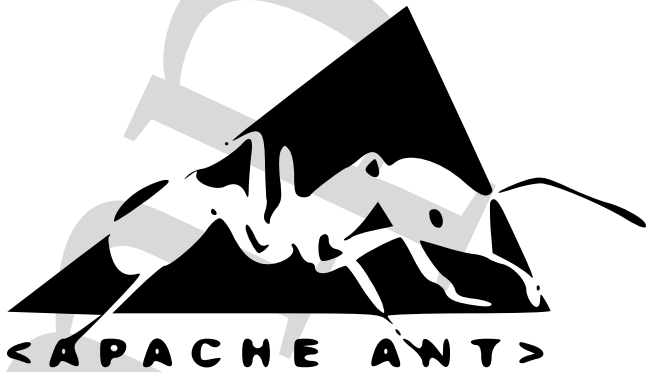

ANTDOCLET EXAMPLE

Fernando Dobladez
fernando@dobladez.com

December 8, 2005

Abstract

This document serves as reference documentation for the core Apache Ant tasks.

This is NOT the official Apache Ant[1] documentation. It is exposed here just as an example of what AntDoclet[2] can generate.

Contents

[1] http://ant.apache.org
[2] http://antdoclet.neuroning.com

1 Introduction

This documentation was generated running AntDoclet[3] over the source code of the core Apache Ant Tasks version 1.6.5.

This NOT the official Apache Ant[4] documentation. It is exposed here just as an example of what AntDoclet[5] can generate.

1.1 About this example

The source code of the core Ant Tasks was written without any knowledge of AntDoclet. Therefore, the quality of the generated documentation for this example is far from perfect.

For instance, the source code of the core Ant Tasks do not use all the tags that AntDoclet supports, and some of the comments use invalid HTML or describe things that AntDoclet already auto-generates.

To get the most out of AntDoclet the source code of your Ant Tasks needs to follow a few simple guidelines described in the AntDoclet documentation.

The following sections describe each of the provided Ant Tasks in more detail, including all the attributes they accept and examples of use.

[3] http://antdoclet.neuroning.com
[4] http://ant.apache.org
[5] http://antdoclet.neuroning.com

2 control Library

2.1 Task control:Ant

Build a sub-project.

```
<target name="foo" depends="init">
  <ant antfile="build.xml" target="bar" >
    <property name="property1" value="aaaaa" />
    <property name="foo" value="baz" />
  </ant>
</target>

<target name="bar" depends="init">
  <echo message="prop is ${property1} ${foo}" />
</target>
```

Parameters:

- **`output`** – String Set the filename to write the output to. This is relative to the value of the dir attribute if it has been set or to the base directory of the current project otherwise.
- **`inheritall`** – boolean If true, pass all properties to the new Ant project. Defaults to true.
- **`dir`** – File The directory to use as a base directory for the new Ant project. Defaults to the current project's basedir, unless inheritall has been set to false, in which case it doesn't have a default value. This will override the basedir setting of the called project.
- **`inheritrefs`** – boolean If true, pass all references to the new Ant project. Defaults to false.
- **`target`** – String The target of the new Ant project to execute. Defaults to the new project's default target.
- **`antfile`** – String The build file to use. Defaults to "build.xml". This file is expected to be a filename relative to the dir attribute given.

Parameters accepted as nested elements:

`<reference>` Helper class that implements the nested <reference>element of <ant>and <antcall>.

Parameters:

- **torefid** – String
- **refid** – String

<propertyset> – *(See ?? on page ??)* A set of properties.

Parameters:

- **refid** – Reference
- **dynamic** – boolean
- **negate** – boolean

<property> – *(See ?? on page ??)* Sets a property by name, or set of properties (from file or resource) in the project. Properties are immutable: whoever sets a property first freezes it for the rest of the build; they are most definitely not variable.

There are five ways to set properties:

- By supplying both the *name* and *value* attribute.
- By supplying both the *name* and *refid* attribute.
- By setting the *file* attribute with the filename of the property file to load. This property file has the format as defined by the file used in the class java.util.Properties.
- By setting the *resource* attribute with the resource name of the property file to load. This property file has the format as defined by the file used in the class java.util.Properties.
- By setting the *environment* attribute with a prefix to use. Properties will be defined for every environment variable by prefixing the supplied name and a period to the name of the variable.

Although combinations of these ways are possible, only one should be used at a time. Problems might occur with the order in which properties are set, for instance.

The value part of the properties being set, might contain references to other properties. These references are resolved at the time these properties are set. This also holds for properties loaded from a property file.

Properties are case sensitive.

Parameters:

- **refid** – Reference
- **url** – URL
- **name** – String
- **classpath** – Path
- **userproperty** – boolean
- **file** – File

- **resource** – String
- **environment** – String
- **prefix** – String
- **classpathref** – Reference
- **value** – String

<target> Helper class that implements the nested <target>element of <ant>and <antcall>.

Parameters:

- **name** – String

2.2 Task control:Available

Will set the given property if the requested resource is available at runtime. This task may also be used as a condition by the condition task.

Parameters:

- **classpath** – Path Set the classpath to be used when searching for classes and resources.
- **file** – File Set the file which must be present in the file system to set the given property.
- **classname** – String Set a classname of a class which must be available to set the given property.
- **resource** – String Set the name of a Java resource which is required to set the property.
- **ignoresystemclasses** – boolean Set whether the search for classes should ignore the runtime classes and just use the given classpath.
- **type** – String [”file”, ”dir”]Set what type of file is required - either directory or file.
- **classpathref** – Reference Set the classpath by reference.
- **value** – String Set the value to be given to the property if the desired resource is available.

- **property** – String Set the name of the property which will be set if the particular resource is available.
- **filepath** – Path Set the path to use when looking for a file.

Parameters accepted as nested elements:

<classpath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<filepath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

2.3 Task control:Checksum

Used to create or verify file checksums.

Parameters:

- **`excludes`** – String
- **`fileext`** – String Sets the file extension that is be to used to create or identify destination file.
- **`totalproperty`** – String Sets the property to hold the generated total checksum for all files.
- **`property`** – String Sets the property to hold the generated checksum.
- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean
- **`casesensitive`** – boolean
- **`readbuffersize`** – int The size of the read buffer to use.
- **`verifyproperty`** – String Sets the verify property. This project property holds the result of a checksum verification - ”true” or ”false”
- **`includes`** – String
- **`todir`** – File Sets the root directory where checksum files will be written/read
- **`algorithm`** – String Specifies the algorithm to be used to compute the checksum. Defaults to ”MD5”. Other popular algorithms like ”SHA” may be used as well.

- **includesfile** – File
- **file** – File Sets the file for which the checksum is to be calculated.
- **excludesfile** – File
- **provider** – String Sets the MessageDigest algorithm provider to be used to calculate the checksum.
- **forceoverwrite** – boolean Whether or not to overwrite existing file irrespective of whether it is newer than the source file. Defaults to false.

Parameters accepted as nested elements:

<patternset> – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference

- **`type`** – String ["file", "dir"]

`<modified>` (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
            >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
                >
                >
```

And the same rewritten as CustomSelector would be

```
                >
                >
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns 'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
	hashvalue - loads the file content into a String and uses its hashValue() method		
	digest - uses java.security.MessageDigest class		no, defaults to digest
comparator	equal — role	which comparator implementation should be used	
	equal - simple comparison using String.equals()		
	role - uses java.text.RuleBasedCollator class		no, defaults to equal
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean

- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don’t get too caught up in boolean, there are other ways you can evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an ”all-but-one” selector, a ”weighted-average” selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference

- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String ["before", "after", "equal"]
- **error** – String

<excludesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String

- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

2.4 Type control:ConditionTask

Task to set a property conditionally using <uptodate>, <available>, and many other supported conditions.

This task supports boolean logic as well as pluggable conditions to decide, whether a property should be set.

This task does not extend Task to take advantage of ConditionBase.

Parameters:

- **value** – String The value for the property to set, if condition evaluates to true. Defaults to "true".
- **property** – String The name of the property to set. Required.
- **else** – String The value for the property to set, if condition evaluates to false. If this attribute is not specified, the property will not be set.

Parameters accepted as nested elements:

`<isfalse>` (Of type IsFalse) Condition that tests whether a given string evals to false

Parameters:

- **`value`** – boolean

`<isreference>` (Of type IsReference) Condition that tests whether a given reference has been defined.

Optionally tests whether it is of a given type/class.

Parameters:

- **`refid`** – Reference
- **`type`** – String

`<equals>` (Of type Equals) Simple String comparison condition.

Parameters:

- **`trim`** – boolean
- **`arg2`** – String
- **`arg1`** – String
- **`casesensitive`** – boolean

`<available>` – *(See Task control:Available on page 9)* Will set the given property if the requested resource is available at runtime. This task may also be used as a condition by the condition task.

Parameters:

- **`classpath`** – Path
- **`file`** – File
- **`classname`** – String
- **`resource`** – String
- **`ignoresystemclasses`** – boolean
- **`type`** – String [”file”, ”dir”]
- **`classpathref`** – Reference
- **`value`** – String
- **`property`** – String
- **`filepath`** – Path

<not> (Of type Not) <not>condition. Evaluates to true if the single condition nested into it is false and vice versa.

Parameters:

<contains> (Of type Contains) Is one string part of another string?

Parameters:

- **casesensitive** – boolean
- **string** – String
- **substring** – String

<os> (Of type Os) Condition that tests the OS type.

Parameters:

- **version** – String
- **name** – String
- **family** – String
- **arch** – String

<or> (Of type Or) <or>condition container.

Iterates over all conditions and returns true as soon as one evaluates to true.

Parameters:

<and> (Of type And) <and>condition container.

Iterates over all conditions and returns false as soon as one evaluates to false.

Parameters:

<filesmatch> (Of type FilesMatch) Compares two files for bitwise equality based on size and content. Timestamps are not at all looked at.

Parameters:

- **file2** – File
- **file1** – File

<checksum> – *(See Task control:Checksum on page 11)* Used to create or verify file checksums.

Parameters:

- **excludes** – String
- **fileext** – String
- **totalproperty** – String
- **property** – String

- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **readbuffersize** – int
- **verifyproperty** – String
- **includes** – String
- **todir** – File
- **algorithm** – String
- **includesfile** – File
- **file** – File
- **excludesfile** – File
- **provider** – String
- **forceoverwrite** – boolean

<istrue> (Of type IsTrue) Condition that tests whether a given string evals to true

Parameters:

- **value** – boolean

<isset> (Of type IsSet) Condition that tests whether a given property has been set.

Parameters:

- **property** – String

<socket> (Of type Socket) Condition to wait for a TCP/IP socket to have a listener. Its attributes are: server - the name of the server. port - the port number of the socket.

Parameters:

- **port** – int
- **server** – String

<http> (Of type Http) Condition to wait for a HTTP request to succeed. Its attribute(s) are: url - the URL of the request. errorsBeginAt - number at which errors begin at; default=400.

Parameters:

- **url** – String
- **errorsbeginat** – int

<uptodate> – *(See Task control:UpToDate on page 30)* Sets the given property if the specified target has a timestamp greater than all of the source files.

Parameters:

- **value** – String
- **property** – String
- **srcfile** – File
- **targetfile** – File

2.5 Task control:ExecTask

Executes a given command if the os platform is appropriate.

Parameters:

- **resultproperty** – String Sets the name of a property in which the return code of the command should be stored. Only of interest if failonerror=false.
- **failifexecutionfails** – boolean Set whether to stop the build if program cannot be started. Defaults to true.
- **command** – Commandline Sets a command line.
- **executable** – String Set the name of the executable program.
- **errorproperty** – String Sets the name of the property whose value should be set to the error of the process.
- **os** – String List of operating systems on which the command may be executed.
- **output** – File File the output of the process is redirected to. If error is not redirected, it too will appear in the output.
- **timeout** – Integer Set the timeout in milliseconds after which the process will be killed.
- **spawn** – boolean Set whether or not you want the process to be spawned. Default is false.
- **input** – File Set the input file to use for the task.
- **inputstring** – String Set the string to use as input.

- **searchpath** – boolean Set whether to search nested, then system PATH environment variables for the executable.
- **logerror** – boolean Controls whether error output of exec is logged. This is only useful when output is being redirected and error output is desired in the Ant log.
- **dir** – File Set the working directory of the process.
- **error** – File Set the File to which the error stream of the process should be redirected.
- **outputproperty** – String Sets the property name whose value should be set to the output of the process.
- **append** – boolean Set whether output should be appended to or overwrite an existing file. Defaults to false.
- **failonerror** – boolean Fail if the command exits with a non-zero return code.
- **resolveexecutable** – boolean Set whether to attempt to resolve the executable to a file.
- **vmlauncher** – boolean Set whether to launch new process with VM, otherwise use the OS's shell. Default value is true.
- **newenvironment** – boolean Do not propagate old environment when new environment variables are specified.

Parameters accepted as nested elements:

<arg> Used for nested xml command line definitions.

Parameters:

- **line** – String
- **file** – File
- **pathref** – Reference
- **value** – String
- **path** – Path

<redirector> – *(See ?? on page ??)* Element representation of a `Redirector`.

Parameters:

- **refid** – Reference
- **inputencoding** – String
- **append** – boolean

- **createemptyfiles** – boolean
- **output** – File
- **outputproperty** – String
- **outputencoding** – String
- **errorproperty** – String
- **error** – File
- **inputstring** – String
- **alwayslog** – boolean
- **input** – File
- **errorencoding** – String
- **logerror** – boolean

<env> representation of a single env value

Parameters:

- **key** – String
- **file** – File
- **value** – String
- **path** – Path

2.6 Task control:ImportTask

Task to import another build file into the current project.

It must be 'top level'. On execution it will read another Ant file into the same Project.

Important: we have not finalized how relative file references will be resolved in deep-/complex build hierarchies -such as what happens when an imported file imports another file. Use absolute references for enhanced build file stability, especially in the imported files. Examples

```
<import file="../common-targets.xml" />
```

Import targets from a file in a parent directory.

```
<import file="${deploy-platform}.xml" />
```

Import the project defined by the property deploy-platform

Parameters:

- **`optional`** – boolean sets the optional attribute
- **`file`** – String the name of the file to import. How relative paths are resolved is still in flux: use absolute paths for safety.

2.7 Task control:Input

Reads an input line from the console.

Parameters:

- **`addproperty`** – String Defines the name of a property to be created from input. Behaviour is according to property task which means that existing properties cannot be overridden.
- **`defaultvalue`** – String Defines the default value of the property to be created from input. Property value will be set to default if not input is received.
- **`validargs`** – String Defines valid input parameters as comma separated strings. If set, input task will reject any input not defined as accepted and requires the user to reenter it. Validargs are case sensitive. If you want 'a' and 'A' to be accepted you need to define both values as accepted arguments.
- **`message`** – String Sets the Message which gets displayed to the user during the build run.

Parameters accepted as nested elements:

This Task accepts text in its element body.

2.8 Task control:Parallel

Executes the contained tasks in separate threads, continuing once all are completed.

New behavior allows for the ant script to specify a maximum number of threads that will be executed in parallel. One should be very careful about using the `waitFor` task when specifying `threadCount` as it can cause deadlocks if the number of threads is too small or if one of the nested tasks fails to execute completely. The task selection

algorithm will insure that the tasks listed before a task have started before that task is started, but it will not insure a successful completion of those tasks or that those tasks will finish first (i.e. it's a classic race condition).

Parameters:

- **`pollinterval`** – int Interval to poll for completed threads when threadCount or threadsPerProcessor is specified. Integer in milliseconds.; optional
- **`threadsperprocessor`** – int Dynamically generates the number of threads to execute based on the number of available processors (via `java.lang.Runtime.availableProcessors()`). Requires a J2SE 1.4 VM, and it will overwrite the value set in threadCount. If used in a 1.1, 1.2, or 1.3 VM then the task will defer to `threadCount`.; optional
- **`timeout`** – long Sets the timeout on this set of tasks. If the timeout is reached before the other threads complete, the execution of this task completes with an exception. Note that existing threads continue to run.
- **`threadcount`** – int Statically determine the maximum number of tasks to execute simultaneously. If there are less tasks than threads then all will be executed at once, if there are more then only `threadCount` tasks will be executed at one time. If `threadsPerProcessor` is set and the JVM is at least a 1.4 VM then this value is ignored.; optional
- **`failonany`** – boolean Control whether a failure in a nested task halts execution. Note that the task will complete but existing threads will continue to run - they are not stopped

Parameters accepted as nested elements:

This Task is a Container (it accepts nested Tasks).

`<daemons>` Class which holds a list of tasks to execute

Parameters:

2.9 Task control:Sequential

Sequential is a container task - it can contain other Ant tasks. The nested tasks are simply executed in sequence. Sequential's primary use is to support the sequential execution of a subset of tasks within the {@link Parallel Parallel Task}

The sequential task has no attributes and does not support any nested elements apart from Ant tasks. Any valid Ant task may be embedded within the sequential task.

Parameters:

Parameters accepted as nested elements:

This Task is a Container (it accepts nested Tasks).

2.10 Task control:UpToDate

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

Parameters:

- **`value`** – String The value to set the named property to if the target file is more up-to-date than (each of) the source file(s). Defaults to 'true'.
- **`property`** – String The property to set if the target file is more up-to-date than (each of) the source file(s).
- **`srcfile`** – File The file that must be older than the target file if the property is to be set.
- **`targetfile`** – File The file which must be more up-to-date than (each of) the source file(s) if the property is to be set.

Parameters accepted as nested elements:

`<srcfiles>` – *(See* **??** *on page* **??***)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **`refid`** – Reference
- **`file`** – File
- **`dir`** – File
- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean

- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<mapper> – *(See ?? on page ??)* Element to define a FileNameMapper.

Parameters:

- **classpath** – Path
- **classpathref** – Reference
- **refid** – Reference
- **type** – String [”identity”, ”flatten”, ”glob”, ”merge”, ”regexp”, ”package”, ”unpackage”]
- **classname** – String
- **from** – String
- **to** – String

2.11 Type control:WaitFor

Wait for an external event to occur. Wait for an external process to start or to complete some task. This is useful with the `parallel` task to synchronize the execution of tests with server startup. The following attributes can be specified on a waitfor task:

- maxwait - maximum length of time to wait before giving up
- maxwaitunit - The unit to be used to interpret maxwait attribute
- checkevery - amount of time to sleep between each check
- checkeveryunit - The unit to be used to interpret checkevery attribute
- timeoutproperty - name of a property to set if maxwait has been exceeded.

The maxwaitunit and checkeveryunit are allowed to have the following values: millisecond, second, minute, hour, day and week. The default is millisecond.

Parameters:

- **`checkeveryunit`** – String [”millisecond”, ”second”, ”minute”, ”hour”, ”day”, ”week”]Set the check every time unit
- **`checkevery`** – long Set the time between each check
- **`maxwaitunit`** – String [”millisecond”, ”second”, ”minute”, ”hour”, ”day”, ”week”]Set the max wait time unit
- **`maxwait`** – long Set the maximum length of time to wait
- **`timeoutproperty`** – String Name the property to set after a timeout.

Parameters accepted as nested elements:

`<isfalse>` (Of type IsFalse) Condition that tests whether a given string evals to false

Parameters:

- **`value`** – boolean

`<isreference>` (Of type IsReference) Condition that tests whether a given reference has been defined.

Optionally tests whether it is of a given type/class.

Parameters:

- **`refid`** – Reference
- **`type`** – String

`<equals>` (Of type Equals) Simple String comparison condition.

Parameters:

- **`trim`** – boolean
- **`arg2`** – String
- **`arg1`** – String
- **`casesensitive`** – boolean

`<available>` – *(See Task control:Available on page 9)* Will set the given property if the requested resource is available at runtime. This task may also be used as a condition by the condition task.

Parameters:

- **`classpath`** – Path

- **file** – File
- **classname** – String
- **resource** – String
- **ignoresystemclasses** – boolean
- **type** – String [”file”, ”dir”]
- **classpathref** – Reference
- **value** – String
- **property** – String
- **filepath** – Path

<not> (Of type Not) <not>condition. Evaluates to true if the single condition nested into it is false and vice versa.

Parameters:

<contains> (Of type Contains) Is one string part of another string?

Parameters:

- **casesensitive** – boolean
- **string** – String
- **substring** – String

<os> (Of type Os) Condition that tests the OS type.

Parameters:

- **version** – String
- **name** – String
- **family** – String
- **arch** – String

<or> (Of type Or) <or>condition container.

Iterates over all conditions and returns true as soon as one evaluates to true.

Parameters:

<and> (Of type And) <and>condition container.

Iterates over all conditions and returns false as soon as one evaluates to false.

Parameters:

<filesmatch> (Of type FilesMatch) Compares two files for bitwise equality based on size and content. Timestamps are not at all looked at.

Parameters:

- **file2** – File
- **file1** – File

<checksum> – *(See Task control:Checksum on page 11)* Used to create or verify file checksums.

Parameters:

- **excludes** – String
- **fileext** – String
- **totalproperty** – String
- **property** – String
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **readbuffersize** – int
- **verifyproperty** – String
- **includes** – String
- **todir** – File
- **algorithm** – String
- **includesfile** – File
- **file** – File
- **excludesfile** – File
- **provider** – String
- **forceoverwrite** – boolean

<istrue> (Of type IsTrue) Condition that tests whether a given string evals to true

Parameters:

- **value** – boolean

<isset> (Of type IsSet) Condition that tests whether a given property has been set.

Parameters:

- **property** – String

<socket> (Of type Socket) Condition to wait for a TCP/IP socket to have a listener. Its attributes are: server - the name of the server. port - the port number of the socket.

Parameters:

- **port** – int
- **server** – String

<http> (Of type Http) Condition to wait for a HTTP request to succeed. Its attribute(s) are: url - the URL of the request. errorsBeginAt - number at which errors begin at; default=400.

Parameters:

- **url** – String
- **errorsbeginat** – int

<uptodate> – *(See Task control:UpToDate on page 30)* Sets the given property if the specified target has a timestamp greater than all of the source files.

Parameters:

- **value** – String
- **property** – String
- **srcfile** – File
- **targetfile** – File

2.12 Task control:antcall

Call another target in the same project.

```
<target name="foo">
  <antcall target="bar">
    <param name="property1" value="aaaaa" />
    <param name="foo" value="baz" />
   </antcall>
</target>

<target name="bar" depends="init">
  <echo message="prop is ${property1} ${foo}" />
</target>
```

This only works as expected if neither property1 nor foo are defined in the project itself.

Parameters:

- **inheritall** – boolean If true, pass all properties to the new Ant project. Defaults to true.

- **inheritrefs** – boolean If true, pass all references to the new Ant project. Defaults to false.
- **target** – String Set target to execute.

Parameters accepted as nested elements:

<reference> Helper class that implements the nested <reference>element of <ant>and <antcall>.

Parameters:

- **torefid** – String
- **refid** – String

<propertyset> – *(See ?? on page ??)* A set of properties.

Parameters:

- **refid** – Reference
- **dynamic** – boolean
- **negate** – boolean

<target> Helper class that implements the nested <target>element of <ant>and <antcall>.

Parameters:

- **name** – String

<param> – *(See ?? on page ??)* Sets a property by name, or set of properties (from file or resource) in the project. Properties are immutable: whoever sets a property first freezes it for the rest of the build; they are most definitely not variable.

There are five ways to set properties:

- By supplying both the *name* and *value* attribute.
- By supplying both the *name* and *refid* attribute.
- By setting the *file* attribute with the filename of the property file to load. This property file has the format as defined by the file used in the class java.util.Properties.
- By setting the *resource* attribute with the resource name of the property file to load. This property file has the format as defined by the file used in the class java.util.Properties.
- By setting the *environment* attribute with a prefix to use. Properties will be defined for every environment variable by prefixing the supplied name and a period to the name of the variable.

Although combinations of these ways are possible, only one should be used at a time. Problems might occur with the order in which properties are set, for instance.

The value part of the properties being set, might contain references to other properties. These references are resolved at the time these properties are set. This also holds for properties loaded from a property file.

Properties are case sensitive.

Parameters:

- **`refid`** – Reference
- **`url`** – URL
- **`name`** – String
- **`classpath`** – Path
- **`userproperty`** – boolean
- **`file`** – File
- **`resource`** – String
- **`environment`** – String
- **`prefix`** – String
- **`classpathref`** – Reference
- **`value`** – String

2.13 Task control:apply

Executes a given command, supplying a set of files as arguments.

Parameters:

- **`resultproperty`** – String
- **`skipemptyfilesets`** – boolean Set whether empty filesets will be skipped. If true and no source files have been found or are newer than their corresponding target files, the command will not be run.
- **`force`** – boolean Set whether to bypass timestamp comparisons for target files.
- **`failifexecutionfails`** – boolean
- **`command`** – Commandline

- **type** – String [”file”, ”dir”, ”both”]Set whether the command works only on files, directories or both.
- **maxparallel** – int Limit the command line length by passing at maximum this many sourcefiles at once to the command.

  Set to <= 0 for unlimited - this is the default.
- **executable** – String
- **errorproperty** – String
- **os** – String
- **output** – File
- **timeout** – Integer
- **dest** – File Specify the directory where target files are to be placed.
- **spawn** – boolean
- **input** – File
- **inputstring** – String
- **addsourcefile** – boolean Set whether to send the source file name on the command line.

  Defaults to `true`.
- **verbose** – boolean Set whether to operate in verbose mode. If true, a verbose summary will be printed after execution.
- **searchpath** – boolean
- **logerror** – boolean
- **dir** – File
- **forwardslash** – boolean Set whether the source and target file names on Windows and OS/2 must use the forward slash as file separator.
- **error** – File
- **outputproperty** – String
- **append** – boolean
- **resolveexecutable** – boolean
- **failonerror** – boolean

- **ignoremissing** – boolean Set whether to ignore nonexistent files from filelists.
- **vmlauncher** – boolean
- **newenvironment** – boolean
- **relative** – boolean Set whether the filenames should be passed on the command line as absolute or relative pathnames. Paths are relative to the base directory of the corresponding fileset for source files or the dest attribute for target files.
- **parallel** – boolean Set whether to execute in parallel mode. If true, run the command only once, appending all files as arguments. If false, command will be executed once for every file. Defaults to false.

Parameters accepted as nested elements:

<mapper> – *(See* **??** *on page* **??***)* Element to define a FileNameMapper.

Parameters:

- **classpath** – Path
- **classpathref** – Reference
- **refid** – Reference
- **type** – String [”identity”, ”flatten”, ”glob”, ”merge”, ”regexp”, ”package”, ”unpackage”]
- **classname** – String
- **from** – String
- **to** – String

<srcfile> Class to keep track of the position of an Argument.

This class is there to support the srcfile and targetfile elements of <execon>and <transform>- don’t know whether there might be additional use cases.

–SB

Parameters:

<targetfile> Class to keep track of the position of an Argument.

This class is there to support the srcfile and targetfile elements of <execon>and <transform>- don’t know whether there might be additional use cases.

–SB

Parameters:

<redirector> – *(See* **??** *on page* **??***)* Element representation of a `Redirector`.

Parameters:

- **refid** – Reference
- **inputencoding** – String
- **append** – boolean
- **createemptyfiles** – boolean
- **output** – File
- **outputproperty** – String
- **outputencoding** – String
- **errorproperty** – String
- **error** – File
- **inputstring** – String
- **alwayslog** – boolean
- **input** – File
- **errorencoding** – String
- **logerror** – boolean

<filelist> – *(See* **??** *on page* **??***)* FileList represents an explicitly named list of files. FileLists are useful when you want to capture a list of files regardless of whether they currently exist. By contrast, FileSet operates as a filter, only returning the name of a matched file if it currently exists in the file system.

Parameters:

- **dir** – File
- **refid** – Reference
- **files** – String

<env> representation of a single env value

Parameters:

- **key** – String
- **file** – File
- **value** – String
- **path** – Path

<fileset> – *(See* **??** *on page* **??***)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File

- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<dirset> – *(See ?? on page ??)* Subclass as hint for supporting tasks that the included directories instead of files should be used.

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<arg> Used for nested xml command line definitions.

Parameters:

- **line** – String
- **file** – File
- **pathref** – Reference
- **value** – String
- **path** – Path

2.14 Task control:fail

Exits the active build, giving an additional message if available. The `if` and `unless` attributes make the failure conditional -both probe for the named property being defined. The `if` tests for the property being defined, the `unless` for a property being undefined. If both attributes are set, then the test fails only if both tests are true. i.e.

```
fail := defined(ifProperty) && !defined(unlessProperty)
```

A single nested<`condition`> element can be specified instead of using `if/unless` (a combined effect can be achieved using `isset` conditions).

Parameters:

- **`status`** – int Set the status code to associate with the thrown Exception.
- **`message`** – String A message giving further information on why the build exited.
- **`unless`** – String Only fail if a property of the given name does not exist in the current project.
- **`if`** – String Only fail if a property of the given name exists in the current project.

Parameters accepted as nested elements:

This Task accepts text in its element body.

`<condition>` (Of type ConditionBase) Baseclass for the <condition>task as well as several conditions - ensures that the types of conditions inside the task and the ”container” conditions are in sync.

Parameters:

2.15 Task control:nice

A task to provide ”nice-ness” to the current thread, and/or to query the current value. Examples:

```
<nice currentPriority="current.value" >
```

Set `currentPriority` to the current priority

```
<nice newPriority="10" >
```

Raise the priority of the build process (But not forked programs)

```
<nice currentPriority="old" newPriority="3" >
```

Lower the priority of the build process (But not forked programs), and save the old value to the property `old`.

Parameters:

- **`newpriority`** – int the new priority, in the range 1-10.
- **`currentpriority`** – String The name of a property to set to the value of the current thread priority. Optional

---

2.16 Task control:subant

Calls a given target for all defined sub-builds. This is an extension of ant for bulk project execution.

Use with directories

subant can be used with directory sets to execute a build from different directories. 2 different options are offered

- run the same build file /somepath/otherpath/mybuild.xml with different base directories use the genericantfile attribute
- if you want to run directory1/build.xml, directory2/build.xml, ... use the antfile attribute. The base directory does not get set by the subant task in this case, because you can specify it in each build file.

Parameters:

- **`genericantfile`** – File This method builds a file path to use in conjunction with directories.

  Use `genericantfile`, in order to run the same build file with different basedirs.

  If this attribute is set, `antfile` is ignored.
- **`output`** – String Corresponds to <`ant`>'s `output` attribute.
- **`inheritall`** – boolean Corresponds to <`ant`>'s `inheritall` attribute.

- **verbose** – boolean Enable/ disable verbose log messages showing when each sub-build path is entered/ exited. The default value is ”false”.
- **inheritrefs** – boolean Corresponds to <ant>'s inheritrefs attribute.
- **antfile** – String This method builds the file name to use in conjunction with directories.

  Defaults to ”build.xml”. If genericantfile is set, this attribute is ignored.
- **buildpathref** – Reference Buildpath to use, by reference.
- **target** – String The target to call on the different sub-builds. Set to ”” to execute the default target.
- **buildpath** – Path Set the buildpath to be used to find sub-projects.
- **failonerror** – boolean Sets whether to fail with a build exception on error, or go on.

Parameters accepted as nested elements:

<reference> Helper class that implements the nested <reference>element of <ant>and <antcall>.

Parameters:

- **torefid** – String
- **refid** – String

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<buildpathelement> Helper class, holds the nested <pathelement> values.

Parameters:

- **path** – String

<dirset> – *(See ?? on page ??)* Subclass as hint for supporting tasks that the included directories instead of files should be used.

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<buildpath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference

- **path** – String

<propertyset> – *(See ?? on page ??)* A set of properties.

Parameters:

- **refid** – Reference
- **dynamic** – boolean
- **negate** – boolean

<filelist> – *(See ?? on page ??)* FileList represents an explicitly named list of files. FileLists are useful when you want to capture a list of files regardless of whether they currently exist. By contrast, FileSet operates as a filter, only returning the name of a matched file if it currently exists in the file system.

Parameters:

- **dir** – File
- **refid** – Reference
- **files** – String

<property> – *(See ?? on page ??)* Sets a property by name, or set of properties (from file or resource) in the project. Properties are immutable: whoever sets a property first freezes it for the rest of the build; they are most definitely not variable.

There are five ways to set properties:

- By supplying both the *name* and *value* attribute.
- By supplying both the *name* and *refid* attribute.
- By setting the *file* attribute with the filename of the property file to load. This property file has the format as defined by the file used in the class java.util.Properties.
- By setting the *resource* attribute with the resource name of the property file to load. This property file has the format as defined by the file used in the class java.util.Properties.
- By setting the *environment* attribute with a prefix to use. Properties will be defined for every environment variable by prefixing the supplied name and a period to the name of the variable.

Although combinations of these ways are possible, only one should be used at a time. Problems might occur with the order in which properties are set, for instance.

The value part of the properties being set, might contain references to other properties. These references are resolved at the time these properties are set. This also holds for properties loaded from a property file.

Properties are case sensitive.

Parameters:

- **refid** – Reference
- **url** – URL
- **name** – String
- **classpath** – Path
- **userproperty** – boolean
- **file** – File
- **resource** – String
- **environment** – String
- **prefix** – String
- **classpathref** – Reference
- **value** – String

3 database Library

3.1 Task database:sql

Executes a series of SQL statements on a database using JDBC.

Statements can either be read in from a text file using the *src* attribute or from between the enclosing SQL tags.

Multiple statements can be provided, separated by semicolons (or the defined *delimiter*). Individual lines within the statements can be commented using either –, // or REM at the start of the line.

The *autocommit* attribute specifies whether auto-commit should be turned on or off whilst executing the statements. If auto-commit is turned on each statement will be executed and committed. If it is turned off the statements will all be executed as one transaction.

The *onerror* attribute specifies how to proceed when an error occurs during the execution of one of the statements. The possible values are: **continue** execution, only show the error; **stop** execution and commit transaction; and **abort** execution and transaction and fail task.

Parameters:

- **onerror** – String [”continue”, ”stop”, ”abort”]Action to perform when statement fails: continue, stop, or abort optional; default ”abort”

- **`userid`** – String
- **`password`** – String
- **`autocommit`** – boolean
- **`rdbms`** – String
- **`print`** – boolean Print result sets from the statements; optional, default false
- **`classpathref`** – Reference
- **`encoding`** – String Set the file encoding to use on the SQL files read in
- **`version`** – String
- **`url`** – String
- **`src`** – File Set the name of the SQL file to be run. Required unless statements are enclosed in the build file
- **`output`** – File Set the output file; optional, defaults to the Ant log.
- **`driver`** – String
- **`showheaders`** – boolean Print headers for result sets from the statements; optional, default true.
- **`classpath`** – Path
- **`delimitertype`** – String ["normal", "row"]Set the delimiter type: "normal" or "row" (default "normal").

  The delimiter type takes two values - normal and row. Normal means that any occurrence of the delimiter terminate the SQL command whereas with row, only a line containing just the delimiter is recognized as the end of the command.
- **`escapeprocessing`** – boolean Set escape processing for statements.
- **`append`** – boolean whether output should be appended to or overwrite an existing file. Defaults to false.
- **`caching`** – boolean
- **`keepformat`** – boolean whether or not format should be preserved. Defaults to false.
- **`delimiter`** – String Set the delimiter that separates SQL statements. Defaults to ";"; optional

  For example, set this to "go" and delimitertype to "ROW" for Sybase ASE or MS SQL Server.

Parameters accepted as nested elements:

This Task accepts text in its element body.

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<classpath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<transaction> Contains the definition of a new transaction element. Transactions allow several files or blocks of statements to be executed using the same JDBC connection and commit operation in between.

Parameters:

- **src** – File

4 filesystem Library

4.1 Task filesystem:Chmod

Chmod equivalent for unix-like environments.

Parameters:

- **addsourcefile** – boolean
- **excludes** – String Sets the set of exclude patterns. Patterns may be separated by a comma or a space.
- **verbose** – boolean
- **parallel** – boolean
- **input** – File
- **executable** – String
- **force** – boolean
- **inputstring** – String
- **searchpath** – boolean
- **relative** – boolean
- **skipemptyfilesets** – boolean
- **command** – Commandline
- **forwardslash** – boolean
- **spawn** – boolean

- **includes** – String Sets the set of include patterns. Patterns may be separated by a comma or a space.
- **vmlauncher** – boolean
- **file** – File The file or single directory of which the permissions must be changed.
- **logerror** – boolean
- **errorproperty** – String
- **failonerror** – boolean
- **output** – File
- **error** – File
- **defaultexcludes** – boolean Sets whether default exclusions should be used or not.
- **os** – String
- **dest** – File
- **resolveexecutable** – boolean
- **resultproperty** – String
- **ignoremissing** – boolean
- **failifexecutionfails** – boolean
- **append** – boolean
- **timeout** – Integer
- **maxparallel** – int
- **outputproperty** – String
- **perm** – String The new permissions.
- **dir** – File The directory which holds the files whose permissions must be changed.
- **newenvironment** – boolean
- **type** – String [”file”, ”dir”, ”both”]

Parameters accepted as nested elements:

<mapper> – *(See ?? on page ??)* Element to define a FileNameMapper.

Parameters:

- **classpath** – Path
- **classpathref** – Reference
- **refid** – Reference
- **type** – String [”identity”, ”flatten”, ”glob”, ”merge”, ”regexp”, ”package”, ”unpackage”]
- **classname** – String
- **from** – String
- **to** – String

<include> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<srcfile> Class to keep track of the position of an Argument.

This class is there to support the srcfile and targetfile elements of <execon>and <transform>- don’t know whether there might be additional use cases.

–SB

Parameters:

<exclude> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<targetfile> Class to keep track of the position of an Argument.

This class is there to support the srcfile and targetfile elements of <execon>and <transform>- don’t know whether there might be additional use cases.

–SB

Parameters:

<patternset> – *(See ?? on page ??)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<redirector> – *(See ?? on page ??)* Element representation of a `Redirector`.

Parameters:

- **refid** – Reference
- **inputencoding** – String
- **append** – boolean
- **createemptyfiles** – boolean
- **output** – File
- **outputproperty** – String
- **outputencoding** – String
- **errorproperty** – String
- **error** – File
- **inputstring** – String
- **alwayslog** – boolean
- **input** – File
- **errorencoding** – String
- **logerror** – boolean

<env> representation of a single env value

Parameters:

- **key** – String
- **file** – File
- **value** – String
- **path** – Path

<filelist> – *(See ?? on page ??)* FileList represents an explicitly named list of files. FileLists are useful when you want to capture a list of files regardless of whether they currently exist. By contrast, FileSet operates as a filter, only returning the name of a matched file if it currently exists in the file system.

Parameters:

- **dir** – File
- **refid** – Reference
- **files** – String

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<dirset> – *(See ?? on page ??)* Subclass as hint for supporting tasks that the included directories instead of files should be used.

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<arg> Used for nested xml command line definitions.

Parameters:

- **line** – String
- **file** – File
- **pathref** – Reference
- **value** – String
- **path** – Path

4.2 Task filesystem:Copy

Copies a file or directory to a new file or directory. Files are only copied if the source file is newer than the destination file, or when the destination file does not exist. It is possible to explicitly overwrite existing files.

This implementation is based on Arnout Kuiper's initial design document, the following mailing list discussions, and the copyfile/copydir tasks.

Parameters:

- **flatten** – boolean When copying directory trees, the files can be "flattened" into a single directory. If there are multiple files with the same name in the source directory tree, only the first file will be copied into the "flattened" directory, unless the forceoverwrite attribute is true.
- **tofile** – File Sets the destination file.
- **encoding** – String Sets the character encoding
- **todir** – File Sets the destination directory.
- **file** – File Sets a single source file to copy.
- **verbose** – boolean Used to force listing of all names of copied files.
- **includeemptydirs** – boolean Used to copy empty directories.
- **overwrite** – boolean Overwrite any existing destination file(s).
- **enablemultiplemappings** – boolean Attribute to handle mappers that return multiple mappings for a given source path.

- **granularity** – long The number of milliseconds leeway to give before deciding a target is out of date.

  Default is 0 milliseconds, or 2 seconds on DOS systems.

- **outputencoding** – String Sets the character encoding for output files.
- **failonerror** – boolean If false, note errors to the output but keep going.
- **preservelastmodified** – boolean Give the copied files the same last modified time as the original files.
- **filtering** – boolean If true, enables filtering.

Parameters accepted as nested elements:

<filterchain> – *(See* **??** *on page* **??***)* FilterChain may contain a chained set of filter readers.

Parameters:

- **refid** – Reference

<fileset> – *(See* **??** *on page* **??***)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<filterset> – *(See* **??** *on page* **??***)* A set of filters to be applied to something. A filter set may have begintoken and endtokens defined.

Parameters:

- **begintoken** – String
- **endtoken** – String

- **refid** – Reference
- **recurse** – boolean
- **filtersfile** – File

<mapper> – *(See ?? on page ??)* Element to define a FileNameMapper.

Parameters:

- **classpath** – Path
- **classpathref** – Reference
- **refid** – Reference
- **type** – String [”identity”, ”flatten”, ”glob”, ”merge”, ”regexp”, ”package”, ”unpackage”]
- **classname** – String
- **from** – String
- **to** – String

4.3 Task filesystem:Delete

Deletes a file or directory, or set of files defined by a fileset. The original delete task would delete a file, or a set of files using the include/exclude syntax. The deltree task would delete a directory tree. This task combines the functionality of these two originally distinct tasks.

Currently Delete extends MatchingTask. This is intended *only* to provide backwards compatibility for a release. The future position is to use nested filesets exclusively.

Parameters:

- **file** – File Set the name of a single file to be removed.
- **dir** – File Set the directory from which files are to be deleted
- **quiet** – boolean If true and the file does not exist, do not display a diagnostic message or modify the exit status to reflect an error. This means that if a file or directory cannot be deleted, then no error is reported. This setting emulates the -f option to the Unix ”rm” command. Default is false meaning things are ”noisy”
- **verbose** – boolean If true, list all names of deleted files.
- **defaultexcludes** – boolean Sets whether default exclusions should be used or not.

- **followsymlinks** – boolean Sets whether or not symbolic links should be followed.
- **includeemptydirs** – boolean If true, delete empty directories.
- **casesensitive** – boolean Sets case sensitivity of the file system
- **failonerror** – boolean If false, note errors but continue.
- **deleteonexit** – boolean If true, on failure to delete, note the error and set the deleteonexit flag, and continue
- **excludes** – String Sets the set of exclude patterns. Patterns may be separated by a comma or a space.
- **includesfile** – File Sets the name of the file containing the includes patterns.
- **excludesfile** – File Sets the name of the file containing the includes patterns.
- **includes** – String Sets the set of include patterns. Patterns may be separated by a comma or a space.

Parameters accepted as nested elements:

<patternset> – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String ["file", "dir"]

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
            >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
                >
                >
```

And the same rewritten as CustomSelector would be

```
                >
                >
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the

selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns 'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean

- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don’t get too caught up in boolean, there are other ways you can evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an ”all-but-one” selector, a ”weighted-average” selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference

- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String ["before", "after", "equal"]
- **error** – String

<excludesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String

- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

4.4 Task filesystem:DependSet

Examines and removes out of date target files. If any of the target files are out of date with respect to any of the source files, all target files are removed. This is useful where dependencies cannot be computed (for example, dynamically interpreted parameters or files that need to stay in synch but are not directly linked) or where the ant task in question could compute them but does not (for example, the linked DTD for an XML file using the style task). nested arguments:

- srcfileset (fileset describing the source files to examine)
- srcfilelist (filelist describing the source files to examine)
- targetfileset (fileset describing the target files to examine)
- targetfilelist (filelist describing the target files to examine)

At least one instance of either a fileset or filelist for both source and target are required.

This task will examine each of the source files against each of the target files. If any target files are out of date with respect to any of the source files, all targets are removed. If any files named in a (src or target) filelist do not exist, all targets are removed. Hint: If missing files should be ignored, specify them as include patterns in filesets, rather than using filelists.

This task attempts to optimize speed of dependency checking. It will stop after the first out of date file is found and remove all targets, rather than exhaustively checking every source vs target combination unnecessarily.

Example uses:

- Record the fact that an XML file must be up to date with respect to its XSD (Schema file), even though the XML file itself includes no reference to its XSD.
- Record the fact that an XSL stylesheet includes other sub-stylesheets
- Record the fact that java files must be recompiled if the ant build file changes

Parameters:

- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean
- **`casesensitive`** – boolean
- **`excludes`** – String
- **`includesfile`** – File
- **`excludesfile`** – File
- **`includes`** – String

Parameters accepted as nested elements:

`<patternset>` – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **`includes`** – String
- **`refid`** – Reference
- **`excludesfile`** – File
- **`includesfile`** – File
- **`excludes`** – String

`<exclude>` inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String ["file", "dir"]

<targetfileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<srcfileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean

- **`casesensitive`** – boolean
- **`excludes`** – String
- **`includesfile`** – File
- **`excludesfile`** – File
- **`includes`** – String

`<modified>` (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
            >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
                >
                >
```

And the same rewritten as CustomSelector would be

```
                >
                >
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns

'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
 generate the site using forrest
>

 upload the changed files


       >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException "Invalid parameter" is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String ["equal", "rule"]
- **`seldirs`** – boolean
- **`algorithm`** – String ["hashvalue", "digest"]
- **`cache`** – String ["propertyfile"]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean
- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don't get too caught up in boolean, there are other ways you can

evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an "all-but-one" selector, a "weighted-average" selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference
- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<srcfilelist> – *(See ?? on page ??)* FileList represents an explicitly named list of files. FileLists are useful when you want to capture a list of files regardless of whether they currently exist. By contrast, FileSet operates as a filter, only returning the name of a matched file if it currently exists in the file system.

Parameters:

- **dir** – File
- **refid** – Reference
- **files** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String [”before”, ”after”, ”equal”]
- **error** – String

<excludesfile> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String

- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

<targetfilelist> – *(See ?? on page ??)* FileList represents an explicitly named list of files. FileLists are useful when you want to capture a list of files regardless of whether they currently exist. By contrast, FileSet operates as a filter, only returning the name of a matched file if it currently exists in the file system.

Parameters:

- **dir** – File
- **refid** – Reference
- **files** – String

4.5 Task filesystem:Filter

Sets a token filter that is used by the file copy tasks to do token substitution. Sets multiple tokens by reading these from a file.

Parameters:

- **token** – String The token string without @ delimiters.
- **value** – String The string that should replace the token during filtered copies.
- **filtersfile** – File The file from which the filters must be read. This file must be a formatted as a property file.

4.6 Task filesystem: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.

This task can take the following arguments:

- srcdir
- destdir
- include
- exclude
- cr
- eol
- tab
- eof
- encoding

Of these arguments, only **sourcedir** is required.

When this task executes, it will scan the srcdir based on the include and exclude properties.

This version generalises the handling of EOL characters, and allows for CR-only line endings (which I suspect is the standard on Macs.) Tab handling has also been generalised to accommodate any tabwidth from 2 to 80, inclusive. Importantly, it will leave untouched any literal TAB characters embedded within string or character constants.

Warning: do not run on binary files. Caution: run with care on carefully formatted files. This may sound obvious, but if you don't specify asis, presume that your files are going to be modified. If "tabs" is "add" or "remove", whitespace characters may be added or removed as necessary. Similarly, for CR's - in fact "eol"="crlf" or cr="add" can result in cr characters being removed in one special case accommodated, i.e., CRCRLF is regarded as a single EOL to handle cases where other programs have converted CRLF into CRCRLF.

Parameters:

- **`tab`** – String ["add", "asis", "remove"]Specify how tab characters are to be handled.
- **`excludes`** – String
- **`encoding`** – String Specifies the encoding Ant expects the files to be in - defaults to the platforms default encoding.
- **`cr`** – String ["add", "asis", "remove"]Specify how carriage return (CR) characters are to be handled.
- **`destdir`** – File Set the destination where the fixed files should be placed. Default is to replace the original file.

- **defaultexcludes** – boolean
- **eol** – String ["asis", "cr", "lf", "crlf", "mac", "unix", "dos"]Specify how EndOfLine characters are to be handled.
- **followsymlinks** – boolean
- **javafiles** – boolean Set to true if modifying Java source files.
- **eof** – String ["add", "asis", "remove"]Specify how DOS EOF (control-z) characters are to be handled.
- **casesensitive** – boolean
- **tablength** – int Specify tab length in characters.
- **includes** – String
- **fixlast** – boolean Specify whether a missing EOL will be added to the final line of a file.
- **includesfile** – File
- **srcdir** – File Set the source dir to find the source text files.
- **excludesfile** – File

Parameters accepted as nested elements:

<patternset> – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String

- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String [”file”, ”dir”]

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
>
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
>
>
```

And the same rewritten as CustomSelector would be

```
>
>
>
>
>
>
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns 'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
	hashvalue - loads the file content into a String and uses its hashValue() method		
	digest - uses java.security.MessageDigest class	no, defaults to digest	
comparator	equal — role	which comparator implementation should be used	
	equal - simple comparison using String.equals()		
	role - uses java.text.RuleBasedCollator class	no, defaults to equal	
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException "Invalid parameter" is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String ["equal", "rule"]
- **`seldirs`** – boolean
- **`algorithm`** – String ["hashvalue", "digest"]
- **`cache`** – String ["propertyfile"]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean
- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don't get too caught up in boolean, there are other ways you can

evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an "all-but-one" selector, a "weighted-average" selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference
- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String [”before”, ”after”, ”equal”]
- **error** – String

<excludesfile> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

4.7 Task filesystem:Mkdir

Creates a given directory. Creates a directory and any non-existent parent directories, when necessary

Parameters:

- **`dir`** – File the directory to create; required.

4.8 Task filesystem:Move

Moves a file or directory to a new file or directory. By default, the destination file is overwritten if it already exists. When *overwrite* is turned off, then files are only moved if the source file is newer than the destination file, or when the destination file does not exist.

Source files and directories are only deleted when the file or directory has been copied to the destination successfully. Filtering also works.

This implementation is based on Arnout Kuiper's initial design document, the following mailing list discussions, and the copyfile/copydir tasks.

Parameters:

- **`flatten`** – boolean
- **`tofile`** – File
- **`encoding`** – String
- **`todir`** – File
- **`file`** – File
- **`verbose`** – boolean
- **`includeemptydirs`** – boolean
- **`overwrite`** – boolean
- **`enablemultiplemappings`** – boolean
- **`granularity`** – long
- **`outputencoding`** – String
- **`failonerror`** – boolean

- **preservelastmodified** – boolean
- **filtering** – boolean

Parameters accepted as nested elements:

<filterchain> – *(See ?? on page ??)* FilterChain may contain a chained set of filter readers.

Parameters:

- **refid** – Reference

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<filterset> – *(See ?? on page ??)* A set of filters to be applied to something. A filter set may have begintoken and endtokens defined.

Parameters:

- **begintoken** – String
- **endtoken** – String
- **refid** – Reference
- **recurse** – boolean
- **filtersfile** – File

<mapper> – *(See ?? on page ??)* Element to define a FileNameMapper.

Parameters:

- **classpath** – Path

- **classpathref** – Reference
- **refid** – Reference
- **type** – String ["identity", "flatten", "glob", "merge", "regexp", "package", "unpackage"]
- **classname** – String
- **from** – String
- **to** – String

4.9 Task filesystem:Replace

Replaces all occurrences of one or more string tokens with given values in the indicated files. Each value can be either a string or the value of a property available in a designated property file. If you want to replace a text that crosses line boundaries, you must use a nested <replacetoken> element.

Parameters:

- **encoding** – String Set the file encoding to use on the files read and written by the task; optional, defaults to default JVM encoding
- **file** – File Set the source file; required unless dir is set.
- **dir** – File The base directory to use when replacing a token in multiple files; required if file is not defined.
- **token** – String Set the string token to replace; required unless a nested replacetoken element or the replacefilterfile attribute is used.
- **propertyfile** – File 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.
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **summary** – boolean Indicates whether a summary of the replace operation should be produced, detailing how many token occurrences and files were processed; optional, default=false
- **casesensitive** – boolean

- **value** – String Set the string value to use as token replacement; optional, default is the empty string ""
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **replacefilterfile** – File Sets the name of a property file containing filters; optional. Each property will be treated as a replacefilter where token is the name of the property and value is the value of the property.
- **includes** – String

Parameters accepted as nested elements:

<patternset> – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String

- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String

- **`refid`** – Reference
- **`type`** – String [”file”, ”dir”]

`<modified>` (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
            >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
                >
                >
```

And the same rewritten as CustomSelector would be

```
                >
                >
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns ’true’ (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<replacefilter>` A filter to apply.

Parameters:

- **`token`** – String
- **`value`** – String
- **`property`** – String

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

<depend> (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<replacetoken> an inline string to use as the replacement text

Parameters:

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean
- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String ["less", "more", "equal"]

- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don’t get too caught up in boolean, there are other ways you can evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an ”all-but-one” selector, a ”weighted-average” selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference
- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<replacevalue> an inline string to use as the replacement text

Parameters:

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String ["before", "after", "equal"]
- **error** – String

<excludesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

4.10 Task filesystem:Sync

Synchronize a local target directory from the files defined in one or more filesets.

Uses a <copy>task internally, but forbidding the use of mappers and filter chains. Files of the destination directory not present in any of the source fileset are removed.

Parameters:

- **verbose** – boolean Used to force listing of all names of copied files.
- **overwrite** – boolean Overwrite any existing destination file(s).
- **todir** – File Sets the destination directory.
- **includeemptydirs** – boolean Used to copy empty directories.
- **failonerror** – boolean If false, note errors to the output but keep going.
- **granularity** – long The number of milliseconds leeway to give before deciding a target is out of date.

  Default is 0 milliseconds, or 2 seconds on DOS systems.

Parameters accepted as nested elements:

<fileset> – *(See* **??** *on page* **??***)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

4.11 Task filesystem:Sync.MyCopy

Subclass Copy in order to access it's file/dir maps.

Parameters:

- **flatten** – boolean
- **tofile** – File
- **encoding** – String
- **todir** – File
- **file** – File
- **verbose** – boolean
- **includeemptydirs** – boolean
- **overwrite** – boolean
- **enablemultiplemappings** – boolean
- **granularity** – long
- **outputencoding** – String
- **failonerror** – boolean
- **preservelastmodified** – boolean
- **filtering** – boolean

Parameters accepted as nested elements:

<filterchain> – *(See ?? on page ??)* FilterChain may contain a chained set of filter readers.

Parameters:

- **refid** – Reference

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<filterset> – *(See ?? on page ??)* A set of filters to be applied to something. A filter set may have begintoken and endtokens defined.

Parameters:

- **begintoken** – String
- **endtoken** – String
- **refid** – Reference
- **recurse** – boolean
- **filtersfile** – File

<mapper> – *(See ?? on page ??)* Element to define a FileNameMapper.

Parameters:

- **classpath** – Path
- **classpathref** – Reference
- **refid** – Reference
- **type** – String [”identity”, ”flatten”, ”glob”, ”merge”, ”regexp”, ”package”, ”unpackage”]

- **classname** – String
- **from** – String
- **to** – String

4.12 Task filesystem:Touch

Touch a file and/or fileset(s) and/or filelist(s); corresponds to the Unix touch command.

If the file to touch doesn't exist, an empty one is created.

Parameters:

- **verbose** – boolean Set whether the touch task will report every file it creates; defaults to `true`.
- **mkdirs** – boolean Set whether nonexistent parent directories should be created when touching new files.
- **millis** – long Set the new modification time of file(s) touched in milliseconds since midnight Jan 1 1970. Optional, default=now.
- **file** – File Sets a single source file to touch. If the file does not exist an empty file will be created.
- **pattern** – String Set the format of the datetime attribute.
- **datetime** – String Set the new modification time of file(s) touched in the format "MM/DD/YYYY HH:MM AM *or* PM" or "MM/DD/YYYY HH:MM:SS AM *or* PM". Optional, default=now.

Parameters accepted as nested elements:

<fileset> – *(See* **??** *on page* **??***)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean

- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<filelist> – *(See ?? on page ??)* FileList represents an explicitly named list of files. FileLists are useful when you want to capture a list of files regardless of whether they currently exist. By contrast, FileSet operates as a filter, only returning the name of a matched file if it currently exists in the file system.

Parameters:

- **dir** – File
- **refid** – Reference
- **files** – String

<mapper> – *(See ?? on page ??)* Element to define a FileNameMapper.

Parameters:

- **classpath** – Path
- **classpathref** – Reference
- **refid** – Reference
- **type** – String [”identity”, ”flatten”, ”glob”, ”merge”, ”regexp”, ”package”, ”unpackage”]
- **classname** – String
- **from** – String
- **to** – String

---

5 internal Library

5.1 Task internal:Taskdef

Adds a task definition to the current project, such that this new task can be used in the current project. Two attributes are needed, the name that identifies this task uniquely, and the full name of the class (including the packages) that implements this task.

You can also define a group of tasks at once using the file or resource attributes. These attributes point to files in the format of Java property files. Each line defines a single task in the format:

```
taskname=fully.qualified.java.classname
```

Parameters:

- **name** – String
- **uri** – String
- **classpath** – Path
- **file** – File
- **classname** – String
- **resource** – String
- **adaptto** – String
- **onerror** – String [”fail”, ”report”, ”ignore”]
- **loaderref** – Reference
- **classpathref** – Reference
- **adapter** – String
- **format** – String [”properties”, ”xml”]
- **reverseloader** – boolean

Parameters accepted as nested elements:

<classpath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

5.2 Task internal:Typedef

Adds a data type definition to the current project. Two attributes are needed, the name that identifies this data type uniquely, and the full name of the class (including the packages) that implements this type.

You can also define a group of data types at once using the file or resource attributes. These attributes point to files in the format of Java property files. Each line defines a single data type in the format:

```
typename=fully.qualified.java.classname
```

Typedef should be used to add your own types to the system. Data types are things likepaths or filesets that can be defined at the project level and referenced via their ID attribute.

Custom data types usually need custom tasks to put them to good use.

Parameters:

- **`name`** – String
- **`uri`** – String
- **`classpath`** – Path
- **`file`** – File
- **`classname`** – String
- **`resource`** – String
- **`adaptto`** – String

- **onerror** – String [”fail”, ”report”, ”ignore”]
- **loaderref** – Reference
- **classpathref** – Reference
- **adapter** – String
- **format** – String [”properties”, ”xml”]
- **reverseloader** – boolean

Parameters accepted as nested elements:

<classpath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

6 java Library

6.1 Task java:Java

Launcher for Java applications. Allows use of the same JVM for the called application thus resulting in much faster operation.

Parameters:

- **`resultproperty`** – String The name of a property in which the return code of the command should be stored. Only of interest if failonerror=false.
- **`jvmargs`** – String Set the command line arguments for the JVM.
- **`args`** – String Deprecated: use nested arg instead. Set the command line arguments for the class.
- **`fork`** – boolean If true, execute in a new VM.
- **`maxmemory`** – String Corresponds to -mx or -Xmx depending on VM version.
- **`classpathref`** – Reference Classpath to use, by reference.
- **`jar`** – File The location of the JAR file to execute.
- **`errorproperty`** – String Property name whose value should be set to the error of the process.
- **`output`** – File File the output of the process is redirected to.
- **`timeout`** – Long Timeout in milliseconds after which the process will be killed.
- **`spawn`** – boolean set whether or not you want the process to be spawned default is not spawned
- **`jvm`** – String Set the command used to start the VM (only if forking).
- **`input`** – File Set the input to use for the task
- **`inputstring`** – String Set the string to use as input
- **`classpath`** – Path Set the classpath to be used when running the Java class
- **`logerror`** – boolean Controls whether error output of exec is logged. This is only useful when output is being redirected and error output is desired in the Ant log
- **`dir`** – File The working directory of the process
- **`error`** – File File the error stream of the process is redirected to.

- **jvmversion** – String Sets the JVM version.
- **outputproperty** – String Property name whose value should be set to the output of the process.
- **failonerror** – boolean If true, then fail if the command exits with a returncode other than 0
- **append** – boolean If true, append output to existing file.
- **newenvironment** – boolean If true, use a completely new environment.

  Will be ignored if we are not forking a new VM.
- **classname** – String Sets the Java class to execute.

Parameters accepted as nested elements:

<classpath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<sysproperty> representation of a single env value

Parameters:

- **key** – String

- **file** – File
- **value** – String
- **path** – Path

<bootclasspath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<redirector> – *(See ?? on page ??)* Element representation of a `Redirector`.

Parameters:

- **refid** – Reference
- **inputencoding** – String
- **append** – boolean
- **createemptyfiles** – boolean
- **output** – File
- **outputproperty** – String
- **outputencoding** – String
- **errorproperty** – String
- **error** – File
- **inputstring** – String

- **alwayslog** – boolean
- **input** – File
- **errorencoding** – String
- **logerror** – boolean

<syspropertyset> – *(See ?? on page ??)* A set of properties.

Parameters:

- **refid** – Reference
- **dynamic** – boolean
- **negate** – boolean

<env> representation of a single env value

Parameters:

- **key** – String
- **file** – File
- **value** – String
- **path** – Path

<permissions> – *(See ?? on page ??)* This class implements a security manager meant for useage by tasks that run inside the ant VM. An examples are the Java Task and JUnitTask. The basic functionality is that nothing (except for a base set of permissions) is allowed, unless the permission is granted either explicitly or implicitly. If an permission is granted this can be overruled by explicitly revoking the permission. It is not permissible to add permissions (either granted or revoked) while the Security Manager is active (after calling setSecurityManager() but before calling restoreSecurityManager()).

Parameters:

<jvmarg> Used for nested xml command line definitions.

Parameters:

- **line** – String
- **file** – File
- **pathref** – Reference
- **value** – String
- **path** – Path

<arg> Used for nested xml command line definitions.

Parameters:

- **line** – String
- **file** – File
- **pathref** – Reference
- **value** – String
- **path** – Path

<assertions> – *(See ?? on page ??)* The assertion datatype. This type describes assertion settings for the <java>task and others. One can set the system assertions, and enable/disable those in packages and classes. Assertions can only be enabled or disabled when forking Java. Example: set system assertions and all org.apache packages except for ant, and the class org.apache.tools.ant.Main.

```
<assertions enableSystemAssertions="true" >
  <enable package="org.apache" />
  <disable package="org.apache.ant" />
  <enable class="org.apache.tools.ant.Main"/>
</assertions>
```

Disable system assertions; enable those in the anonymous package

```
<assertions enableSystemAssertions="false" >
  <enable package="..." />
</assertions>
```

enable assertions in a class called Test

```
<assertions >
  <enable class="Test" />
</assertions>
```

This type is a datatype, so you can declare assertions and use them later

```
<assertions id="project.assertions" >
  <enable project="org.apache.test" />
</assertions>

<assertions refid="project.assertions" />
```

Parameters:

- **refid** – Reference
- **enablesystemassertions** – Boolean

6.2 Task java:Javac

Compiles Java source files. This task can take the following arguments:

- sourcedir
- destdir
- deprecation
- classpath
- bootclasspath
- extdirs
- optimize
- debug
- encoding
- target
- depend
- verbose
- failonerror
- includeantruntime
- includejavaruntime
- source
- compiler

Of these arguments, the **sourcedir** and **destdir** are required.

When this task executes, it will recursively scan the sourcedir and destdir looking for Java source files to compile. This task makes its compile decision based on timestamp.

Parameters:

- **`target`** – String Sets the target VM that the classes will be compiled for. Valid values depend on the compiler, for jdk 1.4 the valid values are ”1.1”, ”1.2”, ”1.3”, ”1.4” and ”1.5”.
- **`excludes`** – String
- **`verbose`** – boolean If true, asks the compiler for verbose output.
- **`bootclasspathref`** – Reference Adds a reference to a classpath defined elsewhere.

- **compiler** – String Choose the implementation for this particular task.
- **executable** – String Sets the name of the javac executable.

  Ignored unless fork is true or extJavac has been specified as the compiler.
- **debuglevel** – String Keyword list to be appended to the -g command-line switch. This will be ignored by all implementations except modern and classic(ver >= 1.2). Legal values are none or a comma-separated list of the following keywords: lines, vars, and source. If debuglevel is not specified, by default, :none will be appended to -g. If debug is not turned on, this attribute will be ignored.
- **memoryinitialsize** – String The initial size of the memory for the underlying VM if javac is run externally; ignored otherwise. Defaults to the standard VM memory setting. (Examples: 83886080, 81920k, or 80m)
- **deprecation** – boolean Indicates whether source should be compiled with deprecation information; defaults to off.
- **includeantruntime** – boolean If true, includes Ant's own classpath in the classpath.
- **sourcepathref** – Reference Adds a reference to a source path defined elsewhere.
- **depend** – boolean Enables dependency-tracking for compilers that support this (jikes and classic).
- **includes** – String
- **source** – String Value of the -source command-line switch; will be ignored by all implementations except modern and jikes. If you use this attribute together with jikes, you must make sure that your version of jikes supports the -source switch. Legal values are 1.3, 1.4 and 1.5 - by default, no -source argument will be used at all.
- **includejavaruntime** – boolean If true, includes the Java runtime libraries in the classpath.
- **failonerror** – boolean Indicates whether the build will continue even if there are compilation errors; defaults to true.
- **destdir** – File Set the destination directory into which the Java source files should be compiled.
- **debug** – boolean Indicates whether source should be compiled with debug information; defaults to off.
- **tempdir** – File Where Ant should place temporary files.

- **`classpath`** – Path Set the classpath to be used for this compilation.
- **`followsymlinks`** – boolean
- **`casesensitive`** – boolean
- **`defaultexcludes`** – boolean
- **`bootclasspath`** – Path Sets the bootclasspath that will be used to compile the classes against.
- **`includesfile`** – File
- **`nowarn`** – boolean If true, enables the -nowarn option.
- **`proceed`** – boolean
- **`classpathref`** – Reference Adds a reference to a classpath defined elsewhere.
- **`fork`** – boolean If true, forks the javac compiler.
- **`srcdir`** – Path Set the source directories to find the source Java files.
- **`memorymaximumsize`** – String The maximum size of the memory for the underlying VM if javac is run externally; ignored otherwise. Defaults to the standard VM memory setting. (Examples: 83886080, 81920k, or 80m)
- **`optimize`** – boolean If true, compiles with optimization enabled.
- **`encoding`** – String Set the Java source file encoding name.
- **`excludesfile`** – File
- **`sourcepath`** – Path Set the sourcepath to be used for this compilation.
- **`listfiles`** – boolean If true, list the source files being handed off to the compiler.
- **`extdirs`** – Path Sets the extension directories that will be used during the compilation.

Parameters accepted as nested elements:

`<patternset>` – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **`includes`** – String
- **`refid`** – Reference

- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<extdirs> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String [”file”, ”dir”]

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
            >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
                >
                >
```

And the same rewritten as CustomSelector would be

```
                >
                >
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns ’true’ (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<src>` – *(See* **??** *on page* **??***)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

<or> (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

<contains> (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **refid** – Reference
- **ignorewhitespace** – boolean
- **text** – String

<depend> (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<classpath> – *(See* **??** *on page* **??***)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

`<sourcepath>` – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

`<different>` (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as

possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean
- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don’t get too caught up in boolean, there are other ways you can evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an ”all-but-one” selector, a ”weighted-average” selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference

- **expression** – String

<bootclasspath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference
- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String

- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<compilerarg> Adds an "compiler" attribute to Commandline$Attribute used to filter command line attributes based on the current implementation.

Parameters:

- **line** – String
- **implementation** – String
- **file** – File
- **compiler** – String
- **pathref** – Reference
- **value** – String
- **path** – Path

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String ["before", "after", "equal"]
- **error** – String

<excludesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

6.3 Task java:Javadoc

Generates Javadoc documentation for a collection of source code.

Current known limitations are:

- patterns must be of the form "xxx.*", every other pattern doesn't work.
- there is no control on arguments sanity since they are left to the javadoc implementation.

If no `doclet` is set, then the `version` and `author` are by default `"yes"`.

Note: This task is run on another VM because the Javadoc code calls `System.exit()` which would break Ant functionality.

Parameters:

- **bottom** – String Set the text to be placed at the bottom of each output file.
- **verbose** – boolean Run javadoc in verbose mode

- **bootclasspathref** – Reference Adds a reference to a CLASSPATH defined elsewhere.
- **executable** – String Sets the actual executable command to invoke, instead of the binary `javadoc` found in Ant's JDK.
- **header** – String Set the header text to be placed at the top of each output file.
- **link** – String Create links to javadoc output at the given URL.
- **noindex** – boolean Control generation of index.
- **access** – String [”protected”, ”public”, ”package”, ”private”]Set the scope to be processed. This is an alternative to the use of the setPublic, setPrivate, etc methods. It gives better build file control over what scope is processed.
- **doclet** – String Set the class that starts the doclet used in generating the documentation.
- **noqualifier** – String Enables the -noqualifier switch, will be ignored if javadoc is not the 1.4 version.
- **author** – boolean Include the author tag in the generated documentation.
- **public** – boolean Indicate whether only public classes and members are to be included in the scope processed
- **footer** – String Set the footer text to be placed at the bottom of each output file.
- **docletpath** – Path Set the classpath used to find the doclet class.
- **package** – boolean Indicate whether only package, protected and public classes and members are to be included in the scope processed
- **useexternalfile** – boolean Work around command line length limit by using an external file for the sourcefiles.
- **sourcepathref** – Reference Adds a reference to a CLASSPATH defined elsewhere.
- **protected** – boolean Indicate whether only protected and public classes and members are to be included in the scope processed
- **breakiterator** – boolean Enables the -linksource switch, will be ignored if javadoc is not the 1.4 version. Default is false
- **nonavbar** – boolean Control generation of the navigation bar.
- **maxmemory** – String Set the maximum memory to be used by the javadoc process
- **group** – String Group specified packages together in overview page.

- **source** – String Enables the -source switch, will be ignored if javadoc is not the 1.4 version.
- **linkoffline** – String Link to docs at "url" using package list at "url2" - separate the URLs by using a space character.
- **additionalparam** – String Set an additional parameter on the command line
- **linksource** – boolean Enables the -linksource switch, will be ignored if javadoc is not the 1.4 version. Default is false
- **failonerror** – boolean Should the build process fail if javadoc fails (as indicated by a non zero return code)?

  Default is false.
- **locale** – String Set the local to use in documentation generation.
- **destdir** – File Set the directory where the Javadoc output will be generated.
- **classpath** – Path Set the classpath to be used for this javadoc run.
- **defaultexcludes** – boolean Sets whether default exclusions should be used or not.
- **includenosourcepackages** – boolean If set to true, Ant will also accept packages that only hold package.html files but no Java sources.
- **bootclasspath** – Path Set the boot classpath to use.
- **private** – boolean Indicate whether all classes and members are to be included in the scope processed
- **nodeprecatedlist** – boolean Control deprecated list generation
- **charset** – String Charset for cross-platform viewing of generated documentation.
- **classpathref** – Reference Adds a reference to a CLASSPATH defined elsewhere.
- **stylesheetfile** – File Specifies the CSS stylesheet file to use.
- **docencoding** – String Output file encoding name.
- **excludepackagenames** – String Set the list of packages to be excluded.
- **docletpathref** – Reference Set the classpath used to find the doclet class by reference.
- **packagenames** – String Set the package names to be processed.

- **windowtitle** – String Set the title to be placed in the HTML <title>tag of the generated documentation.
- **notree** – boolean Control class tree generation.
- **splitindex** – boolean Generate a split index
- **packagelist** – String The name of a file containing the packages to process.
- **encoding** – String Set the encoding name of the source files,
- **doctitle** – String Set the title of the generated overview page.
- **serialwarn** – boolean Control warnings about serial tag.
- **old** – boolean Indicate whether Javadoc should produce old style (JDK 1.1) documentation. This is not supported by JDK 1.1 and has been phased out in JDK 1.4
- **use** – boolean Generate the "use" page for each package.
- **sourcepath** – Path Specify where to find source file
- **helpfile** – File Specifies the HTML help file to use.
- **version** – boolean Include the version tag in the generated documentation.
- **sourcefiles** – String Set the list of source files to process.
- **extdirs** – Path Set the location of the extensions directories.
- **nohelp** – boolean Control generation of help link.
- **nodeprecated** – boolean Control deprecation infromation
- **overview** – File Specify the file containing the overview to be included in the generated documentation.

Parameters accepted as nested elements:

<taglet> – *(See Type java:Javadoc.ExtensionInfo on page 137)* A project aware class used for Javadoc extensions which take a name and a path such as doclet and taglet arguments.

Parameters:

- **name** – String
- **pathref** – Reference
- **path** – Path

<doclet> – *(See Type java:Javadoc.DocletInfo on page 136)* This class stores info about doclets.

Parameters:

- **name** – String
- **pathref** – Reference
- **path** – Path

<package> Used to track info about the packages to be javadoc'd

Parameters:

- **name** – String

<arg> Used for nested xml command line definitions.

Parameters:

- **line** – String
- **file** – File
- **pathref** – Reference
- **value** – String
- **path** – Path

<link> Represents a link triplet (href, whether link is offline, location of the package list if off line)

Parameters:

- **href** – String
- **offline** – boolean
- **packagelistloc** – File
- **resolvelink** – boolean

<doctitle> An HTML element in the javadoc. This class is used for those javadoc elements which contain HTML such as footers, headers, etc.

Parameters:

<bottom> An HTML element in the javadoc. This class is used for those javadoc elements which contain HTML such as footers, headers, etc.

Parameters:

<group> – *(See Type java:Javadoc.GroupArgument on page 138)*

Parameters:

- **packages** – String

- **title** – String

<packageset> – *(See ?? on page ??)* Subclass as hint for supporting tasks that the included directories instead of files should be used.

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<classpath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
```

```
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

`<sourcepath>` – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

`<footer>` An HTML element in the javadoc. This class is used for those javadoc elements which contain HTML such as footers, headers, etc.

Parameters:

<source> This class is used to manage the source files to be processed.

Parameters:

- **file** – File

<bootclasspath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<header> An HTML element in the javadoc. This class is used for those javadoc elements which contain HTML such as footers, headers, etc.

Parameters:

<excludepackage> Used to track info about the packages to be javadoc'd

Parameters:

- **name** – String

<tag> – *(See Type java:Javadoc.TagArgument on page 138)* Class representing a -tag argument.

Parameters:

- **refid** – Reference
- **name** – String

- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **enabled** – boolean
- **casesensitive** – boolean
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **scope** – String
- **includes** – String

6.4 Type java:Javadoc.DocletInfo

This class stores info about doclets.

Parameters:

- **name** – String
- **pathref** – Reference
- **path** – Path

Parameters accepted as nested elements:

<path> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

`<param>` Inner class used to manage doclet parameters.

Parameters:

- **`name`** – String
- **`value`** – String

6.5 Type java:Javadoc.ExtensionInfo

A project aware class used for Javadoc extensions which take a name and a path such as doclet and taglet arguments.

Parameters:

- **`name`** – String Set the name of the extension
- **`pathref`** – Reference Adds a reference to a CLASSPATH defined elsewhere.
- **`path`** – Path Set the path to use when loading the component.

Parameters accepted as nested elements:

`<path>` – *(See* **??** *on page* **??***)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
```

```
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

6.6 Type java:Javadoc.GroupArgument

Parameters:

- **`packages`** – String
- **`title`** – String

Parameters accepted as nested elements:

`<package>` Used to track info about the packages to be javadoc'd

Parameters:

- **`name`** – String

`<title>` An HTML element in the javadoc. This class is used for those javadoc elements which contain HTML such as footers, headers, etc.

Parameters:

6.7 Type java:Javadoc.TagArgument

Class representing a -tag argument.

Parameters:

- **`refid`** – Reference
- **`name`** – String Sets the name of the tag.
- **`file`** – File
- **`dir`** – File
- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean
- **`enabled`** – boolean Sets whether or not the tag is enabled.
- **`casesensitive`** – boolean
- **`excludesfile`** – File
- **`includesfile`** – File
- **`excludes`** – String
- **`scope`** – String Sets the scope of the tag. This is in comma-separated form, with each element being one of ”all” (the default), ”overview”, ”packages”, ”types”, ”constructors”, ”methods”, ”fields”. The elements are treated in a case-insensitive manner.
- **`includes`** – String

Parameters accepted as nested elements:

`<patternset>` – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **`includes`** – String
- **`refid`** – Reference
- **`excludesfile`** – File
- **`includesfile`** – File
- **`excludes`** – String

`<exclude>` inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String [”file”, ”dir”]

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
>
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
>
>
```

And the same rewritten as CustomSelector would be

```
>
>
>
>
>
>
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns 'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean
- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don't get too caught up in boolean, there are other ways you can

evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an "all-but-one" selector, a "weighted-average" selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **`error`** – String
- **`refid`** – Reference
- **`allowtie`** – boolean

`<containsregexp>` (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **`error`** – String
- **`refid`** – Reference
- **`expression`** – String

`<filename>` (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`name`** – String
- **`refid`** – Reference
- **`negate`** – boolean

`<selector>` (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **`error`** – String
- **`refid`** – Reference
- **`unless`** – String
- **`if`** – String

`<includesfile>` inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String [”before”, ”after”, ”equal”]
- **error** – String

<excludesfile> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

6.8 Task java:ManifestTask

Creates a manifest file for inclusion in a JAR, Ant task wrapper around {@link Manifest Manifest}. This task can be used to write a Manifest file, optionally replacing or updating an existing file.

Parameters:

- **`file`** – File The name of the manifest file to create/update. Required if used as a task.
- **`encoding`** – String The encoding to use for reading in an existing manifest file
- **`mode`** – String [”update”, ”replace”]Update policy: either ”update” or ”replace”; default is ”replace”.

Parameters accepted as nested elements:

`<attribute>` An attribute for the manifest. Those attributes that are not nested into a section will be added to the ”Main” section.

Parameters:

- **`name`** – String
- **`value`** – String

`<section>` – *(See* **??** *on page* **??***)* A manifest section - you can nest attribute elements into sections. A section consists of a set of attribute values, separated from other sections by a blank line.

Parameters:

- **`name`** – String

6.9 Task java:Rmic

Runs the rmic compiler against classes.

Rmic can be run on a single class (as specified with the classname attribute) or a number of classes at once (all classes below base that are neither _Stub nor _Skel classes). If you want to rmic a single class and this class is a class nested into another class, you have to specify the classname in the form `Outer$$Inner` instead of `Outer.Inner`.

It is possible to refine the set of files that are being rmiced. This can be done with the *includes*, *includesfile*, *excludes*, *excludesfile* and *defaultexcludes* attributes. With the *includes* or *includesfile* attribute you specify the files you want to have included by using patterns. The *exclude* or *excludesfile* attribute is used to specify the files you want to have excluded. This is also done with patterns. And finally with the *defaultexcludes* attribute, you can specify whether you want to use default exclusions or not. See the section on directory based tasks, on how the inclusion/exclusion of files works, and how to write patterns.

This task forms an implicit FileSet and supports all attributes of <`fileset`> (`dir` becomes `base`) as well as the nested <`include`>, <`exclude`> and <`patternset`> elements.

It is possible to use different compilers. This can be selected with the "build.rmic" property or the `compiler` attribute. :

- sun (the standard compiler of the JDK)
- kaffe (the standard compiler of {@link Kaffe (at http://www.kaffe.org)})
- weblogic

The miniRMI (at http://dione.zcu.cz/ toman40/miniRMI/) project contains a compiler implementation for this task as well, please consult miniRMI's documentation to learn how to use it.

Parameters:

- **`verify`** – boolean Flag to enable verification so that the classes found by the directory match are checked to see if they implement java.rmi.Remote. optional; This defaults to false if not set.
- **`excludes`** – String
- **`iiop`** – boolean Indicates that IIOP compatible stubs should be generated; optional, defaults to false if not set.
- **`idlopts`** – String pass additional arguments for IDL compile
- **`extdirs`** – Path Sets the extension directories that will be used during the compilation; optional.
- **`sourcebase`** – File optional directory to save generated source files to.
- **`classpathref`** – Reference Adds to the classpath a reference to a <path>defined elsewhere.
- **`base`** – File Sets the location to store the compiled files; required

- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **includeantruntime** – boolean Sets whether or not to include ant's own classpath in this task's classpath. Optional; default is `true`.
- **includejavaruntime** – boolean task's classpath. Enables or disables including the default run-time libraries from the executing VM; optional, defaults to false
- **classpath** – Path Set the classpath to be used for this compilation.
- **casesensitive** – boolean
- **debug** – boolean Generate debug info (passes -g to rmic); optional, defaults to false
- **compiler** – String Sets the compiler implementation to use; optional, defaults to the value of the `build.rmic` property, or failing that, default compiler for the current VM
- **includes** – String
- **includesfile** – File
- **idl** – boolean Indicates that IDL output should be generated. This defaults to false if not set.
- **iiopopts** – String Set additional arguments for iiop
- **classname** – String Sets the class to run `rmic` against; optional
- **excludesfile** – File
- **filtering** – boolean Sets token filtering [optional], default=false
- **stubversion** – String Specify the JDK version for the generated stub code. Specify "1.1" to pass the "-v1.1" option to rmic.

Parameters accepted as nested elements:

<patternset> – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference

- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<extdirs> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String [”file”, ”dir”]

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
            >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
                >
                >
```

And the same rewritten as CustomSelector would be

```
                >
                >
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns ’true’ (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<classpath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean
- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don’t get too caught up in boolean, there are other ways you can evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an ”all-but-one” selector, a ”weighted-average” selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference
- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<compilerarg> Adds an "compiler" attribute to Commandline$Attribute used to filter command line attributes based on the current implementation.

Parameters:

- **line** – String
- **implementation** – String
- **file** – File
- **compiler** – String
- **pathref** – Reference
- **value** – String
- **path** – Path

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String [”before”, ”after”, ”equal”]
- **error** – String

<excludesfile> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

6.10 Task java:SignJar

Signs JAR or ZIP files with the javasign command line tool. The tool detailed dependency checking: files are only signed if they are not signed. The `signjar` attribute can point to the file to generate; if this file exists then its modification date is used as a cue as to whether to resign any JAR file.

Parameters:

- **`storepass`** – String password for keystore integrity; required
- **`lazy`** – boolean flag to control whether the presence of a signature file means a JAR is signed; optional, default false
- **`jar`** – File the jar file to sign; required
- **`verbose`** – boolean Enable verbose output when signing ; optional: default false
- **`keypass`** – String password for private key (if different); optional
- **`maxmemory`** – String Set the maximum memory to be used by the jarsigner process
- **`alias`** – String the alias to sign under; required
- **`sectionsonly`** – boolean flag to compute hash of entire manifest; optional, default false
- **`keystore`** – String keystore location; required
- **`sigfile`** – String name of .SF/.DSA file; optional
- **`internalsf`** – boolean Flag to include the .SF file inside the signature; optional; default false
- **`storetype`** – String keystore type; optional
- **`signedjar`** – File name of signed JAR file; optional

Parameters accepted as nested elements:

`<fileset>` – *(See* **??** *on page* **??***)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **`refid`** – Reference
- **`file`** – File
- **`dir`** – File
- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean
- **`casesensitive`** – boolean
- **`excludes`** – String
- **`includesfile`** – File

- **excludesfile** – File
- **includes** – String

6.11 Task java:genkey

Generates a key in a keystore.

Parameters:

- **storepass** – String Password for keystore integrity. Must be at least 6 characters long.
- **sigalg** – String The algorithm to use in signing.
- **verbose** – boolean If true, verbose output when signing.
- **keypass** – String Password for private key (if different).
- **dname** – String The distinguished name for entity.
- **keysize** – String Indicates the size of key generated.
- **alias** – String The alias to add under.
- **keyalg** – String The method to use when generating name-value pair.
- **validity** – String Indicates how many days certificate is valid.
- **keystore** – String Keystore location.
- **storetype** – String Keystore type.

Parameters accepted as nested elements:

<dname> – *(See Type java:genkey.DistinguishedName on page 163)*

Parameters:

6.12 Type java:genkey.DistinguishedName

Parameters:

Parameters accepted as nested elements:

<param> (Of type Object)
Parameters:

---

7 network Library

7.1 Task network:Get

Gets a particular file from a URL source. Options include verbose reporting, timestamp based fetches and controlling actions on failures. NB: access through a firewall only works if the whole Java runtime is correctly configured.

Parameters:

- **src** – URL Set the URL to get.
- **verbose** – boolean If true, show verbose progress information.
- **ignoreerrors** – boolean If true, log errors but do not treat as fatal.
- **username** – String Username for basic auth.
- **usetimestamp** – boolean If true, conditionally download a file based on the timestamp of the local copy.

  In this situation, the if-modified-since header is set so that the file is only fetched if it is newer than the local file (or there is no local file) This flag is only valid on HTTP connections, it is ignored in other cases. When the flag is set, the local copy of the downloaded file will also have its timestamp set to the remote file time.

  Note that remote files of date 1/1/1970 (GMT) are treated as 'no timestamp', and web servers often serve files with a timestamp in the future by replacing their timestamp with that of the current time. Also, inter-computer clock differences can cause no end of grief.
- **dest** – File Where to copy the source file.
- **password** – String password for the basic authentication.

7.2 Task network:mail

A task to send SMTP email. This task can send mail using either plain text, UU encoding or Mime format mail depending on what is available. Attachments may be sent using nested FileSet elements.

Parameters:

- **`cclist`** – String
- **`messagefile`** – File
- **`password`** – String
- **`encoding`** – String [”auto”, ”mime”, ”uu”, ”plain”]
- **`charset`** – String
- **`files`** – String
- **`subject`** – String
- **`message`** – String
- **`ssl`** – boolean
- **`tolist`** – String
- **`mailport`** – int Sets the mailport parameter of this build task.
- **`messagemimetype`** – String
- **`from`** – String
- **`bcclist`** – String
- **`replyto`** – String
- **`mailhost`** – String
- **`failonerror`** – boolean
- **`includefilenames`** – boolean
- **`user`** – String

Parameters accepted as nested elements:

`<fileset>` – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **`refid`** – Reference
- **`file`** – File
- **`dir`** – File
- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean
- **`casesensitive`** – boolean
- **`excludes`** – String
- **`includesfile`** – File
- **`excludesfile`** – File
- **`includes`** – String

`<bcc>` (Of type EmailAddress) Holds an email address.

Parameters:

- **`address`** – String
- **`name`** – String

`<message>` (Of type Message) Class representing an email message.

Parameters:

- **`mimetype`** – String
- **`src`** – File
- **`charset`** – String

`<replyto>` (Of type EmailAddress) Holds an email address.

Parameters:

- **`address`** – String
- **`name`** – String

`<cc>` (Of type EmailAddress) Holds an email address.

Parameters:

- **`address`** – String
- **`name`** – String

<to> (Of type EmailAddress) Holds an email address.

Parameters:

- **address** – String
- **name** – String

<from> (Of type EmailAddress) Holds an email address.

Parameters:

- **address** – String
- **name** – String

---

8 packaging Library

8.1 Task packaging:BUnzip2

Expands a file that has been compressed with the BZIP2 algorithm. Normally used to compress non-compressed archives such as TAR files.

Parameters:

- **dest** – File
- **src** – File

8.2 Task packaging:BZip2

Compresses a file with the BZIP2 algorithm. Normally used to compress non-compressed archives such as TAR files.

Parameters:

- **zipfile** – File
- **src** – File
- **destfile** – File

8.3 Task packaging:Ear

Creates a EAR archive. Based on WAR task

Parameters:

- **`keepcompression`** – boolean
- **`duplicate`** – String [”add”, ”preserve”, ”fail”]
- **`whenempty`** – String [”fail”, ”skip”, ”create”]
- **`comment`** – String
- **`roundup`** – boolean
- **`excludes`** – String
- **`basedir`** – File
- **`encoding`** – String
- **`manifest`** – File
- **`jarfile`** – File
- **`defaultexcludes`** – boolean
- **`index`** – boolean
- **`update`** – boolean
- **`followsymlinks`** – boolean
- **`destfile`** – File
- **`casesensitive`** – boolean
- **`earfile`** – File
- **`includes`** – String
- **`includesfile`** – File
- **`appxml`** – File File to incorporate as application.xml.
- **`compress`** – boolean
- **`manifestencoding`** – String
- **`filesonly`** – boolean

- **file** – File
- **excludesfile** – File
- **filesetmanifest** – String [”skip”, ”merge”, ”mergewithoutmain”]
- **zipfile** – File

Parameters accepted as nested elements:

<patternset> – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String [”srconly”, ”both”]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<manifest> – *(See* **??** *on page* **??***)* Holds the data of a jar manifest. Manifests are processed according to the {@link Jar file specification. (at http://java.sun.com/j2se/1.4/docs/guide/jar/jar.html)}. Specifically, a manifest element consists of a set of attributes and sections. These sections in turn may contain attributes. Note in particular that this may result in manifest lines greater than 72 bytes being wrapped and continued on the next line. If an application can not handle the continuation mechanism, it is a defect in the application, not this task.

Parameters:

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String ["file", "dir"]

<metainf> – *(See ?? on page ??)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File
- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **includes** – String
- **dirmode** – String

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
          >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
              >
              >
```

And the same rewritten as CustomSelector would be

```
              >
              >
              >
              >
              >
              >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns 'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean

- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don’t get too caught up in boolean, there are other ways you can evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an ”all-but-one” selector, a ”weighted-average” selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<archives> – *(See* **??** *on page* **??***)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **includes** – String
- **dirmode** – String

<indexjars> – *(See* **??** *on page* **??***)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference
- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<zipgroupfileset> – *(See* **??** *on page* **??***)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File

- **includes** – String

<zipfileset> – *(See ?? on page ??)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File
- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **includes** – String
- **dirmode** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String [”before”, ”after”, ”equal”]
- **error** – String

<excludesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

8.4 Task packaging:GUnzip

Expands a file that has been compressed with the GZIP algorithm. Normally used to compress non-compressed archives such as TAR files.

Parameters:

- **dest** – File
- **src** – File

8.5 Task packaging:GZip

Compresses a file with the GZIP algorithm. Normally used to compress non-compressed archives such as TAR files.

Parameters:

- **`zipfile`** – File
- **`src`** – File
- **`destfile`** – File

8.6 Task packaging:Jar

Creates a JAR archive.

Parameters:

- **`whenempty`** – String [”fail”, ”skip”, ”create”]
- **`keepcompression`** – boolean
- **`comment`** – String
- **`duplicate`** – String [”add”, ”preserve”, ”fail”]
- **`roundup`** – boolean
- **`excludes`** – String
- **`basedir`** – File
- **`encoding`** – String
- **`manifest`** – File The manifest file to use. This can be either the location of a manifest, or the name of a jar added through a fileset. If its the name of an added jar, the task expects the manifest to be in the jar at META-INF/MANIFEST.MF.
- **`jarfile`** – File
- **`defaultexcludes`** – boolean
- **`index`** – boolean Set whether or not to create an index list for classes. This may speed up classloading in some cases.
- **`update`** – boolean
- **`followsymlinks`** – boolean
- **`destfile`** – File
- **`casesensitive`** – boolean

- **includes** – String
- **includesfile** – File
- **compress** – boolean
- **manifestencoding** – String Set whether or not to create an index list for classes. This may speed up classloading in some cases.
- **filesonly** – boolean
- **file** – File
- **excludesfile** – File
- **filesetmanifest** – String [”skip”, ”merge”, ”mergewithoutmain”]Behavior when a Manifest is found in a zipfileset or zipgroupfileset file. Valid values are ”skip”, ”merge”, and ”mergewithoutmain”. ”merge” will merge all of manifests together, and merge this into any other specified manifests. ”mergewithoutmain” merges everything but the Main section of the manifests. Default value is ”skip”. Note: if this attribute’s value is not ”skip”, the created jar will not be readable by using java.util.jar.JarInputStream
- **zipfile** – File

Parameters accepted as nested elements:

<patternset> – *(See ?? on page ??)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String

- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<manifest> – *(See ?? on page ??)* Holds the data of a jar manifest. Manifests are processed according to the {@link Jar file specification. (at http://java.sun.com/j2se/1.4/docs/guide/jar/jar.html)}. Specifically, a manifest element consists of a set of attributes and sections. These sections in turn may contain attributes. Note in particular that this may result in manifest lines greater than 72 bytes being wrapped and continued on the next line. If an application can not handle the continuation mechanism, it is a defect in the application, not this task.

Parameters:

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String [”file”, ”dir”]

<metainf> – *(See ?? on page ??)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File
- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File

- **excludes** – String
- **includes** – String
- **dirmode** – String

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
            >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
                >
                >
```

And the same rewritten as CustomSelector would be

```
                >
                >
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns 'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean

- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don’t get too caught up in boolean, there are other ways you can evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an ”all-but-one” selector, a ”weighted-average” selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<indexjars> – *(See* **??** *on page* **??***)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
```

```
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

`<filename>` (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`name`** – String
- **`refid`** – Reference
- **`negate`** – boolean

`<selector>` (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **`error`** – String
- **`refid`** – Reference
- **`unless`** – String
- **`if`** – String

`<includesfile>` inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **`name`** – String

- **unless** – String
- **if** – String

<zipgroupfileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<zipfileset> – *(See ?? on page ??)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File
- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

- **includes** – String
- **dirmode** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String [”before”, ”after”, ”equal”]
- **error** – String

<excludesfile> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

8.7 Task packaging:Tar

Creates a tar archive.

Parameters:

- **`tarfile`** – File Set is the name/location of where to create the tar file.
- **`destfile`** – File Set is the name/location of where to create the tar file.
- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean
- **`basedir`** – File This is the base directory to look in for things to tar.
- **`casesensitive`** – boolean
- **`excludesfile`** – File
- **`excludes`** – String
- **`includesfile`** – File
- **`compression`** – String [”none”, ”gzip”, ”bzip2”]Set compression method. Allowable values are
  - none - no compression
  - gzip - Gzip compression
  - bzip2 - Bzip2 compression
- **`longfile`** – String [”warn”, ”fail”, ”truncate”, ”gnu”, ”omit”]Set how to handle long files, those with a path>100 chars. Optional, default=warn.

  Allowable values are
  - truncate - paths are truncated to the maximum length
  - fail - paths greater than the maximum cause a build exception
  - warn - paths greater than the maximum cause a warning and GNU is used
  - gnu - GNU extensions are used for any paths greater than the maximum.
  - omit - paths greater than the maximum are omitted from the archive
- **`includes`** – String

Parameters accepted as nested elements:

<patternset> – *(See ?? on page ??)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<tarfileset> – *(See Type packaging:Tar.TarFileSet on page 204)* This is a FileSet with the option to specify permissions and other attributes.

Parameters:

- **excludes** – String
- **preserveleadingslashes** – boolean
- **username** – String
- **defaultexcludes** – boolean
- **uid** – int
- **fullpath** – String
- **group** – String
- **followsymlinks** – boolean
- **refid** – Reference
- **casesensitive** – boolean
- **mode** – String
- **gid** – int
- **dir** – File
- **includes** – String
- **prefix** – String
- **dirmode** – String
- **includesfile** – File
- **file** – File
- **excludesfile** – File

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String [”file”, ”dir”]

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
        >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
            >
            >
```

And the same rewritten as CustomSelector would be

```
            >
```

```
                >
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns 'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


            >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean
- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don't get too caught up in boolean, there are other ways you can

evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an "all-but-one" selector, a "weighted-average" selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference
- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String [”before”, ”after”, ”equal”]
- **error** – String

<excludesfile> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

8.8 Type packaging:Tar.TarFileSet

This is a FileSet with the option to specify permissions and other attributes.

Parameters:

- **`excludes`** – String
- **`preserveleadingslashes`** – boolean Flag to indicates whether leading '/'s should be preserved in the file names. Optional, default is `false`.
- **`username`** – String The username for the tar entry This is not the same as the UID.
- **`defaultexcludes`** – boolean
- **`uid`** – int The uid for the tar entry This is not the same as the User name.
- **`fullpath`** – String If the fullpath attribute is set, the file in the fileset is written with that path in the archive. The prefix attribute, if specified, is ignored. It is an error to have more than one file specified in such a fileset.
- **`group`** – String The groupname for the tar entry; optional, default="" This is not the same as the GID.
- **`followsymlinks`** – boolean
- **`refid`** – Reference
- **`casesensitive`** – boolean
- **`mode`** – String A 3 digit octal string, specify the user, group and other modes in the standard Unix fashion; optional, default=0644
- **`gid`** – int The GID for the tar entry; optional, default="0" This is not the same as the group name.
- **`dir`** – File
- **`includes`** – String
- **`prefix`** – String If the prefix attribute is set, all files in the fileset are prefixed with that path in the archive. optional.
- **`dirmode`** – String A 3 digit octal string, specify the user, group and other modes in the standard Unix fashion; optional, default=0755
- **`includesfile`** – File
- **`file`** – File
- **`excludesfile`** – File

Parameters accepted as nested elements:

<patternset> – *(See ?? on page ??)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String ["file", "dir"]

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
          >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
              >
              >
```

And the same rewritten as CustomSelector would be

```
              >
              >
              >
              >
              >
              >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns 'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
    generate the site using forrest
  >

    upload the changed files


          >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean
- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don't get too caught up in boolean, there are other ways you can

evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an "all-but-one" selector, a "weighted-average" selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference
- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String [”before”, ”after”, ”equal”]
- **error** – String

<excludesfile> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

8.9 Task packaging:Untar

Untar a file.

For JDK 1.1 ”last modified time” field is set to current time instead of being carried from the archive file.

PatternSets are used to select files to extract *from* the archive. If no patternset is used, all files are extracted.

FileSet>s may be used to select archived files to perform unarchival upon.

File permissions will not be restored on extracted files.

The untar task recognizes the long pathname entries used by GNU tar.

Parameters:

- **`compression`** – String [”none”, ”gzip”, ”bzip2”]Set decompression algorithm to use; default=none. Allowable values are
  - none - no compression
  - gzip - Gzip compression
  - bzip2 - Bzip2 compression
- **`overwrite`** – boolean
- **`dest`** – File
- **`encoding`** – String No encoding support in Untar.
- **`src`** – File

Parameters accepted as nested elements:

`<fileset>` – *(See* **??** *on page* **??***)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **`refid`** – Reference
- **`file`** – File
- **`dir`** – File
- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean
- **`casesensitive`** – boolean
- **`excludes`** – String

- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<patternset> – *(See ?? on page ??)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

8.10 Task packaging:War

An extension of <jar>to create a WAR archive. Contains special treatment for files that should end up in the `WEB-INF/lib`, `WEB-INF/classes` or `WEB-INF` directories of the Web Application Archive.

(The War task is a shortcut for specifying the particular layout of a WAR file. The same thing can be accomplished by using the *prefix* and *fullpath* attributes of zipfilesets in a Zip or Jar task.)

The extended zipfileset element from the zip task (with attributes *prefix*, *fullpath*, and *src*) is available in the War task.

Parameters:

- **keepcompression** – boolean
- **duplicate** – String [”add”, ”preserve”, ”fail”]
- **whenempty** – String [”fail”, ”skip”, ”create”]
- **comment** – String
- **roundup** – boolean
- **excludes** – String

- **`basedir`** – File
- **`encoding`** – String
- **`manifest`** – File
- **`jarfile`** – File
- **`defaultexcludes`** – boolean
- **`index`** – boolean
- **`update`** – boolean
- **`followsymlinks`** – boolean
- **`destfile`** – File
- **`casesensitive`** – boolean
- **`includes`** – String
- **`webxml`** – File set the deployment descriptor to use (WEB-INF/web.xml); required unless `update=true`
- **`includesfile`** – File
- **`compress`** – boolean
- **`manifestencoding`** – String
- **`filesonly`** – boolean
- **`warfile`** – File *Deprecatedname of the file to create -use* `destfile` *instead.*
- **`file`** – File
- **`excludesfile`** – File
- **`filesetmanifest`** – String [”skip”, ”merge”, ”mergewithoutmain”]
- **`zipfile`** – File

Parameters accepted as nested elements:

`<patternset>` – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference

- **present** – String [”srconly”, ”both”]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<manifest> – *(See* **??** *on page* **??***)* Holds the data of a jar manifest. Manifests are processed according to the {@link Jar file specification. (at http://java.sun.com/j2se/1.4/docs/guide/jar/jar.html)}. Specifically, a manifest element consists of a set of attributes and sections. These sections in turn may contain attributes. Note in particular that this may result in manifest lines greater than 72 bytes being wrapped and continued on the next line. If an application can not handle the continuation mechanism, it is a defect in the application, not this task.

Parameters:

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String [”file”, ”dir”]

<metainf> – *(See* **??** *on page* **??***)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File
- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **includes** – String
- **dirmode** – String

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
            >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
                >
                >
```

And the same rewritten as CustomSelector would be

```
                >
                >
```

```
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns 'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<lib> – *(See ?? on page ??)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File
- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **includes** – String
- **dirmode** – String

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String

- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean
- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don't get too caught up in boolean, there are other ways you can

evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an "all-but-one" selector, a "weighted-average" selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<indexjars> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference
- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<zipgroupfileset> – *(See* **??** *on page* **??***)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean

- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<zipfileset> – *(See ?? on page ??)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File
- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **includes** – String
- **dirmode** – String

<classes> – *(See ?? on page ??)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File
- **refid** – Reference

- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **includes** – String
- **dirmode** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String [”before”, ”after”, ”equal”]
- **error** – String

<excludesfile> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<webinf> – *(See ?? on page ??)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File
- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **includes** – String
- **dirmode** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

8.11 Task packaging:Zip

Create a Zip file.

Parameters:

- **`whenempty`** – String ["fail", "skip", "create"]Sets behavior of the task when no files match. Possible values are: `fail` (throw an exception and halt the build); `skip` (do not create any archive, but issue a warning); `create` (make an archive with no entries). Default for zip tasks is `skip`; for jar tasks, `create`.
- **`keepcompression`** – boolean Whether the original compression of entries coming from a ZIP archive should be kept (for example when updating an archive).
- **`comment`** – String Comment to use for archive.
- **`duplicate`** – String ["add", "preserve", "fail"]Sets behavior for when a duplicate file is about to be added - one of `keep`, `skip` or `overwrite`. Possible values are: `keep` (keep both of the files); `skip` (keep the first version of the file found); `overwrite` overwrite the file with the new file Default for zip tasks is `keep`
- **`roundup`** – boolean Whether the file modification times will be rounded up to the next even number of seconds.

  Zip archives store file modification times with a granularity of two seconds, so the times will either be rounded up or down. If you round down, the archive will always seem out-of-date when you rerun the task, so the default is to round up. Rounding up may lead to a different type of problems like JSPs inside a web archive that seem to be slightly more recent than precompiled pages, rendering precompilation useless.
- **`excludes`** – String
- **`basedir`** – File Directory from which to archive files; optional.
- **`encoding`** – String Encoding to use for filenames, defaults to the platform's default encoding.

  For a list of possible values see http://java.sun.com/products/jdk/1.2/docs/guide/internat/encodir
  (at http://java.sun.com/products/jdk/1.2/docs/guide/internat/encoding.doc.html).
- **`defaultexcludes`** – boolean
- **`update`** – boolean If true, updates an existing file, otherwise overwrite any existing one; optional defaults to false.
- **`followsymlinks`** – boolean
- **`destfile`** – File The file to create; required.

- **casesensitive** – boolean
- **includes** – String
- **includesfile** – File
- **compress** – boolean Whether we want to compress the files or only store them; optional, default=true;
- **filesonly** – boolean If true, emulate Sun's jar utility by not adding parent directories; optional, defaults to false.
- **file** – File This is the name/location of where to create the file.
- **excludesfile** – File
- **zipfile** – File This is the name/location of where to create the .zip file.

Parameters accepted as nested elements:

<patternset> – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String [”file”, ”dir”]

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
            >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
                >
                >
```

And the same rewritten as CustomSelector would be

```
                >
                >
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class with its MD5-Algorithm and its standard provider. The new computed hashvalue and the stored one are compared by the *EqualComparator* which returns ’true’ (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException ”Invalid parameter” is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String [”equal”, ”rule”]
- **`seldirs`** – boolean
- **`algorithm`** – String [”hashvalue”, ”digest”]
- **`cache`** – String [”propertyfile”]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

`<depend>` (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **`error`** – String

- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<fileset> – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean

- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don’t get too caught up in boolean, there are other ways you can evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an ”all-but-one” selector, a ”weighted-average” selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference

- **negate** – boolean

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<zipgroupfileset> – *(See* **??** *on page* **??***)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **casesensitive** – boolean
- **excludes** – String
- **includesfile** – File
- **excludesfile** – File
- **includes** – String

<zipfileset> – *(See* **??** *on page* **??***)* A ZipFileSet is a FileSet with extra attributes useful in the context of Zip/Jar tasks. A ZipFileSet extends FileSets with the ability to extract a subset of the entries of a Zip file for inclusion in another Zip file. It also includes a prefix attribute which is prepended to each entry in the

output Zip file. Since ant 1.6 ZipFileSet can be defined with an id and referenced in packaging tasks

Parameters:

- **src** – File
- **refid** – Reference
- **file** – File
- **dir** – File
- **defaultexcludes** – boolean
- **followsymlinks** – boolean
- **filemode** – String
- **prefix** – String
- **casesensitive** – boolean
- **fullpath** – String
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String
- **includes** – String
- **dirmode** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String [”before”, ”after”, ”equal”]
- **error** – String

<excludesfile> inner class to hold a name on list. ”If” and ”Unless” attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String

- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

8.12 Task packaging:unwar

Unzip a file.

Parameters:

- **overwrite** – boolean Should we overwrite files in dest, even if they are newer than the corresponding entries in the archive?
- **dest** – File Set the destination directory. File will be unzipped into the destination directory.
- **encoding** – String Sets the encoding to assume for file names and comments.

  Set to `native-encoding` if you want your platform's native encoding, defaults to UTF8.
- **src** – File Set the path to zip-file.

Parameters accepted as nested elements:

`<fileset>` – *(See ?? on page ??)* Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **`refid`** – Reference
- **`file`** – File
- **`dir`** – File
- **`defaultexcludes`** – boolean
- **`followsymlinks`** – boolean
- **`casesensitive`** – boolean
- **`excludes`** – String
- **`includesfile`** – File
- **`excludesfile`** – File
- **`includes`** – String

`<patternset>` – *(See ?? on page ??)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **`includes`** – String
- **`refid`** – Reference
- **`excludesfile`** – File
- **`includesfile`** – File
- **`excludes`** – String

9 property Library

9.1 Task property:Basename

Sets a property to the base name of a specified file, optionally minus a suffix. This task can accept the following attributes:

- file
- property

- suffix

The **file** and **property** attributes are required. The **suffix** attribute can be specified either with or without the ".", and the result will be the same (ie., the returned file name will be minus the .suffix).

When this task executes, it will set the specified property to the value of the last element in the specified file. If file is a directory, the basename will be the last directory element. If file is a full-path filename, the basename will be the simple file name. If a suffix is specified, and the specified file ends in that suffix, the basename will be the simple file name without the suffix.

Parameters:

- **`file`** – File file or directory to get base name from
- **`property`** – String Property to set base name to.
- **`suffix`** – String Optional suffix to remove from base name.

9.2 Task property:Dirname

Determines the directory name of the specified file. This task can accept the following attributes:

- file
- property

Both **file** and **property** are required.

When this task executes, it will set the specified property to the value of the specified file up to, but not including, the last path element. If file is a file, the directory will be the current directory.

Parameters:

- **`file`** – File Path to take the dirname of.
- **`property`** – String The name of the property to set.

---

10 scm Library

10.1 Task scm:CVSPass

Adds an new entry to a CVS password file.

Parameters:

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

10.2 Task scm:Cvs

Performs operations on a CVS repository. original 1.20 NOTE: This implementation has been moved to AbstractCvsTask with the addition of some accessors for extensibility.

Parameters:

- **`cvsrsh`** – String
- **`quiet`** – boolean
- **`passfile`** – File
- **`command`** – String
- **`package`** – String
- **`cvsroot`** – String
- **`reallyquiet`** – boolean
- **`output`** – File
- **`port`** – int
- **`compressionlevel`** – int
- **`dest`** – File
- **`noexec`** – boolean
- **`error`** – File

- **failonerror** – boolean
- **append** – boolean
- **date** – String
- **compression** – boolean
- **tag** – String

Parameters accepted as nested elements:

<commandline> – *(See ?? on page ??)* Commandline objects help handling command lines specifying processes to execute. The class can be used to define a command line as nested elements or as a helper to define a command line by an application.

```
<someelement>
  <acommandline executable="/executable/to/run">
    <argument value="argument 1" />
    <argument line="argument_1 argument_2 argument_3" />
    <argument value="argument 4" />
  </acommandline>
</someelement>
```

The element `someelement` must provide a method `createAcommandline` which returns an instance of this class.

Parameters:

- **executable** – String

---

11 utility Library

11.1 Task utility:DefaultExcludes

Alters the default excludes for the **entire** build..

Parameters:

- **remove** – String Pattern to remove from the default excludes.
- **echo** – boolean If true, echo the default excludes.
- **default** – boolean go back to standard default patterns
- **add** – String Pattern to add to the default excludes

11.2 Task utility:Echo

Writes a message to the Ant logging facilities.

Parameters:

- **`append`** – boolean If true, append to existing file.
- **`file`** – File File to write to.
- **`message`** – String Message to write.
- **`level`** – String ["error", "warning", "info", "verbose", "debug"]Set the logging level. Level should be one of
  - error
  - warning
  - info
  - verbose
  - debug

  The default is "warning" to ensure that messages are displayed by default when using the -quiet command line option.

Parameters accepted as nested elements:

This Task accepts text in its element body.

11.3 Task utility:LoadFile

Load a file into a property

Parameters:

- **`encoding`** – String Encoding to use for input, defaults to the platform's default encoding.

  For a list of possible values see http://java.sun.com/products/jdk/1.2/docs/guide/internat/encodir (at http://java.sun.com/products/jdk/1.2/docs/guide/internat/encoding.doc.html).
- **`property`** – String Property name to save to.
- **`srcfile`** – File Sets the file to load.
- **`failonerror`** – boolean If true, fail on load error.

Parameters accepted as nested elements:

`<filterchain>` – *(See ?? on page ??)* FilterChain may contain a chained set of filter readers.

Parameters:

- **`refid`** – Reference

11.4 Task utility:LoadProperties

Load a file's contents as Ant properties.

Parameters:

- **`classpath`** – Path Set the classpath to use when looking up a resource.
- **`classpathref`** – Reference Set the classpath to use when looking up a resource, given as reference to a <path>defined elsewhere
- **`encoding`** – String Encoding to use for input, defaults to the platform's default encoding.

  For a list of possible values see http://java.sun.com/products/jdk/1.2/docs/guide/internat/encodir (at http://java.sun.com/products/jdk/1.2/docs/guide/internat/encoding.doc.html).
- **`resource`** – String Set the resource name of a property file to load.
- **`srcfile`** – File Set the file to load.

Parameters accepted as nested elements:

`<filterchain>` – *(See ?? on page ??)* FilterChain may contain a chained set of filter readers.

Parameters:

- **`refid`** – Reference

`<classpath>` – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
```

```
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

11.5 Task utility:Patch

Patches a file by applying a 'diff' file to it; requires "patch" to be on the execution path.

Parameters:

- **`strip`** – int Strip the smallest prefix containing *num* leading slashes from filenames.

  patch's *-p* option.
- **`dir`** – File The directory to run the patch command in, defaults to the project's base directory.
- **`destfile`** – File The name of a file to send the output to, instead of patching the file(s) in place; optional.
- **`quiet`** – boolean Work silently unless an error occurs; optional, default=false
- **`originalfile`** – File The file to patch; optional if it can be inferred from the diff file
- **`patchfile`** – File The file containing the diff output; required.

- **reverse** – boolean Assume patch was created with old and new files swapped; optional, default=false
- **backups** – boolean flag to create backups; optional, default=false
- **ignorewhitespace** – boolean flag to ignore whitespace differences; default=false

11.6 Task utility:PathConvert

Converts path and classpath information to a specific target OS format. The resulting formatted path is placed into the specified property.

Parameters:

- **pathsep** – String Set the default path separator string; defaults to current JVM {@link java.io.File#pathSeparator File.pathSeparator}.
- **refid** – Reference Add a reference to a Path, FileSet, DirSet, or FileList defined elsewhere.
- **dirsep** – String Set the default directory separator string; defaults to current JVM {@link java.io.File#separator File.separator}.
- **setonempty** – boolean Set whether the specified property will be set if the result is the empty string.
- **property** – String Set the name of the property into which the converted path will be placed.
- **targetos** – String [”windows”, ”unix”, ”netware”, ”os/2”, ”tandem”]Set targetos to a platform to one of ”windows”, ”unix”, ”netware”, or ”os/2”; current platform settings are used by default.

Parameters accepted as nested elements:

<path> – *(See* **??** *on page* **??***)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
```

```
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **`refid`** – Reference
- **`path`** – String

`<map>` Helper class, holds the nested <map>values. Elements will look like this: <map from="d:" to="/foo"/>When running on windows, the prefix comparison will be case insensitive.

Parameters:

- **`from`** – String
- **`to`** – String

`<mapper>` – *(See ?? on page ??)* Element to define a FileNameMapper.

Parameters:

- **`classpath`** – Path
- **`classpathref`** – Reference
- **`refid`** – Reference
- **`type`** – String ["identity", "flatten", "glob", "merge", "regexp", "package", "unpackage"]
- **`classname`** – String
- **`from`** – String
- **`to`** – String

11.7 Task utility:Sleep

Sleep, or pause, for a period of time. A task for sleeping a short period of time, useful when a build or deployment process requires an interval between tasks.

A negative value can be supplied to any of attributes provided the total sleep time is positive, pending fundamental changes in physics and JVM execution times

Note that sleep times are always hints to be interpreted by the OS how it feels small times may either be ignored or rounded up to a minimum timeslice. Note also that the system clocks often have a fairly low granularity too, which complicates measuring how long a sleep actually took.

Parameters:

- **`hours`** – int hours to add to the sleep time.
- **`milliseconds`** – int milliseconds to add to the sleep time
- **`minutes`** – int minutes to add to the sleep time
- **`failonerror`** – boolean flag controlling whether to break the build on an error.
- **`seconds`** – int seconds to add to the sleep time

11.8 Task utility:Tstamp

Sets properties to the current time, or offsets from the current time. The default properties are TSTAMP, DSTAMP and TODAY;

Parameters:

- **`prefix`** – String Set a prefix for the properties. If the prefix does not end with a "." one is automatically added

Parameters accepted as nested elements:

`<format>` This nested element that allows a property to be set to the current date and time in a given format. The date/time patterns are as defined in the Java SimpleDateFormat class. The format element also allows offsets to be applied to the time to generate different time values.

Parameters:

- **`locale`** – String

- **unit** – String [”millisecond”, ”second”, ”minute”, ”hour”, ”day”, ”week”, ”month”, ”year”]
- **timezone** – String
- **pattern** – String
- **property** – String
- **offset** – int

11.9 Task utility:record

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

Several recorders can exist at the same time. Each recorder is associated with a file. The filename is used as a unique identifier for the recorders. The first call to the recorder task with an unused filename will create a recorder (using the parameters provided) and add it to the listeners of the build. All subsequent calls to the recorder task using this filename will modify that recorders state (recording or not) or other properties (like logging level).

Some technical issues: the file’s print stream is flushed for ”finished” events (buildFinished, targetFinished and taskFinished), and is closed on a buildFinished event.

Parameters:

- **append** – boolean Whether or not the logger should append to a previous file.
- **name** – String Sets the name of the file to log to, and the name of the recorder entry.
- **loglevel** – String [”error”, ”warn”, ”info”, ”verbose”, ”debug”]Sets the level to which this recorder entry should log to.
- **emacsmode** – boolean Set emacs mode.
- **action** – String [”start”, ”stop”]Sets the action for the associated recorder entry.

---

12 xml Library

12.1 Task xml:AntStructure

Creates a partial DTD for Ant from the currently known tasks.

Parameters:

- **`output`** – File The output file.

12.2 Task xml:xmlproperty

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

Example:

```
<root-tag myattr="true">
  <inner-tag someattr="val">Text</inner-tag>
  <a2><a3><a4>false</a4></a3></a2>
  <x>x1</x>
  <x>x2</x>
</root-tag>
```

this generates the following properties:

```
root-tag(myattr)=true
root-tag.inner-tag=Text
root-tag.inner-tag(someattr)=val
root-tag.a2.a3.a4=false
root-tag.x=x1,x2
```

The *collapseAttributes* property of this task can be set to true (the default is false) which will instead result in the following properties (note the difference in names of properties corresponding to XML attributes):

```
root-tag.myattr=true
root-tag.inner-tag=Text
root-tag.inner-tag.someattr=val
root-tag.a2.a3.a4=false
root-tag.x=x1,x2
```

Optionally, to more closely mirror the abilities of the Property task, a selected set of attributes can be treated specially. To enable this behavior, the "semanticAttributes" property of this task must be set to true (it defaults to false). If this attribute is specified, the following attributes take on special meaning (setting this to true implicitly sets collapseAttributes to true as well):

- **value**: Identifies a text value for a property.
- **location**: Identifies a file location for a property.
- **id**: Sets an id for a property
- **refid**: Sets a property to the value of another property based upon the provided id

- **pathid**: Defines a path rather than a property with the given id.

For example, with keepRoot = false, the following properties file:

```
<root-tag>
  <build>
  <build folder="build">
    <classes id="build.classes" location="${build.folder}/classes"/>
    <reference refid="build.classes"/>
  </build>
  <compile>
    <classpath pathid="compile.classpath">
      <pathelement location="${build.classes}"/>
    </classpath>
  </compile>
  <run-time>
    <jars>*.jar</jars>
    <classpath pathid="run-time.classpath">
      <path refid="compile.classpath"/>
      <pathelement path="${run-time.jars}"/>
    </classpath>
  </run-time>
</root-tag>
```

is equivalent to the following entries in a build file:

```
<property name="build" location="build"/>
<property name="build.classes" location="${build.location}/classes"/>
<property name="build.reference" refid="build.classes"/>

<property name="run-time.jars" value="*.jar/>

<classpath id="compile.classpath">
  <pathelement location="${build.classes}"/>
</classpath>

<classpath id="run-time.classpath">
  <path refid="compile.classpath"/>
  <pathelement path="${run-time.jars}"/>
</classpath>
```

This task *requires* the following attributes:

- **file**: The name of the file to load.

This task supports the following attributes:

- **prefix**: Optionally specify a prefix applied to all properties loaded. Defaults to an empty string.
- **keepRoot**: Indicate whether the root xml element is kept as part of property name. Defaults to true.

- **validate**: Indicate whether the xml file is validated. Defaults to false.
- **collapseAttributes**: Indicate whether attributes are stored in property names with parens or with period delimiters. Defaults to false, meaning properties are stored with parens (i.e., foo(attr)).
- **semanticAttributes**: Indicate whether attributes named "location", "value", "refid" and "path" are interpreted as ant properties. Defaults to false.
- **rootDirectory**: Indicate the directory to use as the root directory for resolving location properties. Defaults to the directory of the project using the task.
- **includeSemanticAttribute**: Indicate whether to include the semantic attribute ("location" or "value") as part of the property name. Defaults to false.

Parameters:

- **`file`** – File The XML file to parse; required.
- **`semanticattributes`** – boolean Attribute to enable special handling of attributes - see ant manual.
- **`keeproot`** – boolean flag to include the xml root tag as a first value in the property name; optional, default is true
- **`prefix`** – String the prefix to prepend to each property
- **`collapseattributes`** – boolean flag to treat attributes as nested elements; optional, default false
- **`validate`** – boolean flag to validate the XML file; optional, default false
- **`rootdirectory`** – File The directory to use for resolving file references. Ignored if semanticAttributes is not set to true.
- **`includesemanticattribute`** – boolean Include the semantic attribute name as part of the property name. Ignored if semanticAttributes is not set to true.

Parameters accepted as nested elements:

`<xmlcatalog>` – *(See* **??** *on page* **??***)*

This data type provides a catalog of resource locations (such as DTDs and XML entities), based on the OASIS "Open Catalog" standard (at http://oasis-open.org/committees/entity/spec-2001-08-06.html). The catalog entries are used both for Entity resolution and URI resolution, in accordance with the {@link org.xml.sax.EntityResolver EntityResolver} and {@link javax.xml.transform.URIResolver URIResolver} interfaces as defined in the Java API for XML Processing Specification (at http://java.sun.com/xml/jaxp).

Resource locations can be specified either in-line or in external catalog file(s), or both. In order to use an external catalog file, the xml-commons resolver library ("resolver.jar") must be in your classpath. External catalog files may be either plain text format (at http://oasis-open.org/committees/entity/background/9401.html) or XML format (at http://www.oasis-open.org/committees/entity/spec-2001-08-06.html). If the xml-commons resolver library is not found in the classpath, external catalog files, specified in <catalogpath> paths, will be ignored and a warning will be logged. In this case, however, processing of inline entries will proceed normally.

Currently, only <dtd> and <entity> elements may be specified inline; these correspond to OASIS catalog entry types PUBLIC and URI respectively.

The following is a usage example:

```
<xmlcatalog>
  <dtd publicId="" location="/path/to/file.jar" />
  <dtd publicId="" location="/path/to/file2.jar" />
  <entity publicId="" location="/path/to/file3.jar" />
  <entity publicId="" location="/path/to/file4.jar" />
  <catalogpath>
    <pathelement location="/etc/sgml/catalog"/>
  </catalogpath>
  <catalogfiles dir="/opt/catalogs/" includes="**\catalog.xml"
/>
</xmlcatalog>
```

Tasks wishing to use <xmlcatalog> must provide a method called createXMLCatalog which returns an instance of XMLCatalog. Nested DTD and entity definitions are handled by the XMLCatalog object and must be labeled dtd and entity respectively.

The following is a description of the resolution algorithm: entities/URIs/dtds are looked up in each of the following contexts, stopping when a valid and readable resource is found:

1. In the local filesystem
2. In the classpath
3. Using the Apache xml-commons resolver (if it is available)
4. In URL-space

See {@link org.apache.tools.ant.taskdefs.optional.XMLValidateTask XMLValidateTask} for an example of a task that has integrated support for XMLCatalogs.

Possible future extension could provide for additional OASIS entry types to be specified inline.

Parameters:

- **catalogpathref** – Reference
- **classpath** – Path
- **classpathref** – Reference
- **refid** – Reference

12.3 Task xml:xslt

Processes a set of XML documents via XSLT. This is useful for building views of XML based documentation.

Parameters:

- **force** – boolean Set whether to check dependencies, or always generate; optional, default is false.
- **excludes** – String
- **processor** – String Set the name of the XSL processor to use; optional, default trax. Other values are "xalan" for Xalan1 and "xslp" for XSL:P, though the later is strongly deprecated.
- **classpathref** – Reference Set the reference to an optional classpath to the XSL processor
- **basedir** – File Set the base directory; optional, default is the project's basedir.
- **destdir** – File Set the destination directory into which the XSL result files should be copied to; required, unless `in` and `out` are specified.
- **defaultexcludes** – boolean
- **scanincludeddirectories** – boolean Whether to style all files in the included directories as well; optional, default is true.
- **followsymlinks** – boolean
- **extension** – String Set the desired file extension to be used for the target; optional, default is html.
- **in** – File specifies a single XML document to be styled. Should be used with the `out` attribute; ; required if `out` is set
- **classpath** – Path Set the optional classpath to the XSL processor

- **casesensitive** – boolean
- **reloadstylesheet** – boolean Controls whether the stylesheet is reloaded for every transform.

  Setting this to true may get around a bug in certain Xalan-J versions, default is false.
- **includes** – String
- **out** – File Specifies the output name for the styled result from the `in` attribute; required if `in` is set
- **includesfile** – File
- **style** – String Name of the stylesheet to use - given either relative to the project's basedir or as an absolute path; required.
- **excludesfile** – File

Parameters accepted as nested elements:

<patternset> – *(See* **??** *on page* **??***)* Named collection of include/exclude tags.

Moved out of MatchingTask to make it a standalone object that could be referenced (by scripts for example).

Parameters:

- **includes** – String
- **refid** – Reference
- **excludesfile** – File
- **includesfile** – File
- **excludes** – String

<exclude> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<include> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<custom> (Of type ExtendSelector) Selector that selects files by forwarding the request on to other classes.

Parameters:

- **classpath** – Path
- **error** – String
- **classpathref** – Reference
- **refid** – Reference
- **classname** – String

<present> (Of type PresentSelector) Selector that filters files based on whether they appear in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **present** – String ["srconly", "both"]

<none> (Of type NoneSelector) This selector has a collection of other selectors. All of those selectors must refuse to select a file before the file is considered selected by this selector.

Parameters:

- **error** – String
- **refid** – Reference

<not> (Of type NotSelector) This selector has one other selectors whose meaning it inverts. It actually relies on NoneSelector for its implementation of the isSelected() method, but it adds a check to ensure there is only one other selector contained within.

Parameters:

- **error** – String
- **refid** – Reference

<type> (Of type TypeSelector) Selector that selects a certain kind of file: directory or regular.

Parameters:

- **error** – String
- **refid** – Reference
- **type** – String [”file”, ”dir”]

<factory> – *(See Type xml:xslt.Factory on page 269)* The factory element to configure a transformer factory

Parameters:

- **name** – String

<modified> (Of type ModifiedSelector)

Selector class that uses *Algorithm*, *Cache* and *Comparator* for its work. The *Algorithm* is used for computing a hashvalue for a file. The *Comparator* decides whether to select or not. The *Cache* stores the other value for comparison by the *Comparator* in a persistent manner.

The ModifiedSelector is implemented as a **CoreSelector** and uses default values for all its attributes therefore the simpliest example is

```
            >
```

The same example rewritten as CoreSelector with setting the all values (same as defaults are) would be

```
                >
                >
```

And the same rewritten as CustomSelector would be

```
                >
                >
                >
                >
                >
                >
```

All these three examples copy the files from *src* to *dest* using the ModifiedSelector. The ModifiedSelector uses the *PropertyfileCache* , the *DigestAlgorithm* and the *EqualComparator* for its work. The PropertyfileCache stores key-value-pairs in a simple java properties file. The filename is *cache.properties*. The *update* flag lets the selector update the values in the cache (and on first call creates the cache). The *DigestAlgorithm* computes a hashvalue using the java.security.MessageDigest class

with its MD5-Algorithm and its standard provider. The new computed hash-value and the stored one are compared by the *EqualComparator* which returns 'true' (more correct a value not equals zero (1)) if the values are not the same using simple String comparison.

A useful scenario for this selector is inside a build environment for homepage generation (e.g. with Apache Forrest (at http://xml.apache.org/forrest/)).

```
     generate the site using forrest
    >

     upload the changed files


           >
```

Here all **changed** files are uploaded to the server. The ModifiedSelector saves therefore much upload time.

This selector supports the following nested param's:

name	values	description	required
cache	propertyfile	which cache implementation should be used • **propertyfile** - using java.util.Properties	no, defaults to 'propertyfile'
algorithm	hashvalue — digest	which algorithm implementation should be used	
hashvalue - loads the file content into a String and uses its hashValue() method			
digest - uses java.security.MessageDigest class	no, defaults to digest		
comparator	equal — role	which comparator implementation should be used	
equal - simple comparison using String.equals()			
role - uses java.text.RuleBasedCollator class	no, defaults to equal		
update	true — false	If set to *true*, the cache will be stored, otherwise the values will be lost.	no, defaults to true
seldirs	true — false	If set to *true*, directories will be selected otherwise not	no, defaults to true
cache.*	depends on used cache	value is stored and given to the CacheObject for initialisation	depends on used cache
algorithm.*	depends on used algorithm	value is stored and given to the Algorithm-Object for initialisation	depends on used algorithm
comparator.*	depends on used comparator	value is stored and given to the Comparator-Object for initialisation	depends on used comparator

If another name is used a BuildException "Invalid parameter" is thrown.

This selector uses reflection for setting the values of its three interfaces (using org.apache.tools.ant.IntrospectionHelper) therefore no special 'configuration interfaces' has to be implemented by new caches, algorithms or comparators. All present *set*XX methods can be used. E.g. the DigestAlgorithm can use a specified provider for computing its value. For selecting this there is a *setProvider(String providername)* method. So you can use a nested >.

Parameters:

- **`comparator`** – String ["equal", "rule"]
- **`seldirs`** – boolean
- **`algorithm`** – String ["hashvalue", "digest"]
- **`cache`** – String ["propertyfile"]
- **`error`** – String
- **`refid`** – Reference
- **`update`** – boolean

`<param>` The Param inner class used to store XSL parameters

Parameters:

- **`name`** – String
- **`unless`** – String
- **`if`** – String
- **`expression`** – String

`<or>` (Of type OrSelector) This selector has a collection of other selectors, any of which have to select a file in order for this selector to select it.

Parameters:

- **`error`** – String
- **`refid`** – Reference

`<contains>` (Of type ContainsSelector) Selector that filters files based on whether they contain a particular string.

Parameters:

- **`error`** – String
- **`casesensitive`** – boolean
- **`refid`** – Reference
- **`ignorewhitespace`** – boolean
- **`text`** – String

<depend> (Of type DependSelector) Selector that filters files based on whether they are newer than a matching file in another directory tree. It can contain a mapper element, so isn't available as an ExtendSelector (since those parameters can't hold other elements).

Parameters:

- **error** – String
- **targetdir** – File
- **refid** – Reference
- **granularity** – int

<classpath> – *(See ?? on page ??)* This object represents a path as used by CLASSPATH or PATH environment variable.

```
<sometask>
  <somepath>
    <pathelement location="/path/to/file.jar" />
    <pathelement path="/path/to/file2.jar:/path/to/class2;/path/to/class3"
/>
    <pathelement location="/path/to/file3.jar" />
    <pathelement location="/path/to/file4.jar" />
  </somepath>
</sometask>
```

The object implemention `sometask` must provide a method called `createSomepath` which returns an instance of `Path`. Nested path definitions are handled by the Path object and must be labeled `pathelement`.

The path element takes a parameter `path` which will be parsed and split into single elements. It will usually be used to define a path from an environment variable.

Parameters:

- **refid** – Reference
- **path** – String

<different> (Of type DifferentSelector) This selector selects files against a mapped set of target files, selecting all those files which are different. Files with different lengths are deemed different automatically Files with identical timestamps are viewed as matching by default, unless you specify otherwise. Contents are compared if the lengths are the same and the timestamps are ignored or the same, except if you decide to ignore contents to gain speed.

This is a useful selector to work with programs and tasks that don't handle dependency checking properly; Even if a predecessor task always creates its output

files, followup tasks can be driven off copies made with a different selector, so their dependencies are driven on the absolute state of the files, not a timestamp.

Clearly, however, bulk file comparisons is inefficient; anything that can use timestamps is to be preferred. If this selector must be used, use it over as few files as possible, perhaps following it with an <uptodate;>to keep the descendent routines conditional.

Parameters:

- **ignorecontents** – boolean
- **error** – String
- **targetdir** – File
- **refid** – Reference
- **ignorefiletimes** – boolean
- **granularity** – int

<size> (Of type SizeSelector) Selector that filters files based on their size.

Parameters:

- **when** – String [”less”, ”more”, ”equal”]
- **units** – String [”K”, ”k”, ”kilo”, ”KILO”, ”Ki”, ”KI”, ”ki”, ”kibi”, ”KIBI”, ”M”, ”m”, ”mega”, ”MEGA”, ”Mi”, ”MI”, ”mi”, ”mebi”, ”MEBI”, ”G”, ”g”, ”giga”, ”GIGA”, ”Gi”, ”GI”, ”gi”, ”gibi”, ”GIBI”, ”T”, ”t”, ”tera”, ”TERA”, ”Ti”, ”TI”, ”ti”, ”tebi”, ”TEBI”]
- **error** – String
- **refid** – Reference
- **value** – long

<majority> (Of type MajoritySelector) This selector is here just to shake up your thinking a bit. Don’t get too caught up in boolean, there are other ways you can evaluate a collection of selectors. This one takes a vote of the selectors it contains, and majority wins. You could also have an ”all-but-one” selector, a ”weighted-average” selector, and so on. These are left as exercises for the reader (as are the usecases where this would be necessary).

Parameters:

- **error** – String
- **refid** – Reference
- **allowtie** – boolean

<containsregexp> (Of type ContainsRegexpSelector) Selector that filters files based on a regular expression.

Parameters:

- **error** – String
- **refid** – Reference
- **expression** – String

<filename> (Of type FilenameSelector) Selector that filters files based on the filename.

Parameters:

- **error** – String
- **casesensitive** – boolean
- **name** – String
- **refid** – Reference
- **negate** – boolean

<xmlcatalog> – *(See ?? on page ??)*

This data type provides a catalog of resource locations (such as DTDs and XML entities), based on the OASIS "Open Catalog" standard (at http://oasis-open.org/committees/entity/spec-2001-08-06.html). The catalog entries are used both for Entity resolution and URI resolution, in accordance with the {@link org.xml.sax.EntityResolver EntityResolver} and {@link javax.xml.transform.URIResolver URIResolver} interfaces as defined in the Java API for XML Processing Specification (at http://java.sun.com/xml/jaxp).

Resource locations can be specified either in-line or in external catalog file(s), or both. In order to use an external catalog file, the xml-commons resolver library ("resolver.jar") must be in your classpath. External catalog files may be either plain text format (at http://oasis-open.org/committees/entity/background/9401.html) or XML format (at http://www.oasis-open.org/committees/entity/spec-2001-08-06.html). If the xml-commons resolver library is not found in the classpath, external catalog files, specified in <catalogpath> paths, will be ignored and a warning will be logged. In this case, however, processing of inline entries will proceed normally.

Currently, only <dtd> and <entity> elements may be specified inline; these correspond to OASIS catalog entry types PUBLIC and URI respectively.

The following is a usage example:

```
<xmlcatalog>
  <dtd publicId="" location="/path/to/file.jar" />
  <dtd publicId="" location="/path/to/file2.jar" />
  <entity publicId="" location="/path/to/file3.jar" />
  <entity publicId="" location="/path/to/file4.jar" />
  <catalogpath>
    <pathelement location="/etc/sgml/catalog"/>
```

```
  </catalogpath>
  <catalogfiles dir="/opt/catalogs/" includes="**\catalog.xml"
/>
</xmlcatalog>
```

Tasks wishing to use <xmlcatalog> must provide a method called createXMLCatalog which returns an instance of XMLCatalog. Nested DTD and entity definitions are handled by the XMLCatalog object and must be labeled dtd and entity respectively.

The following is a description of the resolution algorithm: entities/URIs/dtds are looked up in each of the following contexts, stopping when a valid and readable resource is found:

1. In the local filesystem
2. In the classpath
3. Using the Apache xml-commons resolver (if it is available)
4. In URL-space

See {@link org.apache.tools.ant.taskdefs.optional.XMLValidateTask XMLValidateTask} for an example of a task that has integrated support for XMLCatalogs.

Possible future extension could provide for additional OASIS entry types to be specified inline.

Parameters:

- **catalogpathref** – Reference
- **classpath** – Path
- **classpathref** – Reference
- **refid** – Reference

<selector> (Of type SelectSelector) This selector just holds one other selector and forwards all requests to it. It exists so that there is a single selector type that can exist outside of any targets, as an element of project. It overrides all of the reference stuff so that it works as expected. Note that this is the only selector you can reference.

Parameters:

- **error** – String
- **refid** – Reference
- **unless** – String
- **if** – String

<includesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<outputproperty> Specify how the result tree should be output as specified in the specification (at http://www.w3.org/TR/xslt).

Parameters:

- **name** – String
- **value** – String

<mapper> – *(See ?? on page ??)* Element to define a FileNameMapper.

Parameters:

- **classpath** – Path
- **classpathref** – Reference
- **refid** – Reference
- **type** – String ["identity", "flatten", "glob", "merge", "regexp", "package", "unpackage"]
- **classname** – String
- **from** – String
- **to** – String

<date> (Of type DateSelector) Selector that chooses files based on their last modified date.

Parameters:

- **refid** – Reference
- **millis** – long
- **checkdirs** – boolean
- **datetime** – String
- **granularity** – int
- **pattern** – String
- **when** – String ["before", "after", "equal"]
- **error** – String

<excludesfile> inner class to hold a name on list. "If" and "Unless" attributes may be used to invalidate the entry based on the existence of a property (typically set thru the use of the Available task).

Parameters:

- **name** – String
- **unless** – String
- **if** – String

<depth> (Of type DepthSelector) Selector that filters files based on the how deep in the directory tree they are.

Parameters:

- **max** – int
- **error** – String
- **min** – int
- **refid** – Reference

<and> (Of type AndSelector) This selector has a collection of other selectors, all of which have to select a file in order for this selector to select it.

Parameters:

- **error** – String
- **refid** – Reference

12.4 Type xml:xslt.Factory

The factory element to configure a transformer factory

Parameters:

- **name** – String Set the name of the factory

Parameters accepted as nested elements:

<attribute> A JAXP factory attribute. This is mostly processor specific, for example for Xalan 2.3+, the following attributes could be set:

- http://xml.apache.org/xalan/features/optimize (true—false)
- http://xml.apache.org/xalan/features/incremental (true—false)

Parameters: