo> Writes a message to the Ant logging facilities. A message may be supplied as nested text to this task.

`append`	If true, append to existing file. [Boolean]
`file`	File to write to. [File]
`level`	Logging level. [`error`, `warning`, `info`, `verbose`, `debug`]
`message`	Message to write. [String]

<echoproperties> Displays all the current properties in the build.

`destfile`	File to store the property output. [File]
`failonerror`	If true, the task will fail if an error occurs while writing the properties file, otherwise errors are just logged. [Boolean]
`prefix`	If the prefix is set, then only properties that start with this prefix string will be recorded. [String]

<ejbc> * Builds EJB support classes using WebLogic's ejbc tool from a directory containing a set of deployment descriptors.

`classpath`	Classpath to be used for this compilation. [String]
`descriptors`	Directory from where the serialized deployment descriptors are to be read. [String]
`dest`	Directory into which the support classes, RMI stubs, etc. are to be written. [String]
`keepgenerated`	If true, ejbc will keep the intermediate Java files used to build the class files. [String]
`manifest`	Name of the generated manifest file. [String]
`src`	Directory containing the source code for the home interface, remote interface, and public key class definitions. [String]

<ejbjar> * Provides automated EJB JAR file creation.

`basejarname`	Base name of the EJB JAR that is to be created if it is not to be determined from the name of the deployment descriptor files. [String]
`basenameterminator`	The string that terminates the base name. [String]
`classpath`	Classpath to use when resolving classes for inclusion in the JAR. [Path]
`dependency`	Analyzer to use when adding in dependencies to the JAR. [String]
`descriptordir`	Descriptor directory. [File]
`destdir`	Destination directory. [File]
`flatdestdir`	Controls whether the destination JARs are written out in the destination directory with the same hierarchical structure from which the deployment descriptors have been read. [Boolean]
`genericjarsuffix`	Suffix for the generated JAR file. [String]
`manifest`	Manifest file to use in the JAR. [File]
`naming`	Naming scheme used to determine the name of the generated JARs from the deployment descriptor. [`ejb-name`, `directory`, `descriptor`, `basejarname`]
`srcdir`	Source directory, which is the directory that contains the classes that will be added to the EJB JAR. [File]
`<borland>`	Adds a deployment tool for Borland server.
`<classpath>`	Adds to the classpath used to locate the super classes and interfaces of the classes that will make up the EJB JAR. [Path]
`<dtd>`	Creates a DTD location record.
`<iplanet>`	Adds a deployment tool for iPlanet Application Server.

`<jboss>`	Adds a deployment tool for JBoss server.
`<jonas>`	Adds a deployment tool for JOnAS server.
`<support>`	Adds a fileset for support elements. [Fileset]
`<weblogic>`	Adds a deployment tool for WebLogic server.
`<weblogictoplink>`	Adds a deployment tool for WebLogic when using the Toplink Object-Relational mapping.
`<websphere>`	Adds a deployment tool for Websphere 4.0 server.

<exec> Executes a given command if the OS platform is appropriate.

`append`	Sets whether output should be appended to or overwrite an existing file. [Boolean]
`dir`	The working directory of the process. [File]
`executable`	The command to execute. [String]
`failifexecution-fails`	Stop the build if program cannot be found or started; default true. [Boolean]
`failonerror`	Fail if the command exits with a nonzero return code. [Boolean]
`newenvironment`	Do not propagate old environment when new environment variables are specified. [Boolean]
`os`	List of operating systems on which the command may be executed. [String]
`output`	File the output of the process is redirected to. [File]
`outputproperty`	Property name whose value should be set to the output of the process. [String]
`resultproperty`	The name of a property in which the return code of the command should be stored. [String]
`vmlauncher`	If true, launch new process with VM, otherwise use the OS's shell. [Boolean]
`<arg>`	Adds a command-line argument.
`<env>`	Adds an environment variable to the launched process.

<fail> Exits the active build, giving an additional message if available. The message may be specified as nested text, or with the `message` attribute.

`if`	Only fail if a property of the given name exists in the current project. [String]
`message`	A message giving further information on why the build exited. [String]
`unless`	Only fail if a property of the given name does not exist in the current project. [String]

<filter> Sets a token filter that is used by the file copy tasks to do token substitution.

`filtersfile`	The file from which the filters must be read. [File]
`token`	The token string without @ delimiters. [String]
`value`	The string that should replace the token during filtered copies. [String]

<fixcrlf> * Converts text source files to local OS formatting conventions, as well as repair text files damaged by misconfigured or misguided editors or file transfer programs.

`destdir`	Destination where the fixed files should be placed. [File]
`encoding`	Specifies the encoding Ant expects the files to be in. Defaults to the platform's default encoding. [String]
`eof`	Specifies how DOS EOF (control-z) characters are to be handled. [`add`, `asis`, `remove`]
`eol`	Specifies how EndOfLine characters are to be handled. [`asis`, `cr`, `lf`, `crlf`]
`javafiles`	Sets to true if modifying Java source files. [Boolean]
`srcdir`	Source dir to find the source text files. [File]
`tab`	Specifies how tab characters are to be handled. [`add`, `asis`, `remove`]
`tablength`	Specifies tab length in characters. [Integer]

<ftp> Uploads or downloads files using FTP.

`action`	FTP action to be taken. [`send`, `put`, `recv`, `get`, `del`, `delete`, `list`, `mkdir`, `chmod`]
`binary`	If true, uses binary mode, otherwise text mode; default is true. [Boolean]
`chmod`	File permission mode (Unix only) for files sent to the server. [String]
`depends`	Sets to true to transmit only files that are new or changed from their remote counterparts. [Boolean]
`ignorenon-criticalerrors`	If true, skip errors on directory creation. [Boolean]
`listing`	The output file for the `list` action. [File]
`newer`	A synonym for `depends`. [Boolean]
`passive`	Specifies whether to use passive mode. [Boolean]
`password`	Login password for the given user ID. [String]
`port`	FTP port used by the remote server. [Integer]
`remotedir`	Remote directory where files will be placed. [String]
`separator`	Remote file separator character. [String]
`server`	FTP server to send files to. [String]
`skipfailed-transfers`	If true, enables unsuccessful file put, deletes, and gets operations to be skipped with a warning and transfers the remainder of the files. [Boolean]
`umask`	Default mask for file creation on a Unix server. [String]
`userid`	Login user ID to use on the specified server. [String]
`verbose`	Set to true to receive notification about each file as it is transferred. [Boolean]
`<fileset>`	A set of files to upload or download. [Fileset]

<genkey> Generates a key in a keystore.

`alias`	The alias to add under. [String]
`dname`	The distinguished name for entity. [String]
`keyalg`	The method to use when generating name-value pair. [String]
`keypass`	Password for private key (if different than `storepass`). [String]
`keysize`	Indicates the size of key generated. [String]
`keystore`	Keystore location. [String]
`sigalg`	The algorithm to use in signing. [String]
`storepass`	Password for Keystore integrity. [String]
`storetype`	Keystore type. [String]
`validity`	Indicates how many days certificate is valid. [String]
`verbose`	If true, enables verbose output when signing. [Boolean]
`<dname>`	Distinguished name list.

<get> Gets a particular file from a URL source, usually a web server.

`dest`	Where to copy the source file. [File]
`ignoreerrors`	If true, log errors but do not treat as fatal. [Boolean]
`password`	Password for basic authentication. [String]
`src`	URL to get. [URL]
`username`	Username for basic authentication. [String]
`usetimestamp`	If true, conditionally download a file based on the timestamp of the local copy. [Boolean]
`verbose`	If true, show verbose progress information. [Boolean]

<gunzip> Expands a file that has been compressed with the GZIP algorithm.

`dest`	The destination file or directory; optional. [File]
`src`	The file to expand; required. [File]

<gzip> Compresses a file with the GZIP algorithm.

`src`	The file to compress; required. [File]
`zipfile`	The required destination file. [File]

<icontract> * Instruments Java classes with iContract DBC preprocessor.

`builddir`	Build directory for instrumented classes. [File]
`classdir`	Class directory (uninstrumented classes). [File]
`classpath`	Classpath to be used for invocation of iContract. [Path]
`classpathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`controlfile`	Control file to pass to iContract. [File]
`failthrowable`	Throwable (Exception) to be thrown on assertion violation. [String]
`instrumentdir`	Instrumentation directory. [File]
`invariant`	Turns on/off invariant instrumentation. [Boolean]

`post`	Turns on/off postcondition instrumentation. [Boolean]
`pre`	Turns on/off precondition instrumentation. [Boolean]
`quiet`	Tells iContract to be quiet. [Boolean]
`repbuilddir`	Build directory for instrumented classes. [File]
`repositorydir`	Build directory for repository classes. [File]
`srcdir`	Source directory. [File]
`targets`	Name of the file where targets will be written. [File]
`updateicontrol`	If true, updates iControl properties file. [Boolean]
`verbosity`	Verbosity level of iContract. [String]
`<classpath>`	Classpath. [Path]

<ilasm> *

Assembles .NET Intermediate Language files.

`debug`	Debug flag on or off. [Boolean]
`extraoptions`	Any extra options that are not explicitly supported by this task. [String]
`failonerror`	If true, fails if ilasm tool fails. [Boolean]
`keyfile`	The name of a file containing a private key. [File]
`listing`	If true, produces a listing; default is false. [Boolean]
`outputfile`	Output file. [File]
`resourcefile`	Name of resource file to include. [File]
`srcdir`	Source directory containing the files to be compiled. [File]
`targettype`	Type of target, either `exe` or `library`. [String]
`verbose`	If true, enables verbose ilasm output. [Boolean]

<input>

Reads an input line from the console. The message can also be specified using nested text.

`addproperty`	Defines the name of a property to be created from input. [String]
`message`	Message that gets displayed to the user during the build run. [String]
`validargs`	Defines valid input parameters as comma-separated strings. [String]

<iplanet-ejbc>

Compiles EJB stubs and skeletons for the iPlanet Application Server.

`classpath`	Classpath to be used when compiling the EJB stubs and skeletons. [Path]
`debug`	If true, debugging output will be generated when `ejbc` is executed. [Boolean]
`dest`	Destination directory where the EJB source classes must exist and where the stubs and skeletons will be written. [File]
`ejbdescriptor`	Location of the standard XML EJB descriptor. [File]
`iasdescriptor`	Location of the iAS-specific XML EJB descriptor. [File]
`iashome`	May be used to specify the "home" directory for this iAS installation. [File]
`keepgenerated`	If true, the Java source files generated by ejbc will be saved. [Boolean]
`<classpath>`	Adds to the classpath used when compiling the EJB stubs and skeletons. [Path]

<jar> *

Creates a JAR archive.

`basedir`	Directory from which to archive files; optional. [File]
`compress`	Sets whether to compress the files or only store them; optional, default is true. [Boolean]
`destfile`	The file to create; required. [File]
`duplicate`	Sets behavior for when a duplicate file is about to be added. [`add`, `preserve`, `fail`]
`encoding`	Encoding to use for file names, defaults to the platform's default encoding. [String]
`filesonly`	If true, emulates Sun's JAR utility by not adding parent directories; optional, defaults to false. [Boolean]
`index`	Sets whether to create an index list for classes. [Boolean]
`manifest`	The manifest file to use. [File]
`update`	If true, updates an existing file, otherwise overwrites any existing one; optional, defaults to false. [Boolean]
`whenempty`	Sets behavior of the task when no files match. [`fail`, `skip`, `create`]
`<fileset>`	Adds a set of files. [Fileset]
`<manifest>`	Allows the manifest for the archive file to be provided inline in the build file rather than in an external file.
`<metainf>`	Adds a zipfileset to include in the META-INF directory. [ZipFileset]
`<zipfileset>`	Adds a set of files that can be read from an archive and be given a prefix/full path. [ZipFileset]
`<zipgroup-fileset>`	Adds a group of Zip files. [Fileset]

<jarlib-available>

Checks whether an extension is present in a fileset or an extension set.

`file`	The JAR library to check. [File]
`property`	The name of property to set if extensions are available. [String]
`<extension>`	Extension to look for.
`<extensionset>`	Adds a set of extensions to search in.

<jarlib-display>

Displays the `Optional Package` and `Package Specification` information contained within the specified JARs.

`file`	The JAR library to display information for. [File]
`<fileset>`	Adds a set of files about which library data will be displayed. [Fileset]

<jarlib-manifest> Generates a manifest that declares all the dependencies.

`destfile`	The location where generated manifest is placed. [File]
`<attribute>`	Adds an attribute that is to be put in main section of manifest.
`<depends>`	Adds a set of extensions that this library requires.
`<extension>`	Adds an extension that this library implements.
`<options>`	Adds a set of extensions that this library optionally requires.

<jarlib-resolve> Tries to locate a JAR to satisfy an extension and place the location of the JAR into a property.

`checkextension`	If true, libraries returned by nested resolvers should be checked to see if they supply an extension. [Boolean]
`failonerror`	If true, failure to locate library should fail build. [Boolean]
`property`	The name of the property in which the location of library is stored. [String]
`<ant>`	Adds Ant resolver to run an Ant build file to generate a library.
`<extension>`	Specifies extension to look for.
`<location>`	Adds location resolver to look for a library in a location relative to project directory.
`<url>`	Adds a URL resolver to download a library from a URL to a local file.

<java> Launcher for Java applications.

`append`	If true, append output to existing file. [Boolean]
`classname`	Java class to execute. [String]
`classpath`	Classpath to be used when running the Java class. [Path]
`classpathref`	Classpath to use, by reference. [Reference]
`dir`	The working directory of the process. [File]
`failonerror`	If true, then fail if the command exits with a return code other than 0. [Boolean]
`fork`	If true, execute in a new VM. [Boolean]
`jar`	The location of the JAR file to execute. [File]
`jvm`	Command used to start the VM (only if not forking). [String]
`jvmargs`	Command-line arguments for the JVM. [String]
`jvmversion`	JVM version. [String]
`maxmemory`	Corresponds to `-mx` or `-Xmx`, depending on VM version. [String]
`newenvironment`	If true, use a completely new environment. [Boolean]
`output`	File the output of the process is redirected to. [File]
`timeout`	Timeout in milliseconds after which the process will be killed. [Long]
`<arg>`	Adds a command-line argument.
`<classpath>`	Adds a path to the classpath. [Path]
`<env>`	Adds an environment variable.
`<jvmarg>`	Adds a JVM argument.
`<sysproperty>`	Adds a system property.

<javac> * Compiles Java source files.

`bootclasspath`	Bootclasspath that will be used to compile the classes against. [Path]
`bootclasspathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`classpath`	Classpath to be used for this compilation. [Path]
`classpathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`compiler`	Chooses the implementation for this particular task. [String]
`debug`	Indicates whether source should be compiled with debug information; defaults to off. [Boolean]
`debuglevel`	Keyword list to be appended to the `-g` command-line switch. [String]
`depend`	Enables dependencytracking for compilers that support this (jikes and classic). [Boolean]
`deprecation`	Indicates whether source should be compiled with deprecation information; defaults to off. [Boolean]
`destdir`	Destination directory into which the Java source files should be compiled. [File]
`encoding`	Java source file encoding name. [String]
`executable`	The name of the javac executable. [String]
`extdirs`	Extension directories that will be used during the compilation. [Path]
`failonerror`	Indicates whether the build will continue even if there are compilation errors; defaults to true. [Boolean]
`fork`	If true, forks the javac compiler. [Boolean]
`includeantruntime`	If true, includes Ant's own classpath in the classpath. [Boolean]
`includejavaruntime`	If true, includes the Java run-time libraries in the classpath. [Boolean]
`listfiles`	If true, lists the source files being handed off to the compiler. [Boolean]
`memoryinitialsize`	The initial size of the memory for the underlying VM if javac is run externally; ignored otherwise. [String]
`memorymaximumsize`	The maximum size of the memory for the underlying VM if javac is run externally; ignored otherwise. [String]
`nowarn`	If true, enables the `-nowarn` option. [Boolean]
`optimize`	If true, compiles with optimization enabled. [Boolean]
`source`	Value of the `-source` command-line switch; will be ignored by all implementations except modern and jikes. [String]
`sourcepath`	Source path to be used for this compilation. [Path]
`sourcepathref`	Adds a reference to a source path defined elsewhere. [Reference]
`srcdir`	Source directories to find the source Java files. [Path]
`target`	Target VM that the classes will be compiled for. [String]
`verbose`	If true, asks the compiler for verbose output. [Boolean]

`<bootclasspath>`	Adds a path to the bootclass path. [Path]
`<classpath>`	Adds a path to the classpath. [Path]
`<compilerarg>`	Adds an implementation-specific command-line argument.
`<extdirs>`	Adds a path to extdirs. [Path]
`<sourcepath>`	Adds a path to source path. [Path]
`<src>`	Adds a path for source compilation. [Path]

<*javacc*> Invokes the JavaCC compiler on a grammar file.

`buildparser`	`BUILD_PARSER` grammar option. [Boolean]
`buildtokenmanager`	`BUILD_TOKEN_MANAGER` grammar option. [Boolean]
`cachetokens`	`CACHE_TOKENS` grammar option. [Boolean]
`choiceambiguitycheck`	`CHOICE_AMBIGUITY_CHECK` grammar option. [Integer]
`commontokenaction`	`COMMON_TOKEN_ACTION` grammar option. [Boolean]
`debuglookahead`	`DEBUG_LOOKAHEAD` grammar option. [Boolean]
`debugparser`	`DEBUG_PARSER` grammar option. [Boolean]
`debugtokenmanager`	`DEBUG_TOKEN_MANAGER` grammar option. [Boolean]
`errorreporting`	`ERROR_REPORTING` grammar option. [Boolean]
`forcelacheck`	`FORCE_LA_CHECK` grammar option. [Boolean]
`ignorecase`	`IGNORE_CASE` grammar option. [Boolean]
`javacchome`	The directory containing the JavaCC distribution. [File]
`javaunicodeescape`	`JAVA_UNICODE_ESCAPE` grammar option. [Boolean]
`lookahead`	`LOOKAHEAD` grammar option. [Integer]
`optimizetokenmanager`	`OPTIMIZE_TOKEN_MANAGER` grammar option. [Boolean]
`otherambiguitycheck`	`OTHER_AMBIGUITY_CHECK` grammar option. [Integer]
`outputdirectory`	The directory to write the generated files to. [File]
`sanitycheck`	`SANITY_CHECK` grammar option. [Boolean]
`static`	`STATIC` grammar option. [Boolean]
`target`	The grammar file to process. [File]
`unicodeinput`	`UNICODE_INPUT` grammar option. [Boolean]
`usercharstream`	`USER_CHAR_STREAM` grammar option. [Boolean]
`usertokenmanager`	`USER_TOKEN_MANAGER` grammar option. [Boolean]

<*javadoc*> Generates Javadoc documentation for a collection of source code.

`access`	Scope to be processed. [`protected`, `public`, `package`, `private`]
`additionalparam`	Sets an additional parameter on the command line. [String]
`author`	Includes the author tag in the generated documentation. [Boolean]
`bootclasspath`	Boot classpath to use. [Path]
`bootclasspathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`bottom`	Text to be placed at the bottom of each output file. [String]

`charset`	Charset for cross-platform viewing of generated documentation. [String]
`classpath`	Classpath to be used for this javadoc run. [Path]
`classpathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`defaultexcludes`	Sets whether default exclusions should be used. [Boolean]
`destdir`	Specifies directory where the Javadoc output will be generated. [File]
`docencoding`	Specifies output file encoding name. [String]
`doclet`	Specifies class that starts the doclet used in generating the documentation. [String]
`docletpath`	Specifies classpath used to find the doclet class. [Path]
`docletpathref`	Specifies classpath used to find the doclet class by reference. [Reference]
`doctitle`	Specifies title of the generated overview page. [String]
`encoding`	Specifies encoding name of the source files. [String]
`excludepackagenames`	Specifies list of packages to be excluded. [String]
`extdirs`	Specifies location of the extensions directories. [Path]
`failonerror`	Specifies the build process to fail if javadoc fails (as indicated by a nonzero return code). [Boolean]
`footer`	Places footer text at the bottom of each output file. [String]
`group`	Groups specified packages together in overview page. [String]
`header`	Places header text at the top of each output file. [String]
`helpfile`	Specifies the HTML help file to use. [File]
`link`	Creates links to javadoc output at the given URL. [String]
`linkoffline`	Links to docs at `url` using package list at `url2`—separates the URLs by using a space character. [String]
`locale`	Locale to use in documentation generation. [String]
`maxmemory`	Maximum memory to be used by the javadoc process. [String]
`nodeprecated`	If true, do not include `@deprecated` information. [Boolean]
`nodeprecatedlist`	If true, do not generate deprecated list. [Boolean]
`nohelp`	If true, do not generate help link. [Boolean]
`noindex`	If true, do not generate index. [Boolean]
`nonavbar`	If true, do not generate navigation bar. [Boolean]
`notree`	If true, do not generate class hierarchy. [Boolean]
`old`	Indicates whether Javadoc should produce old style (JDK 1.1) documentation. [Boolean]
`overview`	Specifies the file containing the overview to be included in the generated documentation. [File]
`package`	Indicates whether only package, protected, and public classes and members are to be included in the scope processed. [Boolean]
`packagelist`	The name of a file containing the packages to process. [String]
`packagenames`	Package names to be processed. [String]

`private`	Indicates whether all classes and members are to be included in the scope processed. [Boolean]
`protected`	Indicates whether only protected and public classes and members are to be included in the scope processed. [Boolean]
`public`	Indicates whether only public classes and members are to be included in the scope processed. [Boolean]
`serialwarn`	If true, generates warning about `@serial` tag. [Boolean]
`source`	Enables the `-source` switch; will be ignored if javadoc is not the 1.4 version or a different doclet than the standard doclet is used. [String]
`sourcefiles`	List of source files to process. [String]
`sourcepath`	Specifies where to find source file. [Path]
`sourcepathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`splitindex`	Generates a split index. [Boolean]
`stylesheetfile`	Specifies the CSS stylesheet file to use. [File]
`use`	Generates the use page for each package. [Boolean]
`useexternalfile`	Works around command-line length limit by using an external file for the sourcefiles. [Boolean]
`verbose`	Runs javadoc in verbose mode. [Boolean]
`version`	Includes the version tag in the generated documentation. [Boolean]
`windowtitle`	Title to be placed in the HTML `<title>` tag of the generated documentation. [String]
`<bootclasspath>`	Creates a path to be configured with the boot classpath. [Path]
`<bottom>`	Text to be placed at the bottom of each output file.
`<classpath>`	Creates a path to be configured with the classpath to use. [Path]
`<doclet>`	Creates a doclet to be used in the documentation generation.
`<doctitle>`	Adds a document title to use for the overview page.
`<excludepackage>`	Adds a package to be excluded from the javadoc run.
`<fileset>`	Adds a fileset. [Fileset]
`<footer>`	Footer text to be placed at the bottom of each output file.
`<group>`	Separates packages on the overview page into whatever groups you specify, one group per table.
`<header>`	Header text to be placed at the top of each output file.
`<link>`	Creates link to javadoc output at the given URL.
`<package>`	Adds a single package to be processed.
`<packageset>`	Adds a packageset. [Dirset]
`<source>`	Adds a single source file.
`<sourcepath>`	Creates a path to be configured with the locations of the source files. [Path]
`<tag>`	Creates and adds a `-tag` argument.
`<taglet>`	Adds a taglet.

<javah> Generates JNI header files using javah.

`bootclasspath`	Location of bootstrap class files. [Path]
`bootclasspathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`class`	The fully qualified name of the class (or classes, separated by commas). [String]
`classpath`	The classpath to use. [Path]
`classpathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`destdir`	Destination directory into which the Java source files should be compiled. [File]
`force`	If true, output files should always be written (JDK1.2 only). [Boolean]
`old`	If true, specifies that old JDK1.0-style header files should be generated. [Boolean]
`outputfile`	Concatenates the resulting header or source files for all the classes listed into this file. [File]
`stubs`	If true, generates C declarations from the Java object file (used with old). [Boolean]
`verbose`	If true, causes javah to print a message concerning the status of the generated files. [Boolean]
`<bootclasspath>`	Adds path to bootstrap class files. [Path]
`<class>`	Adds class to process.
`<classpath>`	Path to use for classpath. [Path]

<jdepend> Runs JDepend tests.

`classpath`	Classpath to be used for this compilation. [Path]
`classpathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`dir`	The directory to invoke the VM in. [File]
`fork`	If true, forks into a new JVM. [Boolean]
`format`	The format to write the output in. [`xml`, `text`]
`haltonerror`	Sets whether to halt on failure. [Boolean]
`jvm`	The command used to invoke a forked Java Virtual Machine. [String]
`outputfile`	The output file name. [File]
`<classpath>`	Adds a path to the classpath. [Path]
`<sourcespath>`	Adds a path to source code to analyze. [Path]

<jjtree> Runs the JJTree preprocessor for the JavaCC compiler compiler.

`buildnodefiles`	`BUILD_NODE_FILES` grammar option. [Boolean]
`javacchome`	The directory containing the JavaCC distribution. [File]
`multi`	`MULTI` grammar option. [Boolean]
`nodedefaultvoid`	`NODE_DEFAULT_VOID` grammar option. [Boolean]
`nodefactory`	`NODE_FACTORY` grammar option. [Boolean]
`nodepackage`	`NODE_PACKAGE` grammar option. [String]

`nodeprefix`	`NODE_PREFIX` grammar option. [String]
`nodescopehook`	`NODE_SCOPE_HOOK` grammar option. [Boolean]
`nodeusesparser`	`NODE_USES_PARSER` grammar option. [Boolean]
`outputdirectory`	The directory to write the generated file to. [File]
`static`	`STATIC` grammar option. [Boolean]
`target`	The jjtree grammar file to process. [File]
`visitor`	`VISITOR` grammar option. [Boolean]
`visitorexception`	`VISITOR_EXCEPTION` grammar option. [String]

<jpcoverage> Runs Sitraka JProbe Coverage analyzer.

`applet`	If true, runs an applet. [Boolean]
`classname`	Classname to run as stand-alone or runner for filesets. [String]
`exitprompt`	Toggles display of the console prompt: `always`, `error`, `never`. [String]
`finalsnapshot`	Type of snapshot to send at program termination: `none`, `coverage`, `all`. [String]
`home`	The directory where JProbe is installed. [File]
`javaexe`	Path to the java executable. [File]
`recordfromstart`	If you want to start analyzing as soon as the program begins, use `all`. If not, select `none`. [`coverage`, `none`, `all`]
`seedname`	Seed name for snapshot file. [String]
`snapshotdir`	The path to the directory where snapshot files are stored. [File]
`tracknatives`	If true, tracks native methods. [Boolean]
`vm`	Indicates which virtual machine to run. [`java2`, `jdk118`, `jdk117`]
`warnlevel`	Sets warning level (0-3, where 0 is the least amount of warnings). [Integer]
`workingdir`	The physical path to the working directory for the VM. [File]
`<arg>`	Adds a command argument.
`<classpath>`	Classpath to run the files. [Path]
`<fileset>`	The classnames to execute. [Fileset]
`<filters>`	Defines class/method filters based on pattern matching.
`<jvmarg>`	Adds a JVM argument.
`<socket>`	Defines a host and port to connect to if you want to do remote viewing.
`<triggers>`	Defines events to use for interacting with the collection of data performed during coverage.

<jpcovmerge> Runs the snapshot merge utility for JProbe Coverage.

`home`	The directory where JProbe is installed. [File]
`tofile`	Output snapshot file. [File]
`verbose`	If true, perform the merge in verbose mode giving details about the snapshot processing. [Boolean]
`<fileset>`	Adds a fileset containing the snapshots to include. [Fileset]

<jpcovreport> Runs the JProbe Coverage 3.0 snapshot merge utility.

`format`	Format of the report. [`html`, `text`, `xml`]
`home`	The directory where JProbe is installed. [File]
`includesource`	If true, include text of the source code lines. [Boolean]
`percent`	A numeric value for the threshold for printing methods. [Integer]
`snapshot`	The name of the snapshot file that is the source to the report. [File]
`tofile`	The name of the generated output file. [File]
`type`	The type of report to be generated. [`executive`, `summary`, `detailed`, `verydetailed`]
`<reference>`	Adds a set of classes whose coverage information will be checked against.
`<sourcepath>`	Adds a path to source files. [Path]

<jspc> * Runs a JSP compiler.

`classpath`	Classpath to be used for this compilation. [Path]
`classpathref`	Adds a reference to a classpath defined elsewhere. [Reference]
`compiler`	Class name of a JSP compiler adapter. [String]
`destdir`	Destination directory into which the JSP source files should be compiled. [File]
`failonerror`	Specifies the build to halt if compilation fails (default is true). [Boolean]
`ieplugin`	Java Plug-in CLASSID for Internet Explorer. [String]
`mapped`	If true, generates separate `write()` calls for each HTML line in the JSP. [Boolean]
`package`	Name of the package the compiled JSP files should be in. [String]
`srcdir`	Path for source JSP files. [Path]
`uribase`	The URI context of relative URI references in the JSP pages. [File]
`uriroot`	The root directory that URI files should be resolved against. [File]
`verbose`	Verbose level of the compiler. [Integer]
`webinc`	Output file name for the fraction of web.xml that lists servlets. [File]
`webxml`	File name for web.xml. [File]
`<classpath>`	Adds a path to the classpath. [Path]
`<webapp>`	Adds a single webapp.

<junit> Runs JUnit tests.

`dir`	The directory to invoke the VM in. [File]
`errorproperty`	Property to set to `true` if there is a error in a test. [String]
`failureproperty`	Property to set to `true` if there is a failure in a test. [String]
`filtertrace`	If true, smartly filter the stack frames of JUnit errors and failures before reporting them. [Boolean]
`fork`	If true, JVM should be forked for each test. [Boolean]
`haltonerror`	If true, stop the build process when there is an error in a test. [Boolean]

`haltonfailure`	If true, stop the build process if a test fails (errors are considered failures as well). [Boolean]
`includeant-runtime`	If true, include ant.jar, optional.jar, and junit.jar in the forked VM. [Boolean]
`jvm`	The command used to invoke the Java Virtual Machine, default is `java`. [String]
`maxmemory`	Maximum memory to be used by all forked JVMs. [String]
`newenvironment`	If true, use a new environment when forked. [Boolean]
`printsummary`	If true, print one-line statistics for each test, or `withOutAndErr` to also show standard output and error. [`true`, `yes`, `false`, `no`, `on`, `off`, `withOutAndErr`]
`showoutput`	If true, send any output generated by tests to Ant's logging system as well as to the formatters. [Boolean]
`timeout`	Timeout value (in milliseconds). [Integer]
`<batchtest>`	Adds a set of tests based on pattern matching.
`<classpath>`	Adds path to classpath used for tests. [Path]
`<env>`	Adds an environment variable; used when forking.
`<formatter>`	Add a new formatter to all tests of this task.
`<jvmarg>`	Adds a JVM argument; ignored if not forking.
`<sysproperty>`	Adds a system property that tests can access.
`<test>`	Adds a new single testcase.

<junitreport>

Aggregates all `<junit>` XML formatter test suite data under a specific directory and transforms the results via XSLT.

`todir`	Destination directory where the results should be written. [File]
`tofile`	Name of the aggregated results file. [String]
`<fileset>`	Adds a new fileset containing the XML results to aggregate. [Fileset]
`<report>`	Generates a report based on the document created by the merge.

<loadfile>

Loads a whole text file into a single property.

`encoding`	Encoding to use for input, defaults to the platform's default encoding. [String]
`failonerror`	If true, fail on load error. [Boolean]
`property`	Property name to save to. [String]
`srcfile`	File to load. [File]
`<filterchain>`	Adds the FilterChain element. [FilterChain]

<loadproperties>

Loads a file's contents as Ant properties.

`srcfile`	File to load. [File]
`<filterchain>`	Adds a FilterChain. [FilterChain]

<*mail*> A task to send SMTP email.

`bcclist`	Adds `bcc` address elements. [String]
`cclist`	Adds `cc` address elements. [String]
`encoding`	Allows the build writer to choose the preferred encoding method. [`auto`, `mime`, `uu`, `plain`]
`failonerror`	Indicates whether BuildExceptions should be passed back to the core. [Boolean]
`files`	Adds a list of files to be attached. [String]
`from`	Shorthand to set the from address element. [String]
`includefilenames`	Sets `Includefilenames` attribute. [Boolean]
`mailhost`	Host. [String]
`mailport`	Mail server port. [Integer]
`message`	Shorthand method to set the message. [String]
`messagefile`	Shorthand method to set the message from a file. [File]
`messagemimetype`	Shorthand method to set type of the text message, text/plain by default, but text/html or text/xml is quite feasible. [String]
`subject`	Subject line of the email. [String]
`tolist`	Adds `to` address elements. [String]
`<bcc>`	Adds `bcc` address element.
`<cc>`	Adds `cc` address element.
`<fileset>`	Adds a set of files (nested fileset attribute). [Fileset]
`<from>`	Adds a `from` address element.
`<message>`	Adds a `message` element.
`<to>`	Adds a `to` address element.

<*manifest*> Creates a manifest file for inclusion in a JAR.

`file`	The name of the manifest file to create/update. [File]
`mode`	Update policy; default is replace. [`update`, `replace`]
`<attribute>`	Adds an attribute to the manifest's main section.
`<section>`	Adds a section to the manifest.

<*maudit*> Invokes the Metamata Audit/Webgain Quality Analyzer on a set of Java files.

`fix`	Automatically fixes certain errors (those marked as fixable in the manual); optional, default false. [Boolean]
`list`	Creates listing file for each audited file; optional, default false. [Boolean]
`maxmemory`	Maximum memory for the JVM; optional. [String]
`metamatahome`	The home directory containing the Metamata distribution; required. [File]
`tofile`	The XML file to which the Audit result should be written to; required. [File]
`unused`	Finds declarations unused in search paths; optional, default false. [Boolean]
`<classpath>`	Classpath (also source path unless one explicitly set). [Path]

`<fileset>`	The Java files or directory to audit. [Fileset]
`<jvmarg>`	Additional optional parameters to pass to the JVM.
`<rulespath>`	Classpath for additional audit rules; these must be placed before metamata.jar. [Path]
`<searchpath>`	Search path to use for unused global declarations; required when `unused` is set. [Path]
`<sourcepath>`	Source path. [Path]

<mimemail>

See <mail>.

<mkdir>

Creates a given directory.

`dir`	The directory to create; required. [File]

<mmetrics>

Computes the metrics of a set of Java files and writes the results to an XML file.

`granularity`	Granularity of the audit. [`compilation-units`, `files`, `methods`, `types`, `packages`]
`maxmemory`	Maximum memory for the JVM; optional. [String]
`metamatahome`	The home directory containing the Metamata distribution; required. [File]
`tofile`	Output XML file. [File]
`<classpath>`	Classpath (also source path unless one explicitly set). [Path]
`<fileset>`	The Java files or directory to audit. [Fileset]
`<jvmarg>`	Additional optional parameters to pass to the JVM.
`<path>`	New path (directory) to measure metrics from. [Path]
`<sourcepath>`	Source path. [Path]

<move>

Moves a file or directory to a new file or directory.

`encoding`	Character encoding. [String]
`failonerror`	If false, notes errors to the output but keeps going. [Boolean]
`file`	Single source file to copy. [File]
`filtering`	If true, enables filtering. [Boolean]
`flatten`	When copying directory trees, the files can be flattened into a single directory. [Boolean]
`includeemptydirs`	Used to copy empty directories. [Boolean]
`overwrite`	Overwrites any existing destination files. [Boolean]
`preservelast-modified`	Gives the copied files the same last modified time as the original files. [Boolean]
`todir`	Destination directory. [File]
`tofile`	Destination file. [File]
`verbose`	Used to force listing of all names of copied files. [Boolean]
`<fileset>`	Adds a set of files to copy. [Fileset]
`<filterchain>`	Adds a FilterChain. [FilterChain]
`<filterset>`	Adds a filterset. [Filterset]
`<mapper>`	Defines the mapper to map source to destination files. [Mapper]

<*mparse*> Invokes the Metamata MParse compiler compiler on a grammar file.

`cleanup`	Remove the intermediate Sun JavaCC file; optional, default false. [Boolean]
`debugparser`	Set parser debug mode; optional, default false. [Boolean]
`debugscanner`	Set scanner debug mode; optional, default false. [Boolean]
`maxmemory`	Maximum memory for the JVM; optional. [String]
`metamatahome`	The home directory containing the Metamata distribution; required. [File]
`target`	The .jj file to process; required. [File]
`verbose`	Set verbose mode; optional, default false. [Boolean]
`<classpath>`	Creates a classpath entry. [Path]
`<jvmarg>`	Additional optional parameters to pass to the JVM.
`<sourcepath>`	Creates a source path entry. [Path]

<*native2ascii*> * Converts files from native encodings to ASCII.

`dest`	Destination directory to place converted files into. [File]
`encoding`	Encoding to translate to/from. [String]
`ext`	Extension which converted files should have. [String]
`reverse`	Flag the conversion to run in the reverse sense, that is ASCII-to-native encoding. [Boolean]
`src`	Source directory in which to find files to convert. [File]
`<mapper>`	Defines the FileNameMapper to use (nested mapper element). [Mapper]

<*netrexxc*> * Compiles NetRexx source files.

`binary`	Sets whether literals are treated as binary, rather than NetRexx types. [Boolean]
`classpath`	Classpath used for NetRexx compilation. [String]
`comments`	Sets whether comments are passed through to the generated Java source. [Boolean]
`compact`	Sets whether error messages come out in compact or verbose format. [Boolean]
`compile`	Sets whether the NetRexx compiler should compile the generated Java code. [Boolean]
`console`	Sets whether messages should be displayed. [Boolean]
`crossref`	Sets whether variable cross-references are generated. [Boolean]
`decimal`	Sets whether decimal arithmetic should be used for the NetRexx code. [Boolean]
`destdir`	Destination directory into which the NetRexx source files should be copied and then compiled. [File]
`diag`	Sets whether diagnostic information about the compile is generated. [Boolean]
`explicit`	Sets whether variables must be declared explicitly before use. [Boolean]

`format`	Sets whether the generated Java code is formatted nicely or left to match NetRexx line numbers for call stack debugging. [Boolean]
`java`	Sets whether the generated Java code is produced. [Boolean]
`keep`	Sets whether the generated Java source file should be kept after compilation. [Boolean]
`logo`	Sets whether the compiler text logo is displayed when compiling. [Boolean]
`replace`	Sets whether the generated .Java file should be replaced when compiling. [Boolean]
`savelog`	Sets whether the compiler messages will be written to NetRexxC.log as well as to the console. [Boolean]
`sourcedir`	Tells the NetRexx compiler to store the class files in the same directory as the source files. [Boolean]
`srcdir`	Source dir to find the source Java files. [File]
`strictargs`	Tells the NetRexx compiler that method calls always need parentheses, even if no arguments are needed. [Boolean]
`strictassign`	Tells the NetRexx compile that assignments must match exactly on type. [Boolean]
`strictcase`	Specifies whether the NetRexx compiler should be case sensitive. [Boolean]
`strictimport`	Sets whether classes need to be imported explicitly using an `import` statement. [Boolean]
`strictprops`	Sets whether local properties need to be qualified explicitly using `this`. [Boolean]
`strictsignal`	Sets whether the compiler should force catching of exceptions by explicitly named types. [Boolean]
`suppress-deprecation`	Sets whether we should filter out any deprecation-messages of the compiler output. [Boolean]
`suppressexcep-tionnotsignalled`	Sets whether the task should suppress the `FooException` is in SIGNALS list but is not signalled within the method, which is sometimes rather useless. [Boolean]
`suppressmethod-argumentnotused`	Sets whether the task should suppress the "Method argument is not used" in strictargs-Mode, which cannot be suppressed by the compiler itself. [Boolean]
`suppressprivate-propertynotused`	Sets whether the task should suppress the "Private property is defined but not used" in strictargs-Mode, which can be quite annoying while developing. [Boolean]
`suppressvariable-notused`	Sets whether the task should suppress the "Variable is set but not used" in strictargs-Mode. [Boolean]
`symbols`	Sets whether debug symbols should be generated into the class file. [Boolean]
`time`	Asks the NetRexx compiler to print compilation times to the console. [Boolean]
`trace`	Turns on or off tracing, and directs the resultant output. [`trace`, `trace1`, `trace2`, `notrace`]
`utf8`	Tells the NetRexx compiler that the source is in UTF8. [Boolean]
`verbose`	Sets whether lots of warnings and error messages should be generated. [`verbose`, `verbose0`, `verbose1`, `verbose2`, `verbose3`, `verbose4`, `verbose5`, `noverbose`]

<p4add> Adds specified files to a Perforce server.

`changelist`	If specified, the open files are associated with the specified pending changelist number; otherwise the open files are associated with the default changelist. [Integer]
`client`	Specifies the p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Sets extra command options; only used on some of the Perforce tasks. [String]
`commandlength`	Positive integer specifying the maximum length of the command line when calling Perforce to add the files. [Integer]
`failonerror`	Sets whether to stop the build or keep going if an error is returned from the p4 command; default is true. [Boolean]
`port`	Specifies the p4d server and port to connect to; optional, default `perforce:1666`. [String]
`user`	Specifies the p4 username; optional, defaults to the current user. [String]
`view`	Specifies the client, branch, or label view to operate upon; optional default `//...`. [String]
`<fileset>`	Files to add. [Fileset]

<p4change> Requests a new changelist from the Perforce server.

`client`	The p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Set extra command options; only used on some of the Perforce tasks. [String]
`description`	Description for ChangeList;optional. [String]
`failonerror`	Sets whether to stop the build (true, default) or keep going if an error is returned from the p4 command. [Boolean]
`port`	The p4d server and port to connect to; optional, default `perforce:1666`. [String]
`user`	The p4 username; optional, defaults to the current user. [String]
`view`	The client, branch, or label view to operate upon; optional default `//...`. [String]

<p4counter> Obtains or sets the value of a Perforce counter.

`client`	The p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Set extra command options; only used on some of the Perforce tasks. [String]
`failonerror`	Sets whether to stop the build (true, default) or keep going if an error is returned from the p4 command. [Boolean]
`name`	The name of the counter; required. [String]
`port`	The p4d server and port to connect to; optional, default `perforce:1666`. [String]
`property`	A property to be set with the value of the counter. [String]
`user`	The p4 username; optional, defaults to the current user. [String]

`value`	The new value for the counter; optional. [Integer]
`view`	The client, branch, or label view to operate upon; optional default `//...`. [String]

<*p4delete*> Checkout Perforce-managed files for deletion.

`change`	An existing changelist number for the deletion; optional but strongly recommended. [String]
`client`	The p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Set extra command options. [String]
`failonerror`	Sets whether to stop the build (true, default) or keep going if an error is returned from the p4 command. [Boolean]
`port`	The p4d server and port to connect to; optional, default `perforce:1666`. [String]
`user`	The p4 username; optional, defaults to the current user. [String]
`view`	The client, branch or label view to operate upon; optional default `//...`. [String]

<*p4edit*> Open Perforce-managed files for editing.

`change`	An existing changelist number to assign files to; optional but strongly recommended. [String]
`client`	The p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Set extra command options; only used on some of the Perforce tasks. [String]
`failonerror`	Sets whether to stop the build (true, default) or keep `going` if an error is returned from the p4 command. [Boolean]
`port`	The p4d server and port to connect to; optional, default `perforce:1666`. [String]
`user`	The p4 username; optional, defaults to the current user. [String]
`view`	The client, branch, or label view to operate upon; optional default `//...`. [String]

<*p4have*> Lists Perforce-managed files currently on the client.

`client`	The p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Set extra command options; only used on some of the Perforce tasks. [String]
`failonerror`	Sets whether to stop the build (true, default) or keep going if an error is returned from the p4 command. [Boolean]
`port`	The p4d server and port to connect to; optional, default `perforce:1666`. [String]
`user`	The p4 username; optional, defaults to the current user. [String]
`view`	The client, branch, or label view to operate upon; optional, default `//...`. [String]

<p4label> Creates a new Perforce label and sets contents to reflect current client file revisions.

`client`	The p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Set extra command options; only used on some of the Perforce tasks. [String]
`desc`	Label description; optional. [String]
`failonerror`	Sets whether to stop the build (true, default) or keep going if an error is returned from the p4 command. [Boolean]
`lock`	When set to `locked`, Perforce will lock the label once created; optional. [String]
`name`	The name of the label; optional, default `AntLabel`. [String]
`port`	The p4d server and port to connect to; optional, default `perforce:1666`. [String]
`user`	The p4 username; optional, defaults to the current user. [String]
`view`	The client, branch, or label view to operate upon; optional, default `//...`. [String]

<p4reopen> Reopens Perforce-managed files.

`client`	The p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Set extra command options; only used on some of the Perforce tasks. [String]
`failonerror`	Sets whether to stop the build (true, default) or keep going if an error is returned from the p4 command. [Boolean]
`port`	The p4d server and port to connect to; optional, default `perforce:1666`. [String]
`tochange`	The changelist to move files to; required. [String]
`user`	The p4 username; optional, defaults to the current user. [String]
`view`	The client, branch, or label view to operate upon; optional default `//...`. [String]

<p4revert> Reverts Perforce open files or files in a changelist

`change`	The changelist to revert; optional. [String]
`client`	The p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Set extra command options; only used on some of the Perforce tasks. [String]
`failonerror`	Sets whether to stop the build (true, default) or keep going if an error is returned from the p4 command. [Boolean]
`port`	The p4d server and port to connect to; optional, default `perforce:1666`. [String]
`revertonly-unchanged`	Flag to revert only unchanged files (p4 revert `-a`); optional, default false. [Boolean]
`user`	The p4 username; optional, defaults to the current user. [String]
`view`	The client, branch, or label view to operate upon; optional default `//...`. [String]

<p4submit> Submits a numbered changelist to Perforce.

`change`	The changelist number to submit; required. [String]
`client`	The p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Set extra command options; only used on some of the Perforce tasks. [String]
`failonerror`	Sets whether to stop the build (true, default) or keep going if an error is returned from the p4 command. [Boolean]
`port`	The p4d server and port to connect to; optional, default `perforce:1666`. [String]
`user`	The p4 username; optional, defaults to the current user. [String]
`view`	The client, branch, or label view to operate upon; optional default `//...`. [String]

<p4sync> Synchronizes client space to a Perforce depot view.

`client`	The p4 client spec to use; optional, defaults to the current user. [String]
`cmdopts`	Set extra command options; only used on some of the Perforce tasks. [String]
`failonerror`	Sets whether to stop the build (true, default) or keep going if an error is returned from the p4 command. [Boolean]
`force`	Force a refresh of files, if this attribute is set; false by default. [String]
`label`	Label to sync client to; optional. [String]
`port`	The p4d server and port to connect to; optional, default `perforce:1666`. [String]
`user`	The p4 username; optional, defaults to the current user. [String]
`view`	The client, branch, or label view to operate upon; optional default `//...`. [String]

<parallel> Executes the contained tasks in separate threads, continuing once all are completed. Any Ant task can be nested inside this task.

<patch> Patches a file by applying a diff file to it; requires `patch` to be on the execution path.

`backups`	Flag to create backups; optional, default=false. [Boolean]
`dir`	The directory to run the patch command in, defaults to the project's base directory. [File]
`ignorewhitespace`	Flag to ignore white space differences; default=false. [Boolean]
`originalfile`	The file to patch; optional if it can be inferred from the diff file. [File]
`patchfile`	The file containing the diff output; required. [File]
`quiet`	Work silently unless an error occurs; optional, default=false. [Boolean]
`reverse`	Assume patch was created with old and new files swapped; optional, default=false. [Boolean]
`strip`	Strip the smallest prefix containing this many leading slashes from file names. [Integer]

<pathconvert> Converts path and classpath information to a specific target OS format.

`dirsep`	Default directory separator string; defaults to current JVM. [String]
`pathsep`	Default path separator string; defaults to current JVM. [String]
`property`	The property into which the converted path will be placed. [String]
`refid`	Adds a reference to a Path, FileSet, DirSet, or FileList defined elsewhere. [Reference]
`setonempty`	If false, don't set the new property if the result is the empty string; default true. [Boolean]
`targetos`	Sets target platform; required unless `pathsep` or `dirsep` are specified. [`windows`, `unix`, `netware`, `os/2`]
`<map>`	Creates a nested MAP element.
`<path>`	Creates a nested PATH element. [Path]

<property> Sets a property by name, or set of properties (from file or resource) in the project.

`classpath`	The classpath to use when looking up a resource. [Path]
`environment`	The prefix to use when retrieving environment variables. [String]
`file`	The file name of a property file to load. [File]
`location`	Property to the absolute file name of the given file. [File]
`name`	Name of the property to set. [String]
`prefix`	Prefix to apply to properties loaded using `file` or `resource`. [String]
`refid`	Reference to an Ant datatype declared elsewhere. [Reference]
`resource`	The resource name of a property file to load. [String]
`value`	Value of the property. [String]
`<classpath>`	The classpath to use when looking up a resource. [Path]

<propertyfile> Modifies settings in a property file.

`comment`	Optional header comment for the file. [String]
`file`	Location of the property file to be edited; required. [File]
`<entry>`	Specifies a property and how to modify it.

<pvcs> Extracts the latest edition of the source code from a PVCS repository.

`filenameformat`	The format of the folder names; optional. [String]
`force`	Specifies the value of the force argument; optional. [String]
`ignorereturncode`	If set to true the return value from executing the PVCS commands are ignored; optional, default false. [Boolean]
`label`	Only files marked with this label are extracted; optional. [String]
`linestart`	What a valid return value from PVCS looks like when it describes a file. [String]
`promotiongroup`	Specifies the name of the `promotiongroup` argument. [String]

`pvcsbin`	Specifies the location of the PVCS bin directory; optional if on the PATH. [String]
`pvcsproject`	The project within the PVCS repository to extract files from; optional, default "/". [String]
`repository`	The network name of the PVCS repository; required. [String]
`updateonly`	If true, files are fetched only if newer than existing local files; optional, default false. [Boolean]
`workspace`	Workspace to use; optional. [String]
`<pvcsproject>`	Specifies a project within the PVCS repository to extract files from.

<record> Adds a listener to the current build process that records the output to a file.

`action`	Action for the associated recorder entry. [`start`, `stop`]
`append`	Sets whether the logger should append to a previous file. [Boolean]
`emacsmode`	No description. [Boolean]
`loglevel`	Level to which this recorder entry should log to. [`error`, `warn`, `info`, `verbose`, `debug`]
`name`	Name of the file to log to, and the name of the recorder entry. [String]

<replace> * Replaces all occurrences of one or more string tokens with given values in the indicated files.

`dir`	The base directory to use when replacing a token in multiple files; required if `file` is not defined. [File]
`encoding`	File encoding to use on the files read and written by the task; optional, defaults to default JVM encoding. [String]
`file`	Source file; required unless `dir` is set. [File]
`propertyfile`	The name of a property file from which properties specified using nested `<replacefilter>` elements are drawn; Required only if `property` attribute of `<replacefilter>` is used. [File]
`replacefilterfile`	Name of a property file containing filters; optional. [File]
`summary`	Indicates whether a summary of the replace operation should be produced, detailing how many token occurrences and files were processed; optional, default is false. [Boolean]
`token`	String token to replace; required unless a nested `replace-token` element or the `replacefilterfile` attribute is used. [String]
`value`	String value to use as token replacement; optional, default is the empty string " ". [String]
`<replacefilter>`	Adds a replacement filter.
`<replacetoken>`	The token to filter as the text of a nested element.
`<replacevalue>`	The string to replace the token as the text of a nested element.

<replaceregexp> Performs regular expression string replacements in a text file.

`byline`	Process the file(s) one line at a time, executing the replacement on one line at a time. [String]
`file`	File for which the regular expression should be replaced; required unless a nested fileset is supplied. [File]
`flags`	The flags to use when matching the regular expression. [String]
`match`	The regular expression pattern to match in the files; required if no nested `<regexp>` is used. [String]
`replace`	The substitution pattern to place in the files in place of the regular expression. [String]
`<fileset>`	Lists files to apply the replacement to. [Fileset]
`<regexp>`	A regular expression.
`<substitution>`	A substitution pattern.

<rmic> * Runs the rmic compiler against classes.

`base`	Location to store the compiled files; required. [File]
`classname`	The class to run rmic against; optional. [String]
`classpath`	Classpath to be used for this compilation. [Path]
`classpathref`	Adds a path to the classpath by reference. [Reference]
`compiler`	Compiler implementation to use; optional, defaults to the value of the `build.rmic` property, or failing that, default compiler for the current VM. [String]
`debug`	Generates debug info (passes `-g` to rmic); optional, defaults to false. [Boolean]
`extdirs`	Extension directories that will be used during the compilation; optional. [Path]
`filtering`	Indicates whether token filtering should take place; optional, default=false. [Boolean]
`idl`	Indicates that IDL output should be generated. [Boolean]
`idlopts`	Passes additional arguments for `idl` compile. [String]
`iiop`	Indicates that IIOP-compatible stubs should be generated; optional, defaults to false if not set. [Boolean]
`iiopopts`	Sets additional arguments for IIOP. [String]
`includeantruntime`	Sets whether to include the Ant run-time libraries; optional defaults to true. [Boolean]
`includejavaruntime`	Task's classpath. [Boolean]
`sourcebase`	Optional directory to save generated source files to. [File]
`stubversion`	Specifies the JDK version for the generated stub code. [String]
`verify`	Flag to enable verification, so that the classes found by the directory match are checked to see if they implement `java.rmi.Remote`. [Boolean]
`<classpath>`	Adds a path to the classpath. [Path]
`<compilerarg>`	Adds an implementation-specific command-line argument.
`<extdirs>`	Adds path to the extension directories path. [Path]

<rpm> Invokes the rpm tool to build a Linux installation file.

`cleanbuilddir`	Flag (optional, default=false) to remove the generated files in the BUILD directory. [Boolean]
`command`	What command to issue to the rpm tool; optional. [String]
`error`	Optional file to save `stderr` to. [File]
`output`	Optional file to save `stdout` to. [File]
`removesource`	Flag (optional, default=false) to remove the sources after the build. [Boolean]
`removespec`	Flag (optional, default=false) to remove the spec file from SPECS. [Boolean]
`specfile`	The name of the spec File to use; required. [String]
`topdir`	The directory which will have the expected subdirectories, SPECS, SOURCES, BUILD, SRPMS ; optional. [File]

<script> Executes a script. The script can be nested as text, or an external file referenced using `src`.

`language`	Defines the language (required). [String]
`src`	Load the script from an external file; optional. [String]

<sequential> Container task to execute all nested tasks sequentially. This is useful when nested within `<parallel>`.

<serverdeploy> Controls hot deployment tools for J2EE servers.

`action`	The action to be performed, usually `deploy`; required. [String]
`source`	The file name of the component to be deployed; optional depending upon the tool and the action. [File]
`<generic>`	Creates a generic deployment tool.
`<jonas>`	Creates a JOnAS deployment tool, for deployment to JOnAS servers.
`<weblogic>`	Creates a WebLogic deployment tool, for deployment to WebLogic servers.

<setproxy> Sets Java's web proxy properties, so that tasks and code run in the same JVM can have through-the-firewall access to remote web sites, and remote ftp sites.

`nonproxyhosts`	A list of hosts to bypass the proxy on. [String]
`proxyhost`	The HTTP/ftp proxy host. [String]
`proxyport`	The HTTP/ftp proxy port number; default is 80. [Integer]
`socksproxyhost`	The name of a Socks server. [String]
`socksproxyport`	ProxyPort for socks connections. [Integer]

<*signjar*> Signs JAR or Zip files with the javasign command-line tool.

`alias`	The alias to sign under; required. [String]
`internalsf`	Flag to include the .SF file inside the signature; optional; default false. [Boolean]
`jar`	The JAR file to sign; required. [File]
`keypass`	Password for private key (if different than `storepass`); optional. [String]
`keystore`	Keystore location; required. [File]
`lazy`	Flag to control whether the presence of a signature file means a JAR is signed; optional, default false. [Boolean]
`sectionsonly`	Flag to compute hash of entire manifest; optional, default false. [Boolean]
`sigfile`	Name of .SF/.DSA file; optional. [File]
`signedjar`	Name of signed JAR file; optional. [File]
`storepass`	Password for Keystore integrity; required. [String]
`storetype`	Keystore type; optional. [String]
`verbose`	Enable verbose output when signing ; optional: default false. [Boolean]
`<fileset>`	Adds a set of files to sign. [Fileset]

<*sleep*> Sleep, or pause, for a period of time.

`failonerror`	Flag controlling whether to break the build on an error. [Boolean]
`hours`	Hours to add to the sleep time. [Integer]
`milliseconds`	Milliseconds to add to the sleep time. [Integer]
`minutes`	Minutes to add to the sleep time. [Integer]
`seconds`	Seconds to add to the sleep time. [Integer]

<*soscheckin*> Commits and unlocks files in Visual SourceSafe via a SourceOffSite server.

`comment`	Comment to apply to all files being labeled; optional, only valid in SOSLabel. [String]
`file`	File name to act upon; optional. [String]
`label`	Labeled version to operate on in SourceSafe. [String]
`localpath`	Override the working directory and get to the specified path; optional. [Path]
`nocache`	Flag to disable the cache when set; optional, needed if SOSHOME is set as an environment variable. [Boolean]
`nocompress`	Flag that disables compression when set; optional. [Boolean]
`password`	SourceSafe password; optional. [String]
`projectpath`	SourceSafe project path without the `$` prefix; required. [String]
`recursive`	Flag to recursively apply the action (not valid on all SOS tasks); optional, default false. [Boolean]
`soscmd`	Directory where soscmd is located; optional, soscmd must be on the path if omitted. [String]
`soshome`	The path to the SourceOffSite home directory. [String]

`sosserverpath`	Address and port of SourceOffSite Server. [String]
`username`	SourceSafe username; required. [String]
`verbose`	Enable verbose output; optional, default false. [Boolean]
`version`	A version number to get—only works with the SOSGet on a file; optional. [String]
`vssserverpath`	Path to the location of the ss.ini file; required. [String]

<soscheckout> Retrieves and locks files in Visual SourceSafe via a SourceOffSite server.

`comment`	Comment to apply to all files being labelled; optional, only valid in SOSLabel. [String]
`file`	File name to act upon; optional. [String]
`label`	Labeled version to operate on in SourceSafe. [String]
`localpath`	Override the working directory and get to the specified path; optional. [Path]
`nocache`	Flag to disable the cache when set; optional needed if `SOSHOME` is set as an environment variable. [Boolean]
`nocompress`	Flag that disables compression when set; optional. [Boolean]
`password`	SourceSafe password; optional. [String]
`projectpath`	SourceSafe project path without the `$` prefix; required. [String]
`recursive`	Flag to recursively apply the action (not valid on all SOS tasks); optional, default false. [Boolean]
`soscmd`	Directory where soscmd is located; optional, soscmd must be on the path if omitted. [String]
`soshome`	The path to the SourceOffSite home directory. [String]
`sosserverpath`	Address and port of SourceOffSite Server, e.g. [String]
`username`	SourceSafe username; required. [String]
`verbose`	Enable verbose output; optional, default false. [Boolean]
`version`	A version number to get—only works with the SOSGet on a file; optional. [String]
`vssserverpath`	Path to the location of the ss.ini file; required. [String]

<sosget> Retrieves a read-only copy of the specified project or file from Visual SourceSafe via a SourceOffSite server.

`comment`	Comment to apply to all files being labelled; optional, only valid in SOSLabel. [String]
`file`	File name to act upon; optional. [String]
`label`	Labeled version to operate on in SourceSafe. [String]
`localpath`	Override the working directory and get to the specified path; optional. [Path]
`nocache`	Flag to disable the cache when set; optional, needed if `SOSHOME` is set as an environment variable. [Boolean]
`nocompress`	Flag that disables compression when set; optional. [Boolean]
`password`	SourceSafe password; optional. [String]
`projectpath`	SourceSafe project path without the `$` prefix; required. [String]

`recursive`	Flag to recursively apply the action (not valid on all SOS tasks); optional, default false. [Boolean]
`soscmd`	Directory where soscmd is located; optional, soscmd must be on the path if omitted. [String]
`soshome`	The path to the SourceOffSite home directory. [String]
`sosserverpath`	Address and port of SourceOffSite Server, e.g. [String]
`username`	SourceSafe username; required. [String]
`verbose`	Enable verbose output; optional, default false. [Boolean]
`version`	A version number to get—only works with the SOSGet on a file; optional. [String]
`vssserverpath`	Path to the location of the ss.ini file; required. [String]

<soslabel> Labels Visual SourceSafe files via a SourceOffSite server.

`comment`	Comment to apply to all files being labelled; optional, only valid in SOSLabel. [String]
`file`	File name to act upon; optional. [String]
`label`	Labeled version to operate on in SourceSafe. [String]
`localpath`	Override the working directory and get to the specified path; optional. [Path]
`nocache`	Flag to disable the cache when set; optional, needed if `SOSHOME` is set as an environment variable. [Boolean]
`nocompress`	Flag that disables compression when set; optional, default. [Boolean]
`password`	SourceSafe password; optional. [String]
`projectpath`	SourceSafe project path without the `$` prefix; required. [String]
`recursive`	Flag to recursively apply the action (not valid on all SOS tasks); optional, default false. [Boolean]
`soscmd`	Directory where soscmd is located; optional, soscmd must be on the path if omitted. [String]
`soshome`	The path to the SourceOffSite home directory. [String]
`sosserverpath`	Address and port of SourceOffSite Server, e.g. [String]
`username`	SourceSafe username; required. [String]
`verbose`	Enable verbose output; optional, default false. [Boolean]
`version`	A version number to get—only works with the SOSGet on a file; optional. [String]
`vssserverpath`	Path to the location of the ss.ini file; required. [String]

<sound> Plays a sound file at the end of the build, according to whether the build failed or succeeded.

`<fail>`	Adds a sound when the build fails.
`<success>`	Adds a sound when the build succeeds.

<splash> Creates a splash screen.

`imageurl`	A URL pointing to an image to display; optional, default antlogo.gif from the classpath. [String]
`password`	Proxy password; required if `user` is set. [String]
`port`	Proxy port; optional, default 80. [String]
`proxy`	Name of proxy; optional. [String]
`showduration`	How long to show the splash screen in milliseconds, optional; default 5000 ms. [Integer]
`user`	Proxy user; optional, default=none. [String]

<sql> Executes a series of SQL statements on a database using JDBC. SQL commands, may optionally be nested as text data.

`append`	Sets whether output should be appended to or overwrite an existing file. [Boolean]
`autocommit`	Auto commit flag for database connection; optional, default false. [Boolean]
`caching`	Caching loaders/driver. [Boolean]
`classpath`	Classpath for loading the driver. [Path]
`classpathref`	Classpath for loading the driver using the classpath reference. [Reference]
`delimiter`	Delimiter that separates SQL statements; optional, default ";". [String]
`delimitertype`	Delimiter type: `normal` or `row` (default `normal`). [`normal`, `row`]
`driver`	Class name of the JDBC driver; required. [String]
`encoding`	File encoding to use on the SQL files read in. [String]
`onerror`	Action to perform when statement fails; default is `abort`. [`continue`, `stop`, `abort`]
`output`	Output file; optional, defaults to the Ant log. [File]
`password`	Password; required. [String]
`print`	Print result sets from the statements; optional, default false. [Boolean]
`rdbms`	Execute task only if the lowercase product name of the DB matches this. [String]
`showheaders`	Print headers for result sets from the statements; optional, default true. [Boolean]
`src`	Name of the SQL file to be run. [File]
`url`	Database connection URL; required. [String]
`userid`	User name for the connection; required. [String]
`version`	Version string, execute task only if rdbms version matches; optional. [String]
`<classpath>`	Adds a path to the classpath for loading the driver. [Path]
`<fileset>`	Adds a set of files (nested fileset attribute). [Fileset]
`<transaction>`	Adds an SQL transaction to execute.

<*stcheckin*> Checks files into a StarTeam project.

`adduncontrolled`	If true, any files or folders NOT in StarTeam will be added to the repository. [Boolean]
`comment`	Optional `checkin` comment to be saved with the file. [String]
`createfolders`	Value of `createFolders`. [Boolean]
`excludes`	Declare files to exclude. [String]
`forced`	Flag to force actions regardless of the status that StarTeam is maintaining for the file; optional, default false. [Boolean]
`includes`	Declare files to include. [String]
`password`	Password to be used for login; required. [String]
`projectname`	Name of the StarTeam project to be acted on; required if `url` is not set. [String]
`recursive`	Flag to set to include files in subfolders in the operation; optional, default true. [Boolean]
`rootlocalfolder`	Local folder that will be the root of the tree to which files are checked out; optional. [String]
`rootstarteam-folder`	Root of the subtree in the StarTeam repository from which to work; optional. [String]
`servername`	Name of StarTeamServer; required if `url` is not set. [String]
`serverport`	Port number of the StarTeam connection; required if `url` is not set. [String]
`unlocked`	Set to do an unlocked checkout; optional, default is false; If true, file will be unlocked so that other users may change it. [Boolean]
`url`	Server name, server port, project name and project folder in one shot; optional, but the server connection must be specified somehow. [String]
`username`	Name of the StarTeam user, needed for the connection. [String]
`viewname`	Name of the StarTeam view to be acted on; required if `url` is not set. [String]

<*stcheckout*> Checks out files from a StarTeam project.

`create-workingdirs`	Flag (defaults to true) to create all directories that are in the Starteam repository even if they are empty. [Boolean]
`delete-uncontrolled`	Should all local files not in StarTeam be deleted? Optional, defaults to `true`. [Boolean]
`excludes`	Declare files to exclude. [String]
`forced`	Flag to force actions regardless of the status that StarTeam is maintaining for the file; optional, default false. [Boolean]
`includes`	Declare files to include. [String]
`label`	Label StarTeam is to use for checkout; defaults to the most recent file. [String]
`locked`	Set to do a locked checkout; optional default is false. [Boolean]
`password`	Password to be used for login; required. [String]
`projectname`	Name of the StarTeam project to be acted on; required if `url` is not set. [String]
`recursive`	Flag to set to include files in subfolders in the operation; optional, default true. [Boolean]

`rootlocalfolder`	Local folder that will be the root of the tree to which files are checked out; optional. [String]
`rootstarteam-folder`	Root of the subtree in the StarTeam repository from which to work; optional. [String]
`servername`	Name of StarTeamServer; required if `url` is not set. [String]
`serverport`	Port number of the StarTeam connection; required if `url` is not set. [String]
`unlocked`	Set to do an unlocked checkout. [Boolean]
`url`	Server name, server port, project name, and project folder in one shot; optional, but the server connection must be specified somehow. [String]
`username`	Name of the StarTeam user, needed for the connection. [String]
`viewname`	Name of the StarTeam view to be acted on; required if `url` is not set. [String]

<*stlabel*> Creates a view label in StarTeam at the specified view.

`description`	Optional description of the label to be stored in the StarTeam project. [String]
`label`	The name to be given to the label; required. [String]
`lastbuild`	The timestamp of the build that will be stored with the label; required. [String]
`password`	Password to be used for login; required. [String]
`projectname`	Name of the StarTeam project to be acted on; required if `url` is not set. [String]
`servername`	Name of StarTeamServer; required if `url` is not set. [String]
`serverport`	Port number of the StarTeam connection; required if `url` is not set. [String]
`url`	Server name, server port, project name and project folder in one shot; optional, but the server connection must be specified somehow. [String]
`username`	Name of the StarTeam user, needed for the connection. [String]
`viewname`	Name of the StarTeam view to be acted on; required if `url` is not set. [String]

<*stlist*> Produces a listing of the contents of the StarTeam repository at the specified view and StarTeamFolder.

`excludes`	Declare files to exclude. [String]
`forced`	Flag to force actions regardless of the status that StarTeam is maintaining for the file; optional, default false. [Boolean]
`includes`	Declare files to include. [String]
`label`	List files, dates, and statuses as of this label; optional. [String]
`password`	Password to be used for login; required. [String]
`projectname`	Name of the StarTeam project to be acted on; required if `url` is not set. [String]
`recursive`	Flag to set to include files in subfolders in the operation; optional, default true. [Boolean]

`rootlocalfolder`	Local folder that will be the root of the tree to which files are checked out; optional. [String]
`rootstarteam-folder`	Root of the subtree in the StarTeam repository from which to work; optional. [String]
`servername`	Name of StarTeamServer; required if `url` is not set. [String]
`serverport`	Port number of the StarTeam connection; required if `url` is not set. [String]
`url`	Server name, server port, project name, and project folder in one shot; optional, but the server connection must be specified somehow. [String]
`username`	Name of the StarTeam user, needed for the connection. [String]
`viewname`	Name of the StarTeam view to be acted on; required if `url` is not set. [String]

<style>

See <xslt>.

<stylebook>

Executes the Apache Stylebook documentation generator.

`append`	If true, append output to existing file. [Boolean]
`book`	The book xml file that the documentation generation starts from; required. [File]
`classname`	Java class to execute. [String]
`classpath`	Classpath to be used when running the Java class. [Path]
`classpathref`	Classpath to use, by reference. [Reference]
`dir`	The working directory of the process. [File]
`failonerror`	If true, then fail if the command exits with a returncode other than 0. [Boolean]
`fork`	If true, execute in a new VM. [Boolean]
`jar`	The location of the JAR file to execute. [File]
`jvm`	Command used to start the VM (only if not forking). [String]
`jvmargs`	Command-line arguments for the JVM. [String]
`jvmversion`	JVM version. [String]
`loaderconfig`	A loader configuration to send to stylebook; optional. [String]
`maxmemory`	Corresponds to `-mx` or `-Xmx` depending on VM version. [String]
`newenvironment`	If true, use a completely new environment. [Boolean]
`output`	File the output of the process is redirected to. [File]
`skindirectory`	The directory that contains the stylebook skin; required. [File]
`targetdirectory`	The destination directory where the documentation is generated; required. [File]
`timeout`	Timeout in milliseconds after which the process will be killed. [Long]
`<arg>`	Adds a command-line argument.
`<classpath>`	Adds a path to the classpath. [Path]
`<env>`	Adds an environment variable.
`<jvmarg>`	Adds a JVM argument.
`<sysproperty>`	Adds a system property.

<*tar*> *

Creates a tar archive.

`basedir`	This is the base directory to look in for things to tar. [File]
`compression`	Set compression method. [`none`, `gzip`, `bzip2`]
`destfile`	Set is the name/location of where to create the tar file. [File]
`longfile`	Set how to handle long files, those with a path>100 chars. [`warn`, `fail`, `truncate`, `gnu`, `omit`]
`<tarfileset>`	Adds a new fileset with the option to specify permissions.

<*taskdef*>

Adds a task definition to the current project, such that this new task can be used in the current project.

`classname`	The full class name of the object being defined. [String]
`classpath`	Classpath to be used when searching for component being defined. [Path]
`classpathref`	Reference to a classpath to use when loading the files. [Reference]
`file`	Name of the property file to load Ant name/classname pairs from. [File]
`loaderref`	Use the reference to locate the loader. [Reference]
`name`	Name of the property resource to load Ant name/classname pairs from. [String]
`resource`	Name of the property resource to load Ant name/classname pairs from. [String]
`<classpath>`	Creates the classpath to be used when searching for component being defined. [Path]

<*telnet*>

Task to automate a telnet session or other TCP connection to a server.

`initialcr`	Send a carriage return after connecting; optional, defaults to false. [Boolean]
`password`	The login password to use; required if `userid` is set. [String]
`port`	TCP port to connect to; default is 23. [Integer]
`server`	Hostname or address of the remote server. [String]
`timeout`	Default timeout in seconds to wait for a response, zero means forever (the default). [Integer]
`userid`	The login ID to use on the server; required if `password` is set. [String]
`<read>`	A string to wait for from the server.
`<write>`	Adds text to send to the server.

<*tempfile*>

This task sets a property to the name of a temporary file.

`destdir`	Destination directory. [File]
`prefix`	Optional prefix string. [String]
`property`	The property you wish to assign the temporary file to. [String]
`suffix`	Suffix string for the temp file (optional). [String]

<touch> Touches a file and/or fileset(s); corresponds to the Unix touch command.

`datetime`	The new modification time of the file in the format `MM/DD/YYYY HH:MM AM` or `PM`; optional, default=now. [String]
`file`	Single source file to touch. [File]
`millis`	The new modification time of the file in milliseconds since midnight Jan 1, 1970. [Long]
`<fileset>`	Adds a set of files to touch. [Fileset]

<translate> * Translates text embedded in files using Resource Bundle files.

`bundle`	Sets family name of resource bundle; required. [String]
`bundlecountry`	Sets locale-specific country of resource bundle; optional. [String]
`bundleencoding`	Sets Resource Bundle file encoding scheme; optional. [String]
`bundlelanguage`	Sets locale-specific language of resource bundle; optional. [String]
`bundlevariant`	Sets locale-specific variant of resource bundle; optional. [String]
`destencoding`	Sets destination file encoding scheme; optional. [String]
`endtoken`	Sets ending token to identify keys; required. [String]
`forceoverwrite`	Sets whether to overwrite existing file irrespective of whether it is newer than the source file as well as the resource bundle file. [Boolean]
`srcencoding`	Sets source file encoding scheme; optional, defaults to encoding of local system. [String]
`starttoken`	Sets starting token to identify keys; required. [String]
`todir`	Sets destination directory; required. [File]
`<fileset>`	Adds a set of files to translate as a nested fileset element. [Fileset]

<tstamp> Sets properties to the current time, or offsets from the current time.

`prefix`	Prefix for the properties. [String]
`<format>`	Creates a custom format with the current prefix.

<typedef> Adds a data type definition to the current project.

`classname`	The full class name of the object being defined. [String]
`classpath`	Classpath to be used when searching for component being defined. [Path]
`classpathref`	Reference to a classpath to use when loading the files. [Reference]
`file`	Name of the property file to load Ant name/classname pairs from. [File]
`loaderref`	Use the reference to locate the loader. [Reference]
`name`	Name of the property file to load Ant name/classname pairs from. [String]
`resource`	Name of the property resource to load Ant name/classname pairs from. [String]
`<classpath>`	Creates the classpath to be used when searching for component being defined. [Path]

<unjar> See `<unzip>`.

<untar> Untars a file.

`compression`	Set decompression algorithm to use; default=none. [`none`, `gzip`, `bzip2`]
`dest`	Destination directory. [File]
`overwrite`	If true, overwrite files in `dest`, even if they are newer than the corresponding entries in the archive. [Boolean]
`src`	Path to tar file. [File]
`<fileset>`	Adds a fileset. [Fileset]
`<patternset>`	Adds a patternset. [Patternset]

<unwar> See `<unzip>`.

<unzip> Unzip a file.

`dest`	Destination directory. [File]
`overwrite`	Should the task overwrite files in `dest`, even if they are newer than the corresponding entries in the archive? [Boolean]
`src`	Path to Zip file. [File]
`<fileset>`	Adds a fileset. [Fileset]
`<patternset>`	Adds a patternset. [Patternset]

<uptodate> Sets the given property if the specified target has a timestamp greater than all of the source files.

`property`	The property to set if the target file is more up-to-date than (each of) the source file(s). [String]
`srcfile`	The file that must be older than the target file if the property is to be set. [File]
`targetfile`	The file which must be more up-to-date than (each of) the source file(s) if the property is to be set. [File]
`value`	The value to set the named property to if the target file is more up-to-date than (each of) the source files. [String]
`<mapper>`	Defines source to target mapping. [Mapper]
`<srcfiles>`	Adds fileset to the source files. [Fileset]

<vajexport> Exports packages from the Visual Age for Java workspace.

`defaultexcludes`	Sets whether default exclusions should be used; default true. [Boolean]
`destdir`	Destination directory into which the selected items should be exported; required. [File]
`excludes`	Set of exclude patterns. [String]
`exportclasses`	Optional flag to export the class files; default false. [Boolean]
`exportdebuginfo`	Optional flag to export the debug info; default false. [Boolean]
`exportresources`	Optional flag to export the resource file; default true. [Boolean]

`exportsources`	Optional flag to export the Java files; default true. [Boolean]
`includes`	Set of include patterns. [String]
`overwrite`	If true, files will be overwritten during export. [Boolean]
`remote`	Name and port of a remote tool server. [String]
`<exclude>`	Adds a name entry on the exclude list.
`<include>`	Adds a name entry on the include list.

<vajimport> Imports source, class files, and resources to the Visual Age for Java workspace.

`defaultexcludes`	Sets whether default exclusions should be used. [Boolean]
`importclasses`	Flag to import .class files; optional, default false. [Boolean]
`importresources`	Imports resource files (anything that doesn't end in .class or .java); optional, default true. [Boolean]
`importsources`	Imports .java files; optional, default true. [Boolean]
`project`	The VisualAge for Java Project name to import into. [String]
`remote`	Name and port of a remote tool server [String]
`<fileset>`	Adds a set of files (nested `fileset` attribute). [Fileset]

<vajload> Loads specific project versions into the Visual Age for Java workspace.

`remote`	Name and port of a remote tool server, optional. [String]
`<vajproject>`	Adds a project description entry on the project list.

<vssadd> Adds files to a Microsoft Visual SourceSafe repository.

`autoresponse`	What to respond with (sets the `-I` option). [String]
`comment`	Comment to apply; optional. [String]
`localpath`	Local path. [Path]
`login`	The login to use when accessing VSS, formatted as `username, password`; optional. [String]
`recursive`	Sets behavior to recursive or nonrecursive. [Boolean]
`serverpath`	Directory where srssafe.ini resides; optional. [String]
`ssdir`	Directory where ss.exe resides; optional. [String]
`vsspath`	SourceSafe path that specifies the project/file(s) you wish to perform the action on; required. [String]
`writable`	Leave added files writable? Default: false. [Boolean]

<vsscheckin> Checks in files to a Microsoft Visual SourceSafe repository.

`autoresponse`	What to respond with (sets the `-I` option). [String]
`comment`	Comment to apply; optional. [String]
`localpath`	Local path. [Path]
`login`	The login to use when accessing VSS, formatted as `username,password`; optional. [String]
`recursive`	Flag to tell the task to recurse down the tree; optional, default false. [Boolean]

`serverpath`	Directory where srssafe.ini resides; optional. [String]
`ssdir`	Directory where ss.exe resides; optional. [String]
`vsspath`	SourceSafe path that specifies the project/file(s) you wish to perform the action on; required. [String]
`writable`	Leave checked in files writable? Default: false. [Boolean]

<vsscheckout> Checks out files from a Microsoft Visual SourceSafe repository.

`autoresponse`	What to respond with (sets the `-I` option). [String]
`date`	Date to get. [String]
`label`	Label to get. [String]
`localpath`	Local path. [Path]
`login`	The login to use when accessing VSS, formatted as `username,password`; optional. [String]
`recursive`	Flag to tell the task to recurse down the tree; optional, default false. [Boolean]
`serverpath`	Directory where srssafe.ini resides; optional. [String]
`ssdir`	Directory where ss.exe resides; optional. [String]
`version`	Version to get; optional. [String]
`vsspath`	SourceSafe path which specifies the project/files you wish to perform the action on; required. [String]

<vsscp> Performs CP (Change Project) commands on a Microsoft Visual SourceSafe repository.

`autoresponse`	What to respond with (sets the `-I` option). [String]
`login`	The login to use when accessing VSS, formatted as `username,password`; optional. [String]
`serverpath`	Directory where srssafe.ini resides; optional. [String]
`ssdir`	Directory where ss.exe resides; optional. [String]
`vsspath`	SourceSafe path that specifies the project/files you wish to perform the action on; required. [String]

<vsscreate> Creates a new project in a Microsoft Visual SourceSafe repository.

`autoresponse`	What to respond with (sets the `-I` option). [String]
`comment`	Comment to apply in SourceSafe. [String]
`failonerror`	Sets whether task should fail if there is an error creating the project; optional, default true. [Boolean]
`login`	The login to use when accessing VSS, formatted as `username,password`; optional. [String]
`quiet`	Sets/clears quiet mode; optional, default false. [Boolean]
`serverpath`	Directory where srssafe.ini resides; optional. [String]
`ssdir`	Directory where ss.exe resides; optional. [String]
`vsspath`	SourceSafe path that specifies the project/files you wish to perform the action on; required. [String]

<*vssget*> Gets files from a Microsoft Visual SourceSafe repository.

`autoresponse`	What to respond with (sets the `-I` option). [String]
`date`	Date to get; optional. [String]
`label`	Label to get; optional. [String]
`localpath`	Overrides the working directory to get to the specified path; optional. [Path]
`login`	The login to use when accessing VSS, formatted as `username,password`; optional. [String]
`quiet`	Flag to suppress output when true; false by default. [Boolean]
`recursive`	Flag to tell the task to recurse down the tree; optional, default false. [Boolean]
`serverpath`	Directory where srssafe.ini resides; optional. [String]
`ssdir`	Directory where ss.exe resides; optional. [String]
`version`	Version number to get; optional. [String]
`vsspath`	SourceSafe path that specifies the project/files you wish to perform the action on; required. [String]
`writable`	Makes fetched files writable; optional, default false. [Boolean]

<*vsshistory*> Gets a change history from a Microsoft Visual SourceSafe repository.

`dateformat`	Format of dates in `fromdate` and `todate`; optional. [String]
`fromdate`	Start date for the comparison of two versions; optional. [String]
`fromlabel`	Start label; optional. [String]
`login`	The login to use when accessing VSS, formatted as `username,password`; optional. [String]
`numdays`	Number of days for comparison; optional. [Integer]
`output`	Output file name for the history; optional. [File]
`recursive`	Flag to tell the task to recurse down the tree; optional, default false. [Boolean]
`serverpath`	Directory where srssafe.ini resides; optional. [String]
`ssdir`	Directory where ss.exe resides; optional. [String]
`style`	Specify the output style; optional. [`brief`, `codediff`, `nofile`, `default`]
`todate`	End date for the comparison of two versions; optional. [String]
`tolabel`	End label; optional. [String]
`user`	Name the user whose changes we would like to see; optional. [String]
`vsspath`	SourceSafe path that specifies the project/files you wish to perform the action on; required. [String]

<*vsslabel*> Labels files in a Microsoft Visual SourceSafe repository.

`autoresponse`	What to respond with (sets the `-I` option). [String]
`comment`	The comment to use for this label; optional. [String]
`label`	Label to apply; required. [String]
`login`	The login to use when accessing VSS, formatted as `username,password`; optional. [String]

`serverpath`	Directory where srssafe.ini resides; optional. [String]
`ssdir`	Directory where ss.exe resides; optional. [String]
`version`	Name of an existing file or project version to label; optional. [String]
`vsspath`	SourceSafe path that specifies the project/files you wish to perform the action on; required. [String]

<*waitfor*> Waits for a nested condition to become valid.

`checkevery`	Time between each check. [Long]
`checkeveryunit`	Check every time unit. [`millisecond`, `second`, `minute`, `hour`, `day`, `week`]
`maxwait`	Maximum length of time to wait. [Long]
`maxwaitunit`	Max wait time unit. [`millisecond`, `second`, `minute`, `hour`, `day`, `week`]
`timeoutproperty`	Name of the property to set after a timeout. [String]
`<and>`	True if all nested conditions evaluate to true.
`<available>`	Identical to the `<available>` task.
`<checksum>`	Identical to the `<checksum>` task.
`<contains>`	Tests whether one string contains another.
`<equals>`	Tests whether two strings are equal.
`<filesmatch>`	Tests that two files match, byte for byte.
`<http>`	Checks for a valid response from a web server of a specified URL.
`<isfalse>`	Tests whether a string value is not `<istrue>`.
`<isset>`	Tests whether a property has been set.
`<istrue>`	Tests whether a string evaluates to "true", "on", or "yes".
`<not>`	Negates results of single nested condition.
`<or>`	True if one nested condition is true.
`<os>`	Tests whether the current operating system is of a given type.
`<socket>`	Checks for the existence of a TCP/IP listener at the specified host and port.
`<uptodate>`	Identical to the `<uptodate>` task.

<*war*> * An extension of `<jar>` to create a WAR archive.

`basedir`	Directory from which to archive files; optional. [File]
`compress`	Sets whether to compress the files or only store them; optional, default=true;. [Boolean]
`destfile`	The file to create; required. [File]
`duplicate`	Sets behavior for when a duplicate file is about to be added. [`add`, `preserve`, `fail`]
`encoding`	Encoding to use for file names, defaults to the platform's default encoding. [String]
`filesonly`	If true, emulates Sun's JAR utility by not adding parent directories; optional, defaults to false. [Boolean]
`index`	Sets whether to create an index list for classes. [Boolean]

`manifest`	The manifest file to use. [File]
`update`	If true, updates an existing file, otherwise overwrites any existing one; optional, defaults to false. [Boolean]
`webxml`	Deployment descriptor to use (WEB-INF/web.xml); required unless `update` is true. [File]
`whenempty`	Sets behavior of the task when no files match. [`fail`, `skip`, `create`]
`<classes>`	Adds files under WEB-INF/classes. [ZipFileset]
`<fileset>`	Adds a set of files. [Fileset]
`<lib>`	Adds files under WEB-INF/lib/. [ZipFileset]
`<manifest>`	Allows the manifest for the archive file to be provided inline in the build file rather than in an external file.
`<metainf>`	Adds a zipfileset to include in the META-INF directory. [ZipFileset]
`<webinf>`	Files to add under WEB-INF. [ZipFileset]
`<zipfileset>`	Adds a set of files that can be read from an archive and be given a prefix/fullpath. [ZipFileset]
`<zipgroup-fileset>`	Adds a group of Zip files. [Fileset]

<wljspc> * Precompiles JSPs using WebLogic's JSP compiler (weblogic.jspc).

`classpath`	Classpath to be used for this compilation. [Path]
`dest`	Directory containing the source JSPs. [File]
`package`	Package under which the compiled classes go. [String]
`src`	Directory containing the source JSPs. [File]
`<classpath>`	Adds a path to the classpath. [Path]

<wlrun> Starts a WebLogic server.

`args`	Additional argument string passed to the WebLogic instance; optional. [String]
`beahome`	The location of the BEA Home; implicitly selects WebLogic 6.0; optional. [File]
`classpath`	The classpath to be used with the Java Virtual Machine that runs the WebLogic Server; required. [Path]
`domain`	Domain to run in; required for WL6.0. [String]
`home`	The location where WebLogic lives. [File]
`jvmargs`	Additional arguments to pass to the WebLogic JVM. [String]
`name`	The name of the WebLogic server within the WebLogic home that is to be run. [String]
`password`	Management password of the server; optional and only applicable to WL6.0. [String]
`pkpassword`	Private key password so the server can decrypt the SSL private key file; optional and only applicable to WL6.0. [String]
`policy`	The name of the security policy file within the WebLogic home directory that is to be used. [String]
`properties`	The name of the server's properties file within the WebLogic home directory used to control the WebLogic instance; required for WL4.5.1. [String]

`username`	Management username to run the server; optional and only applicable to WL6.0. [String]
`weblogicmain-class`	Name of the main class for WebLogic; optional. [String]
`wlclasspath`	WebLogic classpath used by the WebLogic server; optional, and only applicable to WL4.5.1. The WebLogic classpath is used by WebLogic to support dynamic class loading. [Path]
`<classpath>`	Adds the classpath for the user classes. [Path]
`<wlclasspath>`	Gets the classpath to the WebLogic classpaths. [Path]

<wlstop> Shuts down a WebLogic server.

`beahome`	The location of the BEA Home; implicitly selects WebLogic 6.0 shutdown; optional. [File]
`classpath`	The classpath to be used with the Java Virtual Machine that runs the WebLogic Shutdown command;. [Path]
`delay`	Delay (in seconds) before shutting down the server; optional. [String]
`password`	The password for the account specified in the user parameter; required. [String]
`url`	URL to which the WebLogic server is listening for T3 connections; required. [String]
`user`	The username of the account that will be used to shut down the server; required. [String]
`<classpath>`	The classpath to be used with the Java Virtual Machine that runs the WebLogic Shutdown command. [Path]

<wsdltodotnet> Converts a WSDL file or URL resource into a .NET language.

`destfile`	Name of the file to generate. [File]
`extraoptions`	Any extra WSDL.EXE options that aren't explicitly supported by the Ant wrapper task; optional. [String]
`failonerror`	Should failure halt the build? Optional, default=true. [Boolean]
`language`	Language; default is CS, generating C# source. [`CS`, `JS`, or `VB`]
`namespace`	Namespace to place the source in. [String]
`server`	Flag to enable server-side code generation; optional, default=false. [Boolean]
`srcfile`	The local WSDL file to parse; either `url` or `srcfile` is required. [File]
`url`	URL to fetch. [String]

<xmlproperty> Loads property values from a valid XML file, generating the property names from the file's element and attribute names.

`collapseat-tributes`	Flag to treat attributes as nested elements; optional, default false. [Boolean]
`file`	The XML file to parse; required. [File]
`keeproot`	Flag to include the XML root tag as a first value in the property name; optional, default is true. [Boolean]
`prefix`	The prefix to prepend to each property. [String]
`validate`	Flag to validate the XML file; optional, default false. [Boolean]

<xmlvalidate> Checks whether XML files are valid (or only well formed).

`classname`	Specify the class name of the SAX parser to be used. [String]
`classpath`	Specify the classpath to be searched to load the parser (optional). [Path]
`classpathref`	Where to find the parser class; optional. [Reference]
`failonerror`	Specify how parser errors are to be handled; optional, default is true. [Boolean]
`file`	Specify the file to be checked; optional. [File]
`lenient`	Specify whether the parser should be validating. [Boolean]
`warn`	Specify how parser error are to be handled. [Boolean]
`<classpath>`	No description. [Path]
`<dtd>`	Creates a DTD location record; optional.
`<fileset>`	Specifies a set of files to be checked. [Fileset]
`<xmlcatalog>`	Adds an XMLCatalog as a nested element; optional. [XMLCatalog]

<xslt> * Processes a set of XML documents via XSLT.

`basedir`	Base directory; optional, default is the project's basedir. [File]
`classpath`	Optional classpath to the XSL processor. [Path]
`classpathref`	Reference to an optional classpath to the XSL processor. [Reference]
`destdir`	Destination directory into which the XSL result files should be copied to; required, unless `in` and `out` are specified. [File]
`extension`	Desired file extension to be used for the target; optional, default is html. [String]
`force`	Sets whether to check dependencies, or always generate; optional, default is false. [Boolean]
`in`	Specifies a single XML document to be styled. [File]
`out`	Specifies the output name for the styled result from the `in` attribute; required if `in` is set. [File]
`processor`	Name of the XSL processor to use; optional, default is trax. [String]
`scanincluded-directories`	Sets whether to style all files in the included directories as well; optional, default is true. [Boolean]
`style`	Name of the stylesheet to use—given either relative to the project's basedir or as an absolute path; required. [String]
`<classpath>`	Optional classpath to the XSL processor. [Path]
`<outputproperty>`	Specifies how you wish the result tree to be output.
`<param>`	Creates an instance of an XSL parameter.
`<xmlcatalog>`	Adds the catalog to our internal catalog. [XMLCatalog]

<zip> * Creates a Zip file.

`basedir`	Directory from which to archive files; optional. [File]
`compress`	Sets whether to compress the files or only store them; optional, default=true;. [Boolean]
`destfile`	The file to create; required. [File]
`duplicate`	Sets behavior for when a duplicate file is about to be added. [`add`, `preserve`, `fail`]
`encoding`	Encoding to use for file names, defaults to the platform's default encoding. [String]
`filesonly`	If true, emulates Sun's JAR utility by not adding parent directories; optional, defaults to false. [Boolean]
`update`	If true, updates an existing file, otherwise overwrites any existing one; optional, defaults to false. [Boolean]
`whenempty`	Sets behavior of the task when no files match. [`fail`, `skip`, `create`]
`<fileset>`	Adds a set of files. [Fileset]
`<zipfileset>`	Adds a set of files that can be read from an archive and be given a prefix/fullpath. [ZipFileset]
`<zipgroup-fileset>`	Adds a group of Zip files. [Fileset]